Gesellschaft für Informatik (GI)

publishes this series in order to make available to a broad public recent findings in informatics (i.e. computer science and information systems), to document conferences that are organized in cooperation with GI and to publish the annual GI Award dissertation.

Broken down into the fields of
• Seminars
• Proceedings
• Dissertations
• Thematics
current topics are dealt with from the fields of research and development, teaching and further training in theory and practice. The Editorial Committee uses an intensive review process in order to ensure the high level of the contributions.

The volumes are published in German or English.

Information: http://www.gi-ev.de/service/publikationen/lni/

ISSN 1617-5468
ISBN 978-3-88579-264-2

This volume contains the contributions to the 5th Conference of the GI special interest group "Sicherheit, Schutz und Zuverlässigkeit" that took place in Berlin on October 5-7, 2010. The main aspects of the conference were secure software development, biometrics, e-commerce, reliability and safety, certification, fault-tolerance, formal methods, critical infrastructure protection, cryptography, network security, privacy enhancing techniques, intrusion detection and prevention, and steganography.

Felix C. Freiling (Hrsg.): Sicherheit 2010

170

GI-Edition

Lecture Notes in Informatics

Felix C. Freiling (Hrsg.)

Sicherheit 2010

Sicherheit, Schutz und Zuverlässigkeit

Beiträge der 5. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)

5.–7. Oktober 2010
Berlin

Proceedings

Felix C. Freiling (Hrsg.)

SICHERHEIT 2010

Sicherheit, Schutz und Zuverlässigkeit

Konferenzband der 5. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)

5.-7. Oktober 2010
in Berlin

Gesellschaft für Informatik e.V. (GI)

Lecture Notes in Informatics (LNI) - Proceedings
Series of the Gesellschaft für Informatik (GI)

Volume P-170

ISBN 978-3-88579-264-2
ISSN 1617-5468

Volume Editor
Prof. Dr. Ing. Felix C. Freiling
Lehrstuhl Praktische Informatik I
Universität Mannheim
A5, 6
68159 Mannheim, Germany
Email: freiling@informatik.uni-mannheim.de

Series Editorial Board
Heinrich C. Mayr, Universität Klagenfurt, Austria (Chairman, mayr@ifit.uni-klu.ac.at)
Hinrich Bonin, Leuphana-Universität Lüneburg, Germany
Dieter Fellner, Technische Universität Darmstadt, Germany
Ulrich Flegel, SAP Research, Germany
Ulrich Frank, Universität Duisburg-Essen, Germany
Johann-Christoph Freytag, Humboldt-Universität Berlin, Germany
Thomas Roth-Berghofer, DFKI, Germany
Michael Goedicke, Universität Duisburg-Essen, Germany
Ralf Hofestädt, Universität Bielefeld
Michael Koch, Universität der Bundeswehr, München, Germany
Axel Lehmann, Universität der Bundeswehr München, Germany
Ernst W. Mayr, Technische Universität München, Germany
Sigrid Schubert, Universität Siegen, Germany
Martin Warnke, Leuphana-Universität Lüneburg, Germany

Dissertations
Dorothea Wagner, Universität Karlsruhe, Germany
Seminars
Reinhard Wilhelm, Universität des Saarlandes, Germany
Thematics
Andreas Oberweis, Universität Karlsruhe (TH)

© Gesellschaft für Informatik, Bonn 2010
printed by Köllen Druck+Verlag GmbH, Bonn

Preface

"Sicherheit, Schutz und Zuverlässigkeit" (SICHERHEIT) is the main conference on safety and security of *Gesellschaft für Informatik e.V.* (GI), the German non-profit organization of computing professionals with around 25.000 members. Having been held before in Frankfurt (2003), Regensburg (2005), Magdeburg (2006) and Saarbrücken (2008), SICHERHEIT 2010 was co-located with the conference *Information Security Solutions Europe* (ISSE) in Berlin and jointly organized by Gesellschaft für Informatik e.V, eema, TeleTrusT and ENISA. By teaming up with ISSE in this one-time endeavour, the official conference language of SICHERHEIT 2010 was English.

The academic program of SICHERHEIT featured 30 papers that were selected in a rigorous peer-review process from 57 submissions by a program committee consisting of 98 experts from the areas of safety and security. All accepted papers are collected within the present volume. Following the tradition, papers could be written in either German or English. Several special interest groups within GI organized Special Sessions that brought additional focus to the program

I would like to express my gratitude to all people who helped to make the event possible. This includes the steering committee and organizational staff of ISSE who mastered the burden of all financial and local arrangments of the joint conference. I am indebted to all reviewers of SICHERHEIT, in particular the members of the program committee and the chairs and co-chairs of the five special sessions. Finally, I would like to thank the organizing committee of SICHERHEIT — Ammar Alkassar, Ulrich Flegel, and Michael Waidner — for sharing their experience in organizing the conference. Last but not least, thanks to Stefan Vömel for his careful help in preparing the proceedings.

Vorwort

Die fünfte Ausgabe der Konferenz "Sicherheit, Schutz und Zuverlässigkeit" (SICHERHEIT) der Gesellschaft für Informatik e.V. fand 2010 in Berlin statt. SICHERHEIT führt als *das* deutsche Diskussionsforum Experten aus Wissenschaft und Wirtschaft für beide Aspekte von Sicherheit zusammen: *Safety* als Schutz vor katastrophalem Fehlverhalten technischer Systeme und *Security* als Schutz informationstechnischer Systeme vor Datenspionage und Datenkorruption. Nach Frankfurt (2003), Regensburg (2005), Magdeburg (2006) und Saarbrücken (2008) wurde SICHERHEIT in 2010 als einmaliges Experiment gemeinsam mit der Konferenz "Information Security Solutions Europe" (ISSE) veranstaltet und von der Gesellschaft für Informatik e.V., eema, TeleTrusT und ENISA gemeinschaftlich organisiert.

Der vorliegende Tagungsband umfasst alle 30 Beiträge des wissenschaftlichen Programms. Diese Beiträge wurden aus insgesamt 57 Einreichungen durch das international besetzte 98-köpfige Programmkomitee ausgewählt. Obwohl die Tagungssprache aufgrund der Kooperation mit ISSE Englisch war, durften traditionsgemäß Beiträge auch auf Deutsch eingereicht werden. Zudem bereicherten mehrere Fachgruppen innerhalb der GI das Programm durch "Special Sessions" zu spezifischen Fragestellungen.

Mein Dank gilt all denen, die sich mit Zeit und Mühe bei der Vorbereitung der Konferenz engagiert haben. Zu ihnen zählen die Mitglieder der ISSE-Leitungsgruppe sowie alle, die vor und hinter den Kulissen für die reibungslose organisatorische und finanzielle Abwicklung der gemeinsamen Tagung sorgten. Außerdem danke ich sämtlichen Mitgliedern des Programmkomitees und allen weiteren Gutachtern. Nicht zuletzt gilt mein herzlicher Dank der Tagungsleitung — Ammar Alkassar, Ulrich Flegel und Michael Waidner — für ihre Unterstützung dieses Projekts sowie Stefan Vömel für seine umsichtige Hilfe bei der Zusammenstellung des Tagungsbandes.

Oktober 2010 *Felix C. Freiling*

Tagungsleitung

- Ammar Alkassar, Sirrix AG, Saarbrücken
- Ulrich Flegel, SAP Research, Karlsruhe
- Felix C. Freiling, Universität Mannheim (Vorsitz)
- Michael Waidner, Fraunhofer SIT und TU Darmstadt, CASED

Programmkomitee

- Ammar Alkassar (Sirrix AG)
- Frederik Armknecht (Universität Bochum)
- Michael Backes (Universität des Saarlandes)
- Thomas Biege (SUSE Linux Products/Novell GmbH/)
- Fevzi Belli (Universität Paderborn)
- Zinaida Benenson (Universität Mannheim)
- Erik-Oliver Blaß (Institut EURECOM, Frankreich)
- Rainer Böhme (ICSI, USA)
- Wolfgang Böhmer (TU Darmstadt)
- Jens Braband (Siemens)
- Arslan Brömme (GI SIG BIOSIG)
- Christoph Busch (Hochschule Darmstadt)
- Jana Dittmann (Universität Magdeburg)
- Falko Dressler (Universität Erlangen)
- Thomas Dübendorfer (Google Switzerland GmbH, Schweiz)
- Peter Ebinger (Fraunhofer IGD)
- Klaus Echtle (Universität Duisburg-Essen)
- Claudia Eckert (TU München)
- Wolfgang Ehrenberger (Hochschule Fulda)
- Bernhard Fechner (FernUniversität Hagen)

- Hannes Federrath (Universität Regensburg)
- Simone Fischer-Hübner (Karlstad University, Schweden)
- Marc Fischlin (TU Darmstadt)
- Ulrich Flegel (SAP Research)
- Felix C. Freiling (Universität Mannheim, Vorsitz)
- Willi Geiselmann (Universität Karlsruhe)
- Sabine Glesner (TU Berlin)
- Dieter Gollmann (TU Hamburg-Harburg)
- Christian Gorecki (Universität Mannheim)
- Ulrich Greveler (FH Münster)
- Rüdiger Grimm (Universität Koblenz)
- Karl-Erwin Großpietsch (Fraunhofer AIS)
- Bernhard Hämmerli (Acris GmbH & Hochschule Luzern, Schweiz)
- Maritta Heisel (Universität Duisburg-Essen)
- Eckehard Hermann (FH Oberösterreich Studienbetriebs GmbH, Österreich)
- Florian Heß (Otto-von-Guericke-Universität Magdeburg)
- Thorsten Holz (TU Wien, Österreich)
- Detlef Hühnlein (secunet Security Networks AG)
- Dieter Hutter (DFKI)
- Luigi Lo Iacono (NEC Labs Europe)
- Jan Jürjens (TU Dortmund und Fraunhofer ISST)
- Jörg Kaiser (Universität Magdeburg)
- Stefan Katzenbeisser (TU Darmstadt)
- Jörg Keller (FernUniversität Hagen)
- Dogan Kesdogan (Universität Siegen)
- Matthias Krause (Universität Mannheim)
- Ioannis Krontiris (Universität Mannheim)

- Ulrich Kühn (DZ Bank AG)
- Ralf Küsters (Universität Trier)
- Hanno Langweg (eQ-3 Entwicklung GmbH)
- Pavel Laskov (Universität Tübingen)
- Kerstin Lemke-Rust (Hochschule Bonn-Rhein-Sieg)
- Stefan Lucks (Universität Weimar)
- Erik Mähle (Universität Lübeck)
- Heiko Mantel (TU Darmstadt)
- Mark Manulis (TU Darmstadt)
- Michael Meier (TU Dortmund)
- Ulrike Meyer (RWTH Aachen)
- Martin Mink (TU Darmstadt)
- Holger Morgenstern (Sachverständigenbüro Morgenstern)
- Günter Müller (Universität Freiburg)
- Isabel Münch (BSI)
- Jens Nedon (ConSecur GmbH)
- Edgar Nett (Universität Magdeburg)
- Alexander Nouak (Fraunhofer IGD)
- Michael Nüsken (Universität Bonn)
- Rolf Oppliger (eSECURITY Technologies, Schweiz)
- Daniel Pähler (Universität Koblenz)
- Günther Pernul (Universität Regensburg)
- Andreas Pfitzmann (TU Dresden)
- Norbert Pohlmann (FH Gelsenkirchen)
- Joachim Posegga (Universität Passau)
- Kai Rannenberg (Goethe-Universität Frankfurt)
- Rolf Reinema (Vodafone D2 GmbH)

- Konrad Rieck (TU Berlin)
- Heiko Roßnagel (Fraunhofer IAO)
- Ahmad-Reza Sadeghi (Universität Bochum)
- Francesca Saglietti (Universität Erlangen-Nürnberg)
- Dirk Schadt (SPOT Consulting)
- Werner Schindler (BSI)
- Sebastian Schmerl (Universität Cottbus)
- Guido Schryen (RWTH Aachen)
- Jörg Schwenk (Universität Bochum)
- Jean-Pierre Seifert (TU Berlin und T-Labs)
- Peter Sobe (Universität Lübeck)
- Hans von Sommerfeld (Rohde und Schwarz)
- Martin Steinebach (Fraunhofer SIT)
- Werner Stephan (DFKI)
- Helmut G. Stiegler (STI Consulting)
- Bernhard Tellenbach (ETH Zürich, Schweiz)
- Roland Vogt (DFKI)
- Melanie Volkamer (TU Darmstadt)
- Horst Wedde (Universität Dortmund)
- Andreas Westfeld (HTW Dresden)
- Carsten Willems (CWSE GmbH)
- Bernhard C. Witt (it.sec GmbH & Co. KG)
- Christopher Wolf (Ruhr-Universität Bochum)
- Xuebing Zhou (Fraunhofer IGD)

Zusätzliche Gutachter

- Timo Bartkewitz (Ruhr-Universität Bochum)
- Sebastian Clauß (TU Dresden)
- Andreas Dewald (Universität Mannheim)
- Christian Dietrich (Fachhochschule Gelsenkirchen)
- Markus Engelberth (Universität Mannheim)
- Stephan Faßbender (TU Dortmund)
- Elke Franz (TU Dresden)
- Jan Göbel (Universität Mannheim)
- Martin Hirsch (Fraunhofer ISST)
- Matthias Kirchner (TU Dresden)
- Stefan Köpsell (TU Dresden)
- Michael Müter (Daimler AG)
- Dennis Royer (Goethe-Universität Frankfurt, Sirrix AG)
- Holger Schmidt (TU Dortmund)
- Jan Stijohann (Universität Duisburg-Essen)
- Henning Sudbrock (TU Darmstadt)
- Philipp Trinius (Universität Mannheim)
- Christian Weber (Goethe-Universität Frankfurt)
- Jan Zibuschka (Fraunhofer IAO)

Organisation der Special Sessions

BIOSIG - Biometrics and Digital Signatures

- Alexander Nouak (Chair)
- Heiko Roßnagel (Co-Chair)

FERS - Dependability and Fault-Tolerance

- Jörg Keller (Chair)
- Karl-Erwin Großpietsch (Co-Chair)

CRYPTO - Theory and Practice of Cryptography

- Christopher Wolf (Chair)
- Frederik Armknecht (Co-Chair)

STEWA - Multimedia Security

- Martin Steinebach (Chair)
- Jana Dittmann (Co-Chair)

SIDAR - Reactive Security

- Michael Meier (Chair)
- Sebastian Schmerl (Co-Chair)

Inhaltsverzeichnis

Biometrics and Digital Signatures

Dependability and Fault-Tolerance

Security Systems

Multimedia Security

Reactive Security

Theory and Practice of Cryptography

Development of Secure Systems

Models and Metrics for Secure Systems

Network Security

Biometrische Template-Protection-Verfahren und Interoperabilitätsstrategien

Christoph Busch [1], Sebastian Abt [1], Claudia Nickel [1], Ulrike Korte [2], Xuebing Zhou [3]

1: Hochschule Darmstadt / CASED
Haardtring 100, 64295 Darmstadt
christoph.busch@h-da.de
2: Bundesamt für Sicherheit in der Informationstechnik (BSI)
3: Fraunhofer-Institut für Graphische Datenverarbeitung (IGD)

Abstract: Biometrische Authentisierung wird häufig zur Verbesserung der Identitätsverifikation eingesetzt. Durch die Nutzung biometrischer Verfahren entstehen neue Herausforderungen an den Schutz der Privatsphäre betroffener Personen. In biometrischen Systemen gespeicherte Referenzdaten enthalten Informationen, die aus den biometrischen Charakteristika einer Person abgeleitet wurden. Das Speichern von Abbildern einer biometrischen Charakteristik (z.B. Fingerbilder) in einer Datenbank ist aus Datenschutzsicht ungeeignet, da die Charakteristik selbst nach einer etwaigen Korrumpierung der Datenbank nicht ersetzt werden kann. Des Weiteren ist die Anzahl der biometrischen Charakteristika eines Nutzers begrenzt. Biometrische Merkmale werden z.B. aus einem Fingerbild extrahiert und in einem Template gespeichert. Eine Mehrfachnutzung von Templates in verschiedenen Anwendungen kann zu sog. Cross-Matching-Problemen führen, wenn Anwendungen miteinander verknüpft werden. Darüber hinaus können Referenzdaten für die Authentisierung irrelevante Informationen enthalten (z.B. ethnische Zugehörigkeit, Krankheiten). Zur Lösung dieser Herausforderungen hat sich mit den Template-Protection-Verfahren eine Technologie entwickelt, die den Anforderungen des Datenschutzes gerecht wird. Offene Systeme erfordern jedoch die Möglichkeit zum Austausch von interoperablen Referenzdatensätzen. Dieser Beitrag betrachtet daher Sicherheitsanforderungen an biometrische Systeme, behandelt die aktuellen Standardisierungsbemühungen zu Biometric-Template-Protection und schlägt eine weitere Vorgehensweise vor.

1 Einführung

Die steigende Nachfrage an Verbesserungen der Sicherheit in der Grenzkontrolle und die zunehmende Anzahl elektronischer Transaktionen, die über kabelgebundene und drahtlose Netzwerke getätigt werden, haben einen starken Bedarf an einem zuverlässigeren Identitätsmanagement erweckt. Existierende, besitzbasierte Identifikationsmethoden (zum Beispiel eine ID Karte) oder wissensbasierte Methoden (z.B. PIN oder Passwort) sind mit Nachteilen verbunden: Diese Authentisierungsfaktoren können vergessen, verloren, verteilt oder gestohlen werden. Biometrie als ergänzender oder ersetzender Faktor kann zu einer höheren Zuverlässigkeit der Verifikation von Identitätsbehauptungen beitragen und im gleichen Zuge einen

höheren Nutzerkomfort bedeuten, da biometrische Charakteristika nur schwer vergessen werden oder verloren gehen können.

Die Nutzung von Biometrie zur Identitätsverifikation hat jedoch auch Bedenken aufgeworfen. Die enge Verbindung biometrischer Verifikationsmethoden zu physikalischen, anatomischen Eigenschaften der betroffenen Personen ermöglicht die Nutzung biometrischer Messdaten für andere als die beabsichtigten Verwendungszwecke und kann somit eine Gefährdung der Privatsphäre darstellen, die sich in die folgenden vier Kategorien unterteilen lässt:

Nichtauthorisierte Erfassung: Erfassung biometrischer Samples ohne das Wissen der betroffenen Person, zum Beispiel durch Verwendung versteckter Kameras.

Unnötige Erfassung: Anwendung biometrischer Methoden ohne oder mit nur wenig zusätzlichem Nutzen im Vergleich zu gewöhnlicher Nutzerverifikation.

Nichtauthorisierte Verwendung und Preisgabe: Nutzung biometrischer Verfahren für andere als die von der betroffenen Person genehmigten Zwecke.

Schleichende Erweiterung des Verwendungsrahmens: Erweiterung eines Systems in Bereiche, in denen die Verwendung ursprünglich nicht vorgesehen war.

Die für das Verarbeiten biometrischer Daten zu beachtende Direktive 95/46/EC über den Schutz personenbezogener Daten und über die freie Verfügbarkeit derartiger Daten gibt keine eindeutige Vorgabe zum Einsatz von biometrischen Verfahren. Der Artikel 29 EU Beratungsausschuss für Datenschutz und Privatsphäre hat daher in seinem im Jahre 2003 veröffentlichten Arbeitspapier über Biometrie [Par03] die Bedeutung von den Schutz der Privatsphäre verbessernden Technologien hervorgehoben, um hierdurch biometrische Systeme zu fördern, die eine dem Schutz der Privatsphäre und dem Schutz der Daten freundliche Architektur aufweisen und übermäßiges Sammeln von Daten und einen ungesetzmäßigen Umgang mit diesen Daten erschweren bzw. verhindern.

Dieser Beitrag widmet sich dem Schutz von biometrischen Referenzdaten und daraus abgeleiteten Interoperabilitätsstrategien. Dazu werden zunächst in Kapitel 2 die Sicherheitseigenschaften biometrischer Systeme betrachtet. In Kapitel 3 wird die für ein offenes, interoperables System notwendige Standardisierung zusammengefasst und in Kapitel 4 besondere Fragen im Zusammenhang mit dem Schutz von Referenzdaten diskutiert. Das Kapitel 5 gibt einen Ausblick auf die weitere Vorgehensweise in der Standardisierung.

2 Sicherheitsaspekte biometrischer Systeme

Die primäre Motivation beim Einsatz biometrischer Verfahren ist die Steigerung der Sicherheit einer Anwendung durch genauere und zuverlässigere Identifikation. Ein möglicher Vorbehalt gegen die Verwendung biometrischer Verfahren ist, dass die erreichte erhöhte Sicherheit mit einem verminderten Schutz der Privatsphäre

einhergehen kann [CS07]. Die Einbeziehung biometrischer Verfahren kann jedoch darüber hinaus in neuen Schwachstellen resultieren. Nach Jain [Jain08] kann das Sicherheitsrisiko eines biometrischen Systems in vier Kategorien unterteilt werden:

- Immanente biometrische Fehler, die häufig durch Wahrscheinlichkeitswerte für Falsch-Akzeptanz und/oder Falsch-Rückweisung ausgedrückt werden.
- Angriff auf die Systemverwaltung.
- Unzulänglich geschützte Infrastruktur, resultierend in Schwachstellen im Zusammenhang mit nicht hinreichend gesicherter Hardware, Software oder Kommunikationskanälen.
- Öffentlichkeit von biometrischen Charakteristika, die versteckte Gewinnung biometrischer Samples und die Erzeugung von Plagiaten ermöglicht.

Die Beständigkeit biometrischer Charakteristika ist eine für die Erkennungsleistung erstrebenswerte Eigenschaft, hat aber auch Auswirkungen auf die eingeschränkten Möglichkeiten der Risikominimierung in Bezug auf einen Identitätsmissbrauch. Sobald ein biometrisches Charakteristikum einem Diebstahl zum Opfer gefallen ist und einem potentiellen Angreifer in Form eines Plagiates zur Verfügung steht, ist es so gut wie unmöglich, dieses Charakteristikum zu erneuern. Eine Verminderung der einhergehenden Risiken kann jedoch dadurch erreicht werden, dass die Erneuerbarkeit biometrischer Templates[1] in einem Identitätsverifikationssystem, sichergestellt wird. Durch die Erneuerbarkeit wird das Risiko einer unzulänglich geschützten Infrastruktur / Systemverwaltung minimiert und damit auch mittelbar das Risiko einer Plagiat-Erzeugung reduziert.

2.1 Sicherheitsanforderungen

Zur Analyse möglicher Angriffsvektoren auf biometrische Systeme sind zunächst grundlegende Sicherheitsanforderungen zu betrachten. Die Sicherheitsanforderungen lassen sich in die Teilaspekte Vertraulichkeit, Integrität, Verfügbarkeit sowie Erneuerbarkeit und Widerrufbarkeit unterteilen.

Vertraulichkeit ist die Eigenschaft, die den Schutz von Informationen vor nicht autorisiertem Zugriff und unerlaubter Veröffentlichung beschreibt.

Integrität ist die Eigenschaft, die die Unversehrtheit und Korrektheit von Daten und Verfahren sicherstellt. Durch die Überprüfung der Integrität wird eine beabsichtigte oder unbeabsichtigte Modifikation einer biometrischen Referenz oder das Ersetzen einer gespeicherten biometrischen Referenz zum Zwecke eines Angriffs ausgeschlossen.

Verfügbarkeit ist die Eigenschaft eines Systems, bei Bedarf zugänglich und funktionsfähig zu sein. Eine Beeinträchtigung der Funktionsfähigkeit eines

[1] ISO/IEC SC37 Harmonized Biometric Vocabulary: http://www.3dface.org/media/vocabulary.html

biometrischen Systems kann z.B. durch von einem Angreifer vorgenommenes Löschen notwendiger, in einer biometrischen Referenzdatenbank gespeicherter, biometrischer Daten erfolgen.

Erneuerbarkeit und Widerrufbarkeit von biometrischen Referenzen bietet einen Schutz bei einer Kompromittierung des biometrischen Datenspeichers.

2.2 Erzeugung von geschützten biometrischen Referenzen

Bei Erneuerbarkeit und Widerrufbarkeit biometrischer Referenzen handelt es sich um bedeutende Maßnahmen zur Wahrung der Privatsphäre eines Individuums durch Vermeidung unerwünschter Verknüpfungen über Datenbanken hinweg. Erneuerbare biometrische Referenzen werden durch Diversifikation im Erstellungsprozess erzeugt und erlauben das Generieren mehrerer unterschiedlicher biometrischer Referenzen, die sämtlich aus einem Charakteristikum abgeleitet wurden [Bre08]. Dazu werden sogenannte Template-Protection-Verfahren eingesetzt. Die Transformation von biometrischen Templates in geschützte Templates (Referenzen) erfolgt mittels einer Einweg-Funktion, sodass eine Rekonstruktion des ursprünglichen Samples unmöglich wird. Somit erlauben Referenzen, die in unterschiedlichen Anwendungen für eine betroffene Person gespeichert werden, keinerlei Querbezüge zwischen den Anwendungen und geben keine unerwünschte Information über die betroffene Person preis.

3 Standardisierung biometrischer Systeme

Bei der gegebenen großen Bandbreite biometrischer Modalitäten, Sensortypen, Merkmalsextraktionsverfahren und Templateformaten ist für große offene Anwendungen (wie zum Beispiel biometrische Reisepässe, Bürgerkarten oder biometrische Bankkarten) die Interoperabilität gleichzeitig eine große Herausforderung und zwingend erforderlich.

Durch den Einsatz standardisierter Technologien reduziert sich für den Betreiber das Risiko einer sich einstellenden Herstellerabhängigkeit. Bei einem ggf. notwendigen Wechsel eines Herstellers kann der Betreiber des biometrischen Systems daher beträchtliche Migrationskosten sparen, wenn etablierte Standards bei der Einrichtung des Systems berücksichtigt wurden.

3.1 Zusammenwirken der Standardisierungsgremien

Die Standardisierung im Bereich der Informationstechnologie wird von einem Joint Technical Committee (JTC) zwischen der International Organization for Standardization (ISO) und der International Electrotechnical Commission (IEC) erarbeitet. Mit der Biometriestandardisierung beauftragt wurde das im Jahr 2002 etablierte Subcommittee SC37. Parallel dazu werden im Subcommittee SC27 die Sicherheitsverfahren sowie im Subcommittee SC17 SmartCards und deren Kommunikationsprotokollen bearbeitet.

Das SC37 formuliert Datenaustauschformate, nach denen die Repräsentation einer biometrischen Charakteristik, z.B. eines Gesichtsbilds, in einem spezifizierten Datensatz kodiert werden kann. Für den Bereich Biometrische Systeme ist die wichtigste Tätigkeit des SC17 die Bearbeitung des On-Card-Comparison Standards ISO/IEC 24787, der für Token-basierte Systeme relevant ist. Das SC27 beschäftigt sich in der Working Group 5 mit Identity Management Systemen und Privacy-Enhancing-Technologies und behandelt in diesem Kontext auch biometrische Verfahren und den Schutz von biometrischen Referenzdaten im Rahmen der Standardentwicklung des ISO/IEC 24745 „Information technology – Security techniques – Biometric template protection“ [ISOtp]. Mit beiden Standards werden wesentliche Grundlagen für die Sicherheitseigenschaften eines biometrischen Systems definiert.

4 Biometrische Systeme nach ISO/IEC 24745

Biometrische Systeme werden im Wesentlichen zur Authentisierung und Identifikation eines Individuums eingesetzt. Hierzu vergleicht ein biometrisches System eine vom Individuum genommene Probe mit einer oder mehreren gespeicherten biometrischen Referenzen. Bei einer biometrischen Referenz (BR) handelt es sich um ein biometrisches Sample, ein biometrisches Template oder ein biometrisches Modell, das ein Individuum eindeutig innerhalb eines bestimmten Kontextes identifizieren kann.

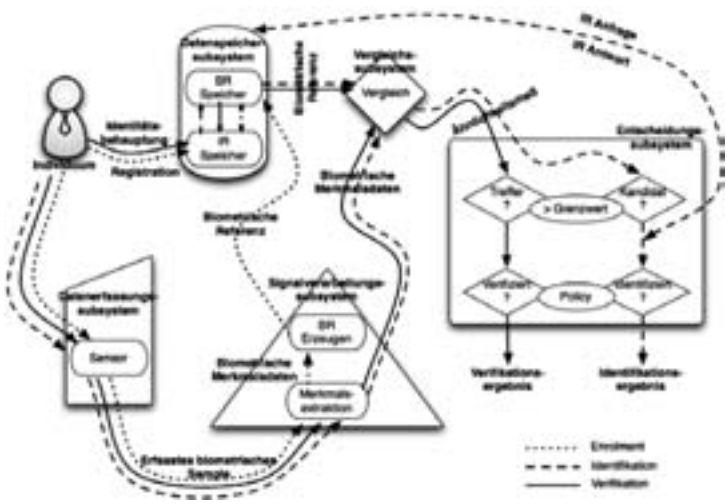


Abbildung 1: Struktur eines biometrischen Systems

Die Architektur eines biometrischen Systems gliedert sich nach [ISOtp] in die folgenden fünf Subsysteme, die in Abbildung 1 dargestellt sind:

Biometrisches Erfassungssubsystem: Beinhaltet Sensoren und bildet die erfassten biometrischen Charakteristika auf biometrische Samples ab.

Signalverarbeitungssubsystem: Extrahiert biometrische Merkmalsdaten aus biometrischen Samples.

Datenspeichersubsystem: Dient zur Speicherung erfasster biometrischer Referenzen und Identitätsreferenzen, meist in separaten Datenbanken.

Vergleichssubsystem: Vergleicht erfasste biometrische Samples mit gespeicherten biometrischen Referenzen und liefert ein Ähnlichkeitsmaß (Vergleichswert).

Entscheidungssubsystem: Entscheidet auf Grund des Ähnlichkeitsmaßes über die Identität des zum erfassten biometrischen Sample gehörenden Individuums.

Biometrische Referenzen stellen sensible personenbezogene Daten dar, die entweder unmittelbar (z.B. Gesichtsfoto) oder indirekt (z.B. Fingerabdruck-Minutien) zur Identifizierung einer Person und auf Grund deren Eindeutigkeit potenziell als eindeutiger Identifikator (Universal Unique Identifier - UUID) zur datenbankübergreifenden Verknüpfung von Daten genutzt werden können. Dies stellt eine Gefährdung der Privatsphäre des Individuums dar. Insbesondere sollten biometrische Daten im Besitz und unter Kontrolle der betroffenen Person bleiben und biometrische Samples von biometrischen Systemen nur gespeichert werden, wenn dies dringend erforderlich ist. Weiterhin sollte ein biometrisches System Mechanismen zum Erzeugen diversifizierbarer Referenzen zur Verfügung stellen, um das Widerrufen und Erneuern biometrischer Referenzen zu ermöglichen.

4.1 Sicherheitsgefährdungen und Gegenmaßnahmen

Jedes der im vorigen Abschnitt beschriebenen Subsysteme (Datenerfassungssubsystem, Signalverarbeitungssubsystem, Datenspeichersubsystem, Vergleichssubsystem, Entscheidungssubsystem) sowie die zwischen den Subsystemen liegenden Kommunikationskanäle besitzen eigene Angriffsvektoren. Tabelle 1 gibt eine Übersicht über Gefährdungen der Subsysteme und die in ISO/IEC 24745 vorgeschlagenen Gegenmaßnahmen.

Subsystem	*Gefährdungen*	*Gegenmaßnahmen*
Erfassungssubsystem	Sensor Spoofing mit Plagiaten	- Lebenderkennung - Multimodale Biometrie - Challenge/Response
Signalverarbeitungs-subsystem	Einfügen gefälschter Daten	- Verwendung geprüfter und freigegebener Algorithmen
Vergleichs-subsystem	Manipulation von Ähnlichkeitsmaßen (berechneten Vergleichswerten)	- Sicherung des Servers und/oder Clients - Geschützte Implementierung (z.B. On- Card-Comparison)
Datenspeicher-subsystem	Kompromittierung der Datenbank	- Verwendung erneuerbarer biometrischer Referenzen

Subsystem	*Gefährdungen*	*Gegenmaßnahmen*
		- Datenseparation - Zugriffskontrolle
	- Unautorisierte Veröffentlichung personenbezogener Daten - Unautorisiertes Austauschen von gespeicherten Daten (BR, IR) - Unautorisierte Modifikation von BR, IR	- Zugriffskontrollen - Sicherung von BR, IR durch elektronische Signaturen - Sicherung von BR, IR durch Verschlüsselung
Entscheidungs-subsystem	Kontinuierliche Modifikation eines biometrischen Samples zur Erreichung der notwendigen Entscheidungsgrenzwerte (Hill-Climbing Attacke)	- Verwendung grob quantisierter Vergleichswerte - Sichere Kommunikationskanäle

Tabelle 1: Gefährdungen biometrischer Subsysteme und Gegenmaßnahmen.

Oft sind biometrische Systeme als verteilte Implementierungen realisiert, so dass sensible Daten zwischen den Subsystemen ausgetauscht werden. Tabelle 2 beschreibt während der Datenübertragung entstehende Gefährdungen und Gegenmaßnahmen.

Kommunikations-verbindung(en)	*Übertragene Daten*	*Gefährdungen*	*Gegenmaßnahmen*
Datenerfassungs-subsystem ↔ Signalverarbeitungs-subsystem ↔ Vergleichs-subsystem	Biometrisches Sample und Merkmalsdaten	Abhören der Daten	Einsatz einer verschlüsselten Kommunikationsverbindung
		Wiederholung (Replay-Attacke)	Einsatz von Challenge-Response-Verfahren
		Brute Force	Fehlbedienungszähler
Datenspeicher-subsystem ↔ Vergleichs-subsystem	Biometrische Referenz	Abhören der Daten	Einsatz einer verschlüsselten Kommunikationsverbindung
		Wiederholte Datenübermittlung (Replay-Attacke)	Einsatz von Challenge-Response-Verfahren
		Man-in-the-middle Angriff	Einsatz einer Ende-zu-Ende verschlüsselten Kommunikationsverbindung und Authentifikation der Teilnehmer
		Hill-Climbing Attacke	Verwendung grob dargestellter Vergleichswerte
Vergleichs-subsystem ↔ Entscheidungs-subsystem	Vergleichswert	Manipulation des Vergleichswertes	Einsatz einer verschlüsselten Kommunikationsverbindung

Tabelle 2: Durch Datenübertragung auftretende Gefährdungen und Gegenmaßnahmen.

Zusätzlich zu den in Tabelle 1 und 2 dargestellten technischen Gegenmaßnahmen existieren weitere administrative Gegenmaßnahmen zum Schutz biometrischer Systeme und Daten. Siehe hierzu ITU-T X.tpp-1 [ISOe] und ISO 19092:2008 [ISOf].

4.2 Erneuerbare biometrische Referenzen

Erneuerbare biometrische Referenzen bestehen aus einem pseudonymen Identifikator (PI) sowie dazugehörenden unterstützenden Daten (Auxilliary Data - AD), die während des Enrolmentprozesses aus einem oder mehreren biometrischen Samples erzeugt werden.

Pseudonyme Identifikatoren (PI) sind diversifizierbare, geschützte binäre Strings. Ein pseudonymer Identifikator gibt keine Informationen preis, die Aufschluss über die ursprünglich erhobenen Daten, das zu Grunde liegende biometrische Template oder die wahre Identität dessen Besitzers geben. Der Prozess zur Erzeugung pseudonymer Identifikatoren wird in Abbildung 2 dargestellt. Während einer Enrolmentphase wird für ein Individuum eine biometrische Referenz generiert. Innerhalb dieses Prozesses werden von einem Sensor ein oder mehrere biometrische Samples erzeugt und im Anschluss von einem Feature-Extraktor zur Erzeugung biometrischer Merkmale verwendet. Abschließend werden von einem Pseudonymous-Identifikator-Encoder (PIE) ein pseudonymer Identifikator sowie möglicherweise benötigte unterstützende Daten erzeugt.

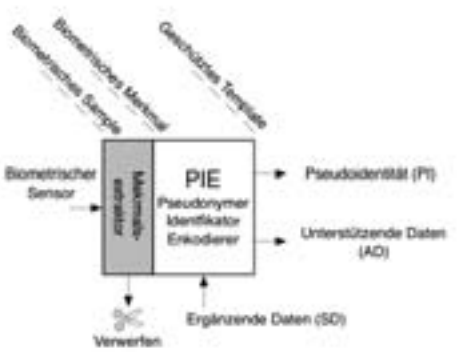


Abbildung 2: Erzeugung geschützter Templates

In den Prozess einfließende ergänzende Daten (Supplementary Data - SD) können Sicherheitsverbesserungen durch besitz- oder wissensbasierte Schlüssel bewirken, die vom Enrollee eingegeben werden müssen (z.B. biometrisch gehärtete Kennwörter). Alternativ können benutzer-spezifische Parameter als SD gespeichert werden.

Die Kombination von pseudonymem Identifikator und unterstützenden Daten wird als ein geschütztes Template bezeichnet. Sowohl der pseudonyme Identifikator, als auch die unterstützenden Daten werden nach deren Erzeugung gespeichert, wohingegen die

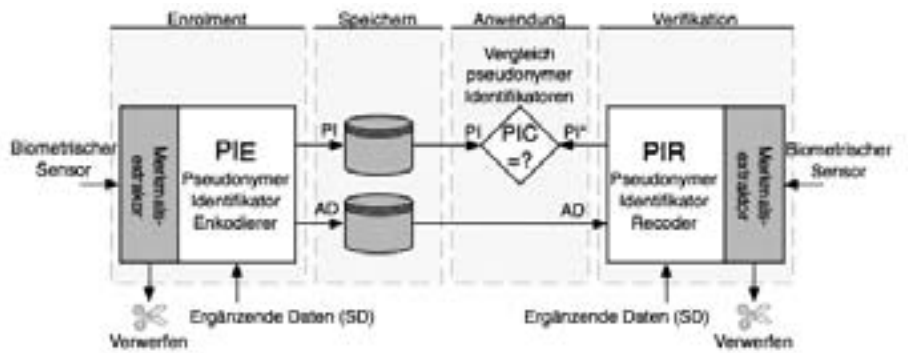


Abbildung 3: Referenzarchitektur eines Systems zum Schutz biometrischer Templates.

erfassten biometrischen Merkmale zerstört werden.

Für die Verifikation wird ein pseudonymer Identifikator neu erzeugt, der dann mit dem während des Enrolments erzeugten PI verglichen wird. Dazu wurden in den letzten zehn Jahren etliche Verfahren vorgeschlagenz.B. [SRS+98], [JW99], [DRS04], [TAK+05], [ST06], [NJP07], [RCCB07]. Die Verifikation wird hierbei durch die Transformation eines Proben-Samples in einen neuen pseudonymen Identifikator PI* unter Verwendung der bereitgestellten unterstützenden Daten erreicht. Ergänzende Daten aus der Enrolmentphase müssen auch dem Pseudonymer-Identifikator-Recoder (PIR) während des Erzeugens des rekonstruierten PI* zur Verfügung gestellt werden. Nach dem Erzeugen von PI* durch den PIR werden alle Eingabedaten, d.h. das biometrische Sample, die Merkmalsdaten und die ergänzenden Daten gelöscht und PI* wird an den Pseudonymen Identifikator-Comparator (PIC) übergeben, der PI mit PI* vergleicht. Abbildung 3 gibt einen Überblick über die Gesamtarchitektur zum Erstellen, Speichern und Verifizieren von pseudonymen Identifikatoren. Der pseudonyme Identifikator und die unterstützenden Daten werden auf einem passenden Medium oder auf unterschiedlichen Medien, wie zum Beispiel Datenbanken, Smartcards, Barcodes, etc., gespeichert.

4.3 Anwendungsmodelle biometrischer Systeme

Das Speichern von PI und AD kann auf unterschiedlichen Wegen stattfinden, die sich wie folgt in drei Kategorien einteilen lassen: zentrales Speichern (sowohl PI, als auch AD werden in einer Datenbank gespeichert), lokales Speichern (PI und AD werden gemeinsam auf einem Token gespeichert) und hybrides Speichern durch Separierung von PI und AD (zum Beispiel durch Speichern der unterstützenden Daten auf einem Token und des pseudonymen Identifikators in einer Datenbank). Vorteile des zentralen Speicherns zumindest einer der beiden Datenelemente liegen in der Möglichkeit des Erstellens einer schwarzen Liste, des Realisierens von Prüf-Funktionalitäten (Audits) und des Ermöglichens eines simplen Widerruf-Prozesses. Die Vorteile des lokalen Speicherns sind das Nichtvorhandensein von für zentrale Datenbanken spezifischen Sicherheitsrisiken sowie der vollständige Besitz der Kontrolle über Referenzdaten bei der betroffenen Person. Das hybride Speichern zeichnet sich dadurch aus, dass sowohl die betroffene Person, als auch der Anbieter Kontrolle über die Nutzung der Templatedaten besitzt und die durch eine zentrale Datenspeicherung potentiell entstehenden Sicherheitsrisiken reduziert werden können.)

Notwendige Schutzmaßnahmen für ein biometrisches System können oft erst in der Analyse des Anwendungskontextes ausgewählt werden. Zu diesem Zweck beschreibt ISO/IEC 24745 verschiedene Anwendungsmodelle und unterscheidet diese auf Basis des Speicherortes der Referenzdaten sowie des Vergleichsortes. Die hierbei verwendeten Standorte lassen sich wie folgt beschreiben:

Client: Bei einem Client handelt es sich um einen Arbeitsplatzrechner oder ein äquivalentes Endgerät (z.B. PDA, Smartphone) und angeschlossene Sensoren.

Server: Ein Server ist ein System, das über ein Netzwerk mit einem Client kommuniziert und Daten zur Verfügung stellt bzw. Operationen ausführt.

Token: Ein Token (z.B. SmartCard) kann biometrische Daten speichern und in manchen Fällen auch vergleichen (z.B. On-Card-Comparison).

Die in ISO/IEC 24745 beschriebenen Modelle A bis F sind anwendbar auf erneuerbare biometrische Referenzen unter der Annahme, dass PI und AD am gleichen Ort gespeichert werden. Die Modelle G und H hingegen sind ausschließlich für den Einsatz mit erneuerbaren biometrischen Referenzen anwendbar und beschreiben Modelle der Separation von PI und AD.

Modell A – Speichern und Vergleich auf Server
Modell B – Speichern auf Token, Vergleich auf Server
Modell C – Speichern auf Server, Vergleich auf Client
Modell D – Speichern und Vergleich auf Client
Modell E – Speichern auf Token, Vergleich auf Client
Modell F – Speichern und Vergleich auf Token
Modell G – Verteiltes Speichern auf Token und Server, Vergleich auf Server
Modell H – Verteiltes Speichern auf Token und Client, Vergleich auf Client

Die Modelle G und H beziehen sich ausschließlich auf erneuerbare biometrische Referenzen. Das Konzept der Datenseparierung wird durch verteiltes Speichern der Komponenten erneuerbarer biometrischer Referenzen (IR, PI, AD) umgesetzt. Nach diesem Modell wird ein pseudonymer Identifikator in einem Server-seitigen Datenspeicher hinterlegt. Die hierzu gehörenden unterstützenden Daten (AD) werden jedoch zusammen mit der Identitätsreferenz (IR) auf einem benutzerspezifischen Token gespeichert. Durch die Verteilung der erneuerbaren biometrischen Referenz auf unterschiedliche Systeme wird bei der Durchführung einer Verifikation zwingend das Vorliegen korrekter Daten von beiden Systemen notwendig. Diese Vorgehensweise setzt zur Authentisierung immer die Zustimmung der betroffenen Person voraus. Darüber hinaus ermöglicht dieses Modell ein serverseitiges Widerrufen biometrischer Referenzdaten (PI), ohne hierzu Zugriff auf das Token zu benötigen.

5 Interoperabilität von erneuerbaren Referenzen

Durch die bisherigen Standardisierungsaktivitäten im Rahmen von ISO/IEC 24745 wurde ein Rahmen für ein biometrisches System mit Mechanismen zum Erzeugen diversifizierbarer Referenzen definiert. Dieser Architekturrahmen ist ein normativer Bestandteil von ISO/IEC 24745 geworden. Konforme Implementierungen müssen daher die beschriebenen Anforderungen zum Widerrufen und Erneuern biometrischer Referenzen erfüllen. Damit ist jedoch nicht sichergestellt, dass Referenzdaten auch zwischen unterschiedlichen Anwendungen ausgetauscht werden können. Die Vielfalt heute angebotener Template-Protection-Verfahren ist per se nicht interoperabel. Um eine Interoperabilität wie bei bildbasierten Referenzen zu erreichen, sollte für die zukünftige Standardisierungsarbeit ein zwei-stufiger Ansatz verfolgt werden.

In der ersten Stufe sollte mittelfristig ein Datenaustauschformat als Element des ISO/IEC 19794 Multipart-Standards definiert werden, der die relevanten Datenelemente PI und AD kodiert, wobei für herstellereigene Kodierungen der beiden Elemente entsprechende dynamische Datenfelder vorgesehen werden sollten, wie dies auch bei ISO/IEC 19794-2 definiert wurde. Eine Interoperabilität kann erreicht werden, wenn der Record zusätzlich eine registrierte Hersteller-Identifikationsnummer enthält, so dass ein Verifikationssystem den zum PIE korrespondierenden PIR auswählen und den PI* an den Comparator übergeben kann. Ein entsprechender pragmatischer Ansatz wurde von SC37 bereits für die Kodierung von Bildqualitätswerten unterschiedlicher Hersteller eingesetzt.

In einer zweiten Stufe sollte langfristig ein Auswahl-Wettbewerb zu einem standardisierten Verfahren gestartet werden, wie er ähnlich in der Vergangenheit mit dem Advanced Encryption Standard (AES) durchgeführt wurde und derzeit mit der Cryptographic Hash Algorithm Competition durchgeführt wird. Ein solcher Auswahl-Wettbewerb verlangt transparente Algorithmen, die eine Sicherheitsevaluierung des Template-Protection-Verfahrens ermöglichen. In der Evaluierung sollte der Nachweis der Einweg-Eigenschaft (ENW) geprüft werden, um eine Rekonstruktion des Samples auszuschließen. In einer vergleichenden Evaluierung sollten mindestens die Kriterien Entropie (ENT) des erzeugten Referenzdatensatzes, die Diversifikationseigenschaft (DIV) sowie die Undichtigkeit (DIC) geprüft und bewertetet werden. Darüber hinaus ist eine Evaluierung der Erkennungsleistung nach ISO/IEC 19795-1 erforderlich, um die Performanz (PER) zu bewerten. Ein Ranking von Kandidaten kann erfolgen, in dem die Systemgüte durch

$$\text{Score} = 0{,}5\ \text{PER} + 0{,}2\ \text{ENT} + 0{,}2\ \text{DIV} + 0{,}1\ \text{DIC}$$

gewichtet bewertet wird. Erkennungsleistung und Sicherheitseigenschaften eines Verfahrens werden dabei gleichgewichtet gefordert. Dieses Maß für die Systemgüte verfolgt den Zweck, einen Zugewinn an Template-Sicherheit nicht auf Kosten der Erkennungsleistung zu bewerten.

6 Zusammenfassung

Mit der Standardisierung in ISO/IEC 24745 wurde ein einheitliches Verständnis für Sicherheitsanforderungen an biometrische Systeme sowie für Risiken und empfohlene Gegenmaßnahmen erreicht. Mit der Integration der normativen Anforderung von erneuerbaren biometrischen Referenzen wurden wesentliche Datenschutzmechanismen im Standard verankert.

Für System-Betreiber ist neben der Sicherheit aber auch die Interoperabilität von Austauschformaten und damit die Reduzierung von Hersteller-Abhängigkeiten ein wichtiges Ziel. Durch den vorgeschlagenen Auswahl-Wettbewerb kann dieses Ziel langfristig verfolgt und sichere, datenschutzfreundliche und performante biometrische Systeme ermöglicht werden.

7 Danksagung

Diese Arbeit wurde durchgeführt im Rahmen des Projekts "BioKeyS- Pilot-DB" des Bundesamtes für Sicherheit in der Informationstechnik.

Literaturverzeichnis

[Bre08] J. Breebaart, C. Busch, J. Grave, E. Kindt: A Reference Architecture for Biometric Template Protection based on Pseudo Identities, in Proceedings BIOSIG 2008

[CS07] A. Cavoukian und A. Stoianov: Biometric encryption: a positive-sum technology that achieves strong authentication, security and privacy. Whitepaper information and Privacy Commissioner/Ontario, 2007. available from www.ipc.on.ca.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, Adam Smith: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In EUROCRYPT, pages 523–540, 2004.

[ISOe] ISO/IEC 9594-2, ITU-T X.tpp-1 (Telebiometric Protection Procedure-Part1): A guideline of technical and managerial countermeasures for biometric data security

[ISOf] ISO 19092:2008, Financial Services – Biometrics – Security framework

[ISOtp] ISO/IEC CD 24745 Information technology - Security techniques - Biometric template protection

[Jain08] A. Jain, K. Nandakumar, A. Nagar: Biometric Template Security, EURASIP Journal on Advances in Signal Processing, Volume 2008

[JW99] A. Juels und M. Wattenberg: A fuzzy commitment scheme. In Proc. 6th ACMCCCS, pages 28–36, 1999.

[NJP07] K. Nandakumar, A.K. Jain, S. Pankanti: Fingerprint-based fuzzy vault: Implementation and performance. Information Forensics and Security, IEEE Transactions on, 2(4):744–757, Dec. 2007.

[Par03] ARTICLE 29 Data Protection Working Party. Working document on biometrics working document on biometrics. http://ec.europa.eu/justice_home/fsj/privacy/docs/wpdocs/2003/wp80_en.pdf, 2003. Last visited: November 26, 2009

[RCCB07] N. K. Ratha, S. Chikkerur, J. H. Connell, R. M. Bolle: Generating cancelable fingerprint templates. IEEE Trans. pattern analysis and machine intelligence, 29(4):561–572, 2007.

[SRS+98] C. Soutar, D. Roberge, A. Stoianov, R. Gilroy, B. V. K. Vijaya Kumar: In R. L. van Renesse, editor, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, volume 3314 of Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, pages 178–188, April 1998.

[ST06] Berry Schoenmakers, Pim Tuyls: Efficient binary conversion for paillier encrypted values. In EUROCRYPT, pages 522–537, 2006.

[TAK+05] P. Tuyls, A. H. M. Akkermans, T. A. M. Kevenaar, G. J. Schrijen, A. M. Bazen, R. N. J. Veldhuis: Practical biometric authentication with template protection. In Audio and video-based biometric person authentication, pages 436–446.

Biometrische Nachrichten-Authentisierung

Christoph Busch[1,2], Daniel Hartung[2]

[1] Hochschule Darmstadt - CASED
[2] Norwegian Information Security Laboratory (NISlab)
Teknologiveien 22, 2821 Gjøvik, Norwegen
christoph.busch@hig.no
daniel.hartung@hig.no

Abstract: Bei vielen Anwendungen ist die Integrität und Authentizität übertragener Nachrichten von Interesse. So sind zum Beispiel im Online-Banking sind die relevanten Informationen i) welches Empfänger–Konto eine Gutschrift erhält, ii) welcher Betrag dem Empfänger gutgeschrieben werden soll, iii)) welches Sender-Konto eine Belastung erhält und schließlich iv) welche natürliche Person die Transaktion initiiert und die Transaktionsdaten bestätigt hat. In derzeitig eingesetzten Protokollen sind die Informationen i), ii) und iii) vielfach nur ungenügend geschützt. In keinem der derzeitigen Protokolle wird die Information iv) ausreichend gesichert. Das hier vorgestellte Protokoll zur Biometrischen Nachrichten-Authentisierung realisiert eine Daten-Authentisierung und gleichzeitig eine Personen-Authentisierung. Damit wird eine starke Bindung zwischen einer natürlichen Person und den anderen relevanten Informationen hergestellt und somit für den Ausführenden der Transaktion gesichert nachgewiesen, dass tatsächlich eine berechtigte natürliche Person die Transaktion initiiert und bestätigt hat.

1 Bedrohungen und Vorfälle mit Identitätsmissbrauch

Ein Identitätsmissbrauch ist definierbar als Nutzung des Identitätsdiebstahls zum Schaden der betroffenen Person, wobei das vorrangige Interesse des Angreifers in aller Regel eine finanzielle Bereicherung ist. Das Risiko, Opfer eines solchen Ereignisses zu werden, ist in den vergangenen Jahren dramatisch gestiegen. Das Identity Theft Resource Center berichtet für das Jahr 2008 eine Zunahme von 47% im Vergleich zum Vorjahr [Idtc2009a]. Die Liste der Einzelvorfälle dokumentiert zum Beispiel Kreditkartenbetrug, Kontenraub und Bankbetrug und zeigt die zur Beschaffung der notwendigen Informationen eingesetzte Spannbreite von Angriffen. Diese reichen von manipulierten Kartenlesern über Phishing-Angriffe bis hin zu ausgefeilten Social-Engineering-Angriffen, die zur unbedachten Preisgabe von sensitiven Daten motivieren. Diese Gefahren sind auch für Deutschland ein größer werdendes Problem, wie die Statistiken des Bundeskriminalamtes belegen [Bka2008]. Eine Studie des Bundesamtes für Sicherheit in der Informationstechnik prognostiziert, dass die Angriffsszenarien in Zukunft deutlich vielfältiger werden [Bsi2010]. Das Potential für Angriffe steigt durch die zunehmende Nutzung von Online-Banking-Diensten. In Deutschland gab es beispielsweise in den letzten Jahren eine Steigerung von 15 Millionen Online-Konten im Jahr 2000 auf 39 Millionen Online-Konten im Jahr 2008 [Bdb2006], [Grud2009]. Nach einer Studie des BITKOM haben sieben Prozent aller Internet-Nutzer über 14 Jahren

bereits einen finanziellen Schaden beispielsweise durch Viren, bei Online-Auktionen oder Online-Banking erlitten [Bit2008]. Neben Online-Banking Transaktionen sind auch andere sicherheitskritische Anwendungen bedroht, wie beispielsweise die authentische Kommunikation zwischen Einsatzeinheiten eines Krisenstabes im Katastrophenmanagement. Insbesondere bei der Koordination von Einheiten, die ad hoc an einem Katastrophenort zusammengezogen werden wie etwa bei einer terroristisch bedingten Katastrophe ist es essentiell, dass die Authentizität von Nachrichten für den Empfänger einer Handlungsanweisung nachweisbar wird.

Nach dem letzten Lagebericht des Bundesamtes für Sicherheit in der Informationstechnik (BSI) geht die Bedrohung weniger von Phishing-Angriffen aus [Bsi2009]. Die Bedrohung wächst vielmehr durch die immer ausgereifteren Mechanismen von bösartiger Software (Malicious Software – Malware), die über verschiedenste Kanäle auf privaten Rechnern installiert wird und dort - ohne Kenntnis des Endanwender - Informationen über verwendete Programme und Nutzdaten wie etwa Finanz-Transaktionen aufzeichnet und an einen entfernten Steuerrechner über das Internet weiterleitet. Zu diesen Malware-Arten zählen Computer-Viren und Trojanische Pferde. Die dabei zum Einsatz kommende Malware ist für das Opfer nur in seltenen Fällen erkennbar. Dies liegt einerseits daran, dass sie ausgefeilte Mechanismen wie Selbstverschlüsselung und Mutation verwenden und somit beim Abgleich mit den Virenmustern in Datenbanken der Anti-Virenhersteller unerkannt bleiben. Andererseits werden Mechanismen wie Rootkits eingesetzt, die das Betriebssystem selbst unterwandern und mit heutigen Methoden kaum zu detektieren sind [Rut2006].

2 Biometrische Authentisierung

Unter Biometrie versteht man ein Messverfahren zur Wiedererkennung von Personen. Die Internationale Standardisierung definiert den Begriff *biometrics* wie folgt: "*automated recognition of individuals based on their behavioural and biological characteristics*" [Iso-sc37]. Biometrische Verfahren analysieren demnach das Verhalten des Menschen und/oder eine Eigenschaft der biologischen Charakteristika. Die biologischen Charakteristika gliedern sich einerseits in anatomische Charakteristika, die geprägt werden durch Strukturen des Körpers und andererseits in physiologische Charakteristika, die geprägt werden durch Funktionen des Körpers wie beispielsweise die Erkennung der Stimme. Der Vorgang der biometrischen Authentisierung bedingt, dass grundsätzlich eine Person vorab eingelernt wurde (Enrolment), um die notwendigen Referenzdaten zu bilden. Biometrische Authentisierungsverfahren werden in sicherheitsrelevanten Anwendungen substituierend oder ergänzend zu anderen Authentisierungsfaktoren wie Wissens-Authentisierung (Passwort) oder Besitz-Authentisierung über Token (Schlüssel) eingesetzt, um deren Nachteile zu kompensieren. Passworte und Token können – meist unter Missachtung einer Sicherheitsrichtlinie – weitergeben werden, sie werden vergessen oder verloren. Um bei der ansteigenden Zahl der logischen und physikalischen Zugangskontrollen dem Verlust vorzubeugen, werden oft ungeeignete Speicherorte oder identische Passworte verwendet. Im Gegensatz dazu können biometrische Charakteristika nicht vergessen gehen und naturgemäß ist keine Delegation möglich.

2.1 Sicherheit biometrischer Systeme

Die Bedrohungen der Sicherheit biometrischer Systeme und geeignete Schutzmaßnahmen sind aus der Literatur hinreichend bekannt. Mögliche Angriffe sind denkbar auf den biometrischen Sensor und auf die gespeicherten Referenzdaten [Iso-sc27]. Die Robustheit des Sensors ist vor allem in einem nicht-überwachten Anwendungsumfeld von Bedeutung; die Sicherheit des Gesamtsystems erfordert, dass von einem Angreifer präsentierte Plagiate (z.B. Gummifinger) einer biometrischen Charakteristik zuverlässig als solche erkannt werden. Alternativ können biometrische Modalitäten wie etwa die Venenerkennung zum Einsatz kommen, bei denen die erfolgreiche Produktion eines Plagiates als unwahrscheinlich eingestuft werden kann.

Aus der Sicht einer Betroffenen Person ist die Sicherheit eines biometrischen Systems jedoch auch verbunden mit den Maßnahmen zum Schutz der biometrischen Referenzdaten. Erwartet werden technische Maßnahmen, die es ermöglichen, Referenzdaten zurückzurufen, einen Querbezug zwischen verschiedenen Anwendungen zu verhindern und potentiell in der biometrischen Charakteristik enthaltene Zusatzinformationen nicht zugänglich werden zu lassen. Zur Lösung dieser Anforderungen gibt es verschiedene Ansätze des *Biometric Template Protection* [Bre2008], die das Speichern von Bild- oder Templatedaten in einer Datenbank entbehrlich machen. Ein bereits kommerziell eingesetztes Verfahren ist das *Helper-Data-Schema* [Tak05], das im Folgenden skizziert wird.

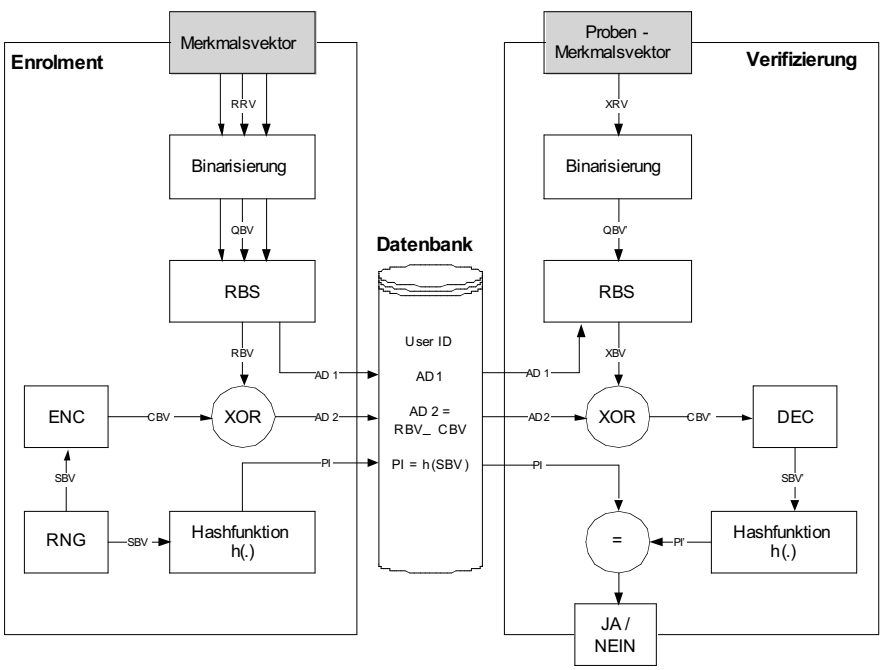


Abbildung 1: Helper-Data-Schema zum Schutz biometrischer Referenzdaten

Um die biometrische Erkennungsleistung zu steigern, wird im Helper-Data-Schema ein Reliable-Bit-Selector (RBS) verwendet. Ein biometrischer Enrolment- und Verifikationsprozess kann in der Folge durch folgende Funktionsblöcke und Variablen beschrieben werden:

2.2 Funktionsblöcke

- Binarisierung – transformiert realwertigen Merkmalsvektor in binäre Form
- RBS – Reliable-Bit-Selector (Analysiert Binärvektor nach stabilen Elementen)
- ECC (ENC/DEC) – Fehlerkorrektur-Block (z.B. BCH-Code) Encoding/Decoding
- XOR – XOR-Operation auf Binärvektoren
- Hashfunktion $h(.)$ – Funktion zur Erzeugung von Hashwerten

2.3 Variablen

- RRV / XRV – realwertiger Merkmalsvektor (Referenz / Probe)
- QBV / QBV‘ – quantisierter Merkmalsvektor
- RBV / XBV – binärer Merkmalsvektor bestehend aus geeigneten Komponenten
- SBV – binärer Zufalls-Geheimnisvektor
- CBV – binärer Codevektor (um Fehlerkorrektur erweitertes SBV)
- AD1 – Auxilliary Data 1: Datensubjekt-spezifischer Indexvektor der geeigneten Komponenten (Reliable Bit Indizes) generiert in RBS
- AD2 – Auxiliary Data 2: berechnet aus Geheimnis SBV und biometrischen Daten (AD2 = RBV XOR CBV)
- PI – Pseudo-Identifikator berechnet aus dem Hashwert $h(.)$ über dem subjekt- und applikations-spezifischem Geheimnisvektor SBV
- Anmerkung: Ein Hochkomma‘ deutet veränderte Version einer Variable an (hervorgerufen durch biometrisches oder sensorbedingtes Rauschen)

Wie in klassischen biometrischen Systemen unterscheiden wir zwei Phasen: Die Registrierungsphase (Enrolment) und die eigentliche Verifikationsphase.

Zum biometrischen Enrolment sind für das Helper-Data-Schema (HDS) mehrere Präsentationen einer biometrischen Charakteristik erforderlich, so dass Merkmalvektoren gleicher Länge gebildet werden können und als Eingabe für das HDS dienen. Im HDS wird eine binäre Repräsentation durch Quantisierung der biometrischen Daten erzeugt. Die entstandenen quantisierten Binärvektoren werden im Reliable-Bit-Selector-Block (RBS) analysiert, diesmal um Positionen in den Vektoren zu identifizieren, an denen sich Bits befinden, die sich einerseits vom Mittel aller Vektoren der Population unterscheiden aber auch stabil und somit reproduzierbar für ein Subjekt in den Merkmalen vorkamen. Der Binärvektor RBV der die |AD1| stabilsten Komponenten enthält, wird nun mit einem Binärvektor CBV gleicher Länger kombiniert, der im zweiten Prozess generiert wird.

In einem parallelen zweiten Prozess wird ein Zufallszahlengenerator (RNG) genutzt um einen binären Geheimnis-Vektor SBV zu erzeugen. Der Hashwert $h(.)$ dieses Vektors wird in der Datenbank gespeichert, ein Berechnen von SBV aus h(SBV) ist somit nicht möglich ohne die Hashfunktion zu brechen. Dieser Wert kann als Pseudo-Identifikator (PI) betrachtet werden. SBV wird nun zusätzlich mit dem Binärvektor aus dem ersten Prozess wie folgt verknüpft: Ein Fehlerkorrekturverfahren (ECC-Encoder ENC, z.B. BCH-Codes) wird genutzt um SBV resistent gegen Einzelbitfehler zu machen, die durch die Variation in der biometrischen Probe verursacht werden. Die Kapazität lässt sich leicht variieren, je mehr Fehler korrigiert werden können sollen, desto niedriger ist der Anteil von SBV in dem entstehenden Codewort CBV. Der Fehlerkorrekturcode wird so gewählt, dass CBV und der Binärvektor RBV die gleiche Länge haben. Eine XOR-Verknüpfung dieser beiden Vektoren sorgt dafür, dass ohne das Wissen eines der beiden Eingaben kein Rückschluss auf die andere Eingabe gemacht werden kann.

Soll eine Verifikation stattfinden, wird der Datensatz eines Nutzers (AD1, AD2, PI=h(SBV)) geladen und ein frischer Probenvektor XRV verarbeitet. Der Binarisierungs -Block erzeugt daraus den Binärvektor QBV‘. Die Bits an den Positionen die in AD1 gespeichert sind werden durch den RBS-Block extrahiert, so entsteht der Binärvektor XBV. Dieser Vektor sollte bei gleichem Datensubjekt dem Vektor RBV aus der Registrierungsphase sehr ähnlich sein. Durch die erneute XOR-Operation auf AD2 und XBV entsteht CBV‘. Wenn die Hamming-Distanz der Codeworte CBV und CBV‘ kleiner ist als die Kapazität des Fehlerkorrekturverfahrens und wenn die Fehler Einzelbitfehler sind, kann SBV‘=SBV im Fehlerkorrektur-Block (DEC) rekonstruiert werden. Die Entscheidung, ob die Verifikation positiv ist, ergibt der Vergleich von PI‘=h(SBV‘) und dem gespeicherten Wert PI=h(SBV).

Um eine Revokation einer biometrischen Referenz durchzuführen, muss ein neuer Geheimnis-Vektor SBV generiert werden, der mit einem frischen Merkmalsvektor (nach Binarisierung und RBS) kombiniert wird. Lediglich AD2 und PI müssen erneut in der Datenbank als Referenz gespeichert werden.

3 Transaktions-Absicherung

Bisher werden für Online-Transaktionen eine Reihe von Verfahren eingesetzt, die als nicht ausreichend sicher eingestuft werden müssen bzw. ungewollt eine Delegation der Authentisierungsfaktoren erlauben. Dieser Abschnitt liefert eine Übersicht bekannter Verfahren. Eine vertiefte Diskussion und Sicherheitsanalyse der gegenwärtig eingesetzten Authentisierungsverfahren im Online-Banking findet sich in [Asit08].

PIN/TAN: *Zwei-Faktoren-Authentisierung* mit Persönlicher Identifikationsnummer (PIN) und Transaktionsnummer (TAN), wobei sich die zu verwendenden TAN’s im Besitz des Bank-Kunden befinden sollen und nach Verwendung (d.h. einer Transkation / Buchung) aus einer Papierliste ausgestrichen werden.

PIN/iTAN: Um die Gefahr durch Phishing Angriffe zu reduzieren, wird seit 2006 eine indizierte TAN-Liste verwendet, so dass der Bank-Kunde vom Bank-Server in der Online-Sitzung aufgefordert wird zur Autorisierung einer gewünschten Transaktion eine bestimmte TAN zu verwenden, deren Index (Position) in einer nummerierten Liste von TANs dem Bank-Kunden mitgeteilt wird.

Mobile TAN (mTAN): Über einen zweiten Kommunikationskanal werden die relevanten Transaktionsdaten, welche bei der Bank eingetroffen sind per *Short Message Service* (SMS) Nachricht zum Mobiltelefon des Bank-Kunden übertragen und von ihm durch visuellen Vergleich mit der intendierten Transaktion geprüft. Die Autorisierung der Transaktion geschieht durch Eingabe einer ebenfalls übermittelten *mTAN,* die nur in einem kurzen Zeitfenster Gültigkeit hat und transaktionsspezifisch ist.

TAN-Generatoren: Beim TAN-Generatoren-Verfahren werden mobile Token verwendet, die sequentiell eine TAN elektronisch erzeugen können. Einige TAN-Generatoren wie der RSA-Token arbeiten zeitgesteuert. Die Ausprägungsformen sind *sm@rt-TAN, eTAN-Generator, chipTAN manuell* und *chipTAN comfort.* Die TAN Generatoren sind dann besonders komfortabel, wenn sie über eine optische Schnittstelle HHD 1.3.2 mit dem Client-PC kommunizieren [Zka2009].

Digitale Signatur / HBCI: Die Digitale Signatur wurde mit dem *Homebanking Computer Interface* (HBCI) seit 1996 entwickelt und standardisiert[1]. Damit steht eine Schnittstelle für ein Chipkarten-basiertes Online-Transaktionsprotokoll zur Verfügung. Das Protokoll wurde als *Financial Transaction Services* (FinTS) vom ZKA weiterentwickelt [Zka2009].

Online-Banking mit USB-Stick: Im Jahr 2009 wurde von IBM mit *Zone Trusted Information Channel* (ZTIC) ein USB-Stick-Verfahren vorgestellt, dass speziell für sicheres Online-Banking auf Malware-betroffenen Client-Rechnern konzipiert wurde [Wei2008]. Ähnliche Produkte gibt es von den Unternehmen KOBIL und Novosec.

4 Biometrische Transaktions-Authentisierung

Vorgestellt wird in diesem Beitrag ein Protokoll zur biometrischen Nachrichten-Authentisierung, exemplarisch dargestellt für die Absicherung von Online-Banking-Diensten. Das Protokoll erfüllt die beiden folgenden wesentlichen Anforderungen:

1.) Eine zuverlässige *Personen-Authentisierung*.
Nur die registrierte natürliche Person hat die Transaktion durchgeführt. Das Abstreiten einer tatsächlich durchgeführten Transaktion durch den registrierten Endkunden wird damit unmöglich.

[1] HBCI als solches ist kein Sicherheitsverfahren per se sondern ein Standard des ZKA zur Abwicklung von Online-Banking-Transaktionen

2.) Eine zuverlässige *Daten-Authentisierung*.
Die registrierte natürliche Person hat die Transaktionsdaten in einer vertrauenswürdigen Umgebung kontrolliert, diese Transaktion autorisiert und die Autorisierung über einen unabhängigen zweiten Kommunikationskanal zum Bank-Server übertragen.

4.1 Annahmen

Auf einem potentiell unsicheren Kundenrechner wird eine Online-Banking-Software (BSW) betrieben, die mit dem Online-Banking-Server (OBS) in der Bank kommuniziert. Die Online-Banking-Software überträgt Transaktionsdaten an den OBS und an das sichere Biometric-Transaction-Device (BTD), auf dem die Bestätigung der Transaktion durch den Endkunden erfolgt. Auf dem BTD wird ein Siegel erzeugt, das als Transaction-Order-Seal (TOS), die Transaktionsdaten mit den biometrischen Daten des Endkunden verknüpft. Für das Biometrische-Nachrichten-Authentisierungs-Protokoll wird von einer Bedrohungs-Situation ausgegangen, die in Abbildung 2 illustriert und im folgenden Abschnitt erläutert wird.

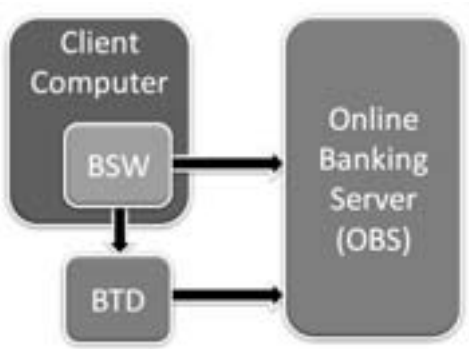


Abbildung 2: Bedrohungs-Situation und Kommunikationswege zwischen Online-Banking-Software (BSW), Online-Banking-Server (OBS) und Biometric-Transaction-Device (BTD).
Die Online-Banking-Software wird auf einem unsicheren Client-Computer betrieben, der Server OBS und das Device BTD werden als sicher eingestuft.

4.2 Komponenten

Zur Umsetzung biometrisch sicherer Online-Transaktionen interagieren die folgenden Komponenten, die sich in Bezug auf die Bedrohungssituation unterscheiden:

1.) Ein sicherer *Online-Banking-Server* (OBS), der folgende Eigenschaften aufweist:
- hat Zugriff auf Kundendaten.
- etabliert eine Kommunikation mit der Online-Banking-Software (BSW), die auf dem unsicheren Rechner des Kunden betrieben wird.
- führt Transaktionen aus.

- kann mit einem *Biometric-Transaction-Device* (BTD) eine Verbindung aufbauen.

2.) Eine *Online-Banking-Software* (BSW) auf einem **unsicherem** Kunden-Rechner (Client-Rechner). Die BSW:
- wird ausgeführt auf einem Kunden-Rechner, der durch Trojanische Pferde, Root-Kits etc. beliebig gefährdet sein kann.
- kann als browserbasierte Applikation ausgeprägt sein.
- kommuniziert mit Online-Banking-Server (OBS) und transferiert Aufträge in Form eines Transaction-Order-Record (TOR). Ein TOR beinhaltet: i) Transaktionsidentifikator (TID), ii) Sender-Account-Number (SAN), iii) Receiver-Account-Number (RAN), iv) Ordered Amount (ORA).
- Verbunden mit dem Kunden-Rechner ist ein vertrauenswürdiges Biometric-Transaction-Device (BTD).

3.) Ein **sicheres** *Biometric-Transaction-Device* (BTD), das mit Kunden-Rechner verbunden ist. Das BTD:
- ist eine vertrauenswürdige Hardware, die idealer Weise sicherheitsgeprüft wurde (z.B. nach Common Criteria). Die Hardware kann eine dedizierte Komponente, wie etwa ein biometrisch erweiterter Secoder nach ZKA-Anforderungen sein.
- kann nicht durch Malware manipuliert werden.
- kann eine biometrische Charakteristik erfassen. Dabei wird als Biometric-Capture-Device (BCD) ein Sensor eingesetzt, der als überwindungssicher eingestuft werden kann und somit für den nicht-überwachten Betrieb im Heimbereich oder Bürobereich geeignet ist.
- kann mit einem Online-Banking-Server (OBS) als Kommunikationspartner eine Verbindung aufbauen.
- kann eine Transaction-Order (TRO) von BSW als Transaction-Order-Record (TOR) empfangen und darstellen.
- Die Kommunikation zwischen BSW und BTD kann in verschiedenen Optionen ausgeprägt sein. Es kann eine kontaktlose Datenanbindung oder eine optische Schnittstelle (z.B. Flicker-Code) sein.

4.3 Enrolment

Zum Enrolment für das Biometrische-Transaktions-Authentisierungs-Protokoll wird das bekannte Helper-Data-Schema wie folgt erweitert (siehe Abbildung 3):

1.) Enrolment-Schritte im Biometric-Transaction-Device (BTD):
- Die selbe biometrische Charakteristik des Bank-Kunden wird mit dem BCD erfasst und in Merkmalsvektoren umgewandelt.
- Das Hilfsdatum AD1 wird aus Enrolment Samples abgeleitet, wobei charakterisierende Daten zur Verteilung über die Population ebenfalls erforderlich sind.
- Ein binarisierter Merkmalsvektor RBV ergibt sich aus den Enrolment-Samples QBV und AD1.
- Der Kunde gibt ein Geheimnis SBV ein, das er zusammen mit der ebenfalls vom Server erzeugten Account-Number auf dem Postwege vom OBS erhalten hat.
- Der Fehlerkorrektur-Codebookvektor CBV ergibt sich aus: CBV = ENC(SBV)
- Die Auxilliary Data AD2 ergibt sich aus CBV und dem binarisierten Referenz-Merkmalsvektor RBV: AD2 = CBV XOR RBV

- AD1 und AD2 sind nicht sonderlich schützenswert und werden im BTD oder auf der persönlichen Chipkarte gespeichert

2.) Enrolment-Schritte des Online-Banking-Server (OBS):
- Generiert und sendet pre-shared secret (Geheimnis SBV)
- Legt Kundenrecord mit den folgenden Daten an: Account-Nummer (AN) und biometrischer Pseudo Identifikator PI = h(SBV) (Hashfunktion h, z.B RIPEMD-160, das bereits in der FinTS-Spezifikation genutzt wird)
- Sendet SBV samt AN zum Kunden (Postweg)

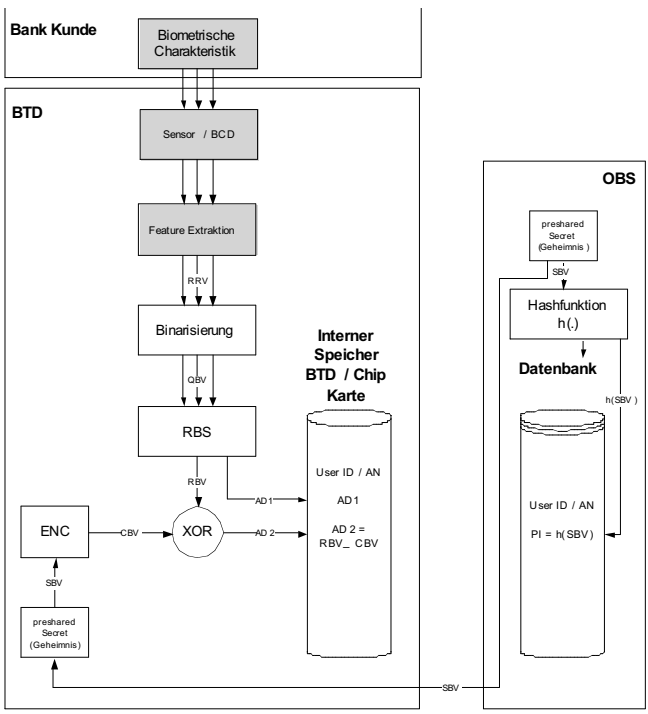


Abbildung 3: Erweitertes Enrolment

4.4 Prüfung der biometrischen Transaktions-Authentisierung

Zur Bestätigung einer vom Bank-Kunden gewünschten Transaktion wird für das Biometrische Nachrichten-Authentisierungs-Protokoll die klassische biometrische Verifikation erweitert. Zur Bestätigung der Transaktion wird lokal, d.h. im Home- oder Office-Umfeld des Bank-Kunden ein Transaction-Order-Seal (TOS‘) berechnet, der dann statt einer TAN an den Bank-Server übertragen wird. Die Arbeitsschritte sind wie folgt (siehe Abbildung 4):

1.) Operationen, die in der unsicheren Online-Banking-Software (BSW) durchgeführt werden sind:
   - Der Kunde erstellt durch Interaktion mit der BSW-Software einen Transaction-Order-Record (TOR).
   - Der BSW überträgt den TOR an den Online-Banking-Server (OBS)
   - Der BSW überträgt den TOR an das Biometric-Transaction-Device (BTD).

2.) Operationen, die im Biometric-Transaction-Device (BTD) durchgeführt werden:
   - Die relevante Information aus dem Transaction-Order-Record (TOR) wird im Display des BTD angezeigt: Receiver-Account-Number (RAN), Ordered-Amount (ORA). Die Ausprägung der Darstellung kann ähnlich wie mit den bereits am Markt erhältlichen chipTAN comfort Token erfolgen.
   - Zur Bestätigung der gewünschten Transaktion präsentiert der Initiator seine nicht-replizierbare biometrische Charakteristik dem Biometric-Capture-Device. Durch diesen Schritt wird ein Probe Image Sample mit dem BCD erfasst.
   - Die Auxilliary Data AD1 wird aus dem BTD-Speicher abgerufen
   - Ein binarisierter Probe-Merkmalsvektor XBV ergibt sich aus dem Probe-Sample QBV‘ und AD1
   - Ein Codebookvektor CBV‘ wird rekonstruiert aus im BTD gespeicherter Auxilliary Data AD2 und dem binarisierten Probe-Merkmalsvektor XBV: CBV‘ = AD2 XOR XBV
   - Das Secret SBV‘ wird aus CBV‘ berechnet: SBV‘ = DEC(CBV‘)
   - Die Pseudo-Identifikator PI‘ wird aus SBV‘ berechnet: PI‘=h(SBV‘)
   - Es wird ein Transaction-Order-Seal (TOS‘) berechnet aus Transaction-Order-Record TOR und rekonstruiertem PI‘: TOS‘ = MAC(h(TOR), PI‘)
   - Das TOS‘ verknüpft als Siegel die Daten eindeutig mit der bestätigenden natürlichen Person. Der berechnete Transaction-Order-Seal kann auch als Message Authentication Code (MAC) bezeichnet werden. Als MAC-Verfahren kann beispielsweise ein HMAC-Verfahren eingesetzt werden, das auf der Hashfunktion h aufbaut. Die Eingabevektoren TOR und PI‘ entsprechen der Nachricht und dem Schlüssel im HMAC-Verfahren. Der Wert TOS‘ lässt sich nach [Rfc2104] z.B. mit RIPEMD-160 als Hashfunktion wie folgt berechnen:
   - TOS‘ = h(PI‘ XOR OPAD, h(PI’ XOR IPAD, TOR))
   - Das Transaction-Order-Seal (TOS‘) wird zum Online-Banking-Server übertragen und gegebenenfalls vorab mit asymmetrischer Kryptographie verschlüsselt. Die Übertragung kann über den unsicheren Client-Rechner getunnelt werden oder alternativ (und bevorzugt) über einen zweiten unabhängigen Kanal. Dieser zweite unabhängige Kanal kann beispielsweise über das GSM-Netz realisiert werden. Diese Variante wird insbesondere dann bevorzugt werden, wenn das BTD in einem marktüblichen Mobiltelefon / Smartphone hardwaretechnisch integriert wird.

3.) Operationen, die im Online-Banking-Server (OBS) durchgeführt werden. Der OBS:
   - hat den Transaction-Order-Record (TOR) von der Banking-Software (BSW) erhalten.
   - hat den Transaction-Order-Seal (TOS‘) von dem Biometric-Transaction-Device (BTD) erhalten
   - hat den PI zum Kunden vorliegen PI=*h*(SBV)
   - rekonstruiert den TOS: TOS = MAC(*h*(TOR), PI)

- vergleicht den rekonstruierten TOS mit dem vom BTD gelieferten TOS‘:
- TOS = = TOS‘?

Die Transaktion ist personen- **und** datenauthentisch, wenn TOS und TOS‘ identisch sind. In diesem Fall und nur in diesem Fall gilt der im Transaktions-Record kodierte Auftrag als authentisch **und** bestätigt und wird vom OBS ausgeführt.

Die notwendigen Verifikations-Schritte im BTD und auf dem OBS zur Bestätigung der Transaktion und zur gleichzeitigen Prüfung von Personen-Authentizität **und** Daten-Authentizität werden in der folgenden Abbildung dargestellt.

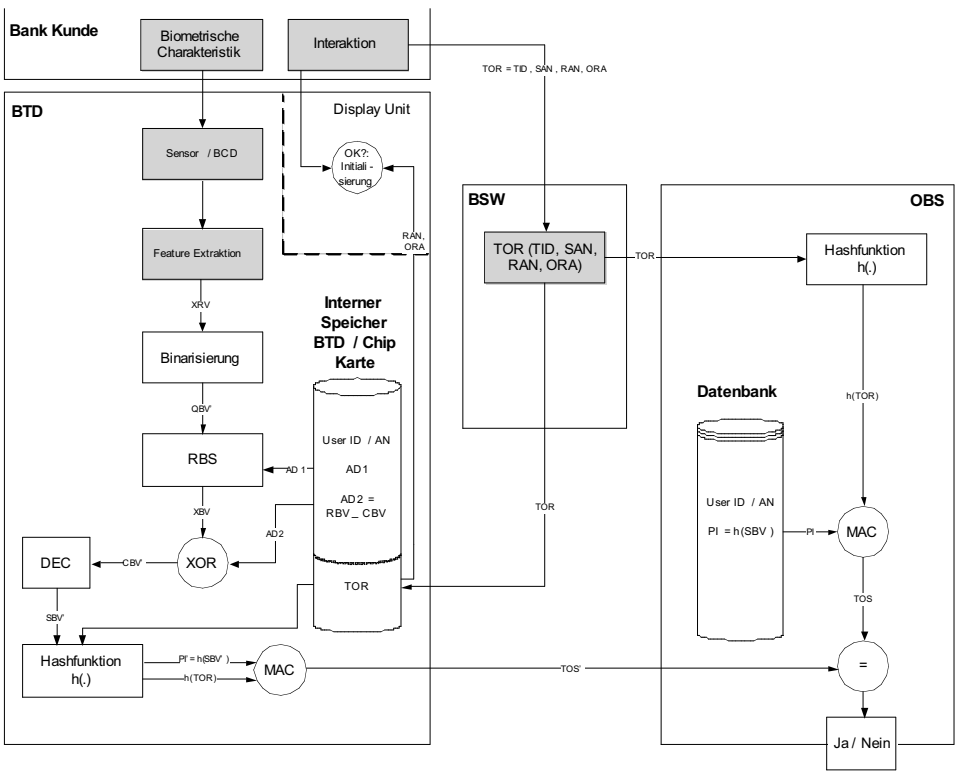


Abbildung 4: Prüfschritte einer biometrischen Transaktions-Authentisierung

5 Zusammenfassung und Ausblick

Durch das hier vorgelegte Transaktionsprotokoll wird die Personen-Authentisierung mit der Daten-Authentisierung verknüpft und damit die Forderung des Bundesverbandes Deutscher Banken erfüllt, dass durch die Nutzung der Biometrie im Online-Banking „eine Bindung des eineindeutigen biometrischen Merkmals des Kunden an seine gewollte Transaktion“ erreicht wird [Grud2009]. Sicherheitsrelevante Funktionalität

wird in einem Biometric-Transaction-Device gekapselt, das auch als biometrischer *Secoder* verstanden werden kann. Der wesentliche Sicherheitsgewinn im Vergleich zu existierenden Protokollen besteht darin, dass eine unerlaubte Delegation von Authentisierungsfaktoren ausgeschlossen werden kann. Weitere Anwendungsfelder ergeben sich z.B. in der Nachrichtenkommunikation im KRITIS- und Katastrophenmanagement. Durch den geringen Funktionsumfang des BTD ist eine Ausprägung in Hardware leicht realisierbar und eine Common-Criteria-konforme Sicherheitsprüfung dieser Komponente möglich.

Literaturverzeichnis

[Asit08] A-SIT: Secure Information Technology Center Austria, Österreichische Nationalbank. Risikoanalyse – E-Banking Angebote Österreichischer Kreditinstitute, 2008

[Bdb2006] Bundesverband Deutscher Banken: „Anzahl der Online-Konten“, http://www.bankenverband.de/downloads/112007/1ta0711-pr-onlinekonten.pdf

[Bit2008] Branchenverband BITKOM: „Fast 4 Millionen Opfer von Computer- und Internet-Kriminalität“, http://www.bitkom.org/de/presse/56204_53100.aspx

[Bka2008] Bundeskriminalamt: „Aktuelle Herausforderungen in der Kriminalitätsbekämpfung“, http://www.bka.de/pressemitteilungen/2008/pm080328.html, März 2008

[Bre2008] J. Breebaart, C. Busch, J. Grave, E. Kindt: „A Reference Architecture for Biometric Template Protection based on Pseudo Identities“, in Proceedings BIOSIG2008, pages 25-37, GI-LNI, (2008)

[Bsi2009] Bundesamt für Sicherheit in der Informationstechnik: „Die Lage der IT-Sicherheit in Deutschland 2009“, https://www.bsi.bund.de/Lageberichte

[Bsi2010] Bundesamt für Sicherheit in der Informationstechnik: „Identitätsdiebstahl und Identitätsmissbrauch im Internet “, https://www.bsi.bund.de/Studien

[Grud2009] W. Grudzien: „Sicherheit in der Kreditwirtschaft“, Vortrag auf der Fachtagung für FinanzDL am 22.09.2009

[Iso-sc37] ISO/IEC JTC1 SC37 SD2 Harmonized Biometric Vocabulary, September 2009 http://www.3dface.org/media/vocabulary.html

[Iso-sc27] ISO/IEC JTC1 2ndCD 24745: Biometric Template Protection, Januar 2010

[Idtc2009a] Identity Theft Resource Center: Security Breaches 2008, http://www.idtheftcenter.org/artman2/publish/lib_survey/Breaches_2008.shtml

[Rfc2104] H. Krawczyk, M. Bellare, R. Canetti. “RFC2104 - HMAC: Keyed-Hashing for Message Authentication”, 1997, http://www.faqs.org/rfcs/rfc2104.html

[Rut2006] J. Rutkowska: „Introducing Stealth Malware Taxonomy“, http://www.invisiblethings.org/papers/malware-taxonomy.pdf

[Tak05] P. Tuyls, A. H. M. Akkermans, T. A. M. Kevenaar, G. J. Schrijen, A. M. Bazen, and R. N. J. Veldhuis. “Practical biometric authentication with template protection.” In Audio and video-based biometric person authentication, pages 436–446. Springer, Berlin, Germany, 2005.

[Wei2008] T. Weigold et al.: „The Zurich Trusted Information Channel – An Efficient Defence against Man-in-the-Middle and Malicious Software Attacks“, in : TRUST 2008, LNCS 4968, pp. 75–91, http://www.zurich.ibm.com/pdf/csc/ZTIC-Trust-2008-final.pdf

[Zka2009] Zentraler Kreditausschuss: „FinTS Spezifikation“, Version vom 02.02.2009, http://www.hbci-zka.de/dokumente/aenderungen/V3.0/HKTAN-4_zur_Unterstuetzung_von_HHD_UC.pdf

Diffusion of Federated Identity Management

Detlef Hühnlein[1], Heiko Roßnagel[2], Jan Zibuschka[2]

[1] secunet Security Networks AG, Sudetenstraße 16, 96247 Michelau,
detlef.huehnlein@secunet.com

[2] Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO), Nobelstr. 12, 70569 Stuttgart
{heiko.rossnagel,jan.zibuschka}@iao.fraunhofer.de

Abstract: In this work, we discuss the diffusion of federated identity management. We base our research on Roger's diffusion of innovation theory, and derive generic factors influencing the diffusion of federated identity management solutions. To validate our model and investigate specific contributions of parameters in specific usage scenarios, we investigate market success of federated identity management systems. We examine several application scenarios in the fields of e-business, Web, and e-government.

1 Introduction

Identity Management (IdM) has emerged as a promising technology to distribute identity information across security domains [MR08]. In e-business scenarios, federated identity management is used to connect enterprises along the value chain and enables them to reduce transaction costs significantly. On the web it offers the promise of single sign on for different domains and service providers, offering a common authentication and authorisation infrastructure that eliminates the neccessity of using passwords. This would on one hand provide improved ease of use for the users and at the same time eliminate problems that are caused by password management issues, password reuse [IWS04], and passwords' security flaws [Neu94]. Therefore, it could make a major contribution to improvement of security on the web. In the e-government domain, Identity Management Systems (IMS) could help to introduce the necessary security infrastructure enabling online services that so far could not have been offered by public administration due to security constraints.

Several different solutions for Federated Identity Management (FIM) have emerged over the last couple of years such as Microsofts Passport and Cardspace, Liberty Alliance and OpenID. The success of these systems in the marketplace has been very diverse. Some Systems, such as Passport, have not been successful and have been replaced [CJ07]. Other systems have been highly succesfull in particular domains, such as SAML in e-business scenarios. In other domains, however, the same systems have not achieved this kind of success.

In this contribution we will examine the market success of FIM systems in different application domains. In particular we want to explore how economic principles affect this success. Since federated identity management is a complex technology that can be applied to several different domains, we distinguish three different usage scenarios and perfom

a cross-case study. To do this we apply Roger's diffusion of innovation theory [Rog03], which is a proven economic theory for explaining the market success or failure of innovations, to each scenario and compare and discuss the results.

The rest of the paper is structured as follows. In section 2 we will provide an overview on FIM and diffusion theory in general. We will apply this diffusion theory to FIM and discuss the factors influencing the adoption in section 3. We then validate our arguments from section 3 by taking a close look at three different usage scenario case studies in section 4. Our results are discussed in section 5, concluding our findings.

2 Background and Related Work

2.1 Federated Identity Management

Among the important Identity Management processes is the authentication, identification and authorization of Users (U) who want to access some resource offered by some Service Provider (SP). For this purpose U is equipped with one or more Credentials, which are presented to the SP. While the SP in a classical IdM scenario must itself understand, verify and accept U's Credentials, those tasks may be delegated to a specialized Identity Provider (IdP) in a federated setup. Hence a FIM system may be viewed as a sequence of entities that transform a Source Credential of U into a Session Credential that is consumable by a SP who makes a decision on whether to grant access to a sensitive service.

Before U is able to use the service offered by the SP, the following steps are typically performed:

1. $UA \rightarrow SP$: The User Agent (UA) contacts the SP and requests some service.
2. $SP \rightarrow UA$: The SP answers the request, possibly including additional information about supported protocols, appropriate IdPs, necessary authentication assurance levels, requested attributes etc.
3. $UA \rightarrow IdP$: The UA connects to the IdP in order to authenticate with the Source Credential and request a Session Credential that can be presented to the SP.
4. IdP: The IdP authenticates U based on the Source Credential, uses an existing session in which authentication has already been performed previously or further delegates the authentication to another IdP.
5. $IdP \rightarrow UA$: If the authentication was successful, the IdP returns a Session Credential to the UA.
6. $UA \rightarrow SP$: The UA sends the Session Credential received from the IdP to the SP.
7. SP: The SP validates the Session Credential and verifies the access rights of the now authenticated U.

8. $SP \rightarrow UA$: The SP serves the requested resource to the UA.

Furthermore, there may be additional steps for identity selection (IS), in which U, UA, and/or the SP interact in order to select an appropriate electronic identity and hence IdP to be contacted in step 3. In addition, there typically is some sort of trust relationship (T) between the SP and the IdP, which allows to verify the integrity and authenticity of the Session Credential.

Federated authentication and SSO systems are around for quite a while [SNS88]. and the most important protocols in this area comprise the Security Assertion Markup Language (SAML) [CKPM05], the WS-* series of standards [NKMHB06] together with the Identity Metasystem Interoperability profile [JM09] and last but not least OpenID [Fou]. Please refer to [LOP04, MR08] for a more complete survey.

2.2 Diffusion Theory

In the information systems literature, a variety of theoretical perspectives have been advanced to provide an understanding of the determinants of usage. An important line of research has examined the adoption and usage of information technology from a diffusion of innovation perspective [Rog03]. This research examines a variety of factors, which have been shown to be determinants of IT adoption and usage, and has been applied to explain the adoption and diffusion of a great variety of innovations ranging from new methods of agriculture to modern communication technology. In his seminal work Rogers defines five attributes of innovations, as perceived by the members of the social system that determine the rate of adoption of an innovation [Rog03]:

Relative advantage. is the degree to which an innovation is perceived as better than the idea it supersedes. It is not so important if the innovation has an objective advantage, but rather if the individual perceives the innovation as advantageous. Advantages can be measured in economic terms, but social prestige, convenience, and satisfaction also can play an important role.

Compatibility. is the degree to which an innovation is perceived as being consistent with the existing values, past experiences, and needs of potential adopters. An Innovation that is consistent with the existing values will diffuse more rapidly than one that is incompatible with the norms and values of the social system.

Complexity. is the degree to which an innovation is perceived as difficult to understand and use. Innovations that are easier to understand will be adopted more rapidly than those which require the adopter to develop new skills and understandings.

Triability. is the degree to which an innovation may be experimented with on a limited basis. New ideas that can be tried before the potential adopter has to make a significant investment in the innovation are adopted more quickly.

Observability. is the degree to which the results of an innovation are visible to others. The easier it is for individual to observe the results of an innovation, the more likely they are to adopt [Rog03].

An interactive innovation is an innovation that is of little use to an adopting individual unless other individuals with whom the adopter wants to communicate also adopt. Thus a critical mass of individuals has to adopt the innovation before it is of use for the average member of the system [MR99]. The individuals who have adopted an innovation form a network and with each new member the overall value of the network increases [MR99]. This fundamental value proposition is being called network effects, network externalities, and demand side economics of scale [SV99]. Until a critical mass occurs in the diffusion process the rate of adoption is relatively slow [MR99]. After the critical mass is achieved the rate of adoption accelerates and leads to a take off in the adoption curve.

The diffusion of security technologies has been discussed by various authors in the context of TOR [ADS03], electronic signatures [Roß06] and smartcards, as well as certificates for web sites [OS06].

Economic aspects of IdM systems have been addressed in the context of possible hindrances for the success of such systems [DD08], and network effects and compatibility have been identified as a decisive factor in the context of certificates and security technologies in general [OS06], however, to our knowledge, no broader discussion of this problem exists, especially covering not only one specific use case (i.e. the Web), but a broader range of identity management deployments.

3 Diffusion of Federated Identity Management

Relative advantage. In the context of identity management, relative advantage is the utility that an IMS can offer to its stakeholders, including both end users, SPs, enterprises, and other players, compared to the previous state of the art (e.g. passwords on the Web). Examples of how a IMS can offer utility include:

- **Reduced sign on** is one of the main benefits provided by IMS. In the enterprise case, it lowers help desk costs, while on the web, it unburdens users of password management worries.
- **Privacy** is also often mentioned as one of the issues IMS can help address. Integration of anonymization and privacy-enhancing technologies into an IMS [HBC+01] offers value to privacy-sensitive users.
- **Reduction of user interaction** Form-filling reduces the time users have to spend while registering. e-government applications may enable users to carry out admin-

istrative tasks online, rather than at the local authority offices. In the context of Web services, this may also lead to a larger registered user base, as the effort necessary for registration is lowered.

- **Security** IMS have the potential to alleviate common security risks of passwords [Neu94]. However, as SSO also adds a single point of failure, IMS also have higher security requirements than analogous decentralized password systems.
- **Identity Intermediation** Identity intermediation, the outsourcing of user identity storage and part of the authentication process, offers cost reductions, compliance risk reduction, and increased identity quality to service providers [ZFR+07].

So, all in all, IMS can offer competitive advantage on several levels. However, an IMS may also decrease utility for several reasons.

- **Liability:** Depending on contract, an identity provider may be held liable for the identities it provides. A user employing a signature card rather than a password may find himself bound by contracts and held liable in a way that would not be possible with password-based authentication.
- **Costs** of certification, implementation and other IdM processes may eat up the advantage gained from the other factors.

It should be noted that network effects play a huge role in each scenario because FIM is by nature an interactive innovation. It is quite obvious that a system with a larger user base would be more appealing to service providers, while a system supported by a larger set of services would be of a higher utility to users, offering a meaningful reduction of sign-on processes.

(Social) Compatibility. In the context of IMS, compatibility refers mainly to privacy/trust questions. Trust is a key inhibitor of the broader success of e-business systems, which may be addressed using privacy-enhancing IMS [HBC+01]. However, trust-related previous work shows that technical measures providing trust through security are dominated by user interface issues: a user may distrust even a secure system because it is very complicated to use, it appeals less to him visually, or it produces errors during usage. Those influences have an impact that is at least as strong as technical security across all user groups [LT01]. Also, trust in the SP influences the trust in the system much stronger than the trustworthiness of the system influences trust towards the SP [MCK02]. With regard to compatibility with user expectations, passwords are still dominant. Especially in the web case, passwords will have to be offered simultaniously to reach compatibility with users' authentication expectations. Compatibility could also be an issue for service providers depending on the business model of the IdP.

Complexity. Usability also has a major influence on the perceived complexity of an innovation by users. Many systems require the users to learn a new authentication interaction

paradigm, causing new usability problems [MR08]. If the users have the impression that the system is difficult to use or to obtain, they are unlikely to adopt unless the perceived relative advantage significally outweighs these hindrances. On the other hand, IMS have the potential to offer a significant reduction of complexity to end users, with SSO relieving the users of password management problems, and form filling reducing the time spent while registering.

For SPs the perceived complexity of FIM is not a question of using this technology but rather a question on how easily such systems can be implemented on the server side. This includes potential sunk cost for installing the infrastructure and the operation of several different authentication systems. Even if SPs switch to FIM for user authentication they will still have to support password authentication, because otherwise they would exclude a huge amount of potential customers of using their services.

For enterprises that want to adopt a IMS, a major factor influencing the perceived complexity will be how easily the system can be adapted to and integrated in existing business processes. If major adjustments to existing processes have to be made, the complexity of adopting the innovation increases significally and the costs of adoption will also increase.

Triability. is generally not an issue, as many IMS can be experimented with on a limited basis. However, there are also areas where triability is problematic. In the context of enterprise identity management, complete deployments cannot be easily evaluated, and assumptions about the cost savings have to be made. In national e-government IMS, such as eIDs, ex ante costs such as the price of card readers may inhibit users from trying out the system. On the SP side, it may not be trivial to integrate IMS. Even where modular encapsulated interoperable IMS modules for SPs exist, usability issues add another layer of cost and risk to lose users.

Observability. Signalling quality is a problem for FIM [BHTB05]. It's hard to understand and to see the differences in applied security solutions for users. This could lead to a lemons market, which was defined in [Ake70]. In this pioneering article the author argues that information asymmetry and uncertainty about the product quality will lead to a market failure, unless appropriate counteracting mechanisms are undertaken. In a market that contains good and bad (lemons) FIMs, imperfect information about service quality causes the user to average the quality and price of the service used. The information gap enables the opportunistic behaviour of the owner of a lemon to sell it at average price. As a result, the better quality service will not be used since the price it deserves can not be obtained. The consequence of such practice will lead to a continuous fall of both quality and value of services. On the other hand, the advantages of SSO are easily observed by users. The same applies for cost reductions in the helpdesk area reached through enterprise SSO systems. However, the lemon market problem applies, at the very least, to certification authorities acting as a stakeholder in IMS.

4 Case Studies

The specific factors influencing diffusion of IMS are dependent on the usage scenario. In the case of Relative Advantage/Usefulness, the main drivers seem to be help desk cost reduction in enterprise settings, SSO in Web scenarios, and security (as enabler for new services) in e-government. Therefore, different systems, which emphasise different characteristics have prevailed in different scenarios. This section investigates e-business, Web and e-government scenarios, and illustrates the effects described in a more general manner in the earlier sections.

4.1 Case 1: E-Business - Circle of Trust

Real world examples can be found in the automotive industry. For example there is a technical recommendation [ODE09] based on SAML v2.0 [CKPM05], which has been developed by an Odette[1] working group consisting of leading automotive enterprises[2] and technical supporters[3] and provides guidance for the implementation of federated SSO scenarios between companies in the automotive sector. In a similar fashion SAML has been proposed for cross-enterprise rights management in the automotive industry [iA09].

In these e-business scenarios federated identity management succeeds along the value-networks forming a circle of trust between the participating enterprises. It is mainly aligned to existing business and trust relationships between enterprises, which act as SP and IdP respectively, and U is typically an employee of either company. In these value-networks dominant stakeholders exist that can act as a champion for the technology influencing other stakeholders to adopt the same technology more rapidly [Rog03].

The dominating driving force for the adoption of federation techniques is the general wish to minimize transaction costs. In particular the SP will spare to issue Credentials to Users at business partners, if there is an IdP, which already did that, supports the same federation protocol as the SP and last but not least may be trusted to perform the authentication on behalf of the SP. Therefore, the perceived relative advantage mainly comprises of cost reduction (due to reduced sign on) and identity intermediation.

4.2 Case 2: Web Identity Management

Passwords have long been the predominant means for user authentication on the web. This is also true for other scenarios, however, in the Web scenario, even adopters of IMS mostly still support passwords. This may be because the lack of coordination in this scenario

[1] See `http://www.odette.org/`.

[2] Including BMW Group, Bosch GmbH, Daimer AG, Hella KGaA Hueck & Co, Volvo Personbilar Sverige AB and ZF Friedrichshafen AG.

[3] Including Covisint Compuware GmbH, iC Consult GmbH, Microsoft AG, PingIdentity Corporation and Siemens AG.

makes a timed adoption across users unlikely, and gaining a user base is central to the business goals of many Web services. Consequently, the advantage offered by an IMS on the Web would have to outweigh lost users who do not use it. As this is unlikely, passwords will likely not be superseded completely.

Microsoft Passport [Mic] was one of the first deployed systems to translate the idea of SSO to the web. It was based on the architecture of centralized intra-organizational SSO systems geared towards managing user access to individual services within an enterprise. Passport was rolled out as part of the infamous HailStorm initiative, and was soon criticized for privacy and security shortcomings. Those problems were insecurities of the system design [KR00], and the centralized storage of identity information requiring trust in Microsoft. Although Passport had originally been adopted by several big players (such as eBay and monster.com), the infrastructure did not take off, and eventually the early adopter services left. The conventional wisdom from this case study is that insecurity and trust issues can make IMS fail, which was one of the motivators for the privacy-enhancing IdM movement [HBC+01]. However, Passport has recently made quite a comeback under the new name of Windows Live ID [Mic], mainly used for internal SSO across all Live services, and has become one of the top 5 most successful Web IMS (see Figure 1).

As a follow-up to Passport, Microsoft presented the CardSpace IMS [CJ07]. CardSpace is based on the concept of InfoCards, which allow the user to choose the identity to present to a service from a set of partial identities for different use cases. The CardSpace user interface presents a rolodex-like screen for selection of the appropriate InfoCard after the user has (typically) clicked on the log-in icon at a web site. While CardSpace is technologically superior to the original Passport [MR08], CardSpace has not fared any better than Passport during the adoption process, and early adopters are already leaving again. Whether it will see a comeback similar to Passport remains to be seen, however, there is no obvious alternative application case.

OpenID [Fou] was originally developed for use in the LiveJournal online community as a lightweight, decentralized way to authenticate commenters [MR08]. It is a web-centric FIM protocol that uses user-supplied web addresses for identifying IdPs, and supports self-hosted IdPs. The user enters the URL of her IdP, and is then logged into a service via the IdP provided. While there have been several security problems with OpenID [MR08], OpenID has still seen quite a broad adoption, easily outperforming Live ID and going head-to-head with Google ID, the SSO system used for Google services such as GMail (see Figure 1). What set OpenID and Passport apart may well have been the ability to freely choose (or self-host) the IdP instead of being bound to Microsoft, a compatibility factor.

However, online community sites like Facebook and Twitter have recently implemented their own SSO solutions, specific to their platforms, and transmitting the user's social graph along with classical identity information. As Figure 1 shows, those systems have gained an even stronger traction than OpenID. As of such, the misfortune of Passport should probably be attributed to distrust in Microsoft, rather than generic compatibility/privacy issues.

The Web is an open scenario, with relatively little coordination and trust between enti-

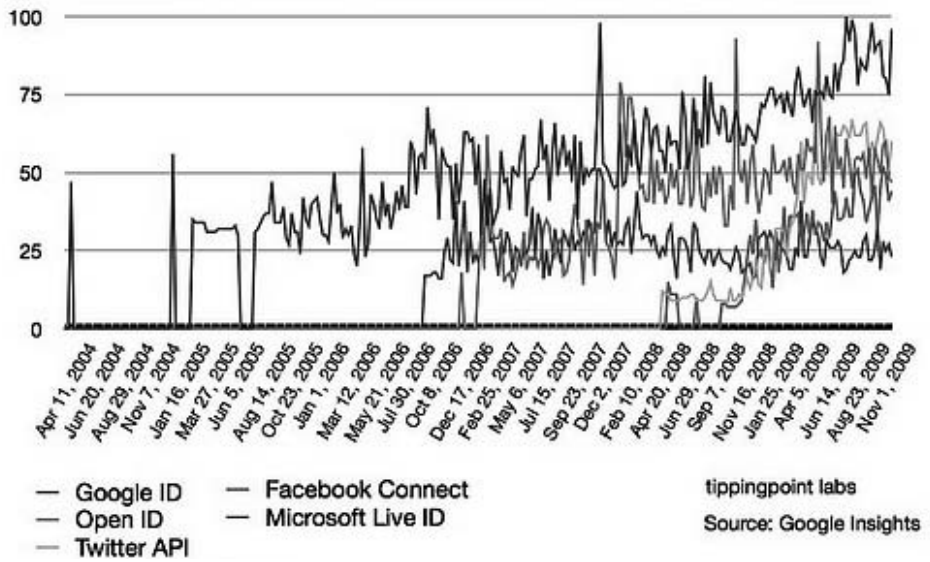


Figure 1: Diffusion of leading Web-scale IMS

ties (compared to the e-business and e-government scenarios). In this application field, lightweight IMS integrating directly with web user interface paradigms have dominated more elaborate systems offering more advanced security guarantees for several generations. Recent developments seem to demonstrate that systems making available the user's social graph to third parties have a considerably higher adoption rate. This supports the theory that in the Web case, compatibility issues are dominated by complexity issues (on the user side) along with the competetive advantage of user information that can be used e.g. for personalization. Additionally, the case of CardSpace may be taken to illustrate that making an IMS more secure cannot replace trust in the operator of the system, even if the system is privacy-friendly (fully client-side anonymous credentials) and open (e.g. with regards to third party certificates). Rather, those technological complexities seem to put extra burden on those systems.

4.3 Case 3: E-Government

There are several services in e-government scenarios that, if they are performed online by citizens, could lead to a major cost reduction within the public administration. So far many of these services can not be offered to a broad audience, because the technology to address the neccessary security requirements has not been adopted by the general population. In the past electronic signatures have been proposed as a suitable solution to address the

demanding security requirements of e-government processes. However, they have not been successful on the market. An analysis of the reasons for their lack of market success can be found in [Roß06]. As is evident by the various real world projects listed in the following, FIM could provide a major contribution to close this gap:

Secure idenTity acrOss boRders linKed (STORK) is a project, which aims at establishing a European eID Interoperability Platform, which allows cross-border recognition of national eID tokens. For this purpose there will be Pan-European Proxy Services (PEPS) in the different EU member states, which are connected through a SAML-based interface as specified in [AMHAJ+09].

SuisseID uses hardware tokens and X.509-based certificates for authentication as core of a more comprehensive "Claim Assertion Infrastructure" based on SAML [CKPM05] and WS-Trust [NGG+07].

German eID-Server is necessary for accessing the identity attributes stored on the forthcoming German eID-card and provides a SAML-based interface [BSI10].

Secure Access to Federated e-Justice/e-Government (S.A.F.E.) aims at providing a scalable Federated Identity Management architecture for the German e-justice sector, which complements the existing OSCI-based infrastructures with components based on international standards.

The success of FIM systems in E-government scenarios will highly depend on the services that can be offered based on this technology and the perceived relative advantage these services can provide to the citizens. In use cases where security requirements and the perceived value are high, FIM solutions that are issued by the government could be very successfull, as from a users perspective the relative advantage of FIM in E-government mainly comprises of form-filling and security. In comparison to solutions that are prevailant in the web scenario, infrastructures that are supported by the government do not face the problem of signaling their quality, as citizens tend to put more trust into those solutions.

There is hope that once adopted in an e-government scenario the same technology will spread to other use cases and in particular to the web use case. From a technological perspective this assumption is very sound, since secure solutions can also be applied to scenarios that do not require such an amount of security. However, from an economic perspective the success of these systems in other domains such as the web scenario is rather unlikely, because they have to compete with incumbent technologies, that already offer similar services. The main relative advantage of e-government solutions compared with other FIM systems will then be the higher degree of security. However, it is rather unlikely that this will be a driving factor for user adoption.

5 Conclusion

Federated identity management systems are a promising technology to achieve cross domain user authentication. Several different systems have emerged and have achieved a

mixed success in the marketplace. We examined the market uptake of FIM, by applying the diffusion of innovations theory to explain the success (and lack thereof) of various FIM-Systems in different usage scenarios. Our results show that in enterprise scenarios the diffusion of FIM follows established trust and business relationships along the value-network. In the web scenario we conclude that reducing complexity is by far more important than achieving a high degree of compatibility. In The most promising e-government scenarios are those that provide a high value to users while at the same time demand high security standards.

References

[ADS03] Alessandro Acquisti, Roger Dingledine, and Paul Syverson. On the Economics of Anonymity. In *Financial Cryptography*, pages 84–102. 2003.

[Ake70] George A. Akerlof. The Market for "Lemons": Quality Uncertainty and the Market Mechanism. *The Quarterly Journal of Economics*, 84(3):488–500, August 1970.

[AMHAJ+09] J. Alcalde-Morańo, J. L. Hernández-Ardieta, A. Johnston, D. Martinez, and B. Zwattendorfer. Interface Specification. STORK Deliverable D5.8.1b, 08.09.2009, September 2009.

[BHTB05] James Backhouse, Carol Hsu, Jimmy C. Tseng, and John Baptista. A question of trust. *Commun. ACM*, 48(9):87–91, 2005.

[BSI10] BSI. eID-Server. Technical Directive (BSI-TR-031030), Version 1.1, 08.02.2010, 2010.

[CJ07] Kim Cameron and Michael B. Jones. Design Rationale behind the Identity Metasystem Architecture. In *ISSE/SECURE 2007 Securing Electronic Business Processes*, pages 117–129. 2007.

[CKPM05] Scott Cantor, John Kemp, Rob Philpott, and Eve Maler. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 15.03.2005, 2005.

[DD08] Rachna Dhamija and Lisa Dusseault. The Seven Flaws of Identity Management: Usability and Security Challenges. *IEEE Security & Privacy Magazine*, 6(2):24–29, 2008.

[Fou] OpenID Foundation. OpenID Authentication 2.0. Final, December 5, 2007. `http://openid.net/specs/openid-authentication-2_0.html`.

[HBC+01] Marit Hansen, Peter Berlich, Jan Camenisch, Sebastian Clauß, Andreas Pfitzmann, and Michael Waidner. Privacy-enhancing identity management. *Information Security Technical Report*, 9(1):35–44, 2001.

[iA09] ProSTEP iViP Assocation. Enterprise Rights Management. PSI-Recommendation 7, Version v0.9, Annex B, Cross-Enterprise-ID, 2009.

[IWS04] Blake Ives, Kenneth R. Walsh, and Helmut Schneider. The domino effect of password reuse. *Commun. ACM*, 47(4):75–78, 2004.

[JM09] Michael B. Jones and Michael McIntosh. Identity Metasystem Interoperability Version 1.0. OASIS Standard, July 2009.

[KR00] David P. Kormann and Aviel D. Rubin. Risks of the Passport single signon protocol. *Computer Networks*, 33(1-6):51–58, June 2000.

[LOP04] Javier Lopez, Rolf Oppliger, and Günther Pernul. Authentication and authorization infrastructures (AAIs): a comparative survey. *Computers & Security*, 23(7):578–590, October 2004.

[LT01] M.K.O. Lee and E. Turban. A trust model for consumer Internet shopping. *International Journal of Electronic Commerce*, 6(1):75–91, 2001.

[MCK02] D. Harrison McKnight, Vivek Choudhury, and Charles Kacmar. Developing and Validating Trust Measures for e-Commerce: An Integrative Typology. *INFORMATION SYSTEMS RESEARCH*, 13(3):334–359, September 2002.

[Mic] Microsoft. Windows Live ID/Passport Network. https://accountservices.passport.net/ppnetworkhome.srf?vv=700&lc=1031.

[MR99] Alwin Mahler and Everett M. Rogers. The diffusion of interactive communication innovations and the critical mass - The adoption of telecommunication services by German banks. *Telecommunications Policy*, (23):719–740, 1999.

[MR08] Eve Maler and Drummond Reed. The Venn of Identity: Options and Issues in Federated Identity Management. *IEEE Security & Privacy Magazine*, 6(2):16–23, 2008.

[Neu94] Peter G. Neumann. Risks of passwords. *Commun. ACM*, 37(4):126, 1994.

[NGG+07] Anthony Nadalin, Marc Goodner, Martin Gudgin, Abbie Barbir, and Hans Granqvist. WS-Trust 1.3. OASIS Standard, 19.03.2007, 2007.

[NKMHB06] Anthony Nadalin, Chris Kaler, Ronald Monzillo, and Phillip Hallam-Baker. Web Services Security: SOAP Message Security 1.1. OASIS Standard, 01.02.2006, 2006.

[ODE09] ODETTE. SESAM specification for building up federated Single-Sign-On (SSO) scenarios between companies in the automotive sector. ODETTE Recommendation, Draft of 15.07.2009, July 2009.

[OS06] A. Ozment and S. E Schechter. Bootstrapping the adoption of Internet security protocols. In *Fifth Workshop on the Economics of Information Security*, Cambridge, UK, 2006.

[Rog03] Everett M. Rogers. *Diffusion of Innovations*. Free Press, New York, 5 edition, 2003.

[Roß06] Heiko Roßnagel. On Diffusion and Confusion - Why Electronic Signatures Have Failed. In *Trust and Privacy in Digital Business*, pages 71–80. Springer, 2006.

[SNS88] J.G. Steiner, B.C. Neuman, and J.I. Schiller. Kerberos: An Authentication Service for Open Network Systems. Usenix Conference Proceedings, 1988.

[SV99] Carl Shapiro and Hal R. Varian. *Information Rules - A Strategic Guide to the Network Economy*. Harvard Business School Press, Boston, 1999.

[ZFR+07] Jan Zibuschka, Lothar Fritsch, Mike Radmacher, Tobias Scherner, and Kai Rannenberg. Enabling Privacy of Real-Life LBS. In *New Approaches for Security, Privacy and Trust in Complex Environments*, pages 325–336. 2007.

Quantitative Model-Based Safety Analysis: A Case Study

Matthias Güdemann*, Frank Ortmeier
matthias.guedemann@ovgu.de, frank.ortmeier@ovgu.de

Abstract:
The rising complexity of many safety-critical systems necessitates new analysis methods. Model-based safety analysis approaches aim at finding critical failure combinations by analysis of models of the whole system (i.e. software, hardware, and failure modes). The big advantage of these methods compared to traditional approaches is that the results are of very high significance.
Until now, model-based approaches have only to a limited extent been applied to answer quantitative questions in safety analysis. Model-based approaches in this context are often limited to analysis of specific failure propagation models. They do not include system dynamics and behavior. A consequence is, that the methods are very error-prone because of wrong assumptions.
New achievements in the domain of (probabilistic) model-checking now allow for overcoming this problem. This paper illustrates how such an approach for quantitative model-based safety analysis is used to model and analyze a real-world case study from the railway domain.

1 Introduction

Due to the rising complexity and basically ubiquitous application, more and more software intensive systems become safety-critical. The amount of software in such systems is increasing at the same time, posing an additional challenge to build these dependable and fault-tolerant.
In order to prevent as many accidents as possible, a lot of effort is being put into safety in many domains. Requirements for the development and life cycle of safety-critical systems are now specified in many different norms like the general IEC 61508 [Lad08], DO178-B [RTC92] for aviation or ISO 26262 [ISO09] for automotive. In addition to requirements, they present guidelines to make systems more fault-tolerant. All these norms require some sort of safety assessment before a system is put into operation. Many of these methods have a long history (some dating back to the 60ies). However, they are more or less only methodologies and purely rely on skill and expertise of the safety engineer. A potential source of safety-critical problems can only be anticipated if an engineer thinks of it a design time. But this foreseeing becomes ever harder, because of rising hardware and software complexity.

*Acknowledgement: Matthias Güdemann is funded by the German Ministry of Education and Science (BMBF) within the ViERforES project (no. 01IM08003C)

To counter these problems, new model-based safety analysis methods get applied. This means that a model of the system under consideration as well as its environment is built. Then the analysis is not solely based on the engineer's skill but also on the formal analysis of this model. Some examples of such methods are explained in [ABB+06, ORS06, ADS+04, BV03]. In some cases it is even possible to semi-automatically deduce cause-consequence relationships between component failures and loss of system.
These are mostly qualitative methods, which can only show fault tolerance by giving critical combinations of failure modes. But for the assessment of the dependability the most important question is: *"What is/are the probabilities of any of the systems hazards?"* For a satisfying answer, accurate quantitative safety analysis methods must be applied.
In this paper we show the application of a new, quantitative model-based safety analysis technique – probabilistic deductive cause-consequence analysis [GO10] – to a real-world case study and report on our experiences. The paper starts with a short presentation of the case study (Sect. 2). Sect. 3 explains the formal modeling and analysis of the case study and observations made from a comparison to a traditional safety analysis, Sect. 4 discusses some related work and Sect. 5 concludes the paper.

2 Case Study

The following case study of a radio-based railroad control was used as a reference case study used in the priority research program 1064 "Integrating software specifications techniques for engineering applications" of the German Research foundation (DFG), it was supplied by the German railway organization, Deutsche Bahn. The scenario addresses a novel technique for controlling railroad crossings. This technique aims at medium speed routes, i.e. routes with maximum speed of 160 km/h. An overview is given in [KT02].

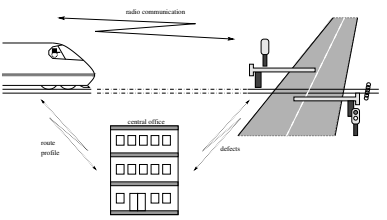


Figure 1: Radio-based railroad crossing

The main difference between this technology and the traditional control of railroad crossings is that signals and sensors on the route are replaced by radio communication and software computations in the train and railroad crossing. This offers cheaper and more flexible solutions, but also shifts safety critical functionality from hardware to software. Instead of detecting an approaching train by a fixed sensor on the track, the train continuously computes the position where it has to send a signal to secure the level crossing. To calculate this activation point the train uses data about its position, maximum deceleration and the position of the crossing. Therefore the train has to know the position of

the railroad crossing, the time needed to secure the railroad crossing, and its current speed and position. The first two items are memorized in a data store and the last two items are measured by an odometer. For safety reasons a safety margin is added to the activation distance. This allows compensating some deviations in the odometer. The system works as follows:
The train continuously computes its position. When it approaches a crossing, it broadcasts a 'secure'-request to the crossing. When the railroad crossing receives the command 'secure', it switches on the traffic lights, first the 'yellow' light, then the 'red' light, and finally closes the barriers. When they are closed, the railroad crossing is 'secured' for a certain period of time. The 'stop' signal on the train route, indicating an insecure crossing, is also substituted by computation and communication. Shortly before the train reaches the 'latest braking point' (latest point, where it is possible for the train to stop in front of the crossing), it requests the status of the railroad crossing. When the crossing is secured, it responds with a 'release' signal which indicates, that the train may pass the crossing. Otherwise the train has to brake and stop before the crossing. Behind the crossing, a sensor detects that the train has passed and an 'open' signal is sent to the crossing. The railroad crossing periodically performs self-diagnosis and automatically informs the central office about defects and problems. The central office is responsible for repair and provides route descriptions for trains.

3 Model-Based Safety Analysis

Our model-based safety analysis consists of building a formal system model which can then be analyzed using formal analysis techniques based on temporal logics and model checking. The accuracy of the formal system model is the most important factor in the accuracy of the whole analysis, especially the modeling of the probabilistic behavior.

3.1 Building a formal model

For model-based safety analysis it is necessary to model (a) the controlling software, (b) the controlled hardware, (c) the environment and (d) possible failure modes. We can not show the full case study here, but only show one part of it. Fig. 2 shows a model of the crossing in state charts notation[1].
Initially the barriers (of the crossing) are *opened*. When the crossing receives a close request from an arriving train - i.e. condition *Close* becomes true, the barriers start *closing*. This process takes some time. After a certain amount of time the barriers are *closed*. They will remain closed until the train has passed the crossing (detected by a sensor). The barriers reopen automatically after a defined time interval. This is a standard procedure in railroad organization, as car drivers tend to ignore closed barriers at a railroad crossing if

[1] In this paper, we use a basic (but very precise) semantics for state charts. Basically no queues are allowed, events do not persist and parallel composition is synchronous. This semantics is very intuitive on the one hand and very easy to translate into model checker input languages on the other hand.

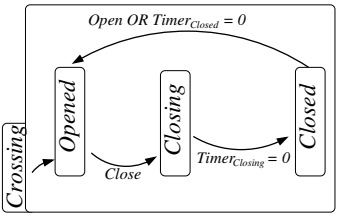


Figure 2: Model of the crossing

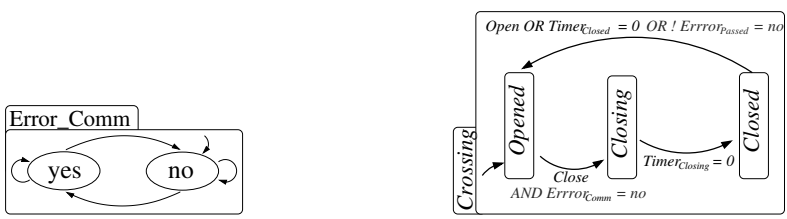
the barriers are closed too long. So it is better to reopen the barriers, than having car drivers slowly driving around the closed barriers. Analogously the other parts of the system (i.e. the train, radio communication, brakes, cars, control software) are modeled. Together this forms a functional model of the system. This means it is a model of the system in an ideal world, where no errors occur.
The safety goal of the system is clear: it must never happen that the train is on the crossing and a car is passing the crossing at the same time. A well designed control system must assure this property at least as long as no component failures occur. The corresponding hazard H is "a train passes the crossing and the crossing is not secured". This is the only hazard which we will consider in this paper.

3.2 Modeling failure modes

The next step is to extend this model such that failures are also correctly described. In this paper we do not address the question which failure modes are necessary to model[2], we assume, that this has already been determined. For the example the following six failure modes are considered: **failure of the brakes** (*error_brake*) which describes the failure of the brakes, **failure of the communication** (*error_comm*) which describes the failure of the radio communication, **failure of the barriers closed sensor** (*error_closed*) which describes that the crossing signals *closed*, although it is not closed, **failure of the barriers' actuator** (*error_actuator*) which describes that the actuator of the crossing fails, **failure of the train passed sensor** (*error_passed*) which describes that the sensor detecting trains which passed the crossing fails and **deviation in the odometer** (*error_odo*) which describes that the odometer does not give 100% precise data.
These failure modes are integrated into the formal model. Modeling of failures can always be split into two (sub-)tasks: modeling of the occurrence pattern and modeling of the direct effect of the failure mode. *Occurrence patterns* describe how and when a given failure mode may occur and are modeled by failure charts.
The most basic failure chart has two states, one state *yes* modeling the presence of the failure and one state *no* modeling its absence. The transitions between these states determine

[2]There are numerous other techniques for answering this question. They range from experienced based approaches like component specific list of failure modes to formally grounded methods like failure-sensitive specification [OR04].

the type of failure mode. For example: if the state *yes* can never be left once it became active, the failure mode is called *persistent*. If the state may non-deterministically switch between *yes* and *no*, it is called *transient*. More complex occurrence patterns (e.g. broken until repair) are also possible. Which occurrence pattern is best fitting for a given failure mode is a design/analysis decision.
Modeling of the *direct effects of the failures* of a failure mode basically comes down to adding transitions and/or states to the functional model, which are activated or in the case of additional states, reachable, if the failure occurs (resp. if the corresponding failure automaton is in state "yes").
Correct modeling of failures is a difficult and error-prone task. From a formal point of view the semantics of the original model (without failure modes) and the extended model have little to do with each other. From an intuitive point of view, one would expect some sort of trace-inclusion, i.e. the original behavior of the functional system must still be possible in the extended system model. This property can be assured, if some syntactic rules are followed during modeling of failure modes' direct effects [OGR07].
Fig. 3(b) shows the modeling of the failure effect of the failure modes *error_comm* and *error_passed* for the crossing, Fig. 3(a) shows the transient failure automaton for *error_comm*. If it is active then state *Opened* will not be left although a *Close* signal was sent, i.e. the communication failed. The failure effect of *error_passed* is modeled such that state *Closed* may be left if the corresponding failure automaton is in state *yes*, i.e. a misdetection of the sensor happened.



(a) Failure Automaton (b) Failure Effects Modeling

Figure 3: Modeling of error_comm failure mode

3.3 Modeling for Quantitative Safety Analysis

For an accurate quantitative model-based analysis, probabilistic information must be added to the extended system model. Accurate modeling of the occurrence pattern and the occurrence probability of failure modes is very important. There exist two main types of failure probabilities. The first is *per demand* failure probability, which is the probability of the system component failing its function at a given demand (comparable to *low demand mode* in IEC 61508). The second is *per time* probability, which is the *rate* of failures over a given time interval (comparable to *high demand or continuous mode* in IEC 61508).
Which type of failure probability is best fitting for a given failure mode can only be decided on a case-by-case basis. Transient sensor failures will very often be modeled as a

per time failure mode, as they are active the whole time. Other failure modes, like the activation of a mechanical device, will often be modeled as a *per demand* failure, as a clear moment of activation exists. Per demand failures often are of persistent nature[3].

3.3.1 Discrete Time Model

The underlying semantics of the formal model is a discrete time model in which the temporal behavior of the failure modes and system model must be integrated. For each time step the system performs, an equal amount of time δt passes, which is called the *temporal resolution*. For the example case study, the formal model consists of $10km$ of railroad tracks. The average speed is $115\frac{km}{h}$. The formal system model in built in a way that this translates to a temporal resolution of $\delta t = 5s$, i.e. for every system time step, $5s$ pass.

3.3.2 Modeling per time failure modes

Transitions of the failure automata are labeled with constraints of the form $(p : \phi)$ which translates to: "If ϕ holds, then the transition is taken with probability p". This can be abbreviated by omitting p which then results in probability 1. In order to be a well defined probabilistic model, for each transition label, the outgoing transition probabilities must always sum up to 1. This assures that a valid probability measure is defined.
A *per time* failure mode can be modeled by adding the failure probability to the transition from state *no* to state *yes* as shown in Fig. 4(a). Making its potentially non-deterministic behavior *probabilistic*. The activation condition is always *true* and the sum of probabilities of outgoing transitions is always 1. In the case shown, the failure automaton enters state *yes* with a probability p, stays in this state with the same probability and enters (and stays in) state *no* with probability $1 - p$.
The parameter for a per-time failure is normally given as a *failure rate* λ and interpreted as the parameter for the exponential distribution function (Eq. 1) which computes the probability that a failure occurs before time t. This continuous function can be approximated in discrete time using a per-time failure automaton as shown in Fig. 4(a). This leads to a geometric distribution function (Eq. 2) which computes the probability that a per-time failure appears in at most k time-steps. Setting time $t = k \cdot \delta t$, gives a good discrete time approximation of the exponential distribution [GO10].

$$P(X \leq t) = \int_0^t e^{-\lambda t} dt \qquad (1) \qquad\qquad P(X \leq k) = 1 - (1-p)^k \quad (2)$$

With the given δt, the failure rates for the per-time failures can be converted to transition probabilities for per-time failure automata. Modeling the $error\_passed$ as *per time* failure mode with a failure rate of $7e^{-9}\frac{1}{s}$ and the failure of the odometer with a failure rate of $6e^{-5}\frac{1}{s}$, the transition failure probabilities are $3.5e^{-8}$ for $error\_passed$ and $3e^{-4}$ for $error\_odo$. The deviation of the odometer is then modeled by choosing a deviation from the set $\{-3, -2, -1, 0, 1, 2, 3\}$. The probability of the deviation is modeled to be normally distributed with $\mu = 0\frac{m}{s}$ and $\sigma = 1\frac{m}{s}$.

[3] But there exist of course other failure modes, which are transient and should be modeled as per demand probabilities resp. persistent failure modes, which should be modeled with per time failure rates.

(a) Per-time (b) Per-demand

Figure 4: Failure Automata for quantitative failure mode modeling

3.3.3 Modeling per-demand failure modes

Correct modeling of a *per demand* failure mode is more complex. The challenge is to specify a correct occurrence pattern such that assures that the failure mode can *only* appear if a demand to the component is given and is then taken only with the given failure probability. To accomplish this, the failure probability is first added to the transitions to state *yes* and the state *no* may only be left if there is a *demand* to the component. In other words, a predicate *demand* is a necessary condition for the occurrence of the failure mode. A failure automaton for a *per demand* failure mode is shown in Fig. 4(b).
The failure mode *error_comm* is integrated in this way. In this case, there is a demand if the crossing is in state *Opened* **and** the *Close* signal is sent. If *error_comm* is not present, then the crossing enters state *Closing*, if the failure mode is present or no *Close* signal is sent, the state *Opened* is not left. When *demand* holds $((Crossing = Opened) \wedge Close)$, the failure automaton can change to state *yes* with probability p or stay in state *no* with probability $1 - p$. This means that in the next time step, it is possible that the crossing is either in *Opened* or *Closing*. To model this correctly, the state *Undecided* is introduced which represent either of these states.
In addition a *decide* automaton as shown in Fig. 5(b) is added, which changes from state *undef* to *[Opn, Clg]* if *demand* holds. Based on these two automata the predicates $in(Opened)$ and $in(Closing)$ are defined as shown in equation (3) and (4) which indicate the "real" state of the crossing, *Crossing'* refers to the model of the crossing with per-demand failure mode modeling.

$$\begin{aligned} in(Opened) &:= Crossing' = Opened \vee (Crossing' = undecided \\ &\quad \wedge decide = [Opn, Clg] \wedge \neg failure = no) \qquad (3) \\ in(Closing) &:= Crossing' = Opened \vee (Crossing' = undecided \\ &\quad \wedge decide = [Opn, Clg] \wedge failure = no) \qquad (4) \end{aligned}$$

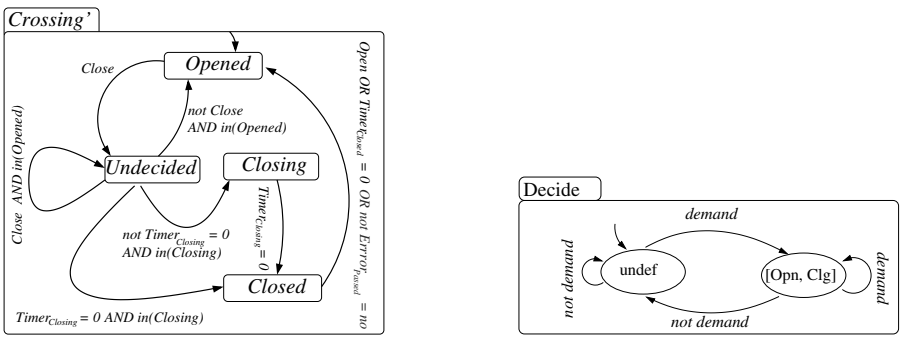
As *Undecided* represents either state *Opened* or *Closing*, depending on the state of the failure automaton, the transitions of these states must be added to the state *Undecided* , in conjunction with the the above predicates.

- For the state *Opened*

  - to state *Opened*, activation condition $\neg Close \wedge in(Opened)$, i.e. in state *Opened* and there is no demand
  - to state *Undecided*, activation condition $Close \wedge in(Opened)$, i.e. there is a demand that possibly fails

- For the state *Closing* the following transitions must be added to *Undecided*:
  - to state *Closing*, activation condition $\neg Timer = 0 \wedge in(Closing)$
  - to state *Closed*, activation condition $Timer = 0 \wedge in(Closing)$

The complete modeling of the *per demand* failure effect for *error_passed* is shown in Fig. 5(a). The decide automaton, *Crossing* and the per-demand failure automaton together show the same *observable* behavior as the qualitative failure modeling. It becomes clear why such a complex construction is necessary: the state *Opened* has successor states if there is a *demand* and successors states if there is no *demand*. Therefore only being in state *Opened* is not sufficient to specify the *demand* predicate, because the failure automaton could then make a transition at the wrong time, resulting in wrong overall probabilities, especially important for persistent failures which cannot disappear after their occurrence. We give an example trace that shows the relevant predicates and states of the automata to illustrate that the resulting traces of the probabilistically modeled system are the same as for the system for qualitative analysis, if the *per demand* failure modes are correctly integrated as described. The first trace shows the behavior of the extended system model



(a) Failure Effects Modeling (*Crossing'*)

(b) Decide Automaton

Figure 5: Modeling of per-demand failure mode error_com

without explicit modeling of the *per demand* failure mode. The failure appears after the first time step, but has no effect there because there no *Close* command was sent. At the third time step, the failure appears again, now it has an effect (*Close* command was sent) causing the crossing to stay in state *Opened*. Afterwards the failure disappears, *Close* is sent again and the crossing enters state *Closing*. The following trace is the corresponding trace of the extended system model with probabilistic modeling of the *per demand* failure. Here the failure can only appear at the fourth time step, as there was no *demand* before. This causes the *decide* automaton to enter the state **[Opn,Clg]** and the crossing to enter

time-step	1	2	3	4	5
error_comm	no	yes	yes	no	no
Crossing	*Opened*	*Opened*	*Opened*	*Opened*	*Closing*
Close	$false$	$false$	$true$	$true$	$false$

state *Undecided*. The failure is active at the fourth time step, therefore *in(Opened)* holds and the "real" state of the crossing is *Opened*. At the fifth time step, the *Close* command is sent again, this time the failure disappears and *in(Closing)* holds. Both traces show the

time-step	1	2	3	4	5
error_comm	no	no	no	**yes**	**no**
Crossing'	*Opened*	*Opened*	*Opened*	***Undecided***	***Undecided***
decide	*undef*	*undef*	*undef*	**[Opn,Clg]**	**[Opn,Clg]**
in(Opened)	$true$	$true$	$true$	**true**	$false$
in(Closing)	$false$	$false$	$false$	$false$	**true**
Close	$false$	$false$	$true$	$true$	$false$

same observable behavior, i.e. the states of the crossing and therefore the failure effects are the same in both cases. Nevertheless it can easily be seen that the failure mode can only appear at the time of a demand, whereas in the modeling without per demand failures the failure can appear at any time, but its effect manifests only at certain time steps (the demands in the probabilistic modeling). This illustrates again that in order to compute accurate probabilities, this distinction is necessary although the correct integration in not trivial. A much more detailed discussion about the integration of both per-demand and per-time failure modes can be found in [GO10].

3.4 Analysis Results

The constructed probabilistic model was analyzed using the PRISM model checker using hybrid models (sparse matrices and MTBDDs) on an Athlon 64 X2 4800+ CPU with 2 Gbyte of RAM. The overall model consisted of 11295021 reachable states and 518674960 transitions resulting from the parallel composition of the state machines of the model description. The duration of the analysis was approx. 190 minutes and 279.7 MByte of RAM were allocated. The hazard H was defined as "the position of the train is on the crossing and the crossing is not in state *closed*". Equation (5) shows the result of the quantitative analysis. The probabilistic temporal logic formula is explained in [GO10].

$$P(H) = 4.47363 \cdot 10^{-6} \tag{5}$$

Compared to a more traditional analysis method based on the a-posteriori estimation of the hazard probability on the resulting critical combinations of failures (for details see [ORS05]), this result is much more accurate. The following equation shows the result using the standard approach for quantitative fault tree analysis (FTA) [VDF+02] as

shown in formula (6).

$$P_{FTA}(H) \leq \sum_{\Delta \in mcs} \prod_{\delta \in \Delta} P(\delta) \quad (6)$$

$$P_{FTA}(H) \leq P(error\_passed) + P(error\_odo) + P(\text{cut sets of size } \geq 2)$$

$$P_{FTA}(H) \leq 2.8 \cdot 10^{-6} + 2.5 \cdot 10^{-2} + O(1 \cdot 10^{-14}) \approx 2.5 \cdot 10^{-2} \quad (7)$$

For this calculations, the probabilities for $error\_passed$ and $error\_odo$ must be estimated for *one* train passing the crossing. In the example, it was assumed that the train needs roughly[4] 400s. These coarse results (Eq. (7)) could then be refined with constraint probabilities. For example, one could deduce that only deviations of at least $2\frac{m}{s}$ are dangerous. This would then lower the probability for $error\_odo$ by the order of 2. It is still rather obvious that this estimation is very coarse compared to quantitative model based analysis. Another drawback of the traditional approach is that if the system is not carefully analyzed for temporal behavior, the estimation can easily be too optimistic. Just try to decide if a deviation of $3\frac{m}{s}$ could also be justified or not.
The biggest advantage of the model-based quantitative analysis is that the stochastic dependencies[5] which often are inherent in such a system are *automatically* considered in the system model, whereas all a-posteriori methods depend on the stochastic *independence* which is often not satisfied.

4 Related Work

Some related model-based analysis methods like [BV03, ABB+06, ADS+04] have already been mentioned. Together with basic fault tree analysis [VDF+02] they have the disadvantage that the *results* of a qualitative analysis are used for quantitative estimations. As already discussed, these estimations rely on the assumption of stochastic independence, which is often not fulfilled.
Notable approaches for doing quantitative safety analysis directly on a formal system model is presented in [BPRW08]. In this work, critical combinations of failures, analogously to the minimal cut sets of FTA or minimal critical sets of DCCA are analyzed for their relative impact. They are sorted according to their contribution to the overall hazard probability. The difference is that the analyzed system behavior is limited to allow *only* failure modes in such a set. This alters the set of possible traces of the system models and the probabilities in such a way that no overall hazard probability can be computed.
Very interesting approaches are developed in the COMPASS [BCK+09a] project with its SLIM modeling language [BCK+09b] and in [GCW07]. Both approaches are similar to ours in analyzing whole system models. The difference is, that solely continuous time models are used. This is well suited to asynchronous interleaved systems but not to syn-

[4] Derived from the formal model of the railroad crossing.

[5] Consider for example that the faster the train, the more likely is the situation that the sensor is passed without sensor failure. A slower train speed translates to a higher probability of a $error\_passed$ failure mode, as the more time passes the more the probability shrinks that the failure does not occur.

chronous parallel systems [HKMKS00]. This also means that per-demand failure modes are not supported, as these are not directly expressible in the continuous time semantics.

5 Conclusion

The quantitative model-based safety analysis method presented in [GO10] was applied to a real world case study of a radio-based railroad crossing. The modeling included both per-time and per-demand failure modes and normally distributed deviations of a sensor. The limiting factor of the analysis method is the size of the state space (which is larger for probabilistic analyses than for qualitative analyses) and the running time of the model-checking. But as we have shown, a realistic case study is analyzable in that way and probabilistic model-checking is an active research area. Different abstractions to reduce the state space have been proposed, multi-terminal BDDs [KNP02] are already in use in PRISM, bisimulation- based abstractions are integrated in MRMC [KKZJ07] and the necessary numerical analysis can be moved to massive parallel GPU computation [BES09]. The analysis results are very promising, as all the stochastic dependencies are automatically reflected in the analysis, resulting from the analysis using probabilistic model-checking techniques. The different types of failure modes can be integrated into the model and although the system is modeled in a discrete time context, accurate continuous time approximation is possible. Future work will include developing a modeling framework for the combined usage of qualitative safety analysis, to compute the critical combinations and quantitative safety analysis to compute the overall hazard probabilities.

References

[ABB+06] O. Akerlund, P. Bieber, E. Boede, M. Bozzano, M. Bretschneider, C. Castel, A. Cavallo, M. Cifaldi, J. Gauthier, A. Griffault, O. Lisagor, A. Luedtke, S. Metge, C. Papadopoulos, T. Peikenkamp, L. Sagaspe, C. Seguin, H. Trivedi, and L. Valacca. ISAAC, a framework for integrated safety analysis of functional, geometrical and human aspects. In *Proceedings of ERTS 2006*, 2006.

[ADS+04] Parosh Aziz Abdulla, Johann Deneux, Gunnar Stalmarck, Herman Agren, and Ove Akerlund. Designing Safe, Reliable Systems using SCADE. In *Proceedings of ISOLA'04*. Springer-Verlag, 2004.

[BCK+09a] Marco Bozzano, Alessandro Cimatti, Joost-Pieter Katoen, Viet Yen Nguyen, Thomas Noll, and Marco Roveri. COMPASS project webpage. http://compass.informatik.rwth-aachen.de/, 2009.

[BCK+09b] Marco Bozzano, Alessandro Cimatti, Joost-Pieter Katoen, Viet Yen Nguyen, Thomas Noll, and Marco Roveri. Model-Based Codesign of Critical Embedded Systems. In *2nd International Workshop on Model Based Architecting and Construction of Embedded Systems*, pages 87–91. CEUR-WS.org, 2009.

[BES09] D. Bosnacki, S. Edelkamp, and D. Sulewski. Efficient Probabilistic Model Checking on General Purpose Graphics Processors. In C. Pasareanu, editor, *Proc. 16th International SPIN Workshop*, volume 5578 of *LNCS*, pages 32–49. Springer, 2009.

[BPRW08] Eckard Böde, Thomas Peikenkamp, Jan Rakow, and Samuel Wischmeyer. Model Based Importance Analysis for Minimal Cut Sets. Reports of SFB/TR 14 AVACS 29, SFB/TR 14 AVACS, Apr 2008. ISSN: 1860-9821, http://www.avacs.org.

[BV03] M. Bozzano and Adolfo Villafiorita. Improving System Reliability via Model Checking: theFSAP/NuSMV-SA Safety Analysis Platform. In *Proceedings of SAFECOMP'03*, pages 49–62. Springer, 2003.

[GCW07] L. Grunske, R. Colvin, and K. Winter. Probabilistic Model-Checking Support for FMEA. In *Proc. 4th International Conference on Quantitative Evaluation of Systems (QEST'07)*, 2007.

[GO10] Matthias Güdemann and Frank Ortmeier. Probabilistic Model-based Safety Analysis. In *Proceedings of the* 8^{th} *Workshop on Quantitative Aspects of Programming Languages (QAPL10)*. EPTCS, 2010.

[HKMKS00] Holger Hermanns, Joost-Pieter Katoen, Joachim Meyer-Kayser, and Markus Siegle. A Markov Chain Model Checker. In *TACAS '00: Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 347–362, London, UK, 2000. Springer-Verlag.

[ISO09] ISO/WD 26262: Road Vehicles-Functional Safety, 2009.

[KKZJ07] Joost-Pieter Katoen, Tim Kemna, Ivan S. Zapreev, and David N. Jansen. Bisimulation Minimisation Mostly Speeds Up Probabilistic Model Checking. In *TACAS*, pages 87–101, 2007.

[KNP02] M. Z. Kwiatkowska, G. Norman, and D. Parker. Probabilistic Symbolic Model Checking with PRISM: A Hybrid Approach. In *TACAS '02: Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 52–66, London, UK, 2002. Springer-Verlag.

[KT02] J. Klose and A. Thums. The STATEMATE Reference Model of the Reference Case Study 'Verkehrsleittechnik'. Technical Report 2002-01, Universität Augsburg, 2002.

[Lad08] Peter B. Ladkin. An Overview of IEC 61508 on EEPE Functional Safety, 2008.

[OGR07] Frank Ortmeier, Matthias Güdemann, and Wolfgang Reif. Formal Failure Models. In *First IFAC Workshop on Dependable Control of Discrete Systems (DCDS 07)*. Elsevier, 2007.

[OR04] F. Ortmeier and W. Reif. Failure-Sensitive Specification: A Formal Method for Finding Failure Modes. Technical Report 3, Institut für Informatik, Universität Augsburg, 2004.

[ORS05] F. Ortmeier, W. Reif, and G. Schellhorn. Formal safety analysis of a radio-based railroad crossing using Deductive Cause-Consequence Analysis (DCCA). In *Proceedings of 5th European Dependable Computing Conference EDCC*, volume 3463 of *LNCS*. Springer, 2005.

[ORS06] F. Ortmeier, W. Reif, and G. Schellhorn. Deductive Cause-Consequence Analysis (DCCA). In *Proceedings of IFAC World Congress*. Elsevier, 2006.

[RTC92] RTCA. DO-178B: Software Considerations in Airborne Systems and Equipment Certification, December, 1st 1992.

[VDF+02] Dr. W. Vesley, Dr. Joanne Dugan, J. Fragole, J. Minarik II, and J. Railsback. *Fault Tree Handbook with Aerospace Applications*. NASA Office of Safety and Mission Assurance, August 2002.

Real-Time Fault-Tolerant Routing in High-Availability Multicast-Aware Video Networks

Roman Messmer
ORF Austrian Broadcasting Corporation
Broadcast Production Systems
A-1136 Vienna, Austria
roman.meszmer@orf.at

Jörg Keller
FernUniversität in Hagen
Dept. of Mathematics and Computer Science
58084 Hagen, Germany
joerg.keller@fernuni-hagen.de

Abstract: Live-videostream networks based on multimedia switches are the most recent products used in television production and distribution facilities to transport the live signal from sources like cameras or microphones to dedicated sinks like video monitors, loudspeakers and transmission lines. To switch signals from a single source to several destinations multicasting or point-to-multipoint technology is considered. To compute multicast trees for multimedia communication, constrained shortest paths algorithms are needed. They are fundamental to important network functionality such as Quality of Service (QoS) routing or Multiprotocol label switching (MPLS) path selection and the problems they attempt to solve are known to be NP-complete. In previous work, we have introduced a heuristic called Multimedia Multicast algorithm (MulMic), which delivers nearly optimal multicast trees in a short time. Here, we propose the combination of MulMic and two models for fault-tolerant routing: disjoint paths and reservation of backup paths. Furthermore we introduce a realtime algorithm we call ZirkumFlex to handle one or even several simultaneous node or line failures in a multicast network by a local search to bypass the failed node or line. We also apply our algorithm to example graphs to demonstrate its feasibility.

1 Introduction

Implementing distributed media applications like high-resolution video routing on switched packet networks faces several obstacles. The most difficult but essential problem concerns ensuring quality of service (QoS). One precondition is knowing the constrained shortest path, i.e. the cheapest path that satisfies a number of constraints, especially the number of routing hops. Depending on the hardware each single hop adds up several microseconds to the total delay time of the signal path and leads to problems in terms of signal synchronization or even causes frame delayed videos which have to be processed together in follow-up working steps in the video and audio domain. The scale of the problem is better understood when the hop time of a switch is compared with the duration of a video line at the usual high definition video format 720p/60 which lasts about 23 microseconds. Most devices can cope with a time window of several lines without having to synchronize the input signal to an external clock and increase the total transfer time of the video signal additionally. Depending on the video product a delay of up to 200 microseconds is acceptable which makes the minimization of hops an important factor. In most cases one

source has to distribute a signal to a multitude of destinations, e.g. a video signal from a camera is transported to studio monitors and live video mixers at the same time. Thus, we have a multicast or point-to-multipoint routing problem with constraints. We will use the two terms interchangeably. In previous work, we have proposed an algorithm for this problem called MulMic that combines preprocessing with very fast path calculation upon connection requests [MK09].

The failure of nodes in switched packet networks often leads to unacceptable delays or total failure of running processes. Therefore, we investigate fault-tolerance measures in this scenario. We present three algorithms to tolerate node or link failures: (1) by allocating with every multicast tree a backup multicast tree on disjoint paths, (2) by reserving shareable backup trees and (3) introduce an online algorithm that on occurrence of a failure performs a graph search from surrounding graph nodes to bypass the failed node as quickly as possible.

To the best of our knowledge all previous work that investigates fault tolerance without extensive a priori resource consumption refer to point-to-point routing only. Of most interest in our paper is the avoidance of perceptible interruptions of the affected media signals with as little overhead as possible. This is what we mean by "real-time" in our scenario.

The remainder of this article is organized as follows. In Section 2 we discuss related work. In Section 3 we briefly review the MulMic algorithm. In Section 4 we present the various fault-tolerance algorithms, and in Section 5 we apply our algorithm to example graphs. We conclude in Section 6.

2 Related Work

In multicast-aware IT networks the fault tolerance aspect differs slightly from high speed media networks in terms of the importance of continuity of streams. Video streams may not be interrupted, frozen displays are very unwanted events. In [UZ06] a heuristic called *Adaptive distributed Concurrent Shortest Path heuristic* is presented which generates delay-aware constrained multicast trees. After removing the failed subtree the source node is informed about the former covered subtree and its destinations. Then a new loop-free subtree from (subtree-)source to destination nodes is established. This procedure delivers a nearly optimal routed multicast tree, but its calculation is very time-consuming. A very effective distributed algorithm was introduced in [Gu03], where parts of the network were analyzed for backup routes which could be used in case of a node failure. This approach extends the resource reservation protocol (RSVP, RFC2208) to improve resource utilization, higher call acceptance rate and better quality of service (QoS). The level of fault tolerance of each connection can be controlled separately. Earlier works as [Gr89], [Ch93a] and [Ch93b] used distributed algorithms to restore point-to-point connections after network failures. Their work does not mention point-to-multipoint connections. [ZS92] presented algorithms for realtime communication together with fault tolerance aspects using the multiple paths between a pair of communicating nodes to make more efficient use of the network. While most of the mentioned papers treat link failures, [Ko90] also an-

alyzes node failures. Communication for restoration bases on network flooding and path route monitoring. [RM03] analyzed multiple segmented backup scenarios for mission critical applications. In [YH88] failure immunization technology for the use in the fiber optic domain is presented and analyzed.

3 Realtime Routing Algorithm

In [MK09] we proposed an algorithm called Multimedia Multicast Algorithm (MulMic) which calculates a multicast tree in $O(|V| + |E| + |M| \cdot \log |M|)$ time where V, E and M represent the set of vertices, edges and multicast destination nodes, respectively. Experiments with different graphs demonstrated not only the favorable runtime of the algorithm but also another important property of the calculated routing. The result of a rerouting after eliminating a failed node is in many cases very similar to the original routing. So there are only few routing commands necessary to restore a multicast path.

In Fig. 1 this property is illustrated. A 10×10 grid network[1] was set up with 14 multicast destination nodes in different portions of the graph. Some are far apart, e.g. (10) or (36), and some form groups, such as (77, 78, 87, 88). Then a MulMic run was started with source node (55). The resulting routing to node (10) starts at (55), leading upwards to (5) and to the right to (10). For destination (91) the routing starts at (55) leading left to (51) and down to (91). Then a node failure at (7) and a line failure (61-71) were simulated. Failure (7) brought up a rerouting of the path to (10) using (15) to (18), then (8) to (10) to reach its destination. The new routing even leaves the routing from (5) to (6) untouched. A similar case appears with the path from (51) to (91). The new routing results in a path (52) to (72), (71) to (91) which leaves the majority of line routings untouched. Combined with the fault tolerance procedures in Section 4, most of the nodes affected by failure of a node can be supported by a new signal path without noticeable interruption. We now present the algorithm in detail. To achieve the necessary realtime behavior, the proposed algorithm is divided into a time-consuming offline part and a fast online part.

The offline part constructs a routing table for each node of the graph beforehand. The local routing table is then computed in parallel at each switch of the network. The data is updated by accessing a common database containing the local network topology information. The switches listen for node-state and line-state information from other links to update the local routing table and bandwidth consumption. After processing a point-to-multipoint connection request, the common database and the local routing information are updated synchronously. The Floyd-Warshall algorithm, an all-pairs shortest path algorithm with complexity $O(n3)$, can be applied to compute the distances and paths between all nodes of the network graph. In our example a cost of 1 is assigned to every link between routing nodes, because the constant major signal delay occurs during the opto-electronic conversion in the switches (several microseconds), not on the fibre-optic line (nanoseconds, less than a microsecond even for long distance links). After finishing the first part every router has knowledge about the network, e.g. the number of nodes and their connections.

[1] The grid topology is used throughout the paper for the sake of simplicity.

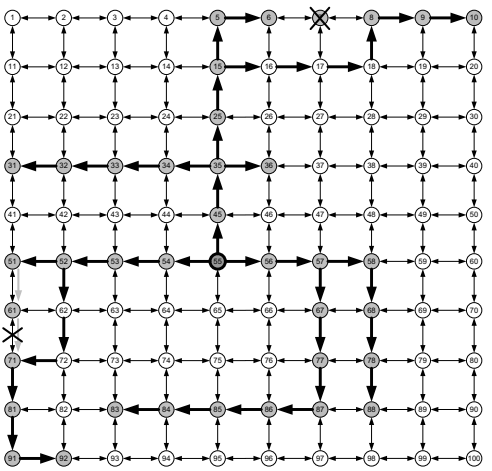


Figure 1: Property of MulMic routing algorithm

The online part starts with a request for a point-to-multipoint path calculation. Let $G = (V, E)$ be the graph representation of the network, V the set of all nodes, E the set of all edges, $M = \{v_1, \ldots, v_m\}$ the set of destination nodes, and s the source node. The online part of the algorithm proceeds as follows:

1. Retrieve all distances $d(s, v_1), \ldots, d(s, v_m)$, i.e. the distances of all destinations from the source, from the routing table, together with the corresponding paths. and sort the distance values in ascending order. From now on we will assume $v_1, \ldots, v_m$ to represent this order, i.e. v_1 has the shortest distance from s, and v_m has the largest distance from s.
2. Start the Multipoint path $MPATH$ with $MPATH = path(s, v_m)$. Let i be a loop index counting down from $m = |M|$ to 2, to add node v_{i-1} to the multicast tree.
   (a) If $dist(v_i, v_{i-1}) < dist(v_{i-1}, s)$
   then add path(v_i, v_{i-1}) to $MPATH$
   else add path(s, v_{i-1}) to $MPATH$. Information about the path length between v_i to v_{i-1} is retrieved from the local routing table.
   (b) (optional) If other destination nodes v_j, where $j < i - 1$, are already covered by the existing path, they are no longer considered for future calculation.
   (c) A variable which calculates the number of hops is increased when the if-case has been taken, i.e. if the shortest path to the next destination is shorter than to the source. The variable is reset to zero otherwise. If the maximum hop count is reached, the next destination node must be connected to the source. So an unacceptable delay time due to the accumulation of hop time is avoided. This way the routing propagation is limited.
3. As there is the possibility of loops in the constructed graph $MPATH$, a general spanning-tree algorithm realized by a depth-first search is used to eliminate them.

The implementation of this algorithm delivers a single point-to-multipoint routing result in record low time $O(|V| + |E| + |M| \cdot \log |M|)$. See [MK09] for a proof.

4 Fault-Tolerance Concepts

We now present three concepts to add fault tolerance to the routing method we mentioned above. The concepts differ in the amount of additional bandwidth that they allocate in the fault-free case, and the amount of calculation necessary in the presence of a fault.

4.1 Backup path model

This approach tries to establish redundancy by a backup multicast tree with paths disjoint from the multicast tree calculated by the MulMic algorithm, as far as is possible, since the graph is not required to be 2-connected. Its advantage is that in case of a failure, shifting transmission to the backup tree is all that needs to be done. This comes at the cost of doubling the bandwidth consumption. First the primary routing path is to be calculated. The MulMic algorithm from Section 3 is used for realtime calculation of the point-to-multipoint path. The calculated path is transformed into routing commands, which are then sent to the routers along the multicast path. Next the backup routing path is calculated. A graph with modified edge cost is used to calculate disjoint paths between source and destination. Links used in step 2 get a high cost value (e.g. edge cost set to number of total graph edges), so that a shortest-path algorithm calculates paths disjoint from the primary path whenever possible. If there is no disjoint path alternative, the original path must be used to transport the media stream. At least the source router as well as the destination routers are always examples of such non-disjoint paths if they don't provide multiple network interfaces. The calculated secondary path is then transformed into routing commands and sent to the routers. After positive switch confirmation the bandwidth management (BM) is called to update the database. Next we concern the failure management of the backup path model. Immediately (usually within microseconds) after a router or a link failure the destination nodes detect an error[2], the redundant secondary channel immediately is being activated for resuming the transfer of media over the backup path. The defective lines or nodes of the graph are removed from the routing table (locally and in the remote database) and an all-pairs shortest path algorithm with complexity $O(n3)$ is called again afterwards, i.e. offline, to update the topological information base for the multicast algorithm. To select disjoint paths we recommended to set the link cost of the primary path to a certain high value. This is also used to recalculate a connected primary path. The algorithm presented in section 3 is appropriate because in most cases it leaves the already working routed links in place and only presents backup routings for the defective line or node.

[2]Physical detection via Loss Of Light (LOL) or CRC error.

4.2 Reservation model

Instead of actually establishing backup paths one can also simply reserve such paths by means of Resource ReSerVation Protocol (RSVP) [RS06]. The advantage of this variant is that backup path reservations can be shared between trees that use disjoint links that never will both be affected by a node or link failure. The disadvantage is the overhead to establish the reserved backup path in case of a failure. The network design which uses a number n of backup paths for m signal paths is called $n : m$ p-cycle and has been studied extensively.

There are concepts like shared link risk groups (SLRG) [Ru05] which consider a set of links that may fail simultaneously because of a common risk they share on the network. For example, they may all be less prioritized destinations. In [RC08] a dual link failure resiliency is considered in a project called backup link mutual exclusion (BLME). P-Cycle in multidomain networks are surveyed in [Dr09].

4.3 Online failure management model

This proposal does not assume any additional reserved or switched redundancy path. Instead it assumes that there is additional bandwidth available in the network that can be used for a failure-triggered routing. The basic idea of this model which we will call *ZirkumFlex* is a combination of the MulMic routing algorithm together with a rapid calculation of a path alternative in case of a node or link failure. The alternative path changes only those routes directly affected by the failure.

In Fig. 2 there is an example for the ZirkumFlex algorithm operating in a common graph. The upper image shows a part of the network graph, the lower the multicast tree only. The node s represents the source node, routing starts from there. Destination nodes $d_1, \ldots, d_6$ are in dark gray. The figure shows the moment when a failure occurs at the node with the flash symbol. Because the multicast tree is loop-free per definition there is only one single predecessor of any node in the tree. We begin at the already established multicast path. An additional feature of the MulMic algorithm sets the variable for the distance from the source node d in each node without extra cost in a depth-first search. Every node holds its number/name, distance d, the multicast call number (to distinguish between several multicast routings per node) and finally a linked list with its source name/number initially filled in plus a rendezvous field containing encountered successor names. A node list holds node names not yet connected to the predecessor.

After a failure has been detected the predecessor and the successors of the failed node(s) is/are determined. A following test of the predecessor (pre) node being a member of a unicast or multicast routing determines if rerouting activity is to be set. If the pre node is not a member, only the central database is to be informed about the failure and no other action is taken. Else if the local node indeed is a member, start the ZirkumFlex-algorithm detailed in Fig. 4. To achieve link and path restoration on large graphs in short time and avoiding the $O(n^3)$ all pairs shortest path algorithm over the whole graph we propose

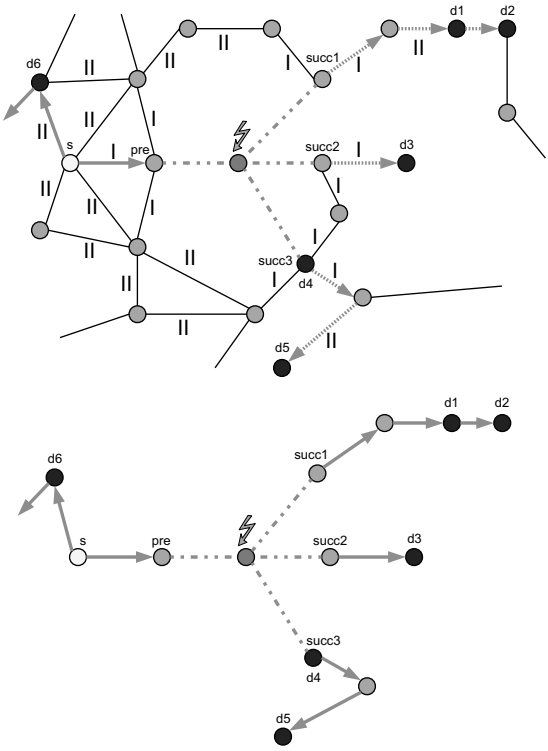


Figure 2: Heuristics example: standard graph and multicast-path

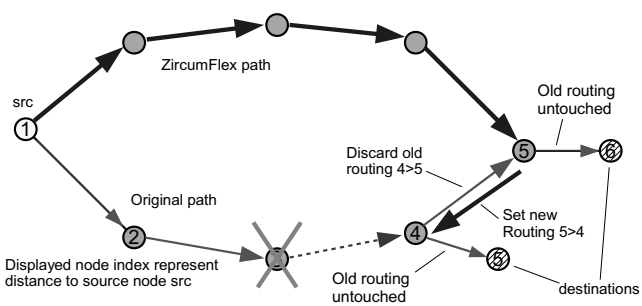


Figure 3: Online Failure Graph

using local graph breadth first searches (BFS) running only on the predecessor node of the failed node/link. As long as there are unreached successors do the following:

- do a simple BFS-run (only direct neighbors) and mark the linked list of the reached

nodes with the path to the origin, i.e. the first detected node gets the predecessor or successor node name in its linked list, respectively. In the next run there are two entries in the list and so on.

- if a previously predecessor-BFS-marked node is reached, the path from the predecessor to the affected successor is taken into an emerging *regional graph*[3]. Then the successor node is marked as reached. Running BFSs of known successor nodes are terminated, their corresponding node names are taken out of the node list. Already visited nodes/edges are then excluded from further BFS-runs.
- if another successor-BFS-marked node is reached the path between both successors is taken into the regional graph again. Already visited nodes/edges are again excluded from further BFS-runs.

This procedure puts up a local subgraph of the network graph (regional graph) and delivers a spanning tree containing shortest paths between successors and predecessor within the regional graph in a few microseconds, therefore we speak of real-time property. Our preliminary tests indicate that only a few BFS steps are necessary in general graphs.

ZirkumFlex allows to eliminate links of pathological (after a failure no longer signal-supported) nodes by iteratively starting at predecessor and successor nodes and eliminating the connected edges as long as the outdeg of the referring nodes equals one[4] or until a multicast-destination is reached. A following DFS-run within the regional graph starting at the pre node sets the correct routing direction in the regional graph.

In some network layouts there appears the effect of the signal path having to be routed forward and backward for a minor number of hops, which consumes unnecessary bandwidth to support nodes which lie closer to the failed node than the bypass. Figure 3 shows the details. This problem can easily be solved by utilizing the already mentioned source-distance information gathered during the MulMic-DFS-run. Routing from a higher node number to a lower one means that the old routing has to be terminated and a reverse link direction has to be applied for this graph edge when applying the new routing. The major advantage for this model is that the successor node and therefore the destination nodes get the original signal as quickly as technically possible and the not directly affected nodes are not disconnected for rerouting purposes.

5 Example Graphs

As an example of the proposed algorithm, Fig. 5 shows a relevant part of a large network graph (hundreds of nodes) and the resulting regional graph with predecessor, failed node and some multicast destinations. In this case (grid-topology) one BFS step (12 links affected) only is necessary to get the whole regional graph. This rerouting can be done in a few microseconds.

[3] A simple concatenation of the reached node's path lists (linked list) and the node-BFS's path lists in inverted visit order delivers the path.

[4] as there is no forking in the network which could be necessary to support another destination node

```
// pre: predecessor node, n: number of successor nodes
// succ(i): i th successor nodes, i=1 to n
// LinkedList PL for the path list information in every node
// nodelist for remaining successor nodes

procedure LBFS (Graph G, Node nod) //LBFS = Labeling BFS

 do a BFS starting at nod; // do only one level of stepping down
 For all nodes discovered during BFS do
   if node has been reached by an already pre-node-derived search
     then build part of MPath by concatenating the meeting PLs.
     // delivers a shortest path from pre to current succ
     Lookup nodes in rendezvous list, terminate their BFS and
     delete them from nodelist.
     Ignore already marked links. If there is no more
     unmarked link available, stop the affected BFS-run.

   elseif reached node is already marked by another succ node then
     build part of MPath by concatenating the meeting PLs.
     Set rendezvous information (local succ-name) in each other node.
     Ignore already marked links. If there is no more unmarked link
     available, stop the affected BFS-run.

   else
       add the already reached nodename(s) to the linked list PL of
       the reached node to setup the routing information.
 od
end LBFS

algorithm online_rerouting(Graph G, Node failed_node, Node pre,
      Node[] succ(i), Integer n) // main algorithm

 remove failed_node from G;
 while nodelist != empty  do
   LBFS(G, pre);  //iterate one step of BFS started from the pre node
   while there are successors j do
        LBFS(G, succ(j));
   od
 od

 // remove pathological edges
 For all succ's and pre do
    remove nodes in direction of old multicast path (in opposite
    direction for pre node) until following node has outdeg != 1
    or until the following node is a destination node.

 Start DFS over MPath to eliminate loops, correct routing directions
 //correct loops
    If the routing for a link would be issued against a falling
    number pair (di, di+1) (i.e. old routing direction was in opp.
dir.)
    then cancel old routing, establish new connection in new routing
dir.

 Calculate routing commands with information gathered in the
linkedLists
 consider that no command necessary for an already switched link
end online_rerouting
```

Figure 4: Pseudo code for ZirkumFlex algorithm

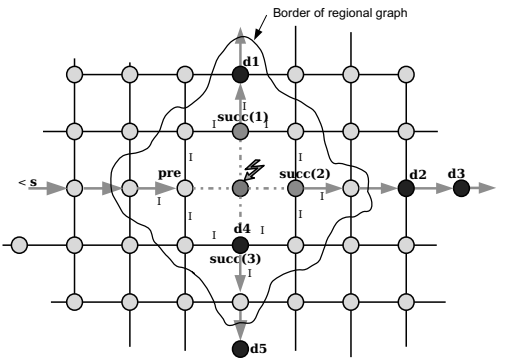


Figure 5: Heuristics example, part of grid graph

The proposed algorithm also handles several simultaneous node or line failures in one pass. In Fig. 6 a grid graph with two simultaneous failures, e.g. a failure of a shared power-supply, is shown. Along the multicast tree path starting at the source node marked "src" there is only one predecessor which is marked with "pre". The possible successor nodes are marked with $d_1, ..., d_3$. Signalling the defective nodes to the predecessor node for instance could be done by a broadcast initiated by the node that detects a loss of light and has been chosen for a multicast route.

In this example, there are three successor nodes visible. It is easy to see that only two BFS steps are needed to connect all of the successor nodes. After the first run which touches 12 nodes there are another runs until all of the common nodes are found. In this example the resulting backup graph is optimal. It follows the shortest path around the failure nodes. The rest of the potential multicast destination paths are untouched which is vital to the overall system performance.

6 Conclusion

We have investigated fault-tolerance measures in switched networks for video multicast. The backup path model combining the MulMic algorithm and choosing disjoint paths gives a very robust solution for multicast routing in multimedia applications. In all examples investigated, node failures did not lead to any noticeable error. Disadvantages of this approach are the immense use of resources as well as a communication overhead. Any routing needs to set up nearly twice the needed nodes and lines to be used. Also if there are two failures in neighboring nodes that are used for disjoint paths there will be a total failure of the signal. A more efficient way is the use of the RSVP-protocol to reserve a disjoint path. The reserved paths can be used as a backup of other nodes as well. If a failure occurs, then all other affected reserved paths have to be recalculated. This leads to a considerable

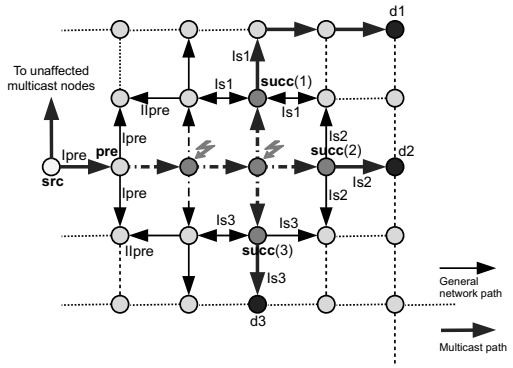


Figure 6: Proposed heuristic handling two simultaneous failures

amount of network communication. Simultaneous failures of two neighboring nodes used for disjoint paths lead to a sudden interruption of the routed multimedia signal as in the backup path model. After calculating and switching another routing the signal can be restored again.

Our proposed *ZirkumFlex* algorithm overcomes this problem. There is no redundant path and no communication overhead. After a failure of one or more neighboring nodes a recalculation of the multicast path is done automatically and in a very short time. Because of using only a so-called regional graph of limited size this algorithm also works very well for large network graphs.

References

[Ch93a] C.-H. E. Chow et al.: A Fast Distributed Network Restoration Algorithm. Proceedings 12th International Phoenix Conference on Computers and Communications, pp. 261–267, 1993

[Ch93b] C.-H. E. Chow et al.: RREACT: A Distributed Protocol for Rapid Restoration of Active Communication Trunks. Proceedings 2nd Network Management and Control Workshop, pp. 391–406, 1993

[Dr09] H. Drid et al.: A Topology Aggregation Model for Survivability in Multi-Domain Optical Networks Using p-Cycles. NPC '09: Proceedings of the 2009 Sixth IFIP International Conference on Network and Parallel Computing, pp. 211–218, 2009

[Gr89] W.D. Grover: SELFHEALING NETWORKS: A Distributed Algorithm for k-shortest link-disjoint paths in a multi-graph with applications in real time network restoration. Doctoral Dissertation, Department of Electrical Engineering, University of Alberta, Fall 1989

[Gu03] P.K. Gummadi et al.: An efficient primary-segmented backup scheme for dependable real-time communication in multihop networks. IEEE/ACM Trans. Netw., vol. 11, no.1, pp. 81–94, Feb. 2003

[Ko90] H. Komine et al.: A distributed restoration algorithm for multiple-link and node failures of transport networks. Proceedings GlobalCom '90, 403.4.1403.4.5, 1990

[MK09] R. Messmer and J. Keller: Real-Time Computation of Point-To-Multipoint Routes in Optical Networks for Digital Television. Proceedings Int. Conference on Parallel and Distributed Computing and Systems 2009, Nov. 2009

[RC08] S. Ramasubramanian and A. Chandak: Dual-link failure resiliency through backup link mutual exclusion. IEEE/ACM Trans. Netw., vol. 16, no. 1, pp. 157–169, 2008

[RM03] G. Ranjith and C. Siva Ram Murthy: A Multiple Segmented Backups Scheme for Dependable Real-Time Communication in Multihop Networks. Proceedings International Parallel and Distributed Processing Symposium, pp. 121b, 2003

[RS06] M. Karsten: Collected experience from implementing RSVP. IEEE/ACM Trans. Netw., vol. 14, no. 4, pp. 767–778, 2006

[Ru05] C. Ruan et al.: p-cycle design in survivable WDM networks with shared risk link groups (SRLGs). Proceedings 5th International Workshop on Design of Reliable Communication Networks, 2005

[UZ06] H. Ural and K. Zhu: Distributed delay constrained multicast routing algorithm with efficient fault recovery. Networks vol. 47, no. 1 , pp. 37–51, 2006

[YH88] C. H. Yang and S. Hasegawa: FITNESS: Failure Immunization Technology for Network Service Survivability. Proceedings GlobalCom '88, 47.3.1-47.3.6, 1988

[ZS92] Q. Zheng and K.G. Shin: Fault-tolerant real-time communication in distributed computing systems. Proceedings IEEE Fault-Tolerant Computing Symposium, pp. 86–93, 1992

State Transfer for Hypervisor-Based Proactive Recovery of Heterogeneous Replicated Services *

Tobias Distler†, Rüdiger Kapitza†, Hans P. Reiser◇
†University of Erlangen-Nürnberg, Germany, ◇University of Lisboa, Portugal
{distler,rrkapitz}@cs.fau.de, hans@di.fc.ul.pt

Abstract: Intrusion-tolerant replication enables the construction of systems that tolerate a finite number of malicious faults. An arbitrary number of faults can be tolerated during system lifetime if faults are eliminated periodically by proactive recovery. The periodic rejuvenation of stateful replicas requires the transfer and validation of the replica state. This paper presents two novel efficient state transfer protocols for a hypervisor-based replication architecture that supports proactive recovery. Our approach handles heterogeneous replicas, and allows changing/updating the replica implementation on each recovery. We harness virtualization for an efficient state transfer between "old" and "new" replicas in virtual machines on the same physical host, and use copy-on-write disk snapshots for low-intrusive recovery of replicas in parallel with service execution. We apply the generic algorithm to a realistic three-tier application (RUBiS) and study the impact of recovery and state transfer on system performance.

1 Introduction

Dependability is an increasingly important requirement for distributed systems. In the extreme case, a system should operate correctly even if some parts of it are compromised by a malicious attacker. Byzantine fault-tolerant (BFT) replication [CL02] is a practical approach to build such systems. Typical BFT systems need $n \geq 3f + 1$ replicas in order to tolerate up to f malicious faults. Replica diversity is necessary to avoid common-mode faults in multiple replicas (achieved either by expensive N-version programming [AC77], or by opportunistic components-off-the-shelf diversity [CRL03, GP07] and system-level diversification such as address-space randomization [SBO08]).

BFT systems also need a mechanism for recovering (refreshing) faulty replicas. Otherwise, given enough time, attackers could eventually succeed in compromising more than f replicas. Proactive recovery [CL02] is a technique for refreshing replicas periodically. It enables systems to tolerate an unlimited number of faults during system lifetime as long as the number of faults remains limited within the recovery period. Recovery operations can reduce system availability, and compensating this impact by increasing the number of replicas [SNVS06] is expensive. In practical systems, a better option is to instead minimize the impact of periodic recovery actions on system operation.

*This work was partially supported by the German Research Council (DFG) under grant no. KA 3171/1.

The VM-FIT project [RK07] has shown that virtualization technology can be harnessed efficiently for building an intrusion-tolerant replication architecture with proactive recovery. The architecture uses a *hybrid failure model*, supporting BFT at the service-replica level, whereas the replication infrastructure is executed in a *trusted computing base* that fails only by crashing. Thus, VM-FIT can be seen as a intermediate step between simple fail-stop systems and pure BFT replication. It requires only $2f + 1$ replicas, which reduces resource requirements and makes it easier to use heterogeneous components-off-the-shelf software. Another benefit of VM-FIT is that the system support for proactive recovery has a negligible impact on system availability, without adding more replicas. However, VM-FIT misses support for state transfer of large application state, which is essential for proactively recovering real-world applications, such as a multi-tier web application.

In this paper, we define two efficient state-transfer strategies and integrate them into VM-FIT. The first approach is based on snapshots and basic conversion routines to transform the service state from/to an abstract format. The second, a log-based variant, optimizes the transfer of large application states and requires minor changes to a service. Both protocols result in a proactive recovery scheme that permits the simultaneous recovery of all replicas with low impact on service performance. We evaluate and analyze our state transfer protocols using a micro benchmark and the three-tier benchmark system RUBiS. Thereby, we apply opportunistic N-version programming using different operating systems and different application servers. Our results show that our protocols are able to recover stateful services with a downtime of a few hundred milliseconds.

The next section provides an overview of the VM-FIT architecture, and Section 3 defines the state transfer protocols. Section 4 experimentally evaluates the current prototype with a micro benchmark and RUBiS. Section 5 discusses related work, and Section 6 concludes.

2 The VM-FIT Infrastructure

VM-FIT [RK07] implements a generic architecture for intrusion-tolerant replication of network-based services using isolated virtual machines on top of a virtual machine monitor. It supports efficient proactive recovery by booting a new replica instance (*shadow replica*) in an additional virtual machine before shutting down the old replica (*senior replica*). Most other systems that support proactive recovery require rebooting the entire physical host from a secure read-only image [CRL03, ASH07], which leads to significantly longer replica unavailability during reboot. Client/service interaction in VM-FIT is exclusively performed via request/reply network messages that can be intercepted at the network level. The remote service is deterministic and replicated using standard state-machine replication techniques.

In VM-FIT, a replica manager runs in a *privileged domain*, whereas service replicas execute in completely separated *application domains* (see Figure 1). The replica manager communicates with clients, delivers requests to replicas in total order using a group communication system, and votes on replies; it also contains the logic for proactive recovery. The virtual machine monitor and the privileged domain with the replica manager form the trusted computing base of VM-FIT that can only fail by crashing. Service replicas, which include a separate operating system, a middleware infrastructure, and a service implementation, are placed in isolated, untrusted application domains. They may fail in arbitrary (Byzan-

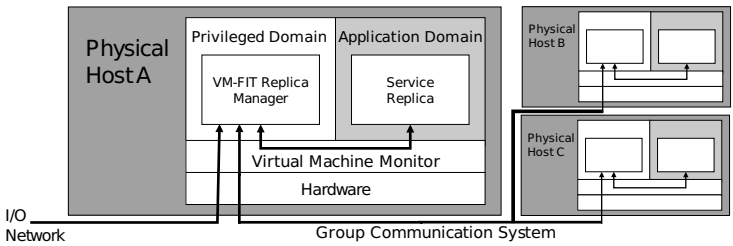


Figure 1: VM-FIT basic replication architecture

tine) ways. This *hybrid fault model* allows us to build services that tolerate any kind of failure in application domains, including malicious faults. This is a significant improvement over applications that do not tolerate intrusions at all. Furthermore, recent advances in software security (e. g., the verification of the virtual machine monitor Hyper-V [CDH+09]) show that it is becoming feasible to verify the correctness of a trusted computing base with the complexity of what we use in VM-FIT.

The basic **request processing workflow** of VM-FIT consists of the following steps: A client sends a service request, which is then received by one of the replica managers. Next, the manager transmits the request to all replica managers using the group communication system. Upon receiving the request, each replica manager passes it to the local application domain, which processes the request and returns a reply. The replica manager that is connected to the client collects the replies from the other replica managers, votes on the replies, and after having obtained $f + 1$ identical values, returns a result to the client. Note that we need only $2f + 1$ replicas to tolerate f malicious faults in the replica domains, as the group communication is part of the trusted computing base of the system. Assuming an asynchronous, reliable network, the voter eventually receives replies from all correct replicas, so it always obtains at least $f + 1$ identical replies, even if f replicas are malicious.

The replica manager provides a system component that controls **proactive recovery** in a reliable and timely way. The basic recovery strategy uses the following algorithm:

1. *Upon trigger of periodic recovery timer:*
   → Start new virtual machine *(shadow replica)* from secure image
   → After startup completion, broadcast `recovery-ready` to all replicas
2. *Upon reception of* k*th* `recovery-ready` *(*k *is the number of non-crashed nodes):*
   → Suspend request processing
   → Create local snapshot of *senior replica*
   → Resume request processing on senior replica
   → Transfer application state to new replica domain
3. *Upon completion of the state transfer:*
   → Process (buffered) client requests in shadow replica
   → Shut down senior replica

Note that as `recovery-ready` messages are totally ordered relative to client requests, all k snapshots represent the same logical point in service history. As soon as the replica manager resumes request processing, all requests are also forwarded to the shadow replica, which processes them as soon as the state transfer is completed.

The outlined algorithm delays execution of requests during snapshot generation and state transfer. For a small application state, we have shown this time to be negligible compared to replica startup [RK07]. For a large application state, however, transfer can take a significant amount of time. In this paper, we propose two optimized state transfer algorithms that tackle this problem and substitute the current simple approach.

3 State Transfer

The recovery of a stateful application's replica requires that the new replica instance receives the current service state. Assuming non-benign faults, the state transfer protocol must ensure transferring a non-corrupted state. Instead of copying the state of a single replica, VM-FIT transfers the state of multiple replicas in parallel and ensures the correctness by voting. Software diversity usually results in different internal state representations in each replica. Nevertheless, all correct replicas will maintain a consistent state from a logical point of view. This means that for voting all replicas need to convert their specific application state into a uniform external representation (*abstract state*), and vice versa. Such an approach requires explicit support for state transformation in the replica implementations.

A straight-forward way for transferring state across virtual machines is a black-box approach that considers the whole memory and disk image of a virtual machine as state. However, this view includes a lot of data that is irrelevant for the logical state of a replicated service, such as internal operating system variables. VM-FIT instead distinguishes between *system state* and *application state*, and transfers only the application state. The system state includes all state of operating system and middleware that requires no consistent replication; we assume that a correct system state can be restored from a secure code storage. In contrast, the application state, composed of *volatile state* (data stored in memory) and *persistent state* (data stored on disk), pertains to the replica implementation.

The efficiency of the state transfer highly influences service availability and performance during recovery. In the following, we present a *snapshot-based* state transfer protocol that exploits the locality provided by virtualization; a preliminary version of this protocol was published in [DKR08]. As most of the state data is transferred directly between the old and the new domain running on the same machine, it is suited for scenarios where large states have to be transmitted. For some applications, state conversion to and/or from the abstract state is costly. In these cases, a benefit can be obtained by differentiation techniques that reduce the amount of transferred data. Unfortunately, direct propagation of state updates in implementation-specific formats such as memory pages is impossible for heterogeneous replicas. To resolve this problem, we integrated an efficient *log-based* state transfer protocol into VM-FIT that copes with diverse replica implementations and yet only transfers state changes made since the previous recovery.

3.1 Snapshot-Based State Transfer

Virtualization can be harnessed for transferring state directly from senior replica to shadow replica on the same physical host. Remote replicas can validate the local state by calculating and transferring checksums of their local abstract state via the network. Unlike traditional reboot approaches, in which only a subset of the replicas can be restarted at a time in order

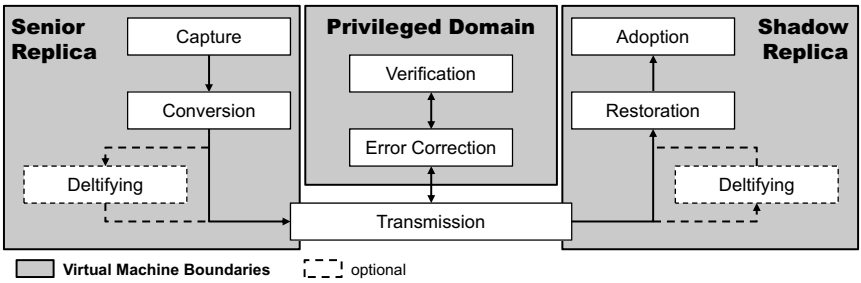


Figure 2: Conceptual phases of a state transfer

to avoid complete unavailability [SNVS06], our virtualization-based approach allows the simultaneous recovery of all replicas. The recovery process in VM-FIT starts with the creation of the shadow domain by the recovery unit (see Section 2). On each machine, the replica manager initiates and controls the subsequent snapshot-based state transfer between senior replica and shadow replica, using the following conceptual phases (see Figure 2):

Capture Phase (Capture, Conversion, optional Deltifying) Prior to the state transfer, senior replicas have to generate a consistent checkpoint. As soon as the replica manager receives a `recovery-ready` message, it suspends the distribution of client requests. After all pending requests have finished, it instructs the replica to write all replication-relevant volatile state to disk. Next, the replica manager creates a snapshot of the disk volume containing the replica application data. Thereby, volatile and persistent state of the application are both captured. After that, the replica manager attaches the snapshot to the senior domain, which now has a read-only view as well as a copy-on-write view of the same logical disk. The read-only view contains the snapshot and is used for state conversion and transfer. The copy-on-write view, which is the still attached but reconfigured replica disk, on the other hand, is used as a basis for further service execution. Thus, the senior replica can resume request processing and change the state of the copy-on-write view without affecting the snapshot. As soon as the snapshot exists, the replica manager resumes forwarding client requests. When the senior replica restarts processing, the replica manager logs these requests in order to re-execute them at the shadow replica. This way, the actual period of replica unavailability can be reduced to the process of taking the snapshot. Optionally, the senior replica can build a delta between the abstract state used for its initialization (as a result of the previous recovery cycle) and the current abstract state. Transferring the delta instead of the full abstract state reduces the data transmission time.

State Transmission and Verification (Transmission, Verification, Error Correction) The senior replica converts the application-specific state that has been captured on the snapshot volume to the abstract format. The abstract state is transferred directly to the shadow replica as a stream of bytes. A replica wrapper (a VM-FIT component fulfilling only this task during state transfer) in the shadow domain calculates block-wise checksums over the received stream data. It then transmits them to the local replica manager, which forwards them to all other replica managers. As soon as f remote checksums that match

the local checksum have been received (i. e., there are $f + 1$ identical values), the replica manager signals the shadow replica that it can safely convert the block from the abstract format to the application-specific representation. The reception of $f + 1$ identical remote values not matching the local value indicates a state corruption. In this case, the correct data block is requested from a remote replica manager that supplied a correct checksum.

Finishing State Transfer (optional Deltifying, Restoration, Adoption) Finally, the shadow replica converts the verified abstract state to the local application-specific representation. If deltifying has been used, the complete abstract state has to be obtained by applying the received change set on the abstract state of the previous recovery provided by the replica manager. Afterwards, the replica manager takes a snapshot that builds the basis of the following recovery. To become up to date, the replica then re-executes all post-snapshot requests, discarding the replies, and takes over the role of the senior replica. After that, the replica manager shuts down the previous senior replica.

3.2 Log-based State Transfer

The snapshot-based state transfer process is generic. It does not require any modifications to the replicated application itself, the only replica-specific requirements are conversion routines for the state. However, state conversion and restoration can take a significant amount of time, thereby delaying replica handover. Thus, we propose another state transfer variant that performs conversion, adoption, and deltifying during normal operation mode and omits state capturing operations. It requires minor changes to the application code.

The core idea is to constantly log all state changes in the abstract state format and apply them during recovery to a pre-initialized replica. This approach requires the identification of certain locations in an application where standardized write operations are performed. For web-based multi-tier applications, an appropriate location is the transition between application layer and data layer, as most of the processing has already been performed. At this point, results are usually written to database, typically using SQL.

We do not directly log the SQL operations of the application. These operations might be crafted maliciously in order to exploit a vulnerability of one of the database replicas. Instead, we retrieve the effective write set of an operation from the database and transform it into a state update operation and append this operation to a log. The log is verified in the same way as the snapshot is verified in the snapshot-based approach. If a malicious request corrupts the state of a database replica, this problem is detected during the verification phase and subsequently corrected by transferring a validated remote log. Additionally, if the same data is changed multiple times, we use online log reduction; that is, during state transfer we only consider the latest update.

In the shadow replica, the log is re-executed on the database that has been initialized using the verified snapshot of the previous recovery, leading to a verified up-to-date replica state. After the replay, but prior to processing further requests, the replica manager takes a file-system snapshot of the current state. This snapshot forms the secure basis to initialize the shadow replica of the next recovery round. This way, we avoid costly conversion operations without changing the processes of state transmission and finishing state transfer.

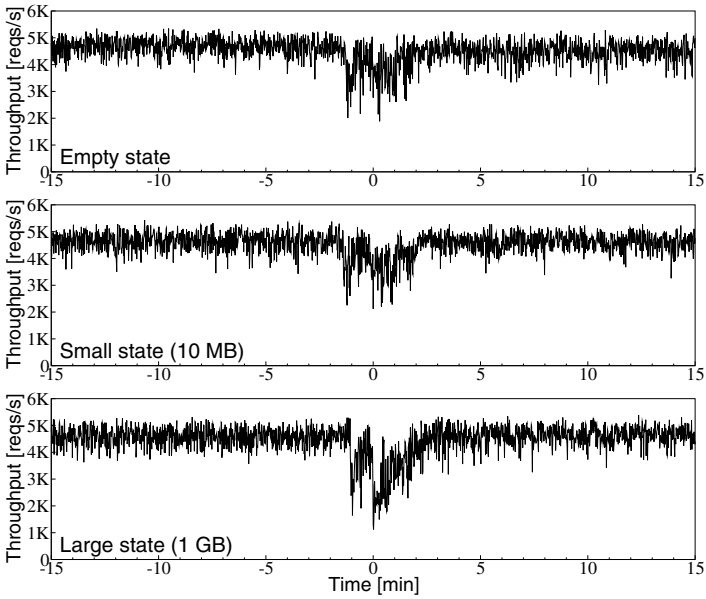


Figure 3: Micro benchmark with snapshot-based proactive recovery for different state sizes.

4 Evaluation

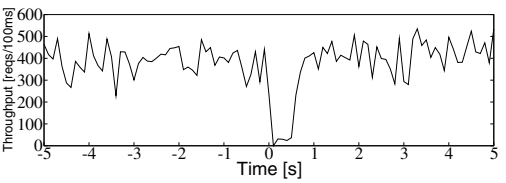
This section demonstrates that proactive recovery in VM-FIT using our state-transfer algorithms causes only negligible downtime for stateful services. We analyze the snapshot-based state transfer with a micro benchmark and the log-based approach with a complex three-tier application. The evaluation uses a cluster of four hosts (Intel Core 2 CPU, 2.4 GHz, 2 GB RAM, Xen 3.1 hypervisor), connected with 1 Gb/s Ethernet. One host simulates clients, the other three hosts execute VM-FIT with identical configurations in the privileged domain and different operating systems in the application domain (Debian Linux, NetBSD, and OpenSolaris). Snapshots of replica application states are created using LVM [LVM10], a logical volume manager for Linux.

4.1 Micro Benchmark

The micro benchmark used to evaluate the snapshot-based state transfer uses 500 clients concurrently retrieving and modifying records in a Java service that implements a simple database. We run the benchmark with empty, small (10 MB) and large (1 GB) application states. Each run takes 30 minutes and includes a recovery after 15 minutes. Furthermore, there is one comparison test run without recovery. Figures 3 and 4 present the results of the micro benchmark. All graphs are synchronized at snapshot creation time ($t = 0$). The measurements show that there are two main factors degrading performance during the recovery process: First, setting up the shadow domains puts additional load on each of the physical hosts. The complete startup usually takes about one minute and results in a 10-20% throughput decrease. This number is independent of state size.

application state size	*state transfer throughput*	*overall throughput*
empty	3459 req/s	4555 req/s
10 MB	3828 req/s	4547 req/s
1 GB	3078 req/s	4431 req/s
no recovery	-	4620 req/s

(a) Average realized throughput of a proactive-recovery micro benchmark for variable state sizes.



(b) Detailed characteristics of a proactive-recovery micro benchmark for a state size of 10 MB during snapshot creation.

Figure 4: Characteristics of a snapshot-based proactive-recovery micro benchmark

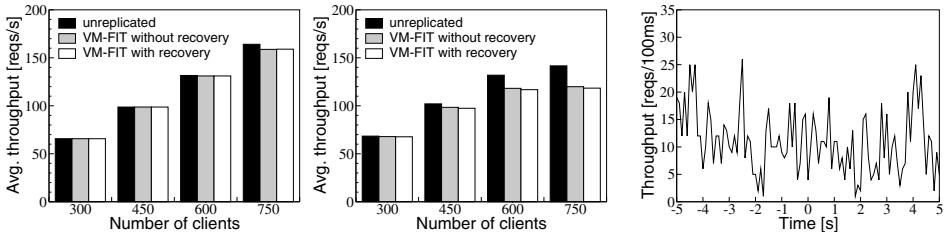
The second factor influencing throughput during recovery is state transfer. In contrast to shadow domain startup, this recovery step is dominated by the characteristics of an application. The results of the benchmark confirm that moving a larger state not only takes more time, but also impacts performance: Transferring a 10 MB state, for example, decreases throughput for 120 seconds by an average of 17%, whereas recovery with 1 GB of state leads to a temporary performance drop of 33% during 169 seconds. There are two reasons for the increase in throughput degradation: More data has to be copied and checked, and an extended transmission and verification phase leads to more requests being buffered, which then have to be re-executed afterwards. As state transfer in the empty-state scenario is limited to the short period of costly snapshot creation, a rather low throughput of 3459 requests per second can be observed during that period. Combining shadow replica startup and state transfer, the results show that proactive recovery in the micro benchmark is very cheap: For the small application state scenario, the average throughput during the test is 4547 req/s which is only 2% below the result of the no-recovery run (4620 req/s). As discussed above, transferring a bigger state further decreases performance. However, the costs for the recovery of a large state only add up to throughput degradation of 4%.

The results of the micro benchmark show no observable service downtime during any stage of the recovery process. Additional measurements conducted with higher sampling rates outline the actual service disruption. Figure 4b presents the throughput development around the snapshot of a 10 MB state. For an interval of 600 milliseconds the throughput drops significantly, with a maximum continuous service downtime of 130 milliseconds.

4.2 Rice University Bidding System (RUBiS)

The Rice University Bidding System (RUBiS) is a web-based auction system built to benchmark middleware infrastructures under realistic load situations [RUB10]. We replicated RUBiS in its Java servlet variant using VM-FIT. In the snapshot-based state transfer, retrieving/restoring the state from/to a database (via plain SQL interface) during the conversion/restauration phase would take several minutes. Therefore, RUBiS is an excellent candidate for the log-based state transfer, as logging state changes during normal operation circumvents the need to retrieve a complete abstract state.

We implemented our approach by intercepting calls at the JDBC layer that represents the interface between application layer and data layer. Each modifying operation is followed by



(a) Realized throughput for the plain service and VM-FIT with and without proactive recovery for database sizes of 100 MB (left) and 1 GB (right).

(b) Detailed characteristics of the state-capture phase (1 GB database).

Figure 5: Characteristics of a log-based proactive-recovery RUBiS benchmark

a read of the affected table entry whose result is then logged in plain SQL[1]. The generated log contains the (differential) abstract state to be distributed and verified during state transfer. Shadow replicas are initialized using a verified snapshot of the previous recovery. Then, they re-execute the operations from the verified log on the database. Before starting to execute client requests, the replica manager takes a file-system snapshot of the database, which serves as the verified snapshot used in the next recovery round.

Besides using different operating systems, additional heterogeneity is achieved by different web/application servers (Jetty 6.1.12rc1 on the Solaris and BSD replicas, Apache Tomcat 5.5.27 in the Debian guest domain). All clients are simulated by the RUBiS client emulator processing the default transition table (which approximates the behaviour of human clients browsing an auction website). Each test run starts with an identical initial state taken from a database dump provided on the RUBiS web site [RUB10]. It includes data sets of about one million users and more than five million bids. The total size of the application state, based on the original SQL dump, exceeds one gigabyte. In addition, we conduct a series of tests with a downscaled database of about 100 MB. Our evaluation compares the unreplicated RUBiS benchmark to VM-FIT with and without proactive recovery, varying the number of clients from 300 to 750. Clients perform a 30 minute runtime session. Proactive recovery is triggered after 15 minutes.

Figures 5a shows the overall throughput results for the RUBiS benchmark. For the small database, the average throughput of VM-FIT without recovery is within 3.2% of the throughput achieved by the unreplicated benchmark, with a maximum of 158.8 requests per second at 750 clients. The results for the tests where VM-FIT proactively recovers using the log-based state transfer are basically equal (less than 0.1% deviation). Using the large database, 750 clients saturate the RUBiS service on our machines due to database operations being more costly, leaving VM-FIT without recovery with an average throughput of 119.9 requests per second. Here, VM-FIT with recovery has a throughput of 118.4 requests per second, a decrease of 1.3%. These numbers show that the log-based proactive recovery has only little effect on the overall service performance.

[1]More efficient approaches for retrieving the write set from a database could also be used [SJPPnMK06], but this simple approach was suitable in the context of our evaluation.

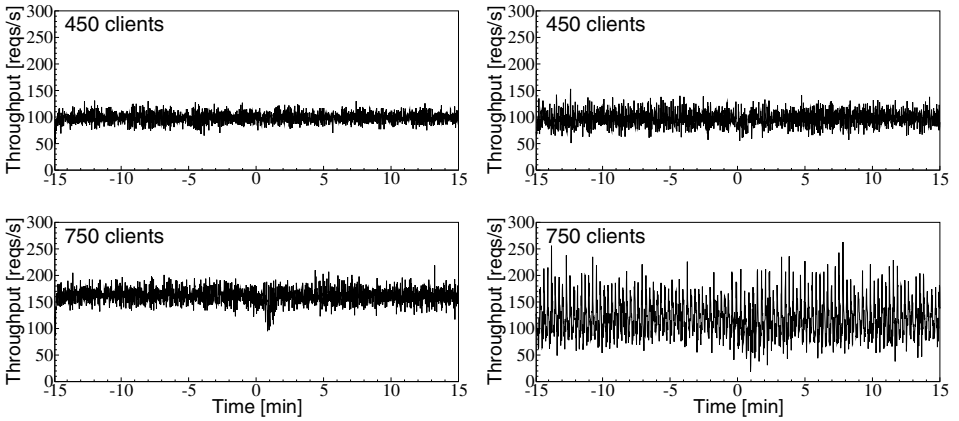


Figure 6: RUBiS benchmark with recovery for database sizes of 100 MB (left) and 1 GB (right).

Figure 6 (we omit the results for 300 and 600 clients due to limited space) gives a more detailed view of the impact of the log-based state transfer. A major difference in comparison to the micro benchmark results is that shadow domain startup has almost no influence on throughput. Note that this circumstance is due to the different application characteristics of the RUBiS service; it is not related to the log-based state transfer. However, applying the log-based approach has a great influence on other recovery phases: Instead of transmitting and verifying the whole application state of 1 GB (in case of the small database: 100 MB), only modified database entries are transferred. For 750 clients, they add up to 2.0 MB (2.1 MB); about 0.2% (2.1%) of state size. As a consequence, state transfer is done in 95 seconds (50 seconds), temporarily decreasing throughput by 11.2% (10.7%).

The most important benefit of the log-based approach is its implementation of the state capture phase. Instead of expensively creating a snapshot of the application state, the log that records state changes is just truncated. Figure 5b shows that this procedure is executed without disrupting the service.

5 Related Work

System recovery is a popular strategy to cope with bugs and intrusions. Recovery-Oriented Computing (ROC) [CBFP04] and Microreboots [CKF+04] represent approaches to cure potential faults by rebooting fine-grained components. Both focus on counteracting hard-to-detect faults that over time cause service degradation, such as memory leaks. Self-Cleansing Intrusion Tolerance [ASH07] targets cleaning intrusions by rebooting services executed in a virtual machine. To prevent service disruption, a backup instance immediately takes over once the primary service instance is rebooted. Similar to this work, the Rx [QTZS07] system goes one step further by modifying the environment after a software failure. This enables Rx to handle certain kinds of deterministic bugs that otherwise would lead to repetitive reboots because of recurring faults. None of these systems is able to handle state corruptions.

The work by Sousa et al. [SNVS06] uses a trusted computing base on the basis of a virtual machine monitor and a trusted virtual machine for proactive and reactive recovery in the context of BFT replication. The trusted computing base only provides functionality for recovery; there is no support for state transfer at the virtualization layer. Our approach is similar to this architecture in the sense that it uses the virtual machine monitor to implement a trusted computing base, but we focus on stateful recovery.

The problem of state transfer has been addressed by the BFT protocol of Castro and Liskov [CL02]. They recognize that efficiency of state transfer is essential proactive recovery systems and propose a solution that creates a hierarchical partition of the state in order to minimize the amount of data to transfer. BASE [CRL03] proposes abstractions that capture parts of the state in an abstract format for use during state transfer in a heterogeneous BFT system. In addition, BASE provides wrapper functions for handling non-determinism that could also be added to our architecture. Our system also uses the concept of an abstract state, but we define two different approaches: first, a generic snapshot-based state transfer, and second, a log-based state transfer that requires slight changes to the replicated application, but reduces overhead. The approach of creating a differential state used in BASE [CRL03] is not suitable for RUBiS, as some requests involve multiple related database operations. The need to reproduce them outside the application would lead to a duplication of great parts of the database layer. Additionally, we exploit virtualization for minimizing state transfer time. Sousa et al. [SNVS06] define requirements on the number of replicas that avoid potential periods of unavailability given maximum numbers of simultaneously faulty and recovering replicas. Our approach instead considers stateful services and reduces unavailability during recovery by performing most of the recovery in parallel to normal system operation on the basis of virtualization technology.

This work expands on preliminary results proposing efficient state transfer for the proactive recovery of stateful services published at the WRAITS workshop [DKR08] by proposing log-based state transfer, a more detailed design as well as extended evaluation results.

6 Conclusion

We have presented two protocols for efficient state transfer between heterogeneous replicas in an intrusion-tolerant replication system with proactive recovery. Snapshot-based state transfer only requires the provision of conversion routines from/to an abstract state format. Creating a new replica instance using virtualization technologies and utilizing copy-on-write techniques for fast generation of atomic state checkpoints enables to continue service provision while the state is transferred. Additionally, we proposed log-based state transfer as a further extension for scenarios where enough knowledge of the service is available to employ a differential state transfer. The results of the micro benchmark and the RUBiS use-case illustrate the efficiency of proactive recovery in VM-FIT. The evaluation shows that there is only a negligible service disruption during recovery. This allows building BFT systems which continuously run stateful applications without needing additional replicas during proactive recovery. Although state size is not a crucial factor regarding service availability, it nevertheless influences performance. Therefore, the abstract state transferred

should be limited to information that is required to initialize the application in the shadow replica. As shown with the RUBiS evaluation, log-based state transfer is an excellent way to approach this task as it reduces the transferred abstract state to the essential updates. This way, we can by simultaneously recovering all replicas of a complex three-tier application with only 2-6% decrease in the overall throughput.

References

[AC77] A. Avižienis and L. Chen. On the Implementation of N-Version Programming for Software Fault Tolerance During Execution. In *Proc. of the IEEE Computer Software and Applications Conf.*, pages 149–155, 1977.

[ASH07] D. Arsenault, A. Sood, and Y. Huang. Secure, Resilient Computing Clusters: Self-Cleansing Intrusion Tolerance with Hardware Enforced Security (SCIT/HES). In *Proc. of the 2nd Int. Conf. on Availability, Reliability, and Security*, pages 343–350, 2007.

[CBFP04] G. Candea, A. B. Brown, A. Fox, and D. Patterson. Recovery-Oriented Computing: Building Multitier Dependability. *Computer*, 37(11):60–67, 2004.

[CDH+09] E. Cohen, M. Dahlweid, M. Hillebrand, D. Leinenbach, M. Moskal, T. Santen, W. Schulte, and S. Tobies. VCC: A Practical System for Verifying Concurrent C. In *Theorem Proving in Higher Order Logics, 22nd Int. Conf.*, pages 23–42, 2009.

[CKF+04] G. Candea, S. Kawamoto, Y. Fujiki, G. Friedman, and A. Fox. Microreboot – A technique for cheap recovery. In *Proc. of the 6th Symp. on Operating Systems Design and Implementation*, pages 31–44, 2004.

[CL02] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[CRL03] M. Castro, R. Rodrigues, and B. Liskov. BASE: Using abstraction to improve fault tolerance. *ACM Trans. on Computer Systems*, 21(3):236–269, 2003.

[DKR08] T. Distler, R. Kapitza, and H. P. Reiser. Efficient State Transfer for Hypervisor-Based Proactive Recovery. In *Proc. of the 2nd Workshop on Recent Advances on Intrusion-Tolerant Systems*, pages 7–12, 2008.

[GP07] I. Gashi and P. Popov. Fault tolerance via diversity for off-the-shelf products: A study with SQL database servers. *IEEE Trans. Dep. Secure Comp.*, 4(4):280–294, 2007.

[LVM10] LVM: Logical Volume Manager. http://sourceware.org/lvm2/, 2010.

[QTZS07] F. Qin, J. Tucek, Y. Zhou, and J. Sundaresan. Rx: Treating bugs as allergies—a safe method to survive software failures. *ACM Trans. on Computer Systems*, 25(3):7, 2007.

[RK07] H. P. Reiser and R. Kapitza. Hypervisor-Based Efficient Proactive Recovery. In *Proc. of the 26th IEEE Int. Symp. on Reliable Distributed Systems*, pages 83–92, 2007.

[RUB10] RUBiS: Rice University Bidding System. http://rubis.ow2.org/, 2010.

[SBO08] P. Sousa, A. N. Bessani, and R. R. Obelheiro. The FOREVER Service for Fault/Intrusion Removal. In *Proc. of the 2nd Workshop on Recent Advances on Intrusion-Tolerant Systems*, pages 13–18, 2008.

[SJPPnMK06] J. Salas, R. Jiménez-Peris, M. Patiño Martínez, and B. Kemme. Lightweight Reflection for Middleware-based Database Replication. In *Proc. of the 25th IEEE Int. Symp. on Reliable Distributed Systems*, pages 377–390, 2006.

[SNVS06] P. Sousa, N. Neves, P. Veríssimo, and W. H. Sanders. Proactive Resilience Revisited: The Delicate Balance Between Resisting Intrusions and Remaining Available. In *Proc. of the 25th IEEE Symp. on Reliable Distributed Systems*, pages 71–82, 2006.

A Parallel Computing System with Specialized Coprocessors for Cryptanalytic Algorithms

Wolfgang Kastl
wolfgang.kastl@students.fh-hagenberg.at

Thomas Loimayr
thomas.loimayr@students.fh-hagenberg.at

Abstract: In this paper we present a scalable, parallel computing system consisting of specialized processors primarily designed for the implementation of cryptanalytic algorithms. Even though the system was developed in regard to solve cryptanalytic problems, it is suitable for many other tasks which can benefit from the enormous computing power of the system (e.g. malware analysis). In addition to the use of multi-core CPUs, the computing system takes advantage of graphic cards (GPUs) and FPGAs as specialized coprocessors. Thus, it gains an edge over other conventional parallel computing systems.

1 Introduction

In the last decade, the clock frequency of traditional processors (CPUs) increased significantly. Due to physical laws, such as thermal density, the increase of clock speed recently hit a wall. Thus, the CPU vendors were forced to accelerate the computation power of CPUs by integrating multiple cores onto a single die.

In recent years, GPUs and FPGAs became very popular as coprocessors in high performance computing systems. These two types of specialized coprocessors can very often achieve much better performance than multi-core CPUs for certain types of computations. Modern GPUs provide a huge amount of massive parallel processing units and are therefore well-suited for high performance computing. Furthermore, they contain up to 4 GB onboard memory and are capable to exceed 100 GB/sec of internal memory bandwidth. In contrast to GPUs, the high performance capability of FPGAs for certain types of applications has been well-known for a long time. Since multi-core CPUs, GPUs and FPGAs are different technologies, they achieve widely different performance on certain tasks. A comparison on different applications between the three technologies is presented in [Shu08].

Nowadays, a fair amount of high performance computing systems already take advantage of coprocessors such as GPUs and FPGAs. Most of them use either GPUs or FPGAs but rarely benefit from both types of coprocessors. In this paper we introduce a scalable, parallel computing system, which takes advantage of GPUs as well as FPGAs as coprocessors. Furthermore, a lot of today's most popular parallel computing systems with focus

on cryptanalysis such as the COPACOBANA [Ruh06] or the PS3-Cluster [Lab] consist of dedicated hardware especially built to solve a small range of specific problems. In contrast to these systems, our cluster exclusively consists of standardized interfaces and components and is freely scalable and configurable. Hence, it is not limited to a small number of problems.

In the following section, an overview of works related to our cluster presented in this paper is given. In Section 3, we present the hardware architecture of the cluster. Section 4 describes the software framework which is necessary to communicate between nodes and to provide the capability of using GPUs and FPGAs as coprocessors. Section 5 explains how the communication between different cluster nodes is handled. Section 6 covers suitable implementations of cryptographic and cryptanalytic algorithms on the cluster. The last section concludes with the current status of the project and gives an outlook on future tasks and improvements of the cluster.

2 Related Work

Obviously, a highly specialized system like COPACOBANA cannot be directly compared to our cluster, as both are completely different systems. However, in recent years, various COTS cluster systems were introduced, which take advantage of coprocessors like GPUs and FPGAs. In 2009, a first prototype of the Quadro Plex (QP) cluster [Mic09] was introduced by NCSA[1]. Each node of the prototype system is equipped with two AMD Opteron CPUs, four NVIDIA G80GL GPUs and one Xilinx Virtex-4 LX100 FPGA. In 2010 the heterogeneous cluster Axel [TL10] was introduced. The first prototype uses 16 nodes, equipped with one AMD Phenom Quad-Core CPU, one NVIDIA Tesla C1060 and one Xilinx Virtex-5 LX330 FPGA hosted on an ADM-XRC-5T2 card.
While these two systems provide a strong basis for developing any new applications, the goal of our cluster is to additionally provide a library of cryptographic modules that can be loaded into the cluster's coprocessors. Hence, the system allows to freely develop new applications or to use existing modules for e.g. cryptanalysis, which makes it easy to configure and easy to use.

3 Physical Architecture

The cluster is composed of a head node that performs management tasks; input nodes that distribute the data to be processed; compute nodes that execute the cryptanalytic algorithms; and output nodes that collect the output from the compute nodes to aggregate and concentrate it. The physical architecture of the cluster is depicted in Figure 1. Each compute node as well as the head node can additionally act as input or output node or both. It is also possible to define separate nodes as input or output nodes which do not

[1] National Center for Supercomputing Applications at the University of Illinois

act as compute nodes. The task of assigning different roles (input, output or compute) to the nodes has to be done on the head node of the cluster. Thus we can see that the cluster is freely configurable on the head node and does not have a predefined and fixed structure. As a result, the structure of the cluster can be configured to best fit to the underlying cryptanalytic algorithm.

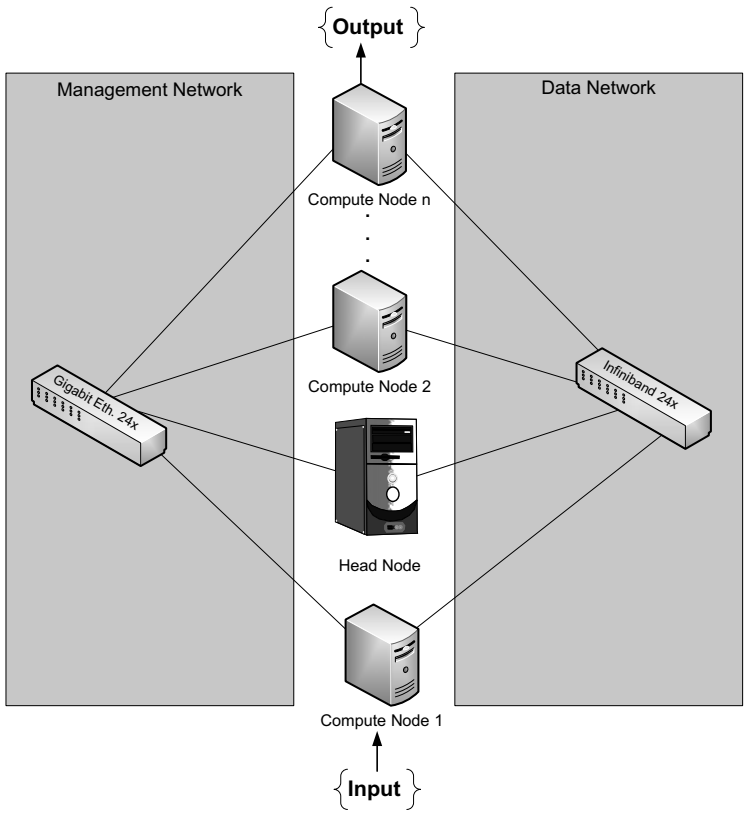


Figure 1: Physical Architecture of the Cluster

3.1 Networks

The architecture of the cluster consists of two different networks - the data network and the management network. The use of two separated networks allows performing management tasks (e.g. monitoring the cluster nodes) over the management network while the actual data throughput over the data network is not affected in any way. Each node in the cluster is consistently connected to both networks. A simplified architecture of the networks is depicted in Figure 1.

Data Network. The data network represents the high-speed Infiniband network. A 24-port Infiniband switch is used to enable a high-speed interconnection between the nodes. The data network transmits data to input nodes, distribute data to compute nodes, eventually exchange data between compute nodes and finally hand over the result to the output nodes. Depending on the type of computation an enormous amount of data might be transmitted over the data network.

Management Network. The management network represents the Gigabit Ethernet network where the nodes are connected through a 24-port Gigabit Ethernet switch. Data that is transmitted over the management network contains status messages, error codes, debug information and any other kind of information necessary for the management of the cluster. Moreover, the management network is responsible for node development and node updates.

3.2 Node Configuration

Each single compute node consists at least of the following standard hardware components:

- Motherboard with at least four PCIe slots which was a prerequisite for our cluster architecture.
- Multi-core CPU and huge main memory: Each node is equipped with a quad-core CPU and 16 GB of main memory.
- Gigabit Ethernet network card: Used to connect to the management network.
- Infiniband Host Channel Adapter (HCA): Used to connect to the data network.

Furthermore, each compute node consists of one or more of the following specialized coprocessors:

- NVIDIA GPUs: Nodes are (at the moment) equipped with NVIDIA graphic cards which come with CUDA[2] support.
- FPGA boards: For current testing a Xilinx ML605 Evaluation Board with one single Virtex-6 FPGA is used. In later versions of the cluster the evaluation board will be replaced by an FPGA board equipped with several FPGAs working in parallel.

3.3 Architecture Constraints

At the moment, the cluster uses a 24-port Infiniband switch for the data network and a 24-port Ethernet switch for the management network allowing to connect a maximum number

[2]Compute Unified Device Architecture

of 24 nodes. This means, the current architecture of the cluster is capable of operating 23 compute nodes and one head node. Since the nodes are equipped with motherboards consisting of 4 PCIe slots, each node can be equipped with a maximum number of three coprocessor boards (one PCIe slot is already used for the Infiniband HCA). Thus, the current cluster architecture is restricted to 69 (23 nodes each with 3 coprocessor boards) coprocessor boards in total.

4 Software Framework

In this section we introduce the software framework which is necessary for the computation of algorithms on the cluster. The software framework provides functionality which allow us to perform the following tasks:

- Configuring the cluster nodes including FPGAs and GPUs in a unitary way,
- assigning the data flow through the cluster and
- operating and monitoring the cluster.

4.1 Management Software

The management software which is running on the head node is responsible for managing the entire cluster. It configures the nodes for the appropriate tasks, sets the data flow within the cluster, initializes the nodes and requests status information regularly to provide a detailed status overview over the whole cluster. Furthermore, the management software is capable of collecting debugging information which is especially useful for the cluster development.

Microsoft HPC Server. We decided to use Microsoft Windows Server 2008 with the Windows High Performance Cluster (HPC) Extension [Mic08] as cluster management software. It provides utile cluster management functions and fully integrates a Microsoft release of the Message Passing Interface (MS-MPI). Due to the fact that the Microsoft HPC Server does not support GPUs and FPGAs as coprocessors, we need additional management software to fully integrate the coprocessors into the cluster environment. The additional management software is part of the MPI program and runs exclusively on the head node.

4.2 Grid Software

The grid software is running on each compute node. It is a main element of the cluster and receives commands from the management software of the head node to configure the

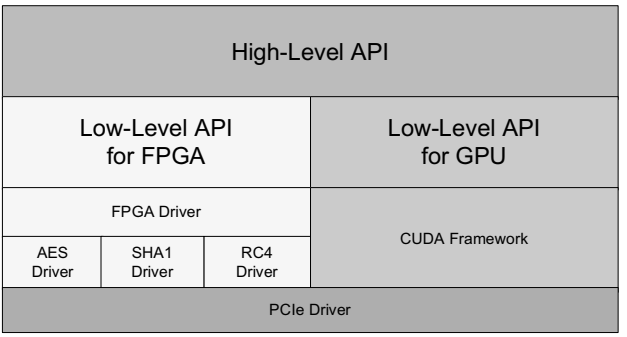


Figure 2: High- and Low-Level API

underlying specialized coprocessors.

The communication between the nodes is accomplished through MS-MPI which allows, in combination with Infiniband, high speed data transfer using Remote Direct Memory Access (RDMA) [LWP08]. Since the Infiniband interface cannot be directly accessed by the coprocessors, the communication has to be initialized by the grid software which has full access to the operating system's Infiniband interfaces.

4.3 Communication Layer

Figure 2 shows the different layers needed to communicate with the underlying coprocessors. As illustrated in Figure 2, the high-level API is situated on the top. The next layer consists of the low-level API which is actually splitted into the FPGA part and the GPU part. Underneath the low-level API, specific drivers required for FPGA development as well as the CUDA framework used for GPU programming can be found. The following two sections describe the high-level API and the low-level API in detail.

High-level API. The high-level API is implemented for abstraction purposes. It provides unified functions for accessing both coprocessor types and is responsible for the following tasks:

- Get the node's hardware information,
- configure the coprocessors related to the given task,
- initiate the coprocessors,
- get status information from the coprocessors and
- get debug information from the coprocessors.

Low-level API. The functions of the low-level API are invoked by the high-level API functions and are directly accessing the drivers of the coprocessors. Basically, the low-level API is divided into two parts to access either FPGAs or GPUs. The FPGA-part and the GPU-part of the low-level API differ in several aspects.
The low-level API for FPGAs is smaller, as any algorithms are directly implemented onto the FPGAs and can be loaded when they are used. The functions of the API are only used for accessing them. Contrary, on the GPU side a library of usable algorithms exists where the algorithms are fully implemented.

Driver Architecture. As illustrated in Figure 2, the low-level API on the side of the GPUs is built on the general CUDA driver for Nvidia graphic cards, which does not need to be modified for a proper use within the cluster.

In contrast, each configuration (actually each algorithm) on an FPGA differs in design, input and output parameters. Thus, a core FPGA driver underneath the low-level API for FPGAs is responsible for loading the specific algorithm driver (e.g. driver for AES, SHA1 or RC4) corresponding to the algorithm. Only if the algorithm driver was initiated successfully, the low-level API is able to communicate with the underlying FPGA. The flexible driver design (the core FPGA driver in combination with the algorithm drivers) enables to run different algorithms or multiple instances of the same algorithm in parallel on a single FPGA board.

5 Inter-Node Communication

This section explains in detail how the communication between two nodes is handled. No matter between which types of nodes (input, output, compute or head node) data is transmitted, any communication is handled the same way via MPI calls. Thereby, MPI is responsible for initiating nodes to send data and allowing nodes to receive data.

While management tasks are handled by the nodes' CPUs, the actual calculations for the cryptographic tasks are executed by the coprocessors. Due to the fact that the coprocessors are not capable of disposing MPI calls, the communication between nodes over the data network has to be triggered by the nodes' CPUs.

The following steps illustrate the principle of node communication between two nodes according to Figure 3.

Node 0. This node plays the role of the sender. It is assumed that the input buffer already contains data and the output buffer is empty. Both memory addresses are known by the coprocessor of the node. The following steps are executed:

1. The coprocessor fetches the input data and processes it.
2. After saving the result into the output buffer an interrupt is provoked.

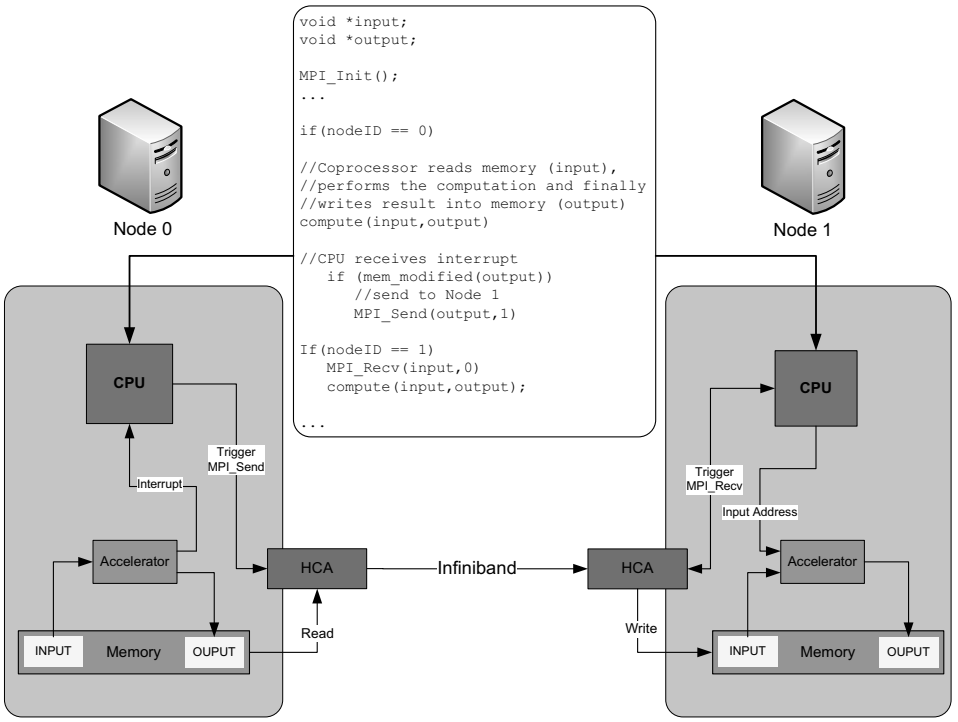


Figure 3: Inter-Node Communication

3. The CPU triggers an MPI call (MPI_Send()), which incites the HCAs to write the content of the first node's output buffer into the second node's input buffer.

Node 1. This node receives data from Node 0. The following are executed:

1. The CPU has to call MPI_Recv() to make the node's HCA ready to receive data.
2. The HCA stores the received data into the input buffer.
3. MPI_Recv() finishes, which indicates a successful data transmission.
4. The CPU sends the memory address of the filled input buffer to the coprocessor.
5. The coprocessor is now capable of processing the input data.

The white box between the nodes in Figure 3 shows a simplified pseudo-code example of how MPI is being used to control the communication. The two basic MPI functions *MPI_Send* and *MPI_Receive* are used to transmit data between the nodes depending on their node IDs. Although the above example shows just one coprocessor per node, the

Pwd: 3rr0r6		Pwd: xavier9	
No. of Nodes	Time [s]	No. of Nodes	Time [s]
1	11	1	367
2	6	2	183
4	1	4	87

Table 1: Results from testing the distributed SHA-1 cracker.

communication principle is very similar for two or more coprocessors included in a single node.

6 Implementations of Algorithms

The presented scalable and parallel computing system is capable of executing various types of algorithms. As this system is developed with focus on cryptographic and cryptanalytic issues, it will mainly deal with tasks of the following sections.

6.1 Brute Force Algorithms

Currently, a distributed SHA-1 cracking algorithm can be executed on the cluster, which uses brute forcing to get the original plaintext from SHA-1 hashes. The algorithm is implemented in CUDA for NVIDIA GPUs and runs in parallel on multiple compute nodes. It generates plaintext strings by using a given character-set. These strings are hashed and compared with the input hash. If both hashes are equal, the plaintext is found. The algorithm requires a SHA-1 hash as input, the length of the plaintext, a character-set and an input-offset for each cluster node. The input-offset defines the partitioning of the input data, which are the generated plaintext strings. The partitioning is carried out by dividing the generated strings by their beginning letter. If a character-set *a-z* would be used for two nodes, the first node would generate strings with the beginning letters *a-m* and the second node with the beginning letters *m-z*. According to that, the input-offset would be *1* for the first node and *2* for the second node, which would divide the input strings into two halfs.

The algorithm was tested for selected passwords on multiple cluster nodes. Two examples are illustrated in Table 1. The character-set *a-z0-9* is used. As the required time shows, the algorithm scales quite well with the amount of nodes. A doubling of the number of nodes, halves the duration on average until the algorithm finishes. In case of the password *3rr0r6* the jump from 2 to 4 nodes decreases the time even to a sixth part of the required time. The reason, why this may happen is illustrated in figure 4, which represents the used character-set. The beginning letter of *3rr0r6* is located in the last half of the specified set. This half pictures the beginning letters that the second node uses for string generation. The node has to generate strings with 11 different beginning letters, before it

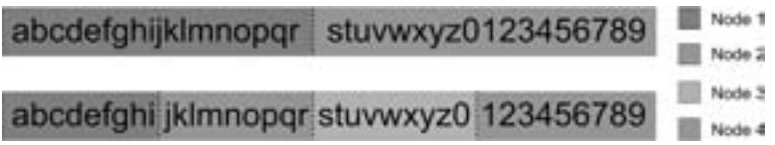


Figure 4: Partitioning of the beginning letters for the cluster nodes.

reaches the ”3”. Using four nodes, the fourth node only has to generate strings with two different beginning letters until it reaches the ”3”. Hence, using four equal nodes, cracking the hashed password *3rr0r6* causes a relatively high decrease of required cracking time, which is a stroke of luck. In average, the speed-up is linear meaning that the calculation time halves, when the amount of nodes is doubled.

6.2 Cryptanalytic Algorithms

Cryptanalytic tasks can be very complex and vary extremely in structure, execution time, required resources, and their capability of parallelization. Only a multi-purpose system that allows free configuration is capable of executing various cryptanalytic algorithms in an efficient way.

7 Future Work

The cluster is still in an early state of development. Recently, we encountered problems concerning the FPGA configuration. At the moment it is not possible to load images onto an FPGA board and use them for calculations within the cluster. The current sample application allows to bruteforce SHA-1 hashes by using only NVIDIA GPUs. Until now, only one algorithm module can be loaded onto the GPUs. In future, multiple crypto-modules shall be developed, which may be used with GPUs and/or FPGAs. Additionally, a graphical user interface shall provide a more intuitive handling of the cluster. The goal of our cluster is to provide a scalable and comprehensive cluster system, which allows easy handling, by the use of prepared and configurable modules.

References

[Lab] Laboratory for cryptologic algorithms. *PlayStation 3 computing breaks 2^{60} barrier*. URL, http://lacal.epfl.ch/page81774.html.

[LWP08] Jiuxing Liu, Jiesheng Wu, and Dhabaleswar K. Panda. *High Performance RDMA-Based MPI Implementation over InfiniBand*. Computer and Information Science, The Ohio State University, June 2008. URL, http://nowlab.cse.ohio-state.edu/publications/journal-papers/2004/liuj-ijpp04.pdf.

[Mic08] Microsoft. *Windows HPC Server 2008: Technical Overview of Windows HPC Server*, June 2008. URL, www.clustervision.com/Windows_HPC_Server_2008_Technical_Overview.pdf.

[Mic09] Micheal Showerman et al. *QP: A Heterogeneous Multi-Accelerator Cluster*. University of Illinois at Urbana-Champaign, Urbana, March 2009. URL, web-test.ncsa.illinois.edu/~kindr/papers/lci09_paper.pdf.

[Ruh06] Ruhr University of Bochum and University of Kiel. *COPACOBANA: A Codebreaker for DES and other Ciphers*, October 2006. URL, http://www.copacobana.org/paper/copacobana_CHES2006.pdf.

[Shu08] Shuai Che, Jie Li et al. *Accelerating Compute-Intensive Applications with GPUs and FPGAs*. Departments of Electrical and Computer Engineering and Computer Science, University of Virginia, May 2008. URL, http://www.nvidia.com/docs/IO/67189/che_sasp08.pdf.

[TL10] Kuen Hung Tsoi and Wayne Luk. *Axel: A Heterogeneous Cluster with FPGAs and GPUs*. Department of Computing, Imperial College London, UK, February 2010. URL, www.doc.ic.ac.uk/~wl/papers/10/fpga10bt.pdf.

Towards Secure and Reliable Firewall Systems based on Minix 3

Rüdiger Weis, Brian Schüler, Stefan Flemming*

Beuth Hochschule für Technik Berlin, University of Applied Sciences
{rcw,bschueler,flemming}@bht-berlin.de

Abstract: Minix 3 is a real micro kernel operation system with a lot of remarkable security features. Two of the main points are size and isolation. The Minix 3 kernel is less than one thousand times the size of Linux. All drivers and the IP stack live in user land. We show a port of the netfilter framework, which leads to a system with better stability and security than the widely used Linux solutions [We07]. Additionally we present some new ideas regarding virtualized systems.

1 Introduction

"your job is being a professor and researcher: That's one hell of a good excuse for some of the brain-damages of minix", Linus Torwalds, 1992.

"I mean, sometimes it's a bit sad and we're definitely not the streamlined hyper-efficient kernel that I had envisioned 15 years ago. The kernel is huge and bloated", Linus Torwalds, 2009.

Security flaws in modern Operating Systems

The security problems facing the current crop of operating systems, including Windows, but also including Linux, are the result of design errors. The errors were inherited for the most part from their predecessors of the 1960s.

Most of these problems can be attributed to the fact that developers aren't perfect. Humans make mistakes. Of course, it would be nice to reduce the numbers and mitigate the effects; however, designers have frequently been far too willing to compromise security and a clean design for speed. Tanenbaum refers to this as a 'Faustian pact'.

In addition to the issues related to sheer size, monolithic designs are also prone to inherent structural problems: Any error is capable of endangering the whole system. A fundamental design error is that current operating systems do not follow the Principle Of Least

*Supported by the Projekt Forschungsassistenz from Europäischer Sozialfonds.

Authority (POLA). To put this simply, POLA states that developers should distribute systems over a number of modules so an error in one module will not compromise the security and stability of other modules. They should also make sure that each module only has the rights that it actually needs to complete its assigned tasks.

The Minix way

Minix is often viewed as the spiritual predecessor of Linux, but these two Unix cousins could never agree on the kernel design. Minix 3 has been designed to reduce the lines of kernel code. With less than 5.000 loc the Minix 3 kernel is less than one thousand times the size of the Linux kernel.

Minix 3 runs on 32-bit x86 CPUs, as well as on a number of virtual machines including Qemu, Xen, and VMware. The operating system includes an X Window System (X11), a number of editors (Emacs and Vi), shells (including bash and Zsh), GCC, script languages such as Python and Perl, and network tools such as SSH. A small footprint and crash-resistant design make Minix a good candidate for embedded systems, and its superior stability has led to a promising new role as a firewall system.

2 The MinixWall

Packet filters are an endangered system component. Despite the pretty good quality of the Linux Netfilter implementation, a number of security issues have surfaced in the past. If a subsystem of this kind is running on the Linux kernel, it will endanger system security. Building on work by the Tanenbaum group, the Technical University of Applied Science Berlin ported the widespread Netfilter framework to Minix 3.

In Minix 3, a single user process could crash without compromising system security. In most cases the reincarnation server would simply restart the process. The differences become even more apparent if an attacker succeeds in executing code. In Minix, a hijacked user process is still a problem, but the effect is far less serious thanks to isolation.

The fact that security-relevant programs such as network packet filters are executed directly inside the kernel space intensifies the problem. This is the cause why a vulnerability in the firewall code does not only effect the filter routines but also influences the whole Linux kernel. A crash of the process thereby not only has an impact on the overall system stability but can be also used in attacks to circumvent system security mechanisms.

2.1 Operating a firewall in Linux and Minix systems

Due to the highly security enhanced and stable design, Minix is predestinated to be run in security critical systems such as firewalls. Figure 1 shows the operation of a packet

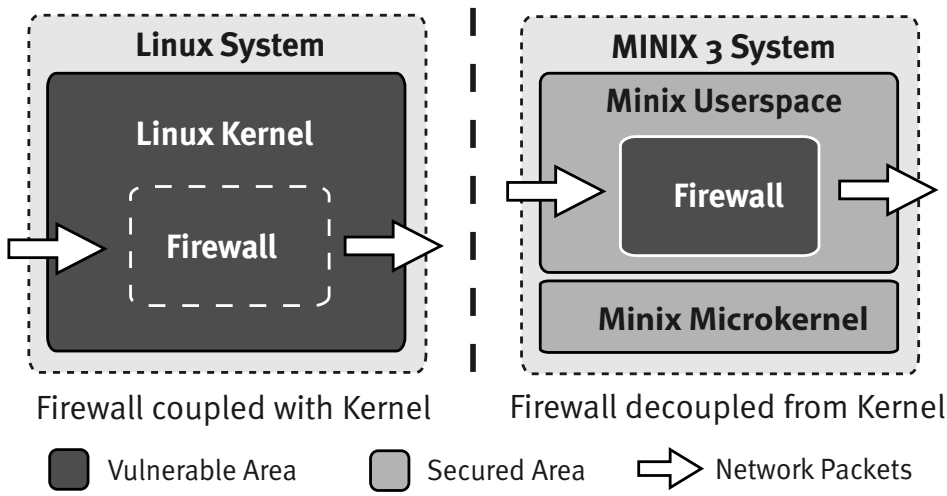


Figure 1: Vulnerability comparison - Linux versus Minix

filter on Linux, compared to one running on Minix. While faulty firewall code in Linux is directly executed inside the kernel space, an exploit will result in an unstable system or even a kernel panic. Beyond that it can be used to hijack the whole kernel. In Minix the execution inside the userspace effectively separates the firewall code from the rest of the system. A crash caused by the packet filter will so only effect the userspace process itself and neither the kernel nor other processes.

2.2 MinixWall - A secure packet filter for Minix

In 2007 the project MinixWall [Sc07] started at the Beuth University of Applied Sciences Berlin. MinixWall ist a port of the iptables/netfilter firewall framework, which is normally part of the Linux kernel. All required sources have been taken from the Linux kernel and freed from Linux-specific structures to make it more portable.

On the Minix side, the network stack daemon *inet* required to be modified by an additional interface that is responsible for the redirection of all network packets to the network filter process. This filter is part of a newly introduced sub-layer for packet filtering tasks. While the filter functions in Linux are executed as sub-routines within the TCP/IP stack, the *MinixWall* filters network packages by a single filter process in user mode and enables to transport them using a device file. In Linux an exploit in the filter is executed inside a sub-routine of the TCP/IP stack and will provoke a kernel panic. A security leak in Minix on the other hand will only crash the single user process without compromising the system stability. In many cases the restart of the service by the *reincarnation server* brings the system again into working state.

To ease the configuration of the netfilter, the userspace application *iptables*, which is well known on Linux environments has been included into the MinixWall distribution in forms of a separate application. This helper tool uses simple device control sequences (ioctl) to configure the firewall rule information. The corresponding filter device is published by */dev/netfilter* and can be made accessible to a specific user and a group that is allowed to administrate the rule set.

Two additional sets of functions have been implemented to deal with the Minix network stack, the *inet* daemon.

- The first set of functions are for exchanging network packet data between the *inet* daemon and the firewall process and are called the backend functions.
  - The functions `inetEthIn(char *ethname)` and `inetEthOut` are for telling the filter which ethernet input and/or output devices are concerned for a network packet (i.e. IN=eth0 and OUT=eth1 in the case of a forwarded packet). One of these functions must be called to initialize the filter state machine that is responsible for the different network packet headers.
  - The functions `inetSetPackSize()` and `inetSetDataSize()` are used to define the total packet size and the data size of the upcoming header respectively.
  - The `inetData()` function transfers a buffer filled with the network packet data to the firewall to be processed later with `inetProcess()`.
- The second set of functions are frontend functions and are designed for setting up and editing the firewall rule set.
  - The command line tool *iptables* from Linux is used here as a frontend. A new filter table is created with `iptablesNewChain(...)` function for example.
  - `iptablesSelectTable(...)` and `iptablesSelectChain(...)` are used for selecting tables and chains.
  - Rules can be appended with `iptablesAppendRule()` to an existing chain.

All these comfortable backend and frontend functions are translated into *ioctl's* sequences that are used for the firewalling device file. The header files which MinixWall uses are similar to these on Linux.

The following example code in the *inet* daemon is passed when a network packet arrives the local machine and performs a full filtering in the INPUT direction:

Protocol	Linux include file	Minix include file
IP header	/usr/include/linux/ip.h	/usr/include/net/gen/ip_hdr.h
ICMP header	/usr/include/net/icmp.h	/usr/include/net/gen/icmp_hdr.h
UDP header	/usr/include/net/udp.h	/usr/include/net/gen/udp_hdr.h
TCP header	/usr/include/net/tcp.h	/usr/include/net/gen/tcp_hdr.h
Routing information	/usr/include/net/route.h	/usr/include/net/gen/route.h

Table 1: networking include files

3 Virtualization

A Minix based system is a reliable and secure platform for the operation of firewalls, but often it is meaningful to take full advantage of the hardware by hosting additional network services on the same machine. Operating all these services directly under Minix already offers several key benefits compared to other operating systems that are not as stable, secure and resource-friendly. Whenever a service crashes, it will only affect one userspace server process that will be restarted immediately.

The classical way to deal with these problems is to operate the *MinixWall* on a standalone network host and forward all filtered traffic to a second computer that is running a different operating system with all required network services such as a mail-, web-, ftp- or file-server. This topology clearly separates the systems with the advantage that exploits do not effect more than one machine but also the disadvantage that additional hardware is required.

3.1 Operation of complete networks on one hardware

An interesting approach is the combination of a Minix 3 based *MinixWall* with other operating systems using virtualization. As shown in figure 2, the hardware of a router is hosting two independent virtualized guest systems. One guest can so operate the *MinixWall* packet filter while the other one serves additional network services. An important advantage of this approach is the need for only one computer while both operating systems are kept separated. Beyond that the user is free to select the best fitting operating system for a specific task.

In respect to the security and stability it is apparent that the virtualized design has advantages caused by this separation. Whenever one guest fails, it can be restarted independently from other systems. If one guest is compromised by an attack, it can be replaced using an image without affecting the other ones. This approach not only enables the separation of critical services in its own virtualization containers but also offers the operation of virtualized guests with different operating systems at the same time. Several combinations of servers and virtualized firewalls are conceivable such as complete networks with DMZs and multiple differently filtered subnets.

```
nf_ioctl(IOCTL_NF_ETH_IN,(void*)ifin);
nf_ioctl(IOCTL_NF_ETH_OUT,(void*)ifout);
nf_ioctl(IOCTL_NF_PACK_SIZE,(void*)pack_size);
nf_ioctl(IOCTL_NF_CONTAINS,(void*)NF_LAYER_IP);
/* sending all buffer parts to the filter */
for (i=0, temppack=pack, count=pack_size; temppack && count>0;)
{
nf_ioctl(IOCTL_NF_SIZE,(void*)temppack->acc_buffer->buf_size);
nf_ioctl(IOCTL_NF_DATA,
(void*) (((vir_bytes)temppack->acc_buffer->buf_data_p)
+temppack->acc_offset)
);
i++; count-=temppack->acc_buffer->buf_size;
temppack=temppack->acc_next;
}
nf_ioctl(IOCTL_NF_HOOK,(void*)NF_IP_LOCAL_IN);

/* do the filtering work */
if (nf_ioctl(IOCTL_NF_PROCESS,NULL)==1)
{
if (broadcast)
ip_arrived_broadcast(ip_port, pack);
else
ip_arrived(ip_port, pack);
}
nf_ioctl(IOCTL_NF_GETDATA,NULL);
```

Table 2: example filter code

3.2 Cascaded firewalls

Figure 2 illustrates that all incoming and unfiltered traffic is forwarded to one *MinixWall*, when it enters the host system. This traffic is guided through the *MinixWall* to be filtered by the packet filter before it is redirected to the Linux guestsystem that acts as a host for all services.

The other way around the packages generated by the *Linux* guest for the outside world can be filtered again by the *MinixWall*, before they leave the router host.

For all router scenarios Minix is equipped with two virtual network interface cards for the internal and external network traffic. One card is directly bridged to the network interface card of the host system while the other one is used for the internal traffic. The hostsystem itself does not require more than one network interface card but can be equipped with a second card if the filtered traffic should be made available to an external system.

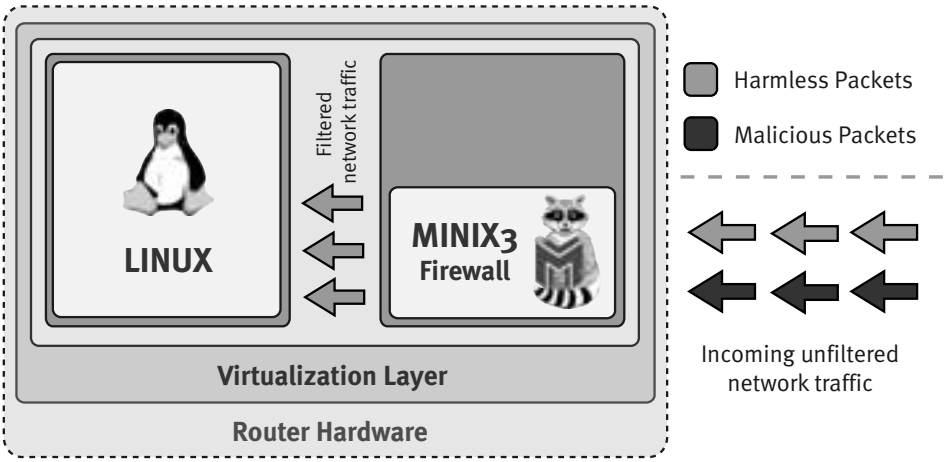


Figure 2: Virtualization of the Minix Packetfilter

While figure 2 shows a solution with only one firewall, it can be extended by additional packet filters to a cascaded firewall solution. Multiple virtualized firewalls can so filter traffic in series to provide redundant security. The risk that an attacker is able to compromise the system is so efficiently reduced, because an attacker needs to break all involved firewalls.

4 Conclusion and Outlook

We have presented a firewall system based on Minix 3 which provides higher security and better reliability by design. Combined with the very low hardware requirements this system has a wide range of applications in the field of embedded hardware and on virtualized systems. It is clear that security researchers would prefer putting the firewall systems onto dedicated hardware, but our solution can be used without additional hardware costs in many scenarios. We are also working on improvements for virtualized networks on one system (e.g. DMZ). All our software is published under der GNU Public Licsence and can be downloaded from: `http://wiki.beuth-hochschule.de/~minixwall`.

References

[Mi05] Minix 3: *The official Minix3 web page*. http://www.minix3.org.

[He06] Herder, Jorrit N.: *TOWARDS A TRUE MICROKERNEL OPERATING SYSTEM*. http://www.minix3.org/doc/herder_thesis.pdf, 2006.

[Sc07] Schüler, Brian : *Analysis and Porting of a network filtering architecture on Minix-3*, Diplomarbeit, TFH Berlin, 2007.

[We07] Weis, Rüdiger : *Linux is obsolete 2.0*, CCCamp 2007, http://public.beuth-hochschule.de/~rweis/vorlesungen/ComputerSicherheit/WeisLinuxIsObsolete2.pdf, 2007.

[We09a] Weis, Rüdiger : *Smart Kernel*, Linux Magazine, 2009. http://www.linux-magazine.com/w3/issue/99/Minix_3_Review.pdf

[We09b] Weis, Rüdiger : *Mit Schutzmantel*, Linux Magazin, 2009. http://www.linux-magazin.de/Heft-Abo/Ausgaben/2009/01/Mit-Schutzmantel?category=0

[Ke06] Kelly, Ivan: *Final Year Project*. Porting Minix to Xen, 2006.

[Ta06] Tanenbaum, Andrew. S. : *Introduction to Minix3*. http://www.osnews.com/story/15960/Introduction-to-Minix-3, 2006.

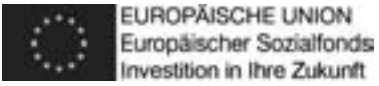
[He06] Herder, J.N. and Bos, H. and Gras, B. and Homburg, P. and Tanenbaum, A.S.: *Minix 3: A highly reliable, self-repairing operating system*. ACM SIGOPS Operating Systems Review, 2006.

[THB06] Tanenbaum, AS and Herder, JN and Bos, H.: *Can we make operating systems reliable and secure?*. Computer Journal Volume 39 Number 5, 2006.

[He07] Herder, J. and Bos, H. and Gras, B. and Homburg, P. and Tanenbaum, A.S.: *Roadmap to a failure-resilient operating system*. https://www.usenix.org/publications/login/2007-02/openpdfs/herder.pdf, 2006.

[Ko09] Van der Kouwe, E.: *Porting the QEMU virtualization software to Minix 3*, 2009.

This work has been supported by the Projekt Forschungsassistenz from Europäischer Sozialfonds (ESF) and the Senat of Berlin.



A transparent bridge for forensic sound network traffic data acquisition

Stefan Kiltz, Mario Hildebrandt, Robert Altschaffel, Jana Dittmann
Otto-von-Guericke University Magdeburg, Germany
Research Group on Multimedia and Security
{kiltz, dittmann}@iti.cs.uni-magdeburg.de, {mhildebr, altschaf}@cs.uni-magdeburg.de

Abstract: In this paper we introduce a prototype that is designed to produce forensic sound network data recordings using inexpensive hard- and software, the Linux Forensic Transparent Bridge (LFTB). It supports the investigation of the network communication parameters and the investigation of the payload of network data. The basis for the LFTB is a self-developed model of the forensic process which also addresses forensically relevant data types and considerations for the design of forensic software using software engineering techniques. LFTB gathers forensic evidence to support cases such as malfunctioning hard- and software and for investigating malicious activity. In the latter application the stealthy design of the proposed device is beneficial. Experiments as part of a first evaluation show its usability in a support case and a malicious activity scenario. Effects to latency and throughput were tested and limitations for packet recording analysed. A live monitoring scheme warning about potential packet loss endangering evidence has been implemented.

Keywords: IT-Forensics, Network security, Reactive security, Intrusion detection

1 Introduction

Computer networks are subject to failure both for unintended and malicious reasons. Such failures are usually attended by incident responders. However, in some cases it is not only important to identify and address the root cause of a malfunction or compromisation of computer networks. Depending on the type of the computer network together with the costs of its downtime, a detailed forensic analysis may become necessary (e.g. to determine which data has been compromised). A forensic model can be used in order to ensure that sound forensic principles are applied throughout the whole investigation. The focus within this paper is the description of a forensic sound network data collection procedure by employing a self-developed tool named "Linux Forensic Transparent Bridge (LFTB)". It is designed to run on inexpensive and readily available hardware and with no costs for software, clearly distinguishing it from costly solutions as provided by e.g. Narus [Nar10]. In the following section the underlying theoretical foundation of our model of the forensic process is introduced. After that, design concepts and their implementation of the LFTB that collects network data with respect to *integrity* and *authenticity* as well as *confidentiality* and *privacy* is shown in detail. Also issues of the deployment of such a device are discussed. Two application examples as part of a first experimental proof of the utility and scalability are shown. The throughput and the latency were tested, addressing the issues of scalability and potential packet loss when employing the LFTB in the field.

2 A holistic model for the forensic process

To gather network data in a forensically sound manner, the security aspects of *integrity*, *authenticity*, *confidentiality* and *privacy* need to be observed (see also [Bis06]). The latter

is relevant because a collection of network data is bound to include data that is protected by data privacy regulations. Different models already exist to describe the forensic process (see for instance [Fre07] and [New07]). Our model extends the one presented in [Fre07] and consists of three main aspects:

- The grouping of investigation steps that belong together logically into **phases**.
- The classification of forensic methods into **classes of forensic methods**.
- The classification of forensically relevant **data types**.

The introduced model covers support cases or hardware/software failures and malicious intent. The three aspects of the model are covered in the following subsections.

2.1 Phases of the forensic process

In our model, six mutually exclusive phases of the forensic process could be identified:

- **Strategic preparation** SP, i.e. measures taken prior to an incident.
- **Operational preparation** OP, i.e measures of preparation for a forensic investigation after a suspected incident.
- **Data gathering** DG, i.e. measures to secure evidence.
- **Data investigation** DI, i.e. measures to evaluate and extract data for further investigation.
- **Data analysis** DA, i.e. measures for detailled analysis and correlation between digital evidence.
- **Documentation** DO, i.e. measures for the detailed documentation of the whole investigation.

The phases of the forensic process do not always follow the chronological order in which they are listed, e.g. the findings during the data investigation phase can lead to subsequent data gathering phases on the same or a different computer system.
During the conduct of each phase, a *process accompanying documentation* should be implemented, i.e. every step taken has to be extensively documented. The final phase of a forensic investigation is the *final documentation*, i.e. the processing of all parts of the process accompanying documentation and the destillation into a report varying in technical detail depending on the audience.
With the Linux Forensic Transparent Bridge (LFTB), the phase of **strategic preparation** SP is of a considerable importance (see also Section 5). Only by looking at IT-forensics from the perspective of the operator of an IT-system, anticipating system failures and malicious activities and therefore practise strategic preparation, the usage of our proposed LFTB becomes useful.
The LFTB can gather data that is relevant to data protection and privacy legislation. The operator of the IT-system has to make sure that he conforms with the legal requirements ahead of any incident.
The decision, what amount of data is to be collected and from which particular LFTB (if multiple instances accross the network exist) form part of the **operational preparation** OP as they rely heavily on the symptom and other incident dependent factors.

2.2 Classes of forensic methods

In our model, forensic methods are not exclusively represented by dedicated forensic toolkits. A lot of software installed on computer systems has forensic capabilities (e.g. logging facilities, anomaly detection). In our model we categorise them according to their type in six classes of forensic methods:

- **Operating system** OS. This class contains methods provided by the operating system (see also [BW06]).
- **File system** FS. This class contains methods provided by the file system (see also [BW06]).
- **Explicit means of intrusion detection** $EMID$. The characteristic of EMID methods provided by extra software is that they are executed autonomously on a routine basis.
- **IT-Application** ITA. This class contains methods provided by IT-Applications run by the user. In addition to their main functionality they also provide forensic methods.
- **Scaling of methods for evidence gathering** SMG. This class contains methods to further collect evidence unsuited for routine usage in a production environment (e.g. false positives, high CPU demands etc.). The LFTB is such an example. It would be undesirable to routinely having the LFTB collect the whole network traffic because of the resulting high data volume.
- **Data processing and evaluation** DPE. This class contains methods which support a detailed forensic investigation, display, processing and documentation.

Those classes of methods are mutually exclusive.

2.3 Data types

Data types are important to describe forensic relevant data of the target computer system, which the forensic tools use as input. The idea is to model forensic data types within a computer system with a layered approach similar to the widely known ISO/OSI network layer model [Zim80]. Although at present our model of data types introduced in the following is considered to cover forensically relevant data within a computer system, it can be extended by new data types in principle. Currently eight data types are suggested:

- **Hardware data** DT_1. Hardware data is data in a system, which is not or only in a limited way influenced by the operating system and applications. Examples are RTC-clock, serial numbers of hardware devices or the code of firmware of hardware devices. This includes virtualisation data, these are easily changed by the host OS but not by the client OS.
- **Raw data** DT_2. Raw data is a sequence of bits (or data streams) of components of the system not (yet) classified. In principle they can contain data of all the other data types. Examples of raw data are memory- and mass-storage dumps. Network packets also constitute raw data.
- **Details about data** DT_3. Details about data constitute meta data added to user data. These can be stored within user data or externally. Details about data can be persistent or volatile. Examples are MAC-times of files or sequence numbers of network packets.
- **Configuration data** DT_4. Configuration data is data that can be changed by the OS or applications, which modify the behaviour of the system, but not its behaviour with regards to communication. That includes the configuration of hardware, of the OS and applications.
- **Communication protocol data** DT_5. Communication protocol data comprises data, which controls the behaviour of the system with regards to communication. That includes, amongst network configuration data, also inter-process communication (e.g. pipes and RPC) of IT-applications.

- **Process data** DT_6. Process data is data about a running process. That includes, for example, the status of the process, the owner of the process, its priority, memory usage or the related application. In IT-applications these can be single threads or data about them.
- **Session data** DT_7. Session data constitutes data collected by a system during a session, regardless of whether the session was initiated by a user, an application or the OS. This includes, for example, started programmes, visited websites or documents within a user session.
- **User data** DT_8. User data are contents created, edited or consumed by the user. This includes, for example, media data such as pictures, texts, audio or video data.

The data represented in the layers, however, is **not** mutually exclusive; the same data that constitutes raw data can also form user data, the difference is the semantics, in which the data is looked at. Although during the usage of the LFTB, in principle, almost all data types can be collected off the raw data DT_2, the main focus is on the Communication protocol data DT_5 and User data DT_8. In the following section we describe the features of the LFTB in detail.

3 Designing forensic software using software engineering techniques

In [Som06] three generic software process models are described. For our project we chose a mixture of the waterfall model and component-based software engineering, since we start from scratch with the design of the LFTB but use parts of the functionality already existing from other authors. For a generic forensic application the requirements to be analysed are methods for $integrity$ and $authenticity$ protection. Depending on the processed data the security aspects of $confidentiality$ and $privacy$ need to be included. Common for all forensic applications is the demand for documentation. Within our model for the forensic process this is the process accompanying documentation (see Section2.1). The resulting software design is shown in Figure 1.

Figure 1: Software design for a generic forensic application

All requirements are incorporated in our LFTB. The process accompanying documentation is a central component for the forensic application, sometimes needing protection for privacy and confidentiality reasons. The implementation and integration phases include the testing of units and the final system (see Section 7).

4 The concept, implementation and usage of a Linux Forensic Transparent Bridge (LFTB)

We designed the Linux Forensic Transparent Bridge to be a universal tool for capturing network data (raw data DT_2) in a sound forensic manner to store evidence on a mass storage device. It is built around the following concepts:

- Integrity of the captured network data
- Integrity of the reports
- Authenticity of the reports and the captured network data

- Integrity of the operating environment
- (Optional) confidentiality of the captured data to preserve privacy
- Stealthy operation within the network

Most of these requirements meet those defined for generic forensic applications. It is of the highest importance, that the source data is not altered by the LFTB during the capturing and storage process. Hence, a cryptographic hash sum is calculated to prove the $integrity$ of the data. The LFTB calculates a cryptographic hash sum using the sha256 algorithm during the capture operation[1]. The LFTB meets the requirement of a accompanying documentation by keeping a detailed log and also ensuring the integrity of that documentation by calculating a cryptographic hash sum, again using the sha256 algorithm. The security aspect of $authenticity$ (i.e. establishing a link of the recorded data to the actual incident) is ensured by a sha256-hmac which uses a user selected passphrase and the current hardware configuration (DMI data) of the computer running LFTB. Hence, our solution ties the gathered network data both to the investigator and the LFTB hardware.We select the sha256-hmac over the use of certificates, which are difficult to employ when using a read-only medium; those certificates would have to be supplied externally.
Instead of developing all functions from scratch the LFTB uses commonly accepted applications and extends them to meet the requirements for forensic applications (see Figure 2).

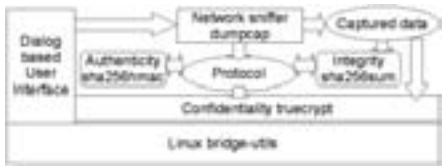


Figure 2: LFTB schematic

The LFTB is a bootable CD-ROM image and comes together with a sha256 publicly comparable cryptographic hash sum, thus ensuring the $integrity$ of the executed code.
We designed the LFTB to be particularly stealthy, which is necessary when recording intended malicious incidents. However, this can only be achieved if the LFTB is deployed in bridging mode or with network taps, since the presence of monitor ports might be detectable via SNMP. Optionally also means of keeping the security aspect of $confidentiality$ and $privacy$ are available. Making this feature optional became necessary, if the hardware platform chosen is not powerful enough to perform both the capturing and encryption processes fast enough. It is, however, positively advised to use the option so as not to collide with data privacy regulations.The following Figure 3 summarises the basic concept.

Figure 3: Functionality of the Linux Forensic Transparent Bridge

We designed the LFTB to record data onto a storage media, together with the documentation script and the cryptographic hash sum, for further analysis. The LFTB is to be used

[1] We select the sha256 algorithm over the popular md5 algorithm despite the higher demands on computational power because the latter is not seen as secure enough with regards to intended collisions.

in the phases of $strategic\ preparation(SP)$, $operational\ preparation(OP)$ and during the $data\ gathering(DG)$ phase according to the model introduced in Section 2. It is a $Scaling\ of\ methods\ for\ evidence\ gathering\ SMG$ method and collects raw data content DT_2 during the data gathering phase DG. The LFTB writes the captured data in a structured file in *pcap* format[2] which can contain all of the forensic data types DT_1 to DT_8. That data is then used in the following phase of data investigation DI to mainly extract Communication protocol data DT_5 and User data DT_8.
Data is captured on a conventional, readily available, x86-based computer system with two network interface adapters installed. The computer has to be equipped with an optical drive capable to boot off a CD-ROM. Mass storage devices need to be attached to the computer (preferably external media such as USB disc drives) to record the captured data together with cryptographic checksums and scripts as part of the accompanying documentation. A generous amount of RAM installed in the LFTB is an advantage, since one option is to use the RAM as intermediate storage for the captured data. The network adapters need to support the promiscous mode for capturing every network packet passing through the network (see also [Tan96]). They should match the surrounding network interface devices (e.g. network speed, duplex operation mode) to prevent a deterioration of the network performance. Data is captured on the data link layer of the ISO/OSI model of network traffic (see [Zim80]). The basis for the Live CD is the Debian Linux Distribution[3] which was hardened with only necessary software present (see [Tur05]). We add the software packages *bridge-utils*, *tshark* and *wireshark-common*, which provides the network packet sniffer *dumpcap*. The first package allows a detailed configuration of two network adapters to form a network bridge. In order to be as stealthy as possible, the spanning tree protocol is disabled. The software $dumpcap$ can act as a network packet capturing tool. Potential alternatives such as *ettercap* were not selected, because they offer the option of actively manipulating network packages and staging man-in-the-middle attacks. To achieve confidentiality and privacy, we use the software *TrueCrypt*[4], which provides a secure virtual drive. A self-developed shell script combines all the software components and provides the user with a screen dialogue based interface.
A useful addition when using the LFTB is the network tap (see also [Lai00]), which separates the upstream and downstream directions of data within the network. Most of the taps offer a read-only access, which is beneficial to forensic investigations, ensuring the stealth functionality and preventing manipulations by the capturing process during the data gathering phase DG (see 2.1). Also placing the network taps as part of the strategic preparation SP allows for a more economical use of the LFTB devices (see Section 5). This enables the investigator to deploy the LFTB without disconnecting the network connection on demand. It also prevents negative impacts to the availability of the network in case of hardware or software failure of the LFTB.
After inserting the LFTB media our boot shell script establishes the bridge (”br0”), using the two network adapters. The script then logs in the user ”lftb” and invokes the execution of the application script that forms the functionality of LFTB. The user is asked to verify the system time to assure a valid time base. The usage of the NTP network time is not an option as it would give away the presence of the LFTB. Although not currently implemented, in principle the employment of radio controlled clocks or GPS modules is possible.
After ensuring the time base, the script asks for the name of the investigator for use in transcripts and reports. In the next step the investigator has to chose a target for storing the captured packets. A menu allows for mass storage devices initialised for use with the LFTB. The script checks and warns the investigator, if data from previous sessions has been detected. With the highly favourable option of using a virtual secure drive being cho-

[2] http://www.winpcap.org/ntar/draft/PCAP-DumpFileFormat.html
[3] http://www.debian.org
[4] http://www.truecrypt.org/downloads

sen, this drive can also be created during this step. After that the investigator currently has five options for recording network data traffic:

- Capture complete traffic.
- Capture traffic to or from a particular MAC.
- Capture traffic using a custom filter.
- Capture traffic using a timer.
- Capture traffic with delayed start and timer.

These options are chosen to reflect different needs when being deployed at different locations (see Section 5). Especially the employment of custom filters (e.g. only recording traffic using the http protocol) allows for scalability with respect to respect to storage space limitations and for application in support cases. The options differ further in the network source and the start and length of the capturing process. All options, however, gather network data starting from the data link layer up to the application layer of the ISO/OSI Model (see also [Zim80]). Further options could easily be added, should the need for extension arise. Every network data recording is then secured in its integrity using the sha256 hashing algorithm. However this can not cover errors during the writing process to the HDD, because the hash is computed afterwards. In future releases the network packet capturing tool should be extended or replaced by a tool which supports internal hashing. The transcript contains timestamps for the start and the termination of the recording (see Figure 4). Those dates are based upon the time the investigator entered when starting the LFTB and are therefore independent of potentially maladjusted system clocks in the network, therefore providing proof within the limits of the correct clock setting of the investigator.

```
Linux forensic transparent bridge evidence storage
Starting time: Sun Mar 28 20:18:51 CEST 2010
Investigator: Hildebrandt
--------------------------------------------------------------------------------
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww

Log item SHA256 hash: 717322afeb1490eaede00b4dfc40215046e60d3d9e3ab75cfe73fdbb1d
053e91
HMAC: 3d161b1c8168a313c8f2cd3a6c5e2b3b58b7507f12e8a091cad81183178b9837
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww

--------------------------------------------------------------------------------
starting time: Sun Mar 28 20:18:55 CEST 2010
action: dumpcap -i br0 -w /root/data/1269800335.cap -a filesize:127004
exit time:Sun Mar 28 20:19:29 CEST 2010
result: /root/data/1269800335.cap
SHA256 hash: eb5b59ecf71ce7adbf9e8c08e1b75c904b9a0387ac73ebc8c324f375a6144295 /r
oot/data/1269800335.cap
dumpcap output:
Packets: 95648 Packets dropped: 0
--------------------------------------------------------------------------------
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww

Log item SHA256 hash: a466dc5cdded8b9c5e1e59e20e54fff7379f6a398e7f652c6f80cbfa4d
92ab26
HMAC: 32066f6d65a8aa4b9b708c2cf13635d48220e8f5294ea1590804bf2eaa3afbd8
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww

--------------------------------------------------------------------------------
MAC-table of br0 at Sun Mar 28 20:19:33 CEST 2010
port no mac addr is local? ageing timer 1 00:0a:8a:a2:30:b5 no 38.69 1 00:0c:29:
3a:86:af yes 0.00 2 00:0c:29:3a:86:b9 yes 0.00 1 00:12:43:30:c1:e7 no 43.66 1 00
:19:66:c3:47:1d no 73.59 1 00:1a:4f:85:0f:6d no 36.79 1 00:30:1b:b8:1e:6c no 8.6
2 1 00:80:c8:d7:ef:c5 no 35.41 1 40:00:04:11:6f:44 no 10.38 1 40:00:04:11:6f:46
no 10.18 1 40:00:04:11:6f:52 no 10.58 1 40:00:04:11:6f:7d no 10.78 1 40:00:04:11
```

Figure 4: Output of the protocol function of the LFTB

In the bottom of the figure the MAC table of the bridging mechanism is shown, which can be crucial when IP addresses need to be correlated to hardware MAC addresses. Confirming with forensic best practices every action the investigator performs is recorded together with command line parameters and the results. A sha256 hash sum and a sha256 hmac hash is being calculated for every item, thus ensuring $integrity$ and $authenticity$ of the collected data. In contrast to the hash of the network dumps, the hashes for each log item are computed internally before the storage process. Thus write errors can be detected.

5 Positioning of the Linux Forensic Transparent Bridge (LFTB)

The data collected by the LFTB depends heavily on the location within the network it is placed in. This placement is a vital part of the strategic preparation SP introduced in Section 2.1. Figure 5 shows a typical local network and potential locations of the LFTB.

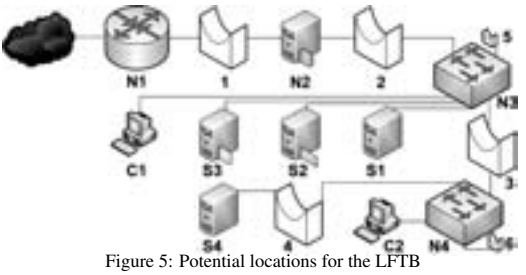


Figure 5: Potential locations for the LFTB

In this setup $N1$ represents a router connected to the Internet. $N2$ is a Firewall/VPN server. $N3$ and $N4$ represent network switches. $S1$ to $S4$ incorporate various dedicated servers. In an exemplary setup, $S2$ has a mail server functionality, with $S3$ acting as a file server. $S4$ is a general purpose server with high protection requirements. In this setup, $C1$ represents a client with normal and $C2$ a client with high protection requirements.
If the LFTB is placed at location 1, it records unfiltered network data traffic to and from the Internet. However, VPN connections are only recorded in their encrypted representation. If the LFTB is stationed at location 2, it captures filtered network data traffic to and from the internet. VPN data is captured in its unencrypted representation. If the functionality of the firewall $N2$ has to be investigated, two LFTBs on location 1 and location 2 are needed. If a LFTB is situated at location 3, the transition of network data between systems with normal protection requirements to systems with high protection requirements can be captured. To exclusively capture network data originating from or destined to server $S4$ with high protection requirements, a LFTB should be placed at location 4. Network taps (see Section 4) at the locations 1 to 4 should to be placed as part of the strategic preparation SP. If the switches $N3$ or $N4$ have a monitoring port, the LFTB could be connected to it.

6 Examples for using a Linux Forensic Transparent Bridge (LFTB)

In this section we show, using two scenarios, how a deployment of LFTBs collecting forensic sound evidence (i.e. adhering to the security aspects of $integrity$, $authenticity$) as well as conforming with $confidentiality$ and $privacy$ can aid in investigating both support cases (malfunctioning hard-/software components and operating errors) and intended malicious activities. The result in both cases is a capture file together with a sha256 hash sum ensuring $integrity$ as well as a report file with a hmac sum for every protocol entry ensuring $authenticity$ and a sha256 sum ensuring the $integrity$ of the report file. The output is placed in a secure virtual drive for $confidentiality$ and $privacy$, allowing access only to the investigator supplying the chosen password. Also we discuss the scalability of the LFTB for the chosen examples as a first evaluation of the concept of the LFTB.

Support case For the first scenario we assume that two systems within a network are accidentally assigned the same IP address. This seemingly trivial problem can prove to be a major problem especially in large intranetworks. In the network (see Figure 5) the task of distribution of IP addresses is assigned to the dhcp server $N2$. It uses static dhcp (i.e. each hardware MAC address is tied to a preconfigured IP address). A client using the Microsoft Windows XP operating system shows multiple alerts of IP address conflicts. A LFTB was

placed at location 3, from which the whole of the network traffic can be captured with all local IP addresses being present. The network traffic (DT_2, see Section 2.3) is recorded as part of the data gathering phase (see Section 2.1) using the LFTB.
Depending on the overall network bandwidth the scalability of the LFTB can be a factor. Since at location 3 the full bandwidth can be required, off-the-shelf PC hardware may not be sufficient to capture the full data content in time, especially if a secure virtual drive is used.
For the following investigation we select the tool "tshark[5]" from the class DPE (see Section 2.2) because it allows the dissection of the captured data in the following phase of data investigation DI to extract data confirming to the ISO/OSI transport layer (see [Zim80]). This extracts the MAC address data (DT_5, see Section 2.3) from the captured network data. The same tool can also be used to extract ISO/OSI gateway layer data (DT_5). In the phase of data analysis DA those two investigation results can be correlated. For this we also need the ARP tables[6] (DT_5) as collected by the LFTB. It in this case an IP address is used by two different MAC addresses. By using this information to correlate it with the data from the dhcp server, it shows which computer was using the IP wrongly. By having the MAC table from the LFTB it can be seen, in which network segment the computer that is using the IP address wrongly is located. This enables the investigator to locate the misconfigured system even without managed switches, witch also maintain a user accessible MAC table. Following the data analysis part, a detailed technical report in the documentation phase DO (see Section 2.1) which also includes all of the accompanying documentation has to be created.

Malicious activity In the second scenario, an unusual network activity has been observed. An LTFB was placed on location 2 (see Figure 5) during the strategic preparation SP (see Section 2.1). This location is chosen because the whole and unfiltered network data traffic (DT_2, see Section 2.3) is needed and no filtering decisions can be made as part of the operational preparation OP. The network data traffic is captured by the LFTB to secure evidence as part of the data gathering phase DG. At the chosen location the scalability of the LFTB is not likely to be an issue since the volume of network traffic passing from the intranet to the Internet will not overstrain the LFTB. In the following phase of data investigation DI, the communication protocol data DT_5 of the captured network data is dissected using the tool $tshark$. The following Figure 6 shows a sample of the output of $tshark$ when used in the data investigation mode, depicting communication protocol data (DT_5, see 2.3) on the transport layer according to [Tan96]).

TCP Conversations
Filter:<No Filter>

		<-		->		Total	
		Frames	Bytes	Frames	Bytes	Frames	Bytes
192.168.3.200:8822	<-> 192.168.1.14:3722	164	27540	163	10997	327	38537
192.168.3.200:8823	<-> 192.168.1.14:3124	71	14930	71	4802	142	19732
192.168.3.200:50251	<-> 192.168.1.14:11457	51	6378	71	4978	122	11356
192.168.3.200:52786	<-> 192.168.1.14:447	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:1398	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:3985	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:1021	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:268	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:479	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:605	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:742	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:288	1	54	1	58	2	112
192.168.3.200:52786	<-> 192.168.1.14:1541	1	54	1	58	2	112

Figure 6: Communication protocol data analysis using $tshark$

Consequently the IP traffic together with port information is visible. It shows that one computer (192.168.3.200) establishes a lot of connections to different TCP ports of another computer (192.168.1.14) only transmitting one single network packet (a typical indicator

[5] see http://www.wireshark.org
[6] see also http://www.ietf.org/rfc/rfc826.txt

of a port scan). Also network traffic using the TCP ports 11457, 3124 and 3722 is detected. Since those are of no regular use on that computer, an investigation of the payload is inevitable as part of the data analysis phase DA. The tool $wireshark$ can be used on the packet level of the captured data from the LFTB. The following Figure 7 shows the output of a packet level analysis to extract user data DT_8 (see 2.3).

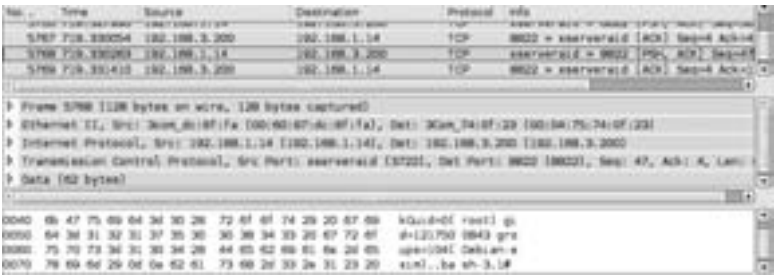


Figure 7: Network data packet level analysis using $wireshark$

The communication protocol data can be seen in the middle of the screenshot, sectioned into the different layers of the ISO/OSI model. In the *Ethernet II* entry the MAC address (DT_5) representing the layer 2 (i.e. the data link layer) is shown. The content of the network layer, the IP address, constituting DT_5 is shown on the entry *Internet Protocol*. The entry *Transport Control Protocol* shows the ports used and represents transport layer content (DT_5). The payload of the selected network data packet is shown on the lower part of the screenshot (DT_8). We select the forensic toolkit $PyFlag$[7], from the class DPE (see Section 2.2) because it enables the restoration of complete sessions (i.e. the combination of payload data).

Figure 8: Network session analysis using $pyflag$

The resulting output (see Figure 8) shows the execution of the script named $make\_me\_root$. Also, a malicious kernel module $enyelkm.ko$ is moved to the /etc directory on the target computer. In the data analysis DA phase it can be established, that shell commands are transmitted using those ports. As shown in Figure 7, this backdoor-ports were also utilised by the attacker. By having the complete network data recording, the progress of the attacker can be reconstructed, including which data was accessed by the intruder. A detailed report in the documentation phase DO (see 2.1) ends the investigation.

[7] http://www.pyflag.net

7 Testing the Linux Forensic Transparent Bridge (LFTB)

Testing of the LFTB covers the impact on the network by deploying one or more LFTB-devices and the quality of the collected evidence. The test concept includes throughput and latency, as well as the expected packet loss depending on the storage destination and the integrity and authenticity of the log-file.

LFTBs impact to the network For the impact the network transfer rate and the latency of packets were tested. Each test was performed for three setups: no LFTB deployed, LFTB connected to the monitor-port of the switch and LFTB in the bridging-mode.

performed test	reference without LFTB	LFTB at monitor port	LFTB in bridging mode
average ping latency	0.258ms	0.252ms	0.414ms
transferring time	22.655s	23.196s	22.918s

Table 1: Average ping latency and transferring time

Table 1 shows the average ping latency in the tests and the time to transfer a 256MB file via netcat through the network. The ping-latency has not changed in the first two cases and got bigger in bridging-mode. This is an expected result, because every OSI-Layer-2 device adds to the total latency. The network transfer rate has not changed significantly.

Quality of the collected evidence When capturing network traffic a certain amount of dropped packets, resulting from insufficient CPU-power or from the bandwith limitation of the storage device. Especially the ad hoc encryption consumes a lot of CPU-time. Our testing system was equipped with two Pentium II CPUs, each running at 450MHz and 384MB of RAM. The network connection was set to 100MBit.

	HDD+TrueCrypt (256MB file)	HDD (256MB file)	RAM-drive (128MB file)
captured/dropped (percentage)	139403 / 150853 (51.97)	210929 / 83075 (28.26)	139170 / 9603 (6.45)

Table 2: Captured and dropped packets

Table 2 shows the percentage of dropped packets. Our conclusion is that the testing system is insufficient for capturing the full 100mbit traffic. In each case a warning message was displayed to warn about the packet loss. In order to be able to capture the full traffic without dropping packets the prior-incident testing is a necessity. However, this packet loss only affected the recording, not the network itself. Further testing showed that newer systems are capable of recording the full 100mbit traffic into a RAM-drive (see Table 3).

System	RAM-drive (256MB file)	HDD (256MB file)	HDD+TrueCrypt (256MB file)
Pentium-M 1.6GHz	0	0.41	6.47
Athlon64 X2 2*2GHz	0	0.37	3.69
PhenomII X4 4*3.2GHz	0	0.01	0.13

Table 3: dropped packet percentage on newer systems (worst results)

The systems in Table 3 used different hard disk drives. For the PhenomII X4 an internal 3.5 inch SATA disk was used. The two other systems utilised a 2.5 inch SATA HDD, which was connected externally via USB. While capturing to the internal drive resulted in no or small amount of dropped packets, the USB device caused constant drop rates around 0.4 percent. However the realtime AES encryption needs sufficient processing power. The five year old hardware of the two slower systems is obviously not sufficient. The PhenomII is still too slow resulting in drop rates between 0.046 and 0.13 percent. The utilisation of special network adapters, which reduce the CPU load, might improve those results.

Reliability of the LFTB log file Although each item is secured by a sha256 hash it is very easy to replace an item with another one. Hence, an additional HMAC is used to provide authenticity. To prove the authenticity of the log item the output of dmidecode of the LFTB and the HMAC password is required.

8 Conclusion

In this paper we have proposed the Linux Transparent Forensic Bride (LFTB) as a network data gathering tool for the investigation of support cases and malicious activities. The LFTB allows for comprehensive and stealthy capturing of network data, ensuring integrity and authenticity of the data and, optionally but highly recommended, confidentiality and privacy. For its construction a self-developed holistic model of the forensic process was used, also firmly integrating accompanying documentation. We discussed the importance of the strategic preparation as an important part of forensic investigations to broaden the view of forensic investigations as a well defined way of data analysis not only limited to criminal proceedings. Especially when using the proposed LFTB the strategic preparation becomes paramount as the location of such a capturing device needs to be determined ahead of a potential incident.
The location management, the scalability to record potential large amount of data in quickly as well as issues with the key management for the hmac and the secure virtual drive containing the evidence constitute further work and testing areas.

Acknowledgement

The work in this paper for IT-forensics has been supported in part by the the Federal Office for Information Security (BSI). The authors wish to thank Mr. Carsten Schulz from the BSI for the initial inspiration and continued support.
The information in this document is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

[Bis06] Matt Bishop. *Computer Security - Art and Science*. Number ISBN 0-201-44099-7. Addison-Wesley, 2006.

[BW06] Steve Bunting and William Wei. *The official EnCE Study Guide*. Number ISBN 978-0-7821-4435-2. Wiley Publishing Inc., 2006.

[Fre07] Felix Freiling. A Common Process Model for Incident Response and Digital Forensics. *Proceedings of the IMF2007*, 2007.

[Lai00] Brian Laing. Intrusion Detection Systems. Technical report, IBM Internet Security Systems, 2000.

[Nar10] Narus. Narus - Leader in Real-Time Traffic Intelligence. http://www.narus.com/, April 2010.

[New07] Robert C. Newman. *Computer Forensics - Evidence collection and Management*. Number ISBN 978-0-8493-0561-0. Auerbach Publications, 2007.

[Som06] Ian Sommerville. *Software Engineering*. Number ISBN 0-321-31379-8. Addison Wesley, 2006.

[Tan96] Andrew S. Tanenbaum. *Computer Networks*. Number ISBN 0-13-394248-1. Prentice Hall International Inc., 1996.

[Tur05] James Turnbull. *Hardening Linux*. Number ISBN 978-1-59059-444-5. Apress, 2005.

[Zim80] Hubert Zimmermann. OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transaction on Communications*, 1980.

A Study on Features for the Detection of Copy-Move Forgeries

Vincent Christlein, Christian Riess, Elli Angelopoulou

{sivichri@stud, riess@i5, elli@i5}.informatik.uni-erlangen.de

Abstract: Blind image forensics aims to assess image authenticity. One of the most popular families of methods from this category is the detection of copy-move forgeries. In the past years, more than 15 different algorithms have been proposed for copy-move forgery detection. So far, the efficacy of these approaches has barely been examined. In this paper, we: a) present a common pipeline for copy-move forgery detection, b) perform a comparative study on 10 proposed copy-move features and c) introduce a new benchmark database for copy-move forgery detection. Experiments show that the recently proposed Fourier-Mellin features perform outstandingly if no geometric transformations are applied to the copied region. Furthermore, our experiments strongly support the use of kd-trees for the matching of similar blocks instead of lexicographic sorting.

1 Introduction

The goal of blind image forensics is to examine image authenticity without the help of an embedded security scheme (like watermarks). In the past few years, a diverse set of methods for blind image forensics has been developed, see e. g. [SM08, NCLS06, Far09a]. The existing methods can be divided into three categories. The first group focuses on detecting the existence of expected artifacts that have been introduced in the process of image sensing. The detection of inconsistensies in such artifacts is a cue for the lack of authenticity of an image. Particular examples include the recovery of a camera-specific sensor noise pattern by Lukas *et al.* [LFG06], the estimation of the camera response function [HC07], demosaicing [GC08], or the extraction of lens properties [JF06].

A second approach is to search for inconsistencies in the scene that have been introduced as a side effect of tampering. Johnson and Farid presented an approach for the examination of the illumination direction [JF05] and for the geometric composition of the light sources in a scene [JF07]. Others like Lalonde and Efros used color distributions in images in order to detect spliced images [LE07].

Finally, many groups considered image artifacts that are introduced by the tampering operations themselves. Examples are artifacts from double JPEG compression (e. g. [HLWT06, Far09b]) or the detection of resampling artifacts [PF05]. One of the most actively researched topics in this area is the detection of copy-move forgeries, see for example [LHQ06, BSN09, LWK09, MS07, KW08, BSM09]. Copy-move forgery detection (CMFD) aims at

finding regions that have been copied and pasted within the same image. The goal of such a tampering operation is to either hide unwanted parts of the image, or to emphasize particular image content, like enlarging a crowd of people. Figure 1 shows an example that was originally presented in [MS07]. In this case, grass has been copied over one of the soldiers.

Unfortunately, although a considerable number of solutions have been proposed for this problem, a thorough comparison of these methods has not been performed so far. We believe that there are two reasons for the lack of such a comparative evaluation: the missing common ground truth data (i. e. standardized benchmark datasets) and the lack of standard metrics for their evaluation.

Figure 1: Example for copy-move forgery from [MS07] (original image left, tampered image right). Grass has been copied to hide one of the soldiers.

In this work, we perform a comparative study on a subset of different feature sets that have been previously proposed. To accomplish this, we: a) formulate existing CMFD methods as concrete instances of a more abstract "copy-move detection pipeline", b) perform a comparison of the methods within this pipeline, and c) present a diverse set of benchmark images for the evaluation of these methods. We believe that this comparison gives further insights into the strengths and weaknesses of particular features for copy-move detection. Furthermore, we propose a common metric for benchmarking copy-move detection methods.

2 Database and Testing Protocol

To create the dataset, we defined 48 manipulation cases. The manipulations were performed by three individuals with different technical skills, so that each person worked on 16 images. Figure 2 and Figure 3 present selected examples from the database. The database is available on our web page[1]. The core manipulation technique that was performed is copy-move forgery. Hence, the major part of every manipulated region stems from another region of the same image. For a more realistic forgery, the boundaries of the copied regions were adjusted with standard image editing methods (like partial trans-

[1] http://www5.informatik.uni-erlangen.de/data

parency, painting and smearing). Finally, per-pixel labels are automatically assigned by a supplementary software. The assigned labels are "original", "altered" and "copy-moved", as shown in Figure 3. For this study, only "original" and "copy-moved" pixels were considered (see below).

2.1 Source Manipulation Properties

We chose six images from the database (originating from different camera types) to compare the features of ten copy-move forgery detection methods. The images have been selected so that they contain representative challenges for copy-move detection algorithms (see also Sect. 4). They contain different numbers of manipulations, as real-world forgeries. The image size is very important for the detection algorithms. Thus, three of the images have a relatively high resolution of more than 2000×1600 pixels, while the other three images have a lower resolution of about 1000×800 pixels. The duplicated regions themselves also vary significantly in size. Thus, feature sets that make assumptions about the size of the manipulated region are at a disadvantage.

The boundaries of the tampered regions are additionally postprocessed with standard image editing methods. The desired effect is that no clear edges appear and the region fits better with its surroundings. Another way is to create shadows, so that the lightning appears correct. This is a common operation in real-life forgeries in order to disguise the copied region. Three of the six images used in this paper are shown in Figure 2. As can be seen, the operations performed in these images vary in type and size of the copied area. For instance, the neck markings of the giraffe and a modified grass patch affect very small regions, while the statues at acropolis involve entire landmarks.

Figure 2: Original images are in the top row, forgeries are in the bottom row. From left to right: *Acropolis* (large copied region), *Cattle* (small region), *Giraffe* (small region)

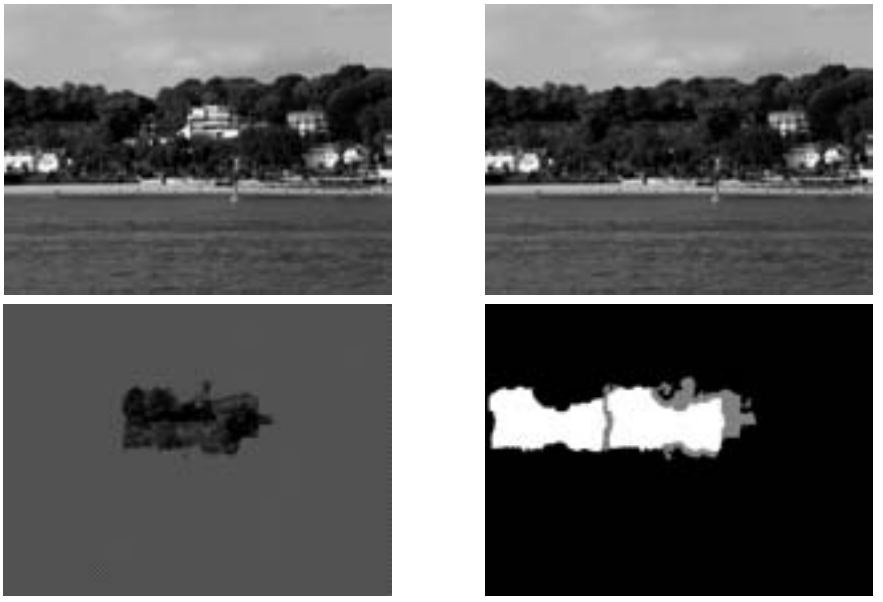


Figure 3: The image *Beachwood* (upper left) is forged with a green patch (bottom left) to conceal a building (upper right). A ground truth map (bottom right) is generated where copy-moved pixels are white, unaltered pixels are black and boundary pixels are gray.

2.2 Testing Protocol

We created ground truth labels for every image. In the ground truth image, the postprocessed boundaries of the duplicated regions are marked differently than the exactly copied pixels (see Figure 3). In our evaluation, we only use exactly copied pixels. Though many methods claim that they can detect such postprocessed parts, we chose to exclude them for two reasons:

1. Most copy-move forgery detection (CMFD) methods mark blocks of copied pixels. Thus, a block on the boundary between copied and non-copied pixels is not well defined as “copied” or “original”.

2. One can not clearly define a set of permissible boundary manipulations, in such a way that these pixels can still be considered as copy-moved.

Hence, we believe that the cleanest solution is to exclude boundary pixels from the evaluation. Note that, although not included in this study, this definition does not hinder us to apply “global” post-processing on the image like additive noise or JPEG compression. The robustness of the methods against various kinds of postprocessing can still be evaluated by introducing these artifacts on the copy-moved regions in a controlled manner.

2.3 Detection Error Measures

We employed two basic error measurements, following the ideas of [BSN09] and [LHQ06]. The percentage of false positive pixels F_{P} and the false negative rate F_{N}. More precisely, let R_1 be the copied region, R_i, $i > 1$ be the i pasted regions and B the unchanged background. Then,

$$F_{\mathrm{P}} = \frac{|\text{matches in } B|}{|B|}$$

and

$$F_{\mathrm{N}} = \frac{|\text{missed matches in } (\bigcup_i R_i)|}{|\bigcup_i R_i|} \quad ,$$

so that lower rates of F_{N} and F_{P} indicate higher accuracy.

Note that, as long as a copied region is detected, we consider a high F_{P} rate to be worse than a high F_{N} rate. The reason is that high F_{P} rates lead to a highly confusing overdetection result, which: a) enforces a man-in-the-loop to examine every result and b) might even conceal the truly tampered regions. Also note that the proposed metrics are only applicable in pixel- or block-based methods. In this work, we considered only block-based methods, so that this metric is appropriate. To compare keypoint based methods like [HGZ08] with each other, the absolute number of false positives and false negatives could be used as an approximate indicator of the algorithm performance.

3 Methods and settings

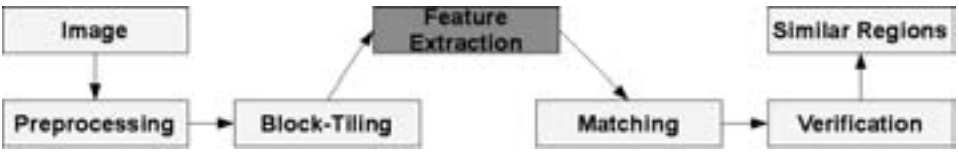


Figure 4: Pipeline of copy-move forgery detection algorithms

The known methods for block-based CMFD follow a similar structure. We built a common framework for CMFD methods that adhere to this structure. Figure 4 shows an overview of the algorithm pipeline containing the main processing steps. After a possible preprocessing step, the image is tiled in small overlapping blocks. For every block a discriminating feature vector is computed. The main difference between the various methods lies in feature extraction.

Similar blocks are pairwise grouped in the matching step. The matching itself is most often done with so-called “lexicographic sorting”. Feature vectors are put row-wise in a matrix and (row-wise) lexicographically sorted. Assuming that similar blocks yield similar features, sufficiently close (w.r.t. a distance measure on the feature vector space) consecutive rows are matched as pairs. Another, not as widely used, approach is to use a kd-tree to

find the nearest neighbor in the d-dimensional feature space directly [MS07]. A block pair is typically subject to further constraints. Most often, two matching blocks must have a minimal Euclidean distance between them. This should avoid matching blocks within the same (homogeneous) region.

The verification step filters matching pairs that follow a common pattern. The most common criterion for the verification step is the recognition of same-shift-vectors. We also use this approach in our evaluation. We sort the found matches by their shift-vectors. All shift-vectors that occur more often than a threshold N_f are assumed to be tampered. Note that this approach is inherently limited when geometric transformations, like rotation and scaling, are applied to the copied regions.

The most distinguishing property of the various CMFD algorithms is the employed feature vector. Features of ten copy-move forgery detection algorithms were evaluated. These features can be divided in four groups: moment-based, dimensionality reduction-based, color-based, and frequency domain-based features. MOMENTS 1 denotes the approach of [MS07]. It uses 24 blur-invariant moments as features. MOMENTS 2 uses the first four Hu-moments as features [WLZ+09]. In [PF04], the feature matching space is reduced with principal component analysis (PCA). [KW08] computes the singular values of a reduced-rank approximation (SVD). A third approach using discrete wavelet transformation (DWT) and Singular Value Decomposition [LWTS07] did not give reliable results in our setup and thus is excluded from the evaluation. The first three features used in [LHQ06] and [BSN09] are the average red, green and blue components. Additionally, Luo et al. [LHQ06] use directional information of blocks (COLOR 1) while Bravo-Solorio et al. [BSN09] consider the entropy of a block as discriminating feature (COLOR 2). COLOR 3 by [LWK09] computes the average grayscale intensities of a block and its sub-blocks. The last three methods we investigated consider features in the frequency domain. The features in [FSL03] are 256 coefficients of the discrete cosine transformation (DCT). Coefficients of a discrete wavelet transform (DWT) using Haar-Wavelets form the features in [BNO+07]. A Fourier-Mellin Transform (FMT) yields the features in [BSM09].

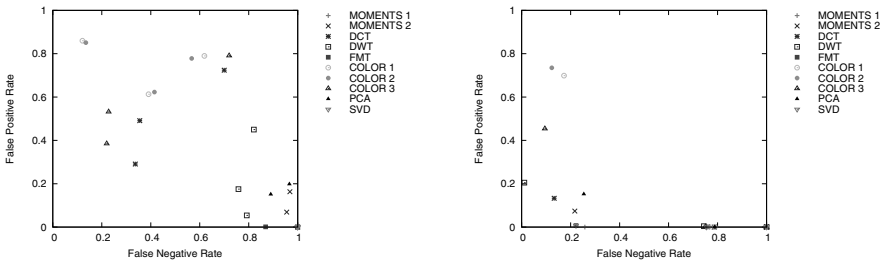
A uniform parameter setting was chosen for better comparison of the individual features. Most of the methods use a block size of 16×16 pixels because it provides a good trade-off between robustness and false positives. We fixed the block size for tiling the image in overlapping blocks accordingly to 16×16 pixels. The step size between every two neighboring blocks was set to 2 pixels to generate fewer blocks and thus decrease the computational complexity. No further preprocessing except grayscale conversion of the image was applied (for algorithms that assume grayscale images).

In order to test the robustness of the features, only the feature sets were exchanged during our different evaluation runs. To decrease the false positive rate, the minimum frequency threshold for the numbers of same-shift-vectors was set to $N_f = 50$. Thus, the minimal area that can be detected is of size 31×31 pixels. The minimum distance between matching blocks was set to 50 pixels, which is also supported by the benchmark images set.

4 Evaluation

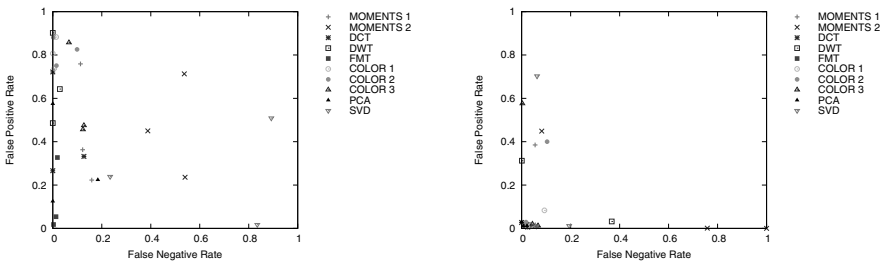
We divided the benchmark images in two groups, large images and small images. Large images are inherently more challenging, since an overall higher number of feature vectors exists, and thus there is a considerably higher probability of matching wrong blocks. On the large image set when using lexicographic sorting, only color features and DCT were able to detect duplicated regions with acceptable error rates, see Figure 5a. We observed that the weakest spot of the other methods is that if too many blocks are wrongly paired, this indeed leads to wrong hypotheses on the dominant shift-vectors, rendering the result barely usable.

Note the comparably weak performance of MOMENTS 1 and MOMENTS 2, on large images as well as on small images. MOMENTS2 was able to detect forgeries in only one small image, MOMENTS 1 always had a false negative rate of 1 and could, hence, detect nothing (Figure 5b).



(a) Large images (b) Small images

Figure 5: Feature Test – Lexicographic sort



(a) Large images (b) Small images

Figure 6: Feature Test – kd-tree sorting

Surprisingly, this behavior completely changes when using a kd-tree. Here, the moment features detect manipulations in all images. In large images (Figure 6a), only FMT and

DCT yield better detection results than MOMENTS 1. PCA gives reliable results, too, while SVD detects duplicated regions in only one large image. Among the color-based methods, COLOR 3 performs best. The features of COLOR 1 and COLOR 2 have very high false positive rates and can not be reliably used for detection.

We assume that this behavior comes from the nature of lexicographic sorting. If the first few elements of the feature vector are indeed the most significant entries, lexicographic sorting might be an appropriate choice. Apparently, the more balanced distance computation of the kd-tree supports the nature of most feature vectors better. On the other hand, many feature vectors exhibit an oversensitivity when using a kd-tree, leading to extremely high F_{P} rates (Figure 6a).

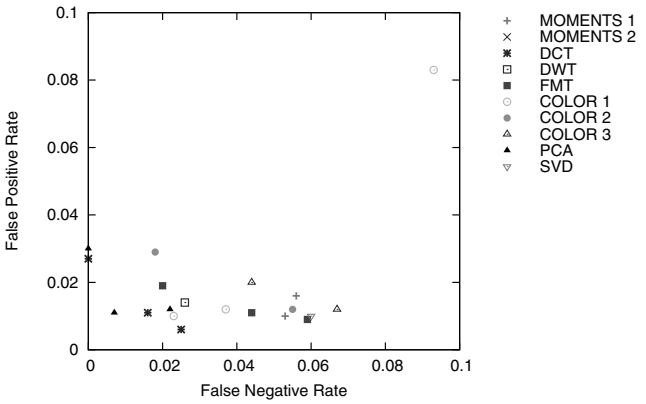


Figure 7: Feature Test – kd-tree representation with small images $[0, 0.1]$

For small images, nearly every feature can be used to detect copy-move forgeries with a kd-tree (Figure 6b). Only MOMENTS 2 and COLOR 3 showed difficulties in a single small image. Upon closer examination, one can notice that the error rates of the small images exhibit a very low false positive rate for all methods (Figure 7). Tables 1b and 1a show the overall average error rates. For the features, FMT has the lowest and, hence, best average error rates when a kd-tree representation is used. On average the F_{P} rate is lower with lexicographic sort. Conversely, the F_{N} rate is typically remarkably better when using a kd-tree representation.

4.1 Geometric Transformation

We also tested the CMFD features on two geometric transformations: scaling and rotation. The copied region in the *Tree* image was rotated up to $20°$ and scaled up to 120% (see Figure 8). The methods SVD, MOMENTS 2 and COLOR 3 could not be tested against geometric transformations due to their high error rates on the untransformed images. We computed the presented results with a kd-tree, since we observed that lexicographic sorting is too sensitive to the transformations and therefore did not produce reliable results.

	F_N	F_P	avg
Color 1	0.515	0.494	0.505
Color 2	0.501	0.498	0.499
Color 3	0.502	0.361	0.431
DCT	0.547	0.273	0.410
DWT	0.688	0.148	0.418
FMT	0.813	0.001	0.407
Moments 1	0.841	0.000	0.421
Moments 2	0.821	0.051	0.436
PCA	0.816	0.084	0.450
SVD	0.830	0.001	0.415
	0.687	0.191	0.439

(a) Lexicographic sort

	F_N	F_P	avg
Color 1	0.029	0.422	0.225
Color 2	0.049	0.483	0.266
Color 3	0.072	0.399	0.235
DCT	0.028	0.227	0.128
DWT	0.071	0.398	0.234
FMT	0.026	0.073	0.050
Moments 1	0.093	0.293	0.193
Moments 2	0.551	0.308	0.429
PCA	0.035	0.163	0.099
SVD	0.380	0.248	0.314
	0.133	0.302	0.217

(b) Kd-tree sorting

Table 1: Overall error rates of the feature test with lexicographic sort on the left side and kd-tree representation on the right side, avg = $(F_N + F_P)/2$.

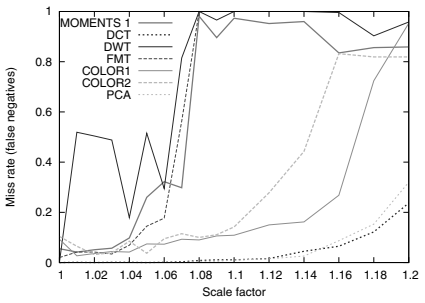
(a) Scaling with 120%

(b) Rotation with 20°

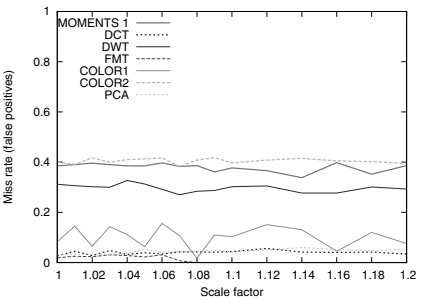
Figure 8: Geometric Transformations adapted to the *Tree* image

Figure 9 shows that the most robust features against scaling were PCA and DCT. They reliably detected copied regions that had been scaled up to 120%. The color based features COLOR 1 and COLOR 2 detected such forgeries up to a scaling of 116% and 114%, respectively. However, it was difficult to distinguish the results of COLOR 2 from the background as the false positive rate was very high at about 0.4. The same problem occured with the features of the MOMENTS 1 method. The false positive rate of DWT was slightly better, but clearly failed to detect anything with scaling more than 106%. Similarly, FMT started degenerating at a scale change of about 106%.

FMT could detect regions rotated up to 3° (see Figure 10). This is not surprising, since the FMT features in [BSM09] are only rotation invariant when a brute-force search over all possible feature vector shifts is done, which we left out. Nearly the same result occured for the DWT and MOMENTS 1 features. As with scaling, the color based features gave an average performance. The best features were once again the ones based on PCA and

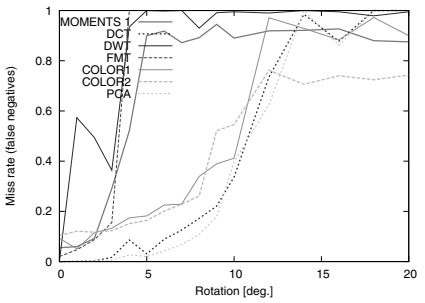


(a) False negative rate

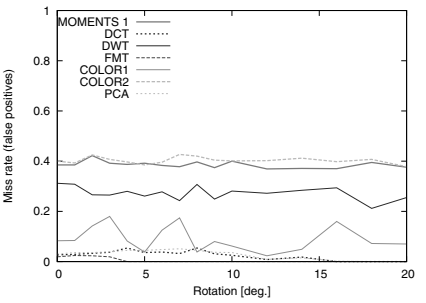


(b) False positive rate

Figure 9: Scale test – kd-tree sorting



(a) False negative rate



(b) False positive rate

Figure 10: Rotation test – kd-tree sorting

DCT. Note that the results on rotational and scaling invariance are additionally hindered by the use of shift vectors in the verification step. With the shift vectors, a scaled or rotated copied region is decomposed in multiple clusters of similar shift vectors, which flattens the distribution of shift vectors over the image.

Consequently, two problems have to be solved to gain better invariance against geometric transformations. On one hand the feature vector itself has to be robust against geometric transformations and on the other hand a different verification method which is not based on using the same shift vectors has to be established.

5 Conclusions

Copy-move forgery detection is a very active field in image forensics, with a large number of proposed features. We believe that the field has matured enough for a thorough evaluation of the advantages and disadvantages of different approaches. In this work, the

features of ten methods were examined on a diverse set of benchmark images. The dataset contains images and copy-moved regions of varying size and texture. We set up a common pipeline for copy-move forgery detection methods and varied intermediate steps, most notably the sorting criterion. We found that, in general, lexicographic sorting yields a lower false positive rate and kd-trees a lower false negative rate. Besides this, lexicographic sorting exhibited severe problems when geometric transformations were applied to the copied region. FMT features showed a very good overall performance. Under geometric transformations, PCA and DCT exhibited remarkably strong results when using same shift vectors as verification criterion.

As part of our future work, we will continue to examine copy-move features under different benchmark conditions. We believe that these findings can be used to create ensembles of copy-move algorithms that are explicitly designed for robustness towards a wide range of postprocessing steps. At the same time, this could help to further reduce the false positive rate, which we identified currently as a major problem in identifying copied regions.

References

[BNO+07] M. K. Bashar, K. Noda, N. Ohnishi, H. Kudo, T. Matsumoto, and Y. Takeuchi. Wavelet-Based Multiresolution Features for Detecting Duplications in Images. In *Conference on Machine Vision Application*, pages 264–267, 2007.

[BSM09] S. Bayram, H. Sencar, and N. Memon. An Efficient and Robust Method for Detecting Copy-Move Forgery. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1053–1056, 2009.

[BSN09] S. Bravo-Solorio and A.K. Nandi. Passive Forensic Method for Detecting Duplicated Regions Affected by Reflection, Rotation and Scaling. In *European Signal Processing Conference*, 2009.

[Far09a] H. Farid. A Survey of Image Forgery Detection. *Signal Processing Magazine*, 26(2):16–25, March 2009.

[Far09b] H. Farid. Exposing Digital Forgeries from JPEG Ghosts. *IEEE Transactions on Information Forensics and Security*, 1(4):154–160, 2009.

[FSL03] J. Fridrich, D. Soukal, and J. Lukáš. Detection of Copy-Move Forgery in Digital Images. In *Proceedings of Digital Forensic Research Workshop*, 2003.

[GC08] A. Gallagher and T. Chen. Image Authentication by Detecting Traces of Demosaicing. In *Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2008.

[HC07] Y. Hsu and S. Chang. Image Splicing Detection using Camera Response Function Consistency and Automatic Segmentation. In *International Conference on Multimedia and Expo*, pages 28–31, 2007.

[HGZ08] H. Huang, W. Guo, and Y. Zhang. Detection of Copy-Move Forgery in Digital Images Using SIFT Algorithm. In *Computational Intelligence and Industrial Application. Pacific-Asia Workshop on*, volume 2, pages 272–276, 2008.

[HLWT06] J. He, Z. Lin, L. Wang, and X. Tang. Detecting Doctored JPEG Images Via DCT Coefficient Analysis. In *European Conference on Computer Vision*, volume 3, pages 423–435, 2006.

[JF05] M. Johnson and H. Farid. Exposing Digital Forgeries by Detecting Inconsistencies in Lighting. In *Workshop on Multimedia and Security*, pages 1–10, 2005.

[JF06] Micah K. Johnson and Hany Farid. Exposing Digital Forgeries through Chromatic Aberration. In *Workshop on Multimedia and Security*, pages 48–55, 2006.

[JF07] M. Johnson and H. Farid. Exposing Digital Forgeries through Specular Highlights on the Eye. In *International Workshop on Information Hiding*, pages 311–325, 2007.

[KW08] X. Kang and S. Wei. Identifying Tampered Regions Using Singular Value Decomposition in Digital Image Forensics. In *International Conference on Computer Science and Software Engineering*, volume 3, pages 926–930, 2008.

[LE07] J. Lalonde and A. Efros. Using Color Compatibility for Assessing Image Realism. In *IEEE International Conference on Computer Vision*, 2007.

[LFG06] J. Lukáš, J. Fridrich, and M. Goljan. Digital Camera Identification From Sensor Pattern Noise. *Information Forensics and Security*, 1(2):205–214, June 2006.

[LHQ06] W. Luo, J. Huang, and G. Qiu. Robust Detection of Region-Duplication Forgery in Digital Images. In *International Conference on Pattern Recognition*, volume 4, pages 746–749, 2006.

[LWK09] H. Lin, C. Wang, and Y. Kao. Fast Copy-Move Forgery Detection. *WSEAS Transactions on Signal Processing*, 5(5):188–197, 2009.

[LWTS07] Guohui Li, Qiong Wu, Dan Tu, and Shaojie Sun. A Sorted Neighborhood Approach for Detecting Duplicated Regions in Image Forgeries Based on DWT and SVD. In *IEEE International Conference on Multimedia and Expo*, pages 1750–1753, 2007.

[MS07] B. Mahdian and S. Saic. Detection of Copy-Move Forgery using a Method Based on Blur Moment Invariants. *Forensic Science International*, 171(2):180–189, December 2007.

[NCLS06] T. Ng, S. Chang, C. Lin, and Q. Sun. Passive-Blind Image Forensics. In *Multimedia Security Technologies for Digital Rights*, chapter 15, pages 383–412. Academic Press, 2006.

[PF04] A.C. Popescu and H. Farid. Exposing Digital Forgeries by Detecting Duplicated Image Regions. Technical Report TR2004-515, Department of Computer Science, Dartmouth College, 2004.

[PF05] A. Popescu and H. Farid. Exposing Digital Forgeries by Detecting Traces of Resampling. *Signal Processing*, 53(2):758–767, February 2005.

[SM08] H. Sencar and N. Memon. Overview of State-of-the-art in Digital Image Forensics. *Algorithms, Architectures and Information Systems Security*, pages 325–344, 2008.

[WLZ+09] J. Wang, G. Liu, Z. Zhang, Y. Dai, and Z. Wang. Fast and Robust Forensics for Image Region-Duplication Forgery. *Acta Automatica Sinica*, 2009.

Lineare Zeilen- und Spaltenprädiktoren zur Erkennung von Bildskalierungen

Matthias Kirchner

Technische Universität Dresden, Fakultät Informatik, 01062 Dresden
matthias.kirchner@inf.tu-dresden.de

Abstract: Geometrische Transformationen von Bildern oder Bildteilen stellen eine wichtige Klasse von Bildmanipulationen dar. In der Literatur diskutierte bildforensische Ansätze zur Erkennung solcher Operationen basieren in der Regel auf periodischen Interpolationsartefakten, die in einem durch lineare Filterung erhaltenen Differenzsignal gemessen werden. Dieser Aufsatz beschreibt einen alternativen Ansatz zur Detektion von Skalierungen. Dabei werden periodische Artefakte unter Ausnutzung der typischen Struktur skalierter Bilder in einer Folge von linearen Prädiktorkoeffizienten gemessen. Experimentelle Ergebnisse zeigen anhand einer großen Bilddatenbank, dass derartige Zeilen- und Spaltenprädiktoren insbesondere Bildverkleinerungen zuverlässiger erkennen können als bisherige Verfahren.

1 Einleitung

Digitale Bildforensik bezeichnet die Wissenschaft zur forensischen Analyse digitaler Bilddaten mit dem Ziel Aussagen zu deren Integrität und Herkunft zu machen [BFGK09]. Bedingt durch die Omnipräsenz digitaler Bilder und der damit einhergegangenen Entwicklung mächtiger Bildbearbeitungsprogramme haben bildforensische Methoden in den letzten Jahren zunehmend an Bedeutung gewonnen. Zahlreiche in der Literatur diskutierte Ansätze bieten inzwischen ein breites Spektrum verschiedenster Verfahren zur Erkennung von Bildmanipulationen oder zur Bestimmung des Aufnahmegerätes [SM08, Far09, u. a.]. Im Gegensatz zu kryptographischen Methoden oder digitalen Wasserzeichen ermöglicht die digitale Bildforensik eine *ex post* Perspektive auf die Analyse digitaler Bilddaten, d. h. ein unmittelbarer Zugriff auf das Originalbild oder Ursprungsgerät ist nicht erforderlich. Während dieser vermeintliche Vorteil nicht ohne Konsequenzen auf die (gerichtliche) Verwertbarkeit forensischer Indizien bleibt [Kno08, Kno09], kann andererseits nicht davon ausgegangen werden, dass in naher Zukunft Digitalkameras mit fälschungssicheren Signaturmodulen zu einem breiten Einsatz kommen. Im Gegenteil kann vermutet werden, dass die praktische Relevanz der digitalen Bildforensik weiter zunimmt, wenn es gelingt robuste und im großen Maßstab getestete [GB10] Verfahren (weiter) zu entwickeln.

Dieser Aufsatz widmet sich der Erkennung von geometrischen Bildtransformationen, genauer von Bildskalierungen. Dabei handelt es sich um eine besonders wichtige Klasse von Bildmanipulationen. So kann die Änderung der Bildgröße als eine Form plausibler Nachbearbeitung aufgefasst werden, die zunächst ohne eigentliche Auswirkung auf die

Integrität des Bildes bleibt. Dass jedoch jegliches Wissen über die Bearbeitungshistorie eines digitalen Bildes von Bedeutung sein kann, wenn es um dessen forensische Analyse geht, liegt auf der Hand. Insbesondere die Erkennung von Bildverkleinerungen ist von großem Interesse, da diese auch als eine mögliche Form des universellen Angriffs zur Vertuschung vorangegangener Bildmanipulationen betrachtet werden kann [BFGK09]. Darüber hinaus stellen geometrische Transformationen von Bildern oder Bildteilen einen wichtigen Baustein von komplexeren Bildmontagen dar, etwa beim Anpassen der Größe eines einzufügenden Bildausschnittes.

Die in der Literatur diskutierten Ansätze zur Erkennung geometrischer Bildtransformationen beruhen im Allgemeinen auf der Existenz periodischer Artefakte in interpolierten Bildern. Interpolation ist immer dann notwendig, wenn transformierte Koordinaten nicht auf dem diskreten kartesischen Bildgitter des Ausgangsbildes liegen. Wie in den grundlegenden Arbeiten von Popescu und Farid [PF05] und Gallagher [Gal05] gezeigt wurde, treten in mit linearen Basisfunktionen interpolierten Bildern periodische lineare Abhängigkeiten zwischen benachbarten Pixeln auf. Für eine Detektion dieser Artefakte kommt bei praktisch allen bekannten Verfahren explizit oder implizit eine Form von linearem Prädiktor zum Einsatz, der einzelne Abtastwerte aus deren unmittelbarer Nachbarschaft schätzt und in dessen Residuen (d. h. Schätzfehler) Interpolationspuren besonders gut messbar sind.

Das Ziel dieses Aufsatzes ist es zu zeigen, dass derartige periodische Artefakte auch in den Prädiktorgewichten selbst gemessen werden können. Dazu wird nach einem knappen Überblick über Interpolationsartefakte in skalierten Bildern im Abschnitt 2 ein geeignetes Verfahren zu deren Erkennung im Abschnitt 3 vorgeschlagen. Anhand umfangreicher Experimente wird anschließend im Abschnitt 4 dessen Eignung zur zuverlässigen Erkennung von Änderungen der Bildgröße demonstriert, bevor Abschnitt 5 den Aufsatz mit einigen zusammenfassenden Bemerkungen beschließt.

2 Interpolationsartefakte in geometrisch transformierten Bildern

Da in der Praxis verwendete Interpolationsbasisfunktionen in der Regel separierbar sind, beschränken sich folgende Betrachtungen zunächst auf den eindimensionalen Fall. Hier lässt sich die Interpolation eines Signals s an einer reellwertigen Position $x \in \mathbb{R}$ als Linearkombination von Signalwerten an ganzzahligen Positionen $\chi' \in \mathbb{Z}$ beschreiben,

$$s(x) = \sum_{\chi'=-\infty}^{\infty} h(x - \chi')s(\chi') \,, \tag{1}$$

wobei die Basisfunktion $h\colon \mathbb{R} \to \mathbb{R}$ die zu verwendenden skalaren Gewichte bestimmt. Für eine Skalierung um einen Faktor $1/\omega$, $\omega > 0$, ergeben sich die transformierten Koordinaten zu $\forall \chi \in \mathbb{Z}\ x = \omega\chi$. Für Vergrößerungen gilt $\omega < 1$ und entsprechend für Verkleinerungen $\omega > 1$.

Es lässt sich leicht veranschaulichen, dass eine Skalierung um den Faktor $1/\omega = p/q$, mit p und q teilerfremd, $p \perp q$, zu periodischen Artefakten mit der Periodendauer p führt. Dazu sei angemerkt, dass die in Gl. (1) verwendeten Werte der Basisfunktion h für eine beliebige

transformierte Koordinate x ohne Einschränkung der Allgemeinheit in Abhängigkeit der Distanz $\delta_x = x - \lfloor x \rfloor$ geschrieben werden können:

$$\sum_{\chi'=-\infty}^{\infty} h(x-\chi')s(\chi') = \sum_{l=-\infty}^{\infty} h\left(l-\delta_x\right) s(\lfloor x \rfloor + l)\,. \tag{2}$$

Aus Gl. (2) folgt, dass zwei zu interpolierende Abtastwerte $s(x_1)$ und $s(x_2)$ immer dann in gleicher Weise aus den jeweilig umliegenden Originalwerten berechnet werden, wenn gilt: $\delta_{x_1} = \delta_{x_2}$. Für skalierte diskrete Koordinaten $x = \omega\chi$, $\omega = q/p$, $p \perp q$ ergibt sich aus

$$x = \frac{q}{p}\left(\frac{pm}{q} + n\right)\,, \qquad \text{mit } m \in \mathbb{Z} \text{ und } n \in \{0, 1, \ldots, p-1\}\,,$$

eine Periodizität von δ_x mit p:

$$\delta_x = m + \frac{qn}{p} - \left\lfloor m + \frac{qn}{p} \right\rfloor = \frac{qn}{p} - \left\lfloor \frac{qn}{p} \right\rfloor\,.$$

Die Periodizität der Interpolationsgewichte führt dazu, dass in skalierten Signalen periodische Korrelationen zwischen benachbarten Abtastwerten auftreten. Die Ausprägung der Abhängigkeit hängt dabei im Allgemeinen von δ_x ab. Unter der Annahme einer symmetrischen Autokorrelation im Orginalsignal und einer symmetrischen Basisfunktion korreliert ein Abtastwert mit $0 < \delta_x < 0{,}5$ stärker mit seinen linken Nachbarn (und umgekehrt für $0{,}5 < \delta_x < 1$). Für $\delta_x = 0$ und $\delta_x = 0{,}5$ sind im gleichen Maße Abhängigkeiten zwischen linken und rechten Nachbarn vorhanden. Da δ_x innerhalb einer Periode monoton steigt, lässt sich in skalierten Signalen bezüglich der Nachbarn von Abtastwerten ein entsprechender Übergang von einer linkslastigen zu einer rechtslastigen Korrelation beobachten.

Die Existenz solcher charakteristischen Abhängigkeiten wurde erstmals von Popescu und Farid [PF05] als Indiz für geometrische Transformationen diskutiert. Zur Detektion schlugen sie einen linearen Prädiktor vor, der einen Abtastwert als Linearkombination seiner Nachbarn beschreibt. Die Residuen e geben dann Aufschluss über Stärke und Ausprägung der linearen Abhängigkeit,

$$e(x) = e(\omega\chi) = s(\omega\chi) - \sum_{k=-K}^{K} \alpha_k s(\omega\chi + \omega k) \qquad (\alpha_0 \equiv 0)\,. \tag{3}$$

Betragsmäßig große Residuen entsprechen dabei einem geringen Grad linearer Abhängigkeit. Im Falle von interpolierten Signalen weisen die Residuen eine mehr oder weniger starke periodische Struktur auf.

Eine zentrale Frage ist hierbei die Parametrisierung des Prädiktors in Form der Koeffizienten $\boldsymbol{\alpha}$. Für die nachfolgenden Betrachtungen ist besonders hervorzuheben, dass in Gl. (3) die gleichen Prädiktorkoeffizienten für *jeden* Abtastwert angewandt werden, was ebenso als Faltung bzw. Filterung mit einem festen Kernel interpretiert werden kann. Popescu und Farid schlagen eine, in ein Expectation/Maximization-Verfahren eingebettete, *weighted least squares* (WLS) Prozedur zur Schätzung der Koeffizienten vor. Gewählt werden die

Koeffizienten, die den gewichteten quadratischen Prädiktorfehler minimieren, wobei die Gewichte in einem iterativen Prozess bestimmt werden [PF05].

Da die Koeffizienten $\boldsymbol{\alpha}$ aus dem gesamten Signal geschätzt werden, können sie als ein Maß der *mittleren* Abhängigkeit zwischen benachbarten Abtastwerten verstanden werden. Es überrascht demnach nicht, dass in späteren Experimenten [Kir08] eine gewisse Unabhängigkeit von den tatsächlichen Transformationsparametern berichtet wurde. Gleichermaßen lässt sich in diesem Zusammenhang vermutlich die beobachtete Symmetrie der Koeffizienten ($\alpha_k \approx \alpha_{-k}$) erkären.

Ein weiterer Ansatz zur Erkennung von Interpolationsartefakten liegt in der Analyse der Varianzen der (partiellen) Ableitungen von interpolierten Signalen begründet. Wie von Gallagher gezeigt [Gal05], und später von Mahdian und Saic verallgemeinert [MS08], weisen diese eine periodische Struktur auf. Da typische Implementierungen von diskreten Ableitungsoperatoren als Faltung mit einem symmetrischen Kernel beschrieben werden können, lassen sich aufschlussreiche Parallelen zu den prädiktorbasierten Ansätzen herstellen [Kir08].

Letztlich bleibt festzuhalten, dass alle bekannten Detektoren die periodischen Interpolationsartefakte in einer Form von Differenzsignal messen, welches durch Filterung des zu analysierenden Signals mit einem festen Kernel erhalten wird. Im Folgenden soll diese Herangehensweise variiert werden, indem nicht ein einzelner Prädiktor zum Einsatz kommt, sondern eine, an die typische Struktur skalierter Bilder angelehnte, Folge von Prädiktoren. Die periodischen Artefakte werden dementsprechend nicht in den Residuen, sondern in den Prädiktorkoeffizienten selbst gemessen.

3 Analyse von Prädiktorkoeffizienten

Aus der Separierbarkeit typischer Basisfunktionen folgt, dass in einem skalierten Bild alle Pixel einer Zeile in gleicher Weise linear mit ihren vertikalen Nachbarn korrelieren (entsprechend für Spalten und horizontale Nachbarn).[1] Schreibt man die Pixel der i-ten Zeile als Vektor $\boldsymbol{z}^{(i)}$, lässt sich das folgende lineare Modell aufstellen:

$$\boldsymbol{z}^{(i)} - \boldsymbol{\eta}^{(i)} = \sum_{\substack{|k|\leq K \\ k\neq 0}} \alpha_k^{(i)} \left(\boldsymbol{z}^{(i-k)} - \boldsymbol{\eta}^{(i-k)} \right) + \boldsymbol{\epsilon}^{(i)} \,, \tag{4}$$

wobei $\boldsymbol{\eta}$ Rundungsfehler bezeichnet und $\boldsymbol{\epsilon}$ ein Fehlerterm zur Beschreibung des „Modellfehlers“ ist. Für Vergrößerungen ($\omega < 1$) entfällt dieser bei einer geeigneten Wahl von K, d. h. jedes Pixel lässt sich (bis auf Rundungsfehler) vollständig als Linearkombination seiner vertikalen Nachbarn ausdrücken. Für Verkleinerungen ist dies im Allgemeinen nicht der Fall und hängt vom Support der Basisfunktion sowie dem Verkleinerungsfaktor ab.

Die zeilenabhängigen Koeffizienten $\boldsymbol{\alpha}^{(i)}$ spiegeln dabei die tatsächlich vorliegende lineare Abhängigkeit zwischen benachbarten Pixeln wider und markieren gleichzeitig den Unter-

[1] Aus Gründen der Lesbarkeit wird im Folgenden nur Bezug auf Zeilen und deren vertikale Nachbarn genommen. Eine Übertragung auf Spalten und horizontale Nachbarn ist jedoch jederzeit problemlos möglich.

schied bei der Modellierung von Interpolationsartefakten verglichen mit bisherigen Arbeiten. Das Modell in Gl. (4) erlaubt eine explizitere Beschreibung der Interpolationsartefakte als Popescu und Farids globaler Prädiktor. Im Gegensatz zu einer impliziten Modellierung der Periodizität als Abweichung vom „mittleren“ Prädiktor erlaubt eine Folge von Zeilenprädiktoren eine Messung der periodischen Artefakte anhand der Prädiktorkoeffizienten selbst. Zur Schätzung der Koeffizienten kann dabei auf Standard-Regressionsverfahren zurückgegriffen werden, wobei pro Zeile eine entsprechende Regression durchgeführt werden muss. Eine konkrete Umsetzung soll im Folgenden diskutiert werden.

3.1 Idealfall: Prädiktion mit Total Least Squares

Im Falle eines vernachlässigbaren Modellfehlers, $\boldsymbol{\epsilon}^{(i)} = \mathbf{0}$, lassen sich Pixel einer Zeile, abgesehen von Rundungsfehlern, vollständig als Linearkombination benachbarter Zeilen beschreiben. Da in Gl. (4) sowohl die abhängige Variable als auch die unabhängigen Variablen durch Rundungsfehler verfälscht sind, bietet sich eine *total least squares* (TLS) Schätzung [VV91] der Prädiktorkoeffizienten $\boldsymbol{\alpha}^{(i)}$ an.

Schreibt man Gl. (4) zu

$$\begin{aligned} \boldsymbol{z}^{(i)} - \boldsymbol{\eta}^{(i)} &= \left(\mathbf{Z}^{(i)} - \mathbf{N}^{(i)}\right) \cdot \boldsymbol{\alpha}^{(i)}\,, \\ \text{mit} \quad \mathbf{Z}^{(i)} &= \left[\boldsymbol{z}^{(i-K)}, \ldots, \boldsymbol{z}^{(i-1)}, \boldsymbol{z}^{(i+1)}, \ldots, \boldsymbol{z}^{(i+K)}\right] \\ \text{und} \quad \mathbf{N}^{(i)} &= \left[\boldsymbol{\eta}^{(i-K)}, \ldots, \boldsymbol{\eta}^{(i-1)}, \boldsymbol{\eta}^{(i+1)}, \ldots, \boldsymbol{\eta}^{(i+K)}\right]\,, \end{aligned}$$

so findet ein TLS-Schätzer den Koeffizientenvektor $\hat{\boldsymbol{\alpha}}^{(i)}$, für den gilt:

$$\hat{\boldsymbol{z}}^{(i)} = \hat{\mathbf{Z}}^{(i)} \cdot \hat{\boldsymbol{\alpha}}^{(i)} \quad \text{und} \quad \left\| \left[\mathbf{Z}^{(i)}, \boldsymbol{z}^{(i)}\right] - \left[\hat{\mathbf{Z}}^{(i)}, \hat{\boldsymbol{z}}^{(i)}\right] \right\|_F \rightarrow \min\,,$$

wobei die Lösung aus der Singulärwertzerlegung (SVD) der Matrix $\left[\mathbf{Z}^{(i)}, \boldsymbol{z}^{(i)}\right]$ folgt.

Bei einer geeigneten Wahl der Nachbarschaftsgröße K erlaubt das TLS-Verfahren eine vergleichsweise genaue Schätzung der tatsächlich vorhandenen linearen Abhängigkeiten zwischen benachbarten Pixeln. Dies zeigt die Abbildung 1 für ein mit bilinearer Interpolation auf 150 % vergrößertes Bild ($\omega = 2/3$) exemplarisch für eine Folge von Zeilenprädiktoren (bei einer Nachbarschaftsgröße von $K = 3$). Die geschätzten Koeffizienten lassen interpolationsbedingt eine deutliche periodische Struktur erkennen und stimmen mit den theoretisch zu erwartenden (im Mittel) überein.

Obwohl die TLS-Schätzung prinzipiell zu äußerst zuverlässigen Ergebnissen führen kann, besteht im Allgemeinen das Problem, dass die Schätzung bei einem zu großen Modellfehler instabil wird. Die bei TLS gemachte Annahme der Homoskedastizität wird mit einem nicht durch lineare Abhängigkeiten erklärbaren Anteil verletzt. Dies kann, wie schon angedeutet, bei Verkleinerungen leicht der Fall sein. Jedoch kann auch die falsche Wahl der Nachbarschaftsgröße zu ungenauen bzw. instabilen Schätzungen führen.[2] Zwar lässt

[2] Dazu sei angemerkt, dass die korrekte bzw. optimale Nachbarschaftsgröße in forensischen Anwendungen

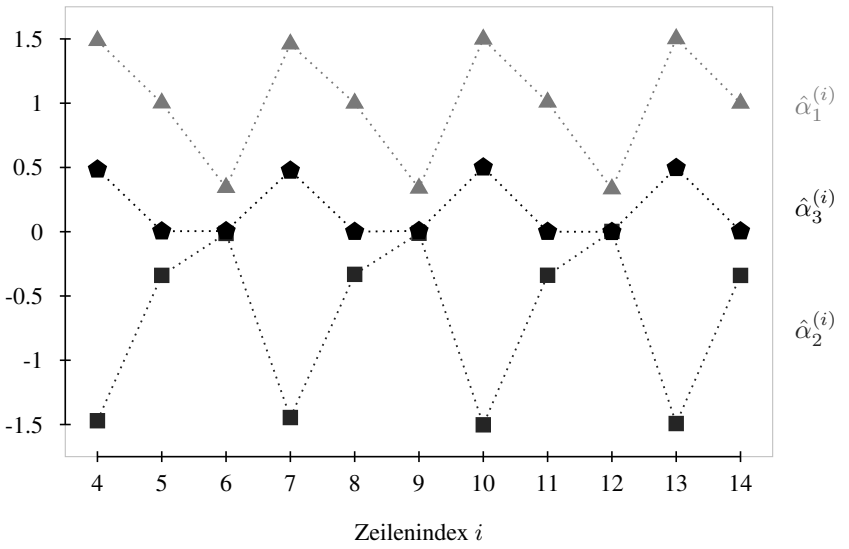


Abbildung 1: Mit TLS geschätze Zeilenprädiktorkoeffizienten für 11 aufeinanderfolgende Zeilen eines mit bilinearer Interpolation auf 150 % vergrößerten Bildes ($K = 3$, Bildgröße 350×350 Pixel). Die theoretisch zu erwartenden Koeffizientenfolgen sind $\{3/2, 1, 1/3\}$ für α_1, $\{-3/2, -1/3, 0\}$ für α_2 und $\{1/2, 0, 0\}$ für α_3.

sich in einem gewissen Rahmen durch Regularisierung oder Skalierung der Fehlerterme gegensteuern [VV91], in der Praxis erwiesen sich jedoch *weighted least squares* (WLS) Schätzer als geeigneter, wenn es lediglich um eine Detektion von Bildskalierungen geht.

3.2 Prädiktion mit Weighted Least Squares

Wie im Abschnitt 2 diskutiert, sind interpolierte Abtastwerte je nach ihrer relativen Position δ_x stärker mit ihren linken bzw. rechten Nachbarn korreliert. Für eine Detektion von Bildskalierungen ist es damit unter Umständen nicht zwangsläufig notwendig die exakten, durch Gl. (4) bestimmten, Koeffizienten $\boldsymbol{\alpha}^{(i)}$ zu ermitteln. Vielmehr kann es genügen den relativen Einfluss linker und rechter Nachbarn zu betrachten.

Ein vereinfachtes lineares Modell beschreibt entsprechend alle Pixel einer Zeile als Linearkombination benachbarter Zeilen, überlagert mit einem additiven Fehlerterm $\boldsymbol{\varepsilon}$,

$$\boldsymbol{z}^{(i)} = \mathbf{Z}^{(i)} \cdot \boldsymbol{\beta}^{(i)} + \boldsymbol{\varepsilon}^{(i)} \,. \tag{5}$$

Zur Bestimmung der Koeffizienten $\hat{\boldsymbol{\beta}}^{(i)}$ soll im Folgenden ein WLS-Schätzer verwendet werden, für den gilt:

$$\hat{\boldsymbol{z}}^{(i)} = \mathbf{Z}^{(i)} \cdot \hat{\boldsymbol{\beta}}^{(i)} \quad \text{und} \quad \left\| \boldsymbol{w}^{(i)} \left(\boldsymbol{z}^{(i)} - \hat{\boldsymbol{z}}^{(i)} \right) \right\| \to \min \,,$$

im Allgemeinen nicht bekannt ist.

wobei die Gewichte $w_j^{(i)}$ aus der Varianz σ_j^2 der umliegenden $2K$ Nachbarn berechnet werden, $w_j^{(i)} \propto 1/(c+\sigma_j^2)$, $c > 0$.

Die relative Abhängigkeit von linken und rechten Nachbarn lässt sich am stärksten an den Koeffizienten $\beta_{-1}^{(i)}$ und $\beta_1^{(i)}$ ablesen, die den beiden direkten Nachbarpixeln entsprechen. Praktische Untersuchungen haben gezeigt, dass eine Analyse der Differenzen d_i,

$$d_i = \beta_{-1}^{(i)} - \beta_1^{(i)} \ , \tag{6}$$

in besonders guten Detektionsergebnissen resultiert.

Bei einer Betrachtung des Verlaufs der Prädiktorkoeffizienten (bzw. deren Differenzen) über mehrere Zeilen hinweg ist in skalierten Bildern wiederum die bekannte periodische Struktur zu erwarten. Für eine Erkennung von Bildskalierungen ist dabei die Existenz einer ausgeprägten spektralen Komponente ausschlaggebend, die im Leistungsdichtespektrum $S(f)$ als starker Peak detektiert werden kann. Erwähnt sei, dass eine Spektralanalyse der Differenzen d_i gegenüber der direkten Untersuchung der Prädiktorkoeffizienten den Vorteil hat, eventuell störende niederfrequente Anteile zu unterdrücken. Der Einsatz von Dämpfungsfunktionen wie bei Popescu und Farid [PF05] oder Gallagher [Gal05] ist damit nicht notwendig.

Vor allem im Fall von Verkleinerungen kann jedoch nicht erwartet werden, dass die erhaltenen Schätzwerte $\hat{\boldsymbol{\beta}}^{(i)}$ eine perfekte Periodizität aufweisen. Aus diesem Grund bietet sich ein robuster Schätzer des Leistungsdichtespektrums an. Ahdesmäki et al. [ALP+05] schlagen diesbezüglich ein auf Spearmans Rangkorrelationskoeffizienten basierendes Verfahren vor, das auch im Folgenden verwendet werden soll.

Abschließend sei angemerkt, dass ein auf Zeilenprädiktoren basierender Detektor mit vergleichbar moderatem Ressourcen-Aufwand auskommt. Zwar steigt die Zahl der durchzuführenden Regressionen mit der Anzahl der Zeilen des zu analysierenden Bildes, jedoch werden pro Regression weniger Messwerte einbezogen als bei Popescu und Farids globalem Schätzer (der schnell zu hohe Speicheranforderungen für Bilder von praktischer Größe stellt). Gegenüber Detektoren auf Basis fester linearer Filter [Kir08] oder Ableitungsoperatoren [Gal05] bleibt dennoch ein stark erhöhter Rechenaufwand festzuhalten, welcher jedoch, wie im folgenden Abschnitt zu sehen sein wird, durch eine erhöhte Zuverlässigkeit ausgeglichen wird.

4 Experimenteller Teil

Die praktische Eignung der beschriebenen Zeilen- bzw. Spaltenprädiktoren zur Erkennung von Bildskalierungen soll anhand eines Teils der „Dresden Image Database“ [GB10] demonstriert werden. Etwa 1100 unkomprimierte Aufnahmen verschiedener Digitalkameras wurden zunächst in Graustufenbilder konvertiert und dann mit dem `convert`-Kommando von ImageMagick unter Verwendung von bilinearer Interpolation und kubischer Spline-Interpolation jeweils um verschiedene Faktoren vergrößert und verkleinert. Zur Analyse wurde anschließend jeweils ein Ausschnitt der Größe 512×512 Pixel aus der Mitte des

skalierten Bildes herangezogen. Die Nachbarschaftsgröße der WLS-Prädiktoren betrug dabei in allen Experimenten $K = 2$.

Als Detektionskriterium wurde das Verhältnis ρ,

$$\rho = \max_f S(f) \,/\, \operatorname*{median}_f S(f)\,,$$

ausgewertet, wobei $S(f)$ das Leistungsdichtespektrum der Koeffizientendifferenzen entsprechend Gl. (6) bezeichnet. Im Falle eines ausgeprägten Peaks im Leistungsdichtespektrum nimmt ρ hohe Werte an. Liegt ρ über einem zu definierenden Schwellwert, $\rho > T$, so wird das zu analysierende Bild als skaliert betrachtet. Da Interpolationsartefakte sowohl mittels Zeilen- als auch Spaltenprädiktoren detektiert werden können, wird die eigentliche Entscheidung anhand des Maximums beider Analyseergebnisse, $\max(\rho_{\text{Zeilen}}, \rho_{\text{Spalten}})$, getroffen.

Für einen Vergleich mit aus der Literatur bekannten Verfahren wurden die Bilder ebenfalls mit Popescu und Farids [PF05] Detektor analysiert, der gemeinhin als am zuverlässigsten betrachtet wird. Wie für die Zeilen- und Spaltenprädiktoren wurde auch hier eine Nachbarschaftsgröße von $K = 2$ gewählt, was einem 5×5 Fenster entspricht. Da Bildvergrößerungen im Allgemeinen als problemlos erkennbar gelten, werden im Folgenden vor allem Ergebnisse für Bildverkleinerungen diskutiert.

Die Abbildung 2 gibt anhand von ROC-Kurven für verschiedene Verkleinerungsfaktoren einen Überblick über typische Detektionsergebnisse. Dabei sind sowohl für bilineare als auch für kubische Interpolation Resultate aufgeführt. Die Falschpositiv-Raten ergeben sich aus der Analyse der unbearbeiteten Bilder im Testdatensatz.

Die bisher unübertroffene Größe des Testdatensatzes erlaubt eine knappe Reflexion der dargestellten Ergebnisse mit Bezug auf bereits aus der Literatur bekannte Resultate. So bestätigt sich im Allgemeinen die verminderte Detektierbarkeit stärkerer Verkleinerungen. Eine Ausnahme bildet die Verkleinerung auf 70 % mit bilinearer Interpolation, die weniger detektierbare Spuren zu hinterlassen scheint als beispielsweise eine Verkleinerung auf 60 %. Ein Grund könnte in der vergleichsweise langen Periodendauer der Interpolationsartefakte bei der Skalierung auf 70 % liegen. Wie zu erwarten, sind zudem bilinear interpolierte Bilder aufgrund des kleineren Supports der Basisfunktion im Allgemeinen deutlich besser detektierbar als kubisch interpolierte Bilder.

Insgesamt scheint jedoch in unserem Zusammenhang interessanter, dass die Detektion auf Basis der vorgestellten Zeilen- und Spaltenprädiktoren zu mindestens gleichwertigen, meist aber zu besseren Erkennungsleistungen führt. Dies lässt sich insbesondere aus den ROC-Kurven für Verkleinerungen mit kubischer Spline-Interpolation ablesen. Hier sind zum Teil deutliche Zuwächse bei der Detektierbarkeit zu verzeichnen. Nichtsdestotrotz erscheinen die Ergebnisse insgesamt wenig überzeugend, auch (oder gerade) im Vergleich zu aus der Literatur bekannten Ergebnissen. Dazu muss jedoch angemerkt werden, das eine Verkleinerung mit ImageMagick stets auch in einer Nachbearbeitung des Bildes resultiert, die visuell ansprechende Ergebnisbilder zum Ziel hat und damit gleichzeitig zu einer Abschwächung der gesuchten Interpolationsartefakte führt. Dieser sonst vernachlässigte Bearbeitungsschritt mag nun einerseits als verfälschend betrachtet werden. Andererseits wurde in den durchgeführten Experimenten bewusst auf eine praktisch relevante Software

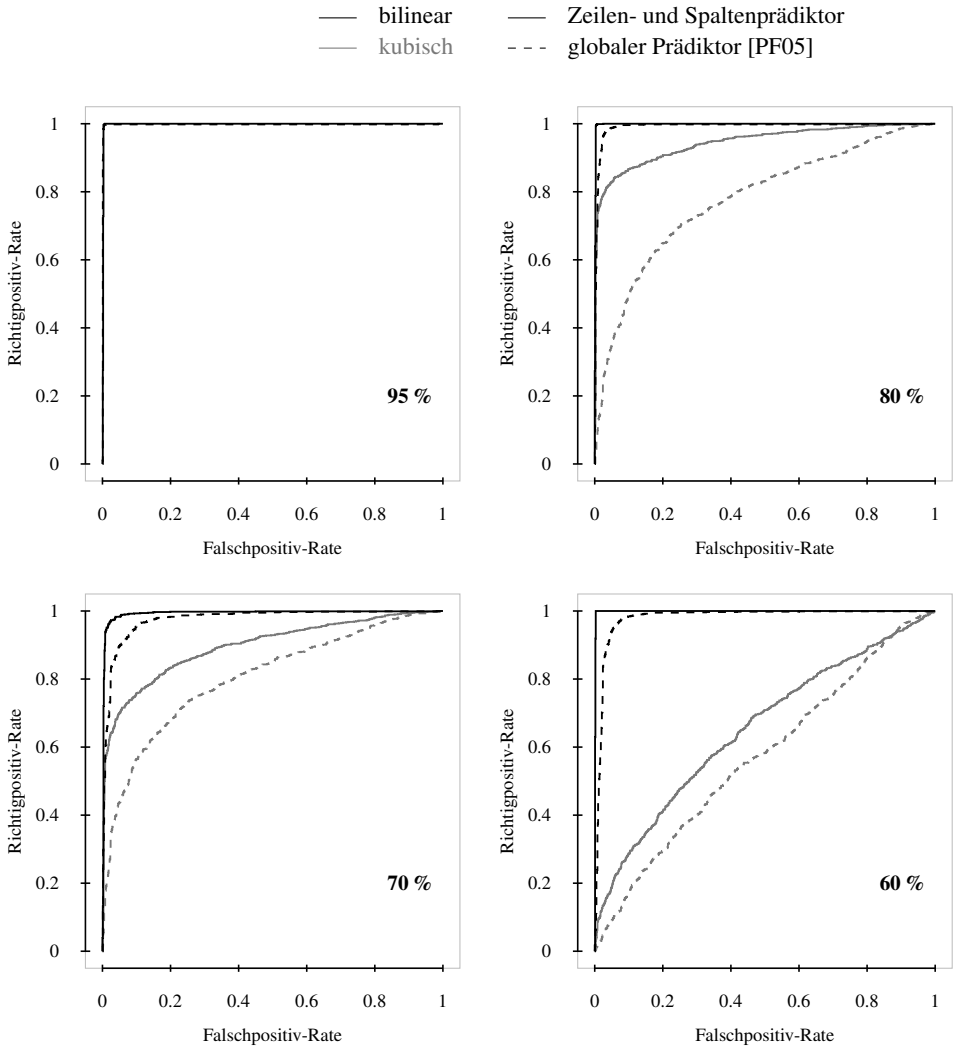


Abbildung 2: ROC-Kurven für die Erkennung von Bildverkleinerung mit bilinearer und kubischer Interpolation. Ergebnisse für die Detektion mit Zeilen- und Spaltenprädiktoren sowie Popescu und Farids [PF05] Verfahren bei einer Verkleinerung auf 95 %, 80 %, 70 % und 60 % der Originalgröße. Eine Analyse anhand von Zeilen- und Spaltenprädiktoren resultiert in jedem der Fälle in einer erhöhten Detektierbarkeit.

zurückgegriffen um möglichst realistische Aussagen zur Erkennbarkeit von Bildverkleinerungen machen zu können.

Nichtsdestotrotz unterstreichen die gezeigten Ergebnisse die prinzipielle Eignung von Zeilen- und Spaltenprädiktoren in Verbindung mit robusten spektralen Schätzern bei der Detektion von Bildskalierungen. Abschließend sei erwähnt, dass größere Nachbarschaften mit $K > 2$ zu vergleichbaren Resultaten führten. Für $K = 1$ musste jedoch eine im Allgemeinen weniger zuverlässige Erkennungsleistung verzeichnet werden.

5 Zusammenfassung und Ausblick

Dieser Aufsatz widmete sich der Erkennung von Bildskalierungen als einer wichtigen Klasse von Bildmanipulationen. Während bisherige Ansätze Interpolationsartefakte in einem durch lineare Filterung erhaltenen Differenzsignal messen, wurde demonstriert, wie diese auch in einer Folge von Prädiktorkoeffizienten detektiert werden können. Dabei spielte die Beobachtung, dass in skalierten Bildern alle Pixel einer Zeile in gleicher Weise von ihren vertikalen Nachbarn abhängen, eine zentrale Rolle (und entsprechend für Spalten). Praktische Tests anhand einer großen Menge von Bildern verdeutlichten, dass eine explizite Modellierung der linearen Abhängigkeiten in skalierten Bildern einen viel versprechenden bildforensischen Ansatz darstellt.

Neben dem direkten Einsatz zur Erkennung von skalierten Bildern oder Bildteilen liegen weitere mögliche Anwendungsszenarien für die diskutierten Zeilen- und Spaltenprädiktoren auf der Hand. So soll in weiteren Untersuchungen geprüft werden, inwieweit anhand der geschätzten Prädiktorkoeffizienten zwischen Bildvergrößerungen und Bildverkleinerungen unterschieden werden kann. Es ist bekannt, dass verschiedene Transformationsparameter zu vergleichbaren periodischen Artefakten im Residuum eines globalen Prädiktors führen können [PF05, Kir08]. Eine Unterscheidung nur anhand der Position ausgeprägter Peaks ist damit nicht möglich. Gleichzeitig ist jedoch zu erwarten, dass unterschiedliche Skalierungsfaktoren in abweichenden Koeffizienten bei der Prädiktion einzelner Zeilen und Spalten resultieren. Eine weitere mögliche Anwendung liegt in der Steganalyse von skalierten Bildern. Hier können zeilen- und spaltenabhängige Prädiktoren unter Umständen bei einer verbesserten Abschätzung des Stegorauschens unterstützen, z. B. [KB08].

Ein Einschränkung des beschriebenen Ansatzes liegt darin, dass eine Verallgemeinerung auf beliebige geometrische Transformationen (etwa der Rotation) schwer oder möglicherweise gar nicht möglich ist. Jedoch stellen Skalierungen, und insbesondere Bildverkleinerungen, den vermutlich wichtigsten Fall von geometrischen Transformationen dar.[3] Bestehen bleibt somit indes die Problematik der zuverlässigen Erkennung starker Bildverkleinerungen bei dem Einsatz von Basisfunktionen jenseits der bilinearen Interpolation. Wie auch die hier gezeigten Ergebnisse trotz sichtlicher Verbesserungen deutlich machen, liegen die erreichbaren Erkennungsraten noch weit von einer praktischen Nützlichkeit entfernt. Es ist eine offene Frage, inwieweit auf periodische Artefakte abzielende Detektoren

[3] Zudem lässt sich die Rotation eines Bildes äußerst zuverlässig mit effizienten Detektoren feststellen [Kir08].

diesbezüglich einen akzeptablen Stand erreichen können, oder letztlich nur eine Kombination einer Vielzahl von Merkmalen zum Erfolg führt.

Literatur

[ALP+05] Miika Ahdesmäki, Harri Lähdesmäki, Ron Pearson, Heikki Huttunen und Olli Yli-Harja. Robust Detection of Periodic Time Series Measured from Biological Systems. *BMC Bioinformatics*, 6(117), 2005.

[BFGK09] Rainer Böhme, Felix Freiling, Thomas Gloe und Matthias Kirchner. Multimedia Forensics is not Computer Forensics. In Zeno J. Geradts, Katrin Y. Franke und Cor. J. Veenman, Hrsg., *Computational Forensics, Third International Workshop, IWCF 2009*, LNCS 5718, Seiten 90–103, Berlin, Heidelberg, 2009. Springer Verlag.

[Far09] Hany Farid. Image Forgery Detection. *IEEE Signal Processing Magazine*, 26(2):16–25, 2009.

[Gal05] Andrew C. Gallagher. Detection of Linear and Cubic Interpolation in JPEG Compressed Images. In *Proceedings of the Second Canadian Conference on Computer and Robot Vision*, Seiten 65–72, 2005.

[GB10] Thomas Gloe und Rainer Böhme. The Dresden Image Database for Benchmarking Digital Image Forensics. In *Proceedings of the 25th Symposium on Applied Computing (ACM SAC 2010)*, Seiten 1585–1591, 2010.

[KB08] Andrew D. Ker und Rainer Böhme. Revisiting Weighted Stego-Image Steganalysis. In Edward J. Delp, Ping Wah Wong, Jana Dittmann und Nasir Memon, Hrsg., *Proceedings of SPIE-IS&T Electronic Imaging: Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, 681905, 2008.

[Kir08] Matthias Kirchner. Fast and Reliable Resampling Detection by Spectral Analysis of Fixed Linear Predictor Residue. In *MM&Sec'08, Proceedings of the Multimedia and Security Workshop 2008*, Seiten 11–20, New York, NY, USA, 2008. ACM Press.

[Kno08] Michael Knopp. Digitalfotos als Beweismittel. *Zeitschrift für Rechtspolitik*, 41(5):156–158, 2008.

[Kno09] Michael Knopp. Rechtliche Perspektiven zur Digitalen Beweisführung. In Stefan Fischer, Erik Maehle und Rüdiger Reischuk, Hrsg., *Informatik 2009*, LNI 154, Seiten 1552–1566. Gesellschaft für Informatik, 2009.

[MS08] Babak Mahdian und Stanislav Saic. Blind authentication using periodic properties of interpolation. *IEEE Transactions on Information Forensics and Security*, 3(3):529–538, 2008.

[PF05] Alin C. Popescu und Hany Farid. Exposing Digital Forgeries by Detecting Traces of Re-sampling. *IEEE Transactions on Signal Processing*, 53(2):758–767, 2005.

[SM08] Husrev T. Sencar und Nasir Memon. Overview of State-of-the-art in Digital Image Forensics. In Bhargab B. Bhattacharya, Susmita Sur-Kolay, Subhas C. Nandy und Aditya Bagchi, Hrsg., *Algorithms, Architectures and Information Systems Security*, Kapitel 15, Seiten 325–348. World Scientific Press, 2008.

[VV91] Sabine Van Huffel und Joos Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis*. Society for Industrial and Applied Mathematics, 1991.

Collusion-Secure Fingerprint Watermarking for Real World Applications

Marcel Schäfer, Waldemar Berchtold, Sascha Zmudzinski, Martin Steinebach
Fraunhofer Institut für Sichere Informationssysteme SIT, Darmstadt, Germany*
{schaefer,berchtold,steinebach,zmudzinski}@sit.fraunhofer.de

Margareta Heilmann
Department of Mathematics, Bergische Universität Wuppertal, Germany
margareta.heilmann@math.uni-wuppertal.de

Stefan Katzenbeisser
Department of Computer Science, Technische Universität Darmstadt, Germany
katzenbeisser@seceng.informatik.tu-darmstadt.de

Abstract: Digital transaction watermarking today is a widely accepted mechanism in multimedia security. One major threat on transaction watermarking are collusion attacks. Here multiple individualized copies of the work are mixed to produce a counterfeited or undetectable watermark. One common countermeasure is the usage of so-called fingerprints. Theoretical fingerprint approaches do not consider the inaccuracy of the detection process of watermarking algorithms. In this work we show how an existing fingerprint code can be optimized with respect to code length in order to collaborate with a watermarking algorithm to provide a maximum of reliability with a minimum of payload.

1 Introduction

In the Internet, copies of digital works, ranging from pictures, music, audio books, videos, movies to software, are distributed illegally via various channels. An answer to this challenge of massive copyright infringement are digital watermarking algorithms. Watermarking imperceptibly hides information in multimedia data. This is done, for example, by changing statistical characteristics or quantization properties of multimedia data in a suitable transformed spectral domain. The hidden information is called *watermark message* or just *watermark*.

If a distributor of multimedia, for example in an online shop, wants to prevent his works from being illegally distributed by its customers, he can use transaction watermarking.

*This work was supported by the *CASED Center for Advanced Security Research Darmstadt*, Germany, funded under the *LOEWE* programme by the Hessian state government (*http://www.cased.de*).

Here, the watermark message is an *individual* identifier, e.g. a binary transaction ID, that can distinguish the distributed copies from each other. If the copy appears in an illegal source, the transaction ID allows to trace back to the customer at any later date. In this way customers are discouraged from illegal distribution of their bought copies in the Internet and reminded of their individual responsibility for respecting the copy right [CFNP00].

One specific security attack in a transaction watermarking scenario representing a significant security threat is the so called *collusion attack*. Here, several customers collaborate and can reveal detail about the watermark algorithm by comparing their individually watermarked user copies. The *colluders* can identify the location of the embedded watermark message and create a new copy of a media file by calculating the average among all user copies. The resulting copy may contain a destroyed or counterfeited watermark message.

One countermeasure to collusion attacks are *collusion-secure fingerprints* [BS98]. These are transaction watermark messages designed to be robust against such attacks. Collusion-secure fingerprints are created under specific additional conditions and assumptions which drastically increases the code length of the embedded message. Thus, a serious challenge most existing fingerprinting algorithms have to face is to provide a code length which is not beyond the limitations of the related embedding algorithms in practice.

One promising example still observing satisfying error probabilities is given by *Skoric* et al. in [SKC08]. However, most existing fingerprint coding approaches are based on simulated results and no specific watermark embedding/detector algorithm is considered. That means they act on the assumption, that every bit of the detected watermark has been detected equally reliably. In practice this assumption is no longer true. Many real world watermark detectors, as for example introduced in an earlier work [Ste03], output a goodness value for every bit, which describes how reliable this bit is retrieved to be '1' or '0'. Thus, combining an existing fingerprinting code directly with a real world watermarking algorithm causes an additional error, respectively the estimated theoretical error probabilities are in doubt.

As a solution to this problem we propose a modified version of *Skoric*'s fingerprinting codes [SKC08] by introducing specific *weights* representing the goodness of the watermark detector and apply them them to *Skoric*'s accusation sums. These weights eliminate the additional error and maintain the error probabilities provided by the earlier theoretical approaches. This is achieved with only a slight increment of *Skoric*'s code length. Therewith we create a new fingerprinting code which can be applied to real world watermarking algorithms.

2 Related Work

2.1 Modified Tardos Code for arbitrary alphabet size

The basic idea of the construction is based on *Skoric* [SKC08] and *Tardos* [Tar03]. Here, *Skoric* et al. provide an ε_1-sound and (ε_2,c_0)-complete fingerprinting code. That is, the scheme provides a false positive error rate ε_1 and a false negative error rate ε_2 (see Defini-

tions 9 and 10). The quantity c_0 denotes the maximum number of colluders for which the scheme works.

The basic restriction for the attackers to perform manipulations we assume in this work is the often made *marking assumption*: during the collusion attack, the colluders are only able to change those message bits in the fingerprint where the bits, i.e. the related watermarked content, is different.

The *Tardos* and *Skoric* fingerprinting scheme generates an $n \times m$ matrix X, where n denotes the number of users to be listed in the system and m stands for the length of the distributed fingerprint. Thus, the jth row of the matrix corresponds to the fingerprint which is later embedded in the copy that is released to customer $j \in \{1, ..., n\}$. The entries X_{ji} of the matrix X are generated in two steps:

First, the distributor picks m independent random numbers $\{p_i\}_{i=1}^m$ over the interval $p_i \in [t, 1-t]$, with $t = (300c)^{-1}$, where c is the number of colluders. Each $p_i = \sin^2(r_i)$ is selected by picking uniformly at random the value $r_i \in \left[\bar{t}, \frac{\pi}{2} - \bar{t}\right]$ with $0 < \bar{t} < \frac{\pi}{4}$, where $\sin^2(\bar{t}) = t$. Second, the matrix X is selected, by picking each entry X_{ji} independently from the binary alphabet $\{0, 1\}$ according to $\mathbb{P}[X_{ji} = 1] = p_i$.

Tardos' choice of the distribution for p_i is biased toward the values close to '0' or '1' which is motivated by the *marking condition*. This distribution is realized by a probability density function f which is symmetric around 0.5 and heavily biased towards values of p_i close to the limits of the interval $[t, 1-t]$.

Definition 1 (Tardos' probability density function)

$$f(p) \;=\; \frac{1}{2\arcsin(1-2t)} \frac{1}{\sqrt{p(1-p)}}, \quad p \in [t, 1-t].a \tag{1}$$

If the distributor then receives an unauthorized copy with embedded fingerprint y by chance, he creates the so called accusation sum A_j for each user j out of his given fingerprint X_j and y to identify at least one of the colluders:

Definition 2 (Skoric's accusation sum A_j)

$$A_j(\bar{\boldsymbol{p}}, X, y) \;:=\; \sum_{i=1}^{m} A_j^{(i)}\,, \quad A_j^{(i)} \;:=\; \delta_{y_i X_{ji}} g_1(p_{y_i}^{(i)}) + [1 - \delta_{y_i X_{ji}}] g_0(p_{y_i}^{(i)}). \tag{2}$$

Here, $\delta_{y_i X_{ji}}$ denotes the Kronecker Delta, and $p_{y_i}^{(i)}$ stands for the probability of the entries of the matrix X. The so called accusation functions g_1 and g_0 are as follows:

Definition 3 (The accusation functions g_1, g_0)

$$g_1(p) \;:=\; \sqrt{\frac{1-p}{p}} \qquad g_0(p) \;:=\; -\sqrt{\frac{p}{1-p}}. \tag{3}$$

The accusation functions g_1 and g_0 have specific properties: The more p increases, the smaller becomes $g_1(p)$. This means, the higher the probability of the symbol at this position, the smaller will be the positive amount given to the according accusation sum A_j, and vice versa.

The distributor accuses user j to be guilty, if his accusation sum A_j exceeds a predefined accusation threshold Z depending on the maximum number of colluders c_0 of the scheme.

To conduct the proof of soundness one has to evaluate over the degrees of freedom of the accusation sum. For the proof of completeness one has to average over the fingerprint matrix X_C of the colluders. This is done by averaging over the so called collective accusation sum A_C. Here the index *C* stands for the set of colluders.

Definition 4 (Skoric's collective accusation sum A_C)

$$A_C \;:=\; \sum_{j\in C} A_j \;=\; \sum_{i=1}^{m} A_C^{(i)}\,; \quad A_C^{(i)} \;:=\; b_{y_i}^{(i)} g_1(p_{y_i}^{(i)}) + [c - b_{y_i}^{(i)}] g_0(p_{y_i}^{(i)}). \tag{4}$$

The parameter $c \leq c_0$ represents the number of colluders, while $b_{y_i}^{(i)}$ stands for the number of occurrences of the symbol $y_i \in \{0,1\}$ in X_C at the position i. Thus holds $b_0 = c - b_1$. The requirement on the code length m for a binary alphabet providing ε_1-soundness and (ε_2,c_0)-completeness and requiring $c_0 \geq 10$, is given by

Definition 5 (Code length m and threshold parameter Z)

$$m \;=\; Ac_0^2 \,\ln\left(\varepsilon_1^{-1}\right), \quad Z \;=\; Bc_0 \ln\left(\varepsilon_1^{-1}\right) \tag{5}$$

with code length parameter $A = \pi^2$ and threshold parameter $B = 2\pi$.

2.2 Audio Watermarking

Digital watermarking schemes have been under research and development for various types of multimedia data for many years, including audio formats like PCM, mp3 or MIDI. In this work we focus on digital PCM audio data. Several approaches for PCM audio watermarking have been introduced in the literature, like in [BTH96], in [CMB02] or also [Ste03]. The latter algorithm is the base of this work.

The technique for incorporating the watermark follows a spread spectrum Patchwork approach [BGML96]:

1. As a first step, the algorithm divides the audio signal into segments, known as windows or frames. Then, the PCM source signal in each frame is transformed to the frequency spectrum using a Fast Fourier Transform (FFT).

2. Then, the watermark is incorporated by the deliberate alternation of FFT energy coefficients according to the Patchwork embedding rule: two disjoint subsets A and B

are selected from all energy coefficients. The selection of the coefficients is done pseudo-randomly dependent on a secret key. Depending on the message bit to be a '0' or '1', respectively, all coefficients in A are increased in energy while all in B are decreased, or vice versa, respectively. This introduces a significant difference in the *mean* energies in the two subsets. The degree of the changes in the frequency domain is controlled by a psycho-acoustic model such that audible distortions are avoided as good as possible.

3. In the final step, the algorithm converts the data back into PCM format. In order to prevent noticeable gaps between the segments, special mechanisms are used to fade between marked and unmarked parts of the audio.

The retrieval process consists of the following steps:

1. The marked audio is divided again into frames and the FFT is applied on the PCM samples. Appropriate synchronizations mechanisms to detect the correct file positions are not discussed here.

2. The watermark message bit is then retrieved as follows: If the correct secret key is available, the same subsets A and B of FFT coefficients used during embedding are selected. Now the difference in the mean energies between the two subsets can be analyzed: the sign of this energy difference indicates if the message bit is a '0' or '1'. The absolute value can be considered as a goodness of the detection. We will refer to this as its *detection score* in the following.

3 The Model

In this chapter we introduce a modification of the accusation sum of [SKC08], which allows to include information on the reliability of a detected watermark bit.

Closer analysis shows that for a fixed code length the actual false positive error rate ε_1 increases about a factor of almost 1.6 when applying for example the *Skoric* code with our real world watermarking algorithm. This is because in fingerprinting algorithms based on the *Tardos* Code, small and large absolute values of the detection score would be evaluated equally. In contrast, we introduce that message bits with smaller detection score contribute less to *Skoric*'s accusation sum, (2). Motivated by this, special weights $w^{(i)} \in [0, 1)$ are introduced, representing the goodness of detection.

Definition 6 (Weight function $w^{(i)}$)

$$w^{(i)} := w(score^{(i)}) := 1 - e^{-|score^{(i)}|}, \tag{6}$$

where the function argument is the output of the watermarking algorithm described in Section 2.2, also referred to as the score.

The choice of the weight function $w^{(i)}$ is such that the weights always lie in the range of $[0, 1)$.

In our construction, we include the weights in the accusation sum as follows:

Definition 7 (Weighted accusation sums)

$$\begin{aligned} \tilde{A}_j &:= \sum_{i=1}^{m} \tilde{A}_j^{(i)}\,, \quad \textit{with } \tilde{A}_j^{(i)} := w^{(i)} \left[\delta_{y_i X_{ji}} g_1(p_{y_i}^{(i)}) + (1 - \delta_{y_i X_{ji}}) g_0(p_{y_i}^{(i)})\right] \\ \tilde{A}_C &:= \sum_{i=1}^{m} \tilde{A}_C^{(i)}\,, \quad \textit{with } \tilde{A}_C^{(i)} := w^{(i)} \left[b_{y_i}^{(i)} g_1(p_{y_i}^{(i)}) + (c - b_{y_i}^{(i)}) g_0(p_{y_i}^{(i)})\right] \end{aligned} \tag{7}$$

It can easily be seen, that $\tilde{A}_j^{(i)}$ equals $w^{(i)} \cdot A_j^{(i)}$ of Equation (2), as well as $\tilde{A}_C^{(i)}$ equals $w^{(i)} \cdot A_C^{(i)}$ of Equation (4).

To keep the generality of the scheme, w and its expectation values are held as fixed parameters during all computations. This is done to remain independent of the particular watermarking algorithm (as long it provides a score for every detected message bit) and to give a general statement of this scheme using weights.

Estimating the conditions of soundness and completeness presented in Sections 4.2 and 4.3 against each other we obtain the following theorem according to *Skoric*'s definition of the code length (5).

Theorem 1 *To satisfy the bounds ε_1 and ε_2, the code length parameter A in the proposed scheme can be bounded as*

$$\frac{\pi^2}{\mathbb{E}[w^2]} \left(1 + \sqrt{\frac{1}{c_0} \frac{\ln(\varepsilon_2)}{\ln(\varepsilon_1)}}\right)^2 \;\leq\; A,$$

where $\mathbb{E}[w^{(i)}]$ denotes the expectation value of the weight function.

The proof of **Theorem 1** is given in Section 4. From this it follows directly from (5):

Corollary 2 *The proposed scheme provides ε_1-soundness and (ε_2,c_0)-completeness with a required minimum code length of*

$$m \;\geq\; \frac{\pi^2}{\mathbb{E}[w^2]} \left(1 + \sqrt{\frac{1}{c_0} \frac{\ln(\varepsilon_2)}{\ln(\varepsilon_1)}}\right)^2 c_0^2 \,\ln\left(\varepsilon_1^{-1}\right).$$

To get a more manageable expression for m, the term within the round brackets can be neglected since ε_1 is assumed to be much smaller than ε_2. Thus we have the estimate

$$m \geq \frac{\pi^2}{\mathbb{E}[w^2]} \cdot c_0^2 \ln\left(\varepsilon_1^{-1}\right) \tag{8}$$

for the minimum code length. The threshold Z stays alike Equation (5).

These weighted accusation sums provide a minimum code length which is longer than the results in [SKC08] about only $1/\mathbb{E}[w^{(i)}]$. The difference to [SKC08] is, if the detection process is not really sure if the detected bit is a '1' or a '0', the weight at that position will be close to zero, and therefore it will only contribute of probably unsafe information to the scheme only a little. The *Skoric* Code does not take this into account, and therewith creates an additional error which might falsifies the results.

An explicit example for the code length m is given in table 1. Here the maximum number of colluders, c_0, is set to 10 and $\mathbb{E}[w^2]$ for our weight function equals 0.888.

4 Proofs

This section gives a short insight into the proofs for correctness of our scheme. For a more detailed description see [Sch09].

4.1 Preliminaries

The following subsection shows up premises for the weighted accusation sums to conduct the proof schemes in 4.2 and 4.3 analogously to [SKC08]. Therefore we adopt:

Definition 8 (Definition of soundness) *Let $\varepsilon_1 \in (0,1)$ be a fixed constant and let j be an arbitrary innocent user. The fingerprinting scheme is ε_1-sound if for all collusions $C \subseteq [n]\backslash j$ and for all C-strategies holds*

$$\mathbb{P}[\textit{False positive}] \;=\; \mathbb{P}[j \in \Sigma] < \varepsilon_1. \tag{9}$$

The parameter C is the set of colluders, whereas Σ means the group that is accused as colluders, and n is the number of all users.

Definition 9 (Definition of completeness) *Let $\varepsilon_2 \in (0,1)$ and $c_0 \in \mathbb{N}^+$ be fixed constants. The fingerprint scheme is (c_0, ε_2)-complete if for all collusions C of size $c \leq c_0$ and all strategies holds*

$$\mathbb{P}[\textit{False negative}] \;=\; \mathbb{P}[C \cap \Sigma = \emptyset] < \varepsilon_2. \tag{10}$$

To prove that our scheme is ε_1-sound and (c_0, ε_2)-complete under certain conditions we need further quantities. In a similar way as in [SKC08] we define:

Definition 10

$$\mu_{j,w} \;:=\; \frac{\mathbb{E}_{wyX_p}[\tilde{A}_j]}{m} \;=\; \mathbb{E}_{wyX_p}\left[\tilde{A}_j^{(i)}\right]; \quad \sigma^2_{j,w} \;:=\; \frac{\mathbb{E}_{wyX_p}[\tilde{A}_j^2] - \mathbb{E}^2_{wyX_p}[\tilde{A}_j]}{m} \tag{11}$$

Definition 11

$$\mu_{c,w} := \frac{\mathbb{E}_{wyXp}[\tilde{A}_C]}{m} = \mathbb{E}_{wyXp}\left[\tilde{A}_C^{(i)}\right] ; \sigma_{c,w}^2 := \frac{\mathbb{E}_{wyXp}[\tilde{A}_C^2] - \mathbb{E}_{wyXp}^2[\tilde{A}_C]}{m} \quad (12)$$

Thus, the summarized expectation value $\mathbb{E}_{wyXp}$ consists of $\mathbb{E}_w$ denoting the expectation value of $w^{(i)}$, and $\mathbb{E}_{X_j}$ which is the evaluation over the distributers choice of generating the columns of the matrix X, and $\mathbb{E}_p$ describing the evaluation over $\mathbf{p}^{(i)}$, and the collusions choice of the symbols of the fingerprint $\mathbb{E}_{y_i}$.

Because of the scaled definitions of the mean values and regarding to [SKC08] we can directly state and prove the following two Lemmas:

Lemma 1 $\mu_{j,w} = 0$.

Proof: Starting by first computing the expectation $\mathbb{E}_{X_j}$, we have

$$\mathbb{E}_{X_j}\left[\tilde{A}_j^{(i)}\right] = \mathbb{E}_{X_j}\left[w^{(i)} \cdot A_j^{(i)}\right] = \mathbb{E}_{X_j}\left[w^{(i)}\right]\mathbb{E}_{X_j}\left[A_j^{(i)}\right] = \mathbb{E}_{X_j}\left[w^{(i)}\right] \cdot 0,$$

where $A_j^{(i)}$ represents *Skoric*'s accusation Sum, (2). Thus holds $\mathbb{E}_{wyXp}[\tilde{A}_j] = 0$. □

Lemma 2 $\mu_{c,w} = \frac{2}{\pi} \cdot \mathbb{E}[w]$.

Proof: With the help of *Skoric*'s collective accusation sum and its mean value, $\tilde{\mu} = 2/\pi$, we can quickly complete the proof:

$$\mathbb{E}_{wyXp}\left[\tilde{A}_c^{(i)}\right] = \mathbb{E}_{wyXp}\left[w^{(i)}A_c^{(i)}\right] = \mathbb{E}\left[w\right]\ \mathbb{E}_{yXp}\left[A_c^{(i)}\right] = \mathbb{E}\left[w\right] \cdot \frac{2}{\pi}.$$ □

Lemma 3 $\sigma_{j,w}^2 = \mathbb{E}[w^2]$.

Proof: $\tilde{A}_j^2$ can be described as:

$$\tilde{A}_j^2 = \sum_{i=1}^{m} \{w^{(i)}\}^2 \cdot A_j^2$$

Because of the independence of the columns of X, and by first evaluating over X_j and w, we get

$$\mathbb{E}_{wX_j}[\tilde{A}_j^2] = \mathbb{E}[w^2] \cdot m + 0.$$

From this follows $\mathbb{E}_{wyXp}[\tilde{A}_j^2] = m \cdot \mathbb{E}[w^2]$. Substituting this into the definition of $\sigma_{j,w}^2$ in equation (11) finishes the proof. □

Lemma 4 *The mean $\mu_{c,w}$ and variance $\sigma_{c,w}$ satisfy*

$$\mu_{c,w}^2 \; + \; \sigma_{c,w}^2 \; = \; \mathbb{E}[w^2] \cdot c.$$

Proof: Straight from the definition of $\sigma_{c,w}$, (12), and applying (7) we get

$$\begin{aligned}\sigma_{c,w}^2 \; &= \; m^{-1}\mathbb{E}_{wyXp}[\tilde{A}_C^2] - m\,\mu_{c,w}^2 \\ &= \; \frac{1}{m}\left(\sum_{i=1}^{m}\mathbb{E}_{wyXp}[\{\tilde{A}_C^{(i)}\}^2] + \sum_{i\neq k}\mathbb{E}_{wyXp}[\tilde{A}_C^{(i)}]\mathbb{E}_{wyXp}[\tilde{A}_C^{(k)}]\right) - m\mu_{c,w}^2. \quad (13)\end{aligned}$$

In order to receive an identity for $\mathbb{E}_{wyXp}[\{\tilde{A}_C^{(i)}\}^2]$, by making use of the column symmetry, the properties $p_1 = 1 - p_0$ and $b_1 = c - b_0$, we get

$$\{\tilde{A}_C^{(i)}\}^2 \; = w^2\big[b_1 g_1(p_1) + (c - b_1)g_0(p_1)\big]^2 \quad ,$$

where the column index i on w, y, p and b is omitted for notational simplicity. Thus it holds

$$\mathbb{E}_{wyXp}\left[\{\tilde{A}_C^{(i)}\}^2\right] = \mathbb{E}[w^2]\mathbb{E}_{yXp}\left[\frac{(b_1 - cp_1)^2}{p_1(1-p_1)}\right] = \mathbb{E}[w^2]\; c\,.$$

Now returning to equation (13), with definition (12), both elements of the second sum exactly equal $\mu_{c,w}$. Thus, the expression becomes

$$\sigma_{c,w}^2 = \frac{1}{m}\left(m\mathbb{E}_{wyXp}\left[\{\tilde{A}_C^{(i)}\}^2\right] + (m^2 - m)\mu_{c,w}^2\right) - m\mu_{c,w}^2 = \mathbb{E}[w^2]c - \mu_{c,w}^2. \qquad \square$$

In the following we will make use of the inequalities

$$1 + a < e^a < 1 + a + a^2, \quad \text{where } 0 < a \leq 1.79 \tag{14}$$

and the *Markov* inequality

$$\mathbb{P}(|X| \geq K) \leq \frac{\mathbb{E}(|X|)}{K}, \quad K > 0. \tag{15}$$

4.2 Proof of soundness

With the results of Section 4.1, we are able to derive the conditions under which soundness for the weighted accusation scheme is achieved similar to [SKC08]. According to (9) we have to determine the conditions under which $\mathbb{P}[j \in \Sigma] < \varepsilon_1$ holds true. With an auxiliary variable $\tilde{\alpha}_1$, by using (15) and as the columns of X are independent, we get

$$\mathbb{P}[j \in \Sigma] \; = \; \mathbb{P}[\tilde{A}_j > Z] \; = \; \mathbb{P}[e^{\tilde{\alpha}_1\tilde{A}_j} > e^{\tilde{\alpha}_1 Z}] \; \leq \; \frac{\mathbb{E}_{X_j}[e^{(\tilde{\alpha}_1\tilde{A}_j)}]}{e^{\tilde{\alpha}_1 Z}} = \frac{\{\mathbb{E}_{X_j}[e^{(\tilde{\alpha}_1\tilde{A}_j^{(i)})}]\}^m}{e^{\tilde{\alpha}_1 Z}}. \tag{16}$$

Now we look at $\mathbb{E}_{X_j}[e^{(\tilde{\alpha}_1 \tilde{A}_j^{(i)})}]$. By restricting $\tilde{\alpha}_1$ to $\tilde{\alpha}_1 \in (0, \tilde{\alpha}_1^{max}]$, $\tilde{\alpha}_1^{\max} = \frac{1.79}{g_1(t)}$ we can apply (14) to $\mathbb{E}_{X_j}[e^{(\tilde{\alpha}_1 \tilde{A}_j^{(i)})}]$. Together with lemmas 1 and 3 this leads to

$$\mathbb{E}_{X_j}[e^{\tilde{\alpha}_1 \tilde{A}_j^{(i)}}] \;<\; 1 \;+\; \tilde{\alpha}_1 \mathbb{E}_{X_j}[\tilde{A}_j^{(i)}] \;+\; \tilde{\alpha}_1^2 \mathbb{E}_{X_j}[\{\tilde{A}_j^{(i)}\}^2] \;<\; e^{\tilde{\alpha}_1^2 \mathbb{E}[w^2]}.$$

Substituting this into (16) we derive

$$\mathbb{P}[j \in \Sigma] \;<\; \min_{\tilde{\alpha}_1 \in (0, \tilde{\alpha}_1^{\max}]} e^{\tilde{\alpha}_1 (m \tilde{\alpha}_1 \mathbb{E}[w^2] - Z)}. \tag{17}$$

With the definitions of m and Z in (5) and using the value $\tilde{\alpha}_1^* = \frac{B}{2Ac_0\mathbb{E}[w^2]}$ which minimizes the right side of (17)we get

$$\mathbb{P}[j \in \Sigma] \;<\; \varepsilon_1^{\frac{B^2}{4A\mathbb{E}[w^2]}}.$$

As $\varepsilon_1 \in (0,1)$ this finally gives the conditions for ε_1-soundness

$$\frac{B^2}{4A\mathbb{E}[w^2]} \;\geq\; 1. \tag{18}$$

4.3 Proof of completeness

According to (10) we estimate $\mathbb{P}[C \cap \Sigma \;=\; \emptyset]$ in an appropriate way. Therefore let C be a coalition of size $c \leq c_0$ and $\tilde{\alpha}_2 > 0$ with $-\tilde{\alpha}_2 \tilde{A}_C^{(i)} \leq 1.79$ be an auxiliary variable. Again using (15) we get

$$\mathbb{P}[C \cap \Sigma = \emptyset] \leq \mathbb{P}[\tilde{A}_C \;<\; c_0 Z] < \frac{\mathbb{E}_{wyXp}\left[e^{-\tilde{\alpha}_2 \tilde{A}_C}\right]}{e^{-\tilde{\alpha}_2 c_0 Z}} = \frac{\left\{\mathbb{E}_{wyXp}\left[e^{-\tilde{\alpha}_2 \tilde{A}_C^{(i)}}\right]\right\}^m}{e^{-\tilde{\alpha}_2 c_0 Z}}.$$

Using the right inequality of Equation (14) and definition 11 we may write

$$\begin{aligned}
\mathbb{E}_{wyXp}\left[\exp\left(-\tilde{\alpha}_2 \tilde{A}_C^{(i)}\right)\right] \;&<\; 1 \;-\; \tilde{\alpha}_2 \mathbb{E}_{wyXp}\left[\tilde{A}_C^{(i)}\right] \;+\; \tilde{\alpha}_2^2 \mathbb{E}_{wyXp}\left[\left(\tilde{A}_C^{(i)}\right)^2\right] \\
&< 1 - \tilde{\alpha}_2 \mu_{c,w} + \tilde{\alpha}_2^2 (\mu_{c,w}^2 + \sigma_{c,w}^2) \\
&\leq 1 - \tilde{\alpha}_2 \mathbb{E}[w^2] \left(\frac{2}{\pi} - c_0 \tilde{\alpha}_2\right),
\end{aligned}$$

where we have made use of lemmas 2 and 4 and of the property $\mathbb{E}[w] > \mathbb{E}[w^2]$. As $-\tilde{\alpha}_2 \tilde{A}_C^{(i)} \leq 1.79$, we choose $\tilde{\alpha}_2^{max} = -1.79 \frac{g_0(t)}{c_0}$ to derive

$$\begin{aligned}
\mathbb{P}[\tilde{A}_C < c_0 Z] \;&<\; \min_{\tilde{\alpha}_2 \in (0, \tilde{\alpha}_2^{max}]} \exp\left(-\tilde{\alpha}_2 \left[m\mathbb{E}[w^2](\frac{2}{\pi} - \tilde{\alpha}_2 c_0) - c_0 Z\right]\right) \\
&=\; \exp\left(\ln(\varepsilon_1)\left(\frac{Ac_0\mathbb{E}[w^2]}{\pi^2} + \frac{c_0 B^2}{4A\mathbb{E}[w^2]} - \frac{Bc_0}{\pi}\right)\right) \\
&=\; \varepsilon_1^{\left(\frac{Ac_0\mathbb{E}[w^2]}{\pi^2} + \frac{c_0 B^2}{4A\mathbb{E}[w^2]} - \frac{Bc_0}{\pi}\right)} \;\leq\; \tilde{\alpha}_2
\end{aligned} \tag{19}$$

where we again made use of (14) and applied the minimizing value $\tilde{\alpha}_2^* = 1/(c_0\pi) - Z/(2m\mathbb{E}[w^2])$ and the definitions of m and Z. In order to get the right side of (19) bounded by ε_2 we finally get the condition for completeness,

$$B \leq \frac{2A\mathbb{E}[w^2]}{\pi} - 2\sqrt{\frac{A\mathbb{E}[w^2]}{c_0}\frac{\ln(\varepsilon_2)}{\ln(\varepsilon_1)}}. \tag{20}$$

5 Discussion

After proving the correctness of our approach, we now discuss its impact on real-world applications. The main question is if the resulting fingerprinting codes are sufficiently short to be used together with the watermarking algorithm. Of course this also depends on the length of the audio material to be protected.

Analogue to Section 6 in [SKC08], by assuming a perfectly Gaussian distribution for the collective accusation sum we are able to reduce our code length in (8) about a factor of more than two. Therewith we create a new requirement on the code length $\tilde{m}$ and a new threshold $\tilde{Z}$:

$$\tilde{m} \geq \frac{\pi^2}{2}\frac{\mathbb{E}[w^2]}{\left[\mathbb{E}(w)\right]^2}c_0^2 \ln\left(\frac{1}{\varepsilon_1\sqrt{2\pi}}\right); \quad \tilde{Z} = 2\pi\mathbb{E}[w^2]c_0 \ln\left(\frac{1}{\varepsilon_1\sqrt{2\pi}}\right).$$

An explicit example for $\tilde{m}$ with a maximum number of colluders of $c_0 = 10$ is given in Table 1. Due to our weight function (6) we estimated $\mathbb{E}[w^2]/[\mathbb{E}(w)]^2 \approx 1.083$, which is empirically computed. Closer analysis shows that an increment of the code length by a factor of only 1.083 compensates the actual value for ε_1 which otherwise would increase by a factor of 1.6.

It is not said that the choice of the weight function w has been optimal. Already directly taking the absolute value of the score as the weights may achieve a shorter code length, but as the expectation values of these weights exceed 1, the proof schemes must be adapted.

FP error probability ε_1	code length m	code length $\tilde{m}$
10^{-3}	7678	3201
10^{-4}	10237	4432
10^{-5}	12796	5662

Table 1: code length for $c_0 = 10$

Given the parameters above, we can distinguish a virtually unlimited number of customers under the assumption that only a limited number of attackers collude to attack the watermark. Using a message length of 3201-12796 message bits, resilience against up to 10 colluders can be achieved.

While the required payload is higher than the typical payloads of our algorithm (which range between 32 and 128 bit in practice), they are suited for many types of audio works:

With a typical payload of 7 bits per second, one complete fingerprint can be embedded in 8 to 30 minutes of audio, depending on the accepted false accusation rate. This means, that the approach is well suited for audio books and movie soundtracks.

6 Conclusions

In our work we proposed a fingerprinting algorithm providing required error probabilities that is collusion-secure for a maximum number of colluders $c_0 \geq 10$. Existing fingerprint coding algorithms in the literature are not able to give a well suited statement about the actual error rates when applied to real world watermarking methods. In contrast, our approach can manage this problem and it gives a reliable and trustful statement in a copyright violation charge. This is done by introducing *weights* that represent the goodness of the watermark detection. As a result, we achieve a significantly higher level of accuracy of the accusation. In our work we show that by combining our audio watermarking algorithm and the *Skoric* fingerprinting scheme, with only a slightly increment in the code length, a reliable and comparatively compact fingerprinting solution can be achieved, allowing the protection of movie soundtracks, audio books and music albums.

References

[BGML96] W. Bender, D. Gruhl, N. Morimoto, and A. Lu. Techniques for Data Hiding. *IBM Systems Journal, MIT Media Lab*, 35(3,4):313–336, 1996.

[BS98] D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, Sept. 1998.

[BTH96] L. Boney, A. H. Tewfik, and K. N. Hamdy. Digital Watermarks for Audio Signals. volume Proceedings of MULTIMEDIA 96, pages 473–480, June 1996.

[CFNP00] B. Chor, A. Fiat, M. Naor, and B. Pinkas. Tracing traitors. *IEEE Transactions on Information Theory*, 46(3):893–910, May 2000.

[CMB02] I. J. Cox, M. L. Miller, and J. A. Bloom. *Digital Watermarking*. Morgan Kaufmann, 2002.

[Sch09] M. Schäfer. Optimization of Fingerprinting Encoding Algorithms in the Context of Digital Audio Watermarking Methods. Master's thesis, Bergische Universität Wuppertal, December 2009.

[SKC08] Boris Skorić, Stefan Katzenbeisser, and Mehmet U. Celik. Symmetric Tardos fingerprinting codes for arbitrary alphabet sizes. *Des. Codes Cryptography*, 46(2):137–166, 2008.

[Ste03] Martin Steinebach. *Digitale Wasserzeichen fuer Audiodaten*. PhD thesis, TU Darmstadt, Germany, 2003. ISBN 3832225072.

[Tar03] G. Tardos. Optimal probabilistic fingerprinting codes. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC), pp. 116-125*, 2003.

Iterative präzisionsbewertende Signaturgenerierung

René Rietz, Sebastian Schmerl, Michael Vogel und Hartmut König

Brandenburgische Technische Universität
Lehrstuhl Rechnernetze und Kommunikationssysteme
03013 Cottbus, Postfach 10 13 44
{rrietz, sbs, mv, koenig}@informatik.tu-cottbus.de

Abstract: Die Wirksamkeit signaturbasierter Intrusion Detection Systeme hängt entscheidend von der Präzision der verwendeten Signaturen ab. Die Ursachen unpräziser Signaturen sind hauptsächlich der Signaturableitung zuzuschreiben. Die Spezifikation einer Signatur ist aufwendig und fehleranfällig. Methoden für ein systematisches Vorgehen existieren bisher kaum. In diesem Papier stellen wir einen Ansatz zur systematischen Ableitung von Signaturen für Host-basierte IDS vor. Ausgehend vom Programmcode und der Verwundbarkeit werden ganze Signaturen oder Signaturfragmente generiert. Wir zeigen, dass durch den Einsatz von statischer Code-Analyse der Entwurfsprozess für Signaturen automatisiert und entscheidend verkürzt werden kann. Ferner ist eine Qualitätsabschätzung der abgeleiteten Signatur möglich.

Keywords: Intrusion Detection, Signaturanalyse, Signature Engineering, systematische Signaturableitung, Quellcodeanalyse

1 Einleitung

Um den Gefahren eines sich ständig vergrößernden Bedrohungspotentials für IT-Infrastrukturen zu begegnen, werden zunehmend Intrusion Detection Systeme (*IDS*) eingesetzt. In der praktischen Anwendung erweist sich die Signaturanalyse im Vergleich zur Anomalieerkennung bisher als die effizientere und zuverlässigere Variante. Sie steht hier im Mittelpunkt der Betrachtung. Signaturbasierte Analyseverfahren untersuchen Protokoll- oder Auditdaten nach Mustern bekannter Sicherheitsverletzungen, den *Signaturen*. Die Wirksamkeit der Analyse hängt entscheidend von der Präzision der verwendeten Signaturen ab. Unpräzise Signaturen schränken die Erkennungsfähigkeit der Analyse stark ein und führen zu den typischen hohen Fehlalarmraten. Die Ursachen dafür sind hauptsächlich in der manuellen Ableitung von Signaturen aus vorliegenden Angriffsszenarien zu finden. Diese Ableitung erfolgt meist empirisch auf der Grundlage des Wissens und der Erfahrung von Experten.

Die in der Forschung untersuchten Ansätze zur Automatisierung der Signaturableitung lassen sich in zwei Klassen gliedern: (1) Black-Box-basierte Verfahren, die auf der Analyse von Angriffsprogrammen basieren und (2) White-Box-basierte Verfahren, die den

Quelltext bzw. Binärcode der verwundbaren Anwendung untersuchen. Bei den Black-Box-basierten Verfahren werden häufig die Angriffsspuren mehrerer realer Angriffe (Netzwerkpakete) auf gemeinsame Substrings untersucht und mittels Graphclusterungsalgorithemen zu Signaturen zusammengefasst. Beispiele hierfür sind POLYGRAPH [NKS05] und NEBULA [WFGPM09]. Die White-Box-basierten Verfahren können weiter unterteilt werden in: (2.1) Verfahren, die aus den Angriffsspuren (hier: *execution traces*) und einem Programm- oder Datenmodell die für eine Verwundbarkeitsausnutzung notwendigen Eingaben berechnen [BNS+06, CCZ+07, CPWL07] sowie (2.2) Policy-basierte Verfahren [FHSL96, WD01, SBD01, GJM04], die das Programmmodell selbst als Signatur auffassen und vom Modell abweichendes Programmverhalten als Angriff interpretieren. Die Verfahren (1) und (2.1) benötigen mindestens ein Angriffsprogramm. Der Signaturgenerierungsprozess beginnt demzufolge erst nach der Verbreitung von Angriffsprogrammen und die generierten Signaturen müssen infolgedessen schneller als die Angriffsprogramme verbreitet werden. Die Verfahren aus (2.2) erfordern kontinuierliche Vergleiche zwischen dem Programmverhalten und dem zugehörigen Programmmodell. Für die Gewährleistung einer geringen Fehlalarmrate bei der Ermittlung von abweichendem Programmverhalten sind sehr genaue Programmmodelle erforderlich, die wegen ihres Umfangs (Ressourcenbedarf) in der Praxis nicht eingesetzt werden können. Eine Qualitätsabschätzung der generierten Signaturen wird in keinem der genannten Verfahren realisiert.

In diesem Beitrag wird eine Methodik zur Automatisierung der Signaturableitung vorgestellt, die keine Angriffsprogramme erfordert und die eine Qualitätsabschätzung der generierten Signaturen ermöglicht. Das vorgestellte Verfahren generiert zu diesem Zweck ein Programmmodell, das ebenfalls für IDS der Klasse 2.2 verwendet werden kann. Voraussetzung für den hier vorgestellten Ansatz ist die Kenntnis über die Programmstellen, an denen eine Verwundbarkeit erfolgreich ausgenutzt werden kann (*Punkte der Verwundbarkeitsausnutzung*). Diese Programmstellen sind der Ausgangspunkt für die Generierung einer Signatur zur Verwendung in Host-basierten Intrusion Detection Systemen. Ziel ist hierbei, dass durch die generierte Signatur erkannt wird, ob das verwundbare Programm durch Eingaben oder Aktionen (z.B. durch Nachrichten oder Kommandos) in einen verwundbaren Programmzustand überführt wird. Das hier vorgestellte Verfahren ist unabhängig von der konkret verwendeten Signaturbeschreibungssprache.

Wir stellen zunächst das allgemeine Konzept in Form der einzelnen Generierungsschritte vor. Anschließend wird eine prototypische Implementierung des Generierungsprozesses anhand von realen Verwundbarkeiten evaluiert. Der Beitrag schließt mit einer Zusammenfassung der Ergebnisse und einem Ausblick auf die zukünftigen Arbeitsschritte.

2 Generierung von Signaturen mittels Quellcodeanalyse

Der bisherige manuelle Entwicklungsprozess einer Signatur für einen neuen Angriff kann grob in vier Schritte unterteilt werden. (1) Zuerst wird der Angriff ausgeführt, um die entstehenden Spuren (*Auditereignisse*) aufzuzeichnen. Spuren sind einzelne sicherheitsrelevante Aktionen, die durch den Sensor eines IDS protokolliert werden. (2) Im Anschluss untersucht der Signaturmodellierer diese Spuren und identifiziert die zum Angriff

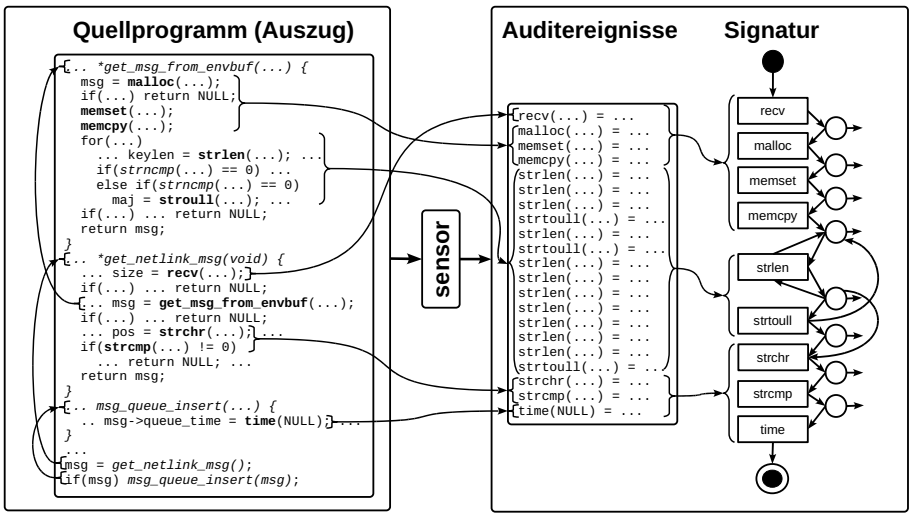


Abbildung 1: Sichtbares Programmverhalten

gehörenden *sicherheitskritischen Aktionen*. Darauf aufbauend entwickelt er (3) schrittweise die neue Signatur. Anschließend wird (4) in einer Testphase ihre Korrektheit und Präzision überprüft sowie die Signatur gegebenenfalls korrigiert. Die Modellierung des Signaturfragments, dass die Überführung eines Programms in den verwundbaren Zustand erkennt, ist dabei besonders aufwendig. Abb. 1 verdeutlicht dieses Problem an einem Beispiel. Der Quelltext beschreibt einen verwundbaren Abschnitt des Gerätedateisystemmanagers *udev* (Version 124) von Linux, der die nicht-autorisierte Erlangung von Administrationsrechten ermöglicht. Die Ereignisse, die ein typischer Host-IDS-Sensor protokollieren kann (Betriebssystem- und Bibliotheksaufrufe), sind fett dargestellt. Bei Ausführung der dargestellten Programmfunktionen entsteht eine zeitlich geordnete Sequenz beobachtbarer Auditereignisse. Das IDS muss auf Basis dieser Ereignisse und mittels der Signatur (EDL-Notation [MSK05] in Abb. 1) erkennen, ob das Programm in einen verwundbaren Zustand überführt wird. Bei der manuellen Signaturmodellierung entstehen oftmals Signaturen, die unpräzise spezifiziert sind und deshalb auch Programmabläufe erkennen, die ein zum verwundbaren Programmabschnitt vergleichbares Verhalten aufweisen, jedoch keine Verwundbarkeit beinhalten. Diese Ungenauigkeiten führen zu Fehlalarmen, die durch eine Automatisierung des Signaturgenerierungsprozesses vermieden werden sollen.

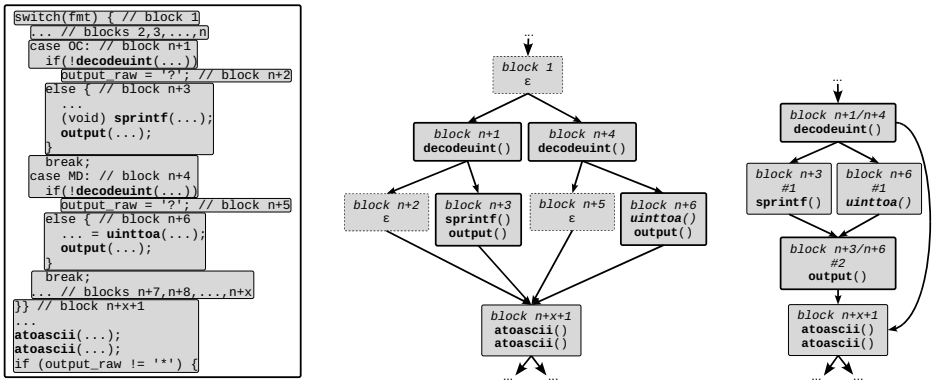
Der in diesem Beitrag vorgestellte automatisierte Signaturgenerierungsprozess lässt sich in die fünf Phasen Verwundbarkeitsanalyse, Programmmodellgenerierung, Programmmodellreduktion, Signaturgenerierung und Präzisionsabschätzung gliedern, die in den folgenden Unterabschnitten im Detail erläutert werden. Der Signaturmodellierer ist lediglich in der ersten und letzten Phase involviert.

2.1 Verwundbarkeitsanalyse

Ein erfolgreicher Angriff auf eine Verwundbarkeit erfordert in der Regel die Kompromittierung des Kontroll- oder Datenflusses der verwundbaren Anwendung mittels fehlerhafter Daten (bspw. durch Versenden von Nachrichten oder direkte Nutzereingaben). In einigen Fällen werden diese eingebrachten Daten auf ihre Integrität überprüft und die Verarbeitung im Fehlerfall abgebrochen. Infolgedessen verzweigen meist einige Kontrollflüsse in Programmabschnitte zur Fehlerbehandlung, in denen eine Ausnutzung der Verwundbarkeit nicht mehr möglich ist. Die verbleibenden Kontrollflüsse erreichen jedoch meist Programmstellen, an der die Verwundbarkeitsausnutzung unmittelbar bevorsteht oder bereits eingeleitet wird. Die entsprechenden Programmstellen werden im Folgenden als Punkte der *Verwundbarkeitsausnutzung* bezeichnet. Alle Programmkontrollflüsse, die durch einen Verwundbarkeitsausnutzungspunkt verlaufen, identifizieren gemeinsam den verwundbaren Programmabschnitt. Anhand dieser Kontrollflüsse und den daraus resultierenden Auditereignissen kann die Überführung eines Programmes in den verwundbaren Programmzustand durch ein IDS verfolgt werden. Für den weiteren Signaturgenerierungsprozess müssen demzufolge die Verwundbarkeitsausnutzungspunkte manuell festgelegt werden, was für den Signaturmodellierer nur mit geringem Aufwand verbunden ist, da sie in den veröffentlichten Sicherheitsmeldungen der Bug-Tracker oder CVE-Reports zumeist konkret benannt werden.

2.2 Programmmodellkonstruktion

Im ersten automatisierten Schritt des Signaturgenerierungsprozesses wird ein Modell des Programmverhaltens generiert. Als Ausgangsbasis dient der Programmquelltext oder der Binärcode der verwundbaren Anwendung. Die Konstruktion von Kontrollflussgraphen aus Binärcode wird bereits in [Fla04] erwähnt, kann aber bedingt durch Probleme bei der Auflösung von Funktionszeigern hier nicht verwendet werden. In Abb. 2a ist ein Beispiel für den Quelltext einer Programmfunktion dargestellt. Eine Funktion besteht aus einer Sequenz von Anweisungen, die den Kontrollfluss (bspw. Schleifen und Verzweigungen) sowie den Datenfluss (Variablenzuweisungen) eines Programms beeinflussen. Diese Anweisungssequenzen können in *Basisblöcke* mit jeweils einer einzigen Kontrollflussanweisung (typischerweise am Ende des Blocks) unterteilt werden. Nach der vollständigen Unterteilung einer Funktion in Basisblöcke (in Abb. 2a grau hinterlegt) wird aus diesen Kontrollflussanweisungen die Ausführungsreihenfolge der Basisblöcke hergeleitet. In ihrer Gesamtheit beschreiben die Vorgänger- und Nachfolgerbeziehungen zwischen den Basisblöcken einen lokalen *Kontrollflussgraph* für die untersuchte Funktion. Die Basisblöcke werden in den Knoten abgebildet und die Nachfolgerbeziehungen werden durch gerichtete Kanten dargestellt. Der Kontrollflussgraph für die Funktion der Abb. 2a ist in Abb. 2b dargestellt. Die generierten Kontrollussgraphen beschreiben zunächst nur das lokale Verhalten einzelner Programmfunktionen. Für eine Beschreibung des globalen Programmverhaltens müssen die in den lokalen Kontrollflussgraphen enthaltenen programminternen Funktionsaufrufe analysiert werden. Eine sofortige Integration der Kontrollfluss-



(a) Quellprogramm (b) Kontrollflussgraph (c) Reduktion

Abbildung 2: Generierung des Programmmodells

graphen aller aufgerufenen Funktionen erschöpft jedoch in vielen Fällen die vorhandenen Speicherressourcen. Demzufolge sind zunächst Reduktionsmaßnahmen notwendig.

2.3 Programmmodellreduktion

Die Reduktion des Programmmodells erfolgt in zwei Phasen. In der ersten Phase werden für typische IDS-Sensoren nicht sichtbare Kontrollflüsse entfernt und verzweigende Kontrollflüsse teilweise zusammengeführt. Die zweite Phase führt die Kontrollflussgraphen zur Beschreibung des lokalen Funktionsverhaltens zu einem globalen Kontrollflussgraph des Anwendungsverhaltens zusammen. Bei der Umsetzung der ersten Phase (*Reduktionsphase*) werden zunächst die für einen IDS-Sensor nicht sichtbaren Anweisungen aus den Basisblöcken der Kontrollflussgraphen entfernt. Falls ein Basisblock nach dieser Reduktion keine Anweisungen mehr enthält, wird er aus dem Kontrollflussgraph entfernt und die Vorgänger dieses Blockes werden mit dessen Nachfolgerblöcken verbunden. Anschließend wird der Kontrollflussgraph nach für IDS-Sensoren nicht zu unterscheidenden Verzweigungen durchsucht. Wenn zwei verzweigende Kontrollflüsse jeweils mit identischen Anweisungen beginnen, ist für ein IDS bei der Beobachtung der resultierenden Betriebssystem- oder Bibliotheksaufrufe zunächst nicht erkennbar, welcher Zweig ausgeführt wird. Kontrollflüsse dieser Art können reduziert werden, indem die identischen Anweisungssequenzen am Anfang der Zweige in einen gemeinsamen Basisblock zusammengefasst werden. In den Abbildungen 2b und 2c ist der beschriebene Reduktionsprozess an einem Beispiel dargestellt. Die Anweisungsblöcke der Abb. 2b enthalten ausschließlich von IDS-Sensoren protokollierbare Anweisungen. Basisblöcke, die keine protokollierbaren Anweisungen enthalten, sind mit gestrichelten Linien umrandet. Die beschriebenen Reduktionen überführen den Kontrollflussgraph der Abb. 2b in den in Abb. 2c dargestellten Kontrollflussgraph. Die Reduktion der Kontrollflussgraphen auf protokol-

lierbare Anweisungen führt dazu, dass in den nachfolgenden Generierungsschritten deutlich weniger Kontrollflüsse analysiert werden müssen.

In der zweiten Phase (*Konstruktionsphase*) werden die lokalen Kontrollflussgraphen der Programmfunktionen zu einem einheitlichen (flachen) Kontrollflussgraph vereint, der das globale Programmverhalten beschreibt. Hierfür müssen sämtliche Anweisungen, die programminterne Funktionen aufrufen, durch Kopien der Kontrollflussgraphen der aufgerufenen Funktionen ersetzt werden. Diese Integration (Abflachung der Aufrufhierarchie) wird wiederholt bis ein Kontrollflussgraph ohne programminterne Funktionsaufrufe entsteht. Um den Ressourcenbedarf (Arbeitsspeicher) der Konstruktionsphase zu minimieren, wird die Reduktionsphase für jeden in dieser Phase entstehenden Kontrollflussgraph ausgeführt. Bei der praktischen Realisierung der Kontrollflussgraphintegration muss zusätzlich beachtet werden, das die programminternen Funktionsaufrufe ebenfalls rekursiv erfolgen können. In einer prototypischen Implementierung der Konstruktionsphase wurden verschiedene Rekursionsprobleme behandelt, die hier jedoch aus Platzgründen nicht weiter erläutert werden. Zuletzt verbleibt ein einzelner Kontrollflussgraph, der das globale, von einem IDS-Sensor protokollierbare Programmverhalten beschreibt. Aus diesem Kontrollflussgraph wird im nächsten Schritt des Generierungsprozesses die Signatur abgeleitet.

2.4 Signaturgenerierung

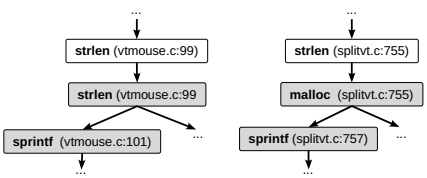
Die Signaturableitung verfolgt das Ziel, die durch die Verwundbarkeitsausnutzungspunkte verlaufenden Programmkontrollflüsse zu identifizieren, da diese einen potenziellen Angriff auf die Anwendung repräsentieren. Gleichzeitig muss sichergestellt werden, dass die generierte Signatur ausschließlich den verwundbaren Programmabschnitt erkennt und nicht die Ausführung anderer Programmabschnitte identifiziert. In Abb. 3 ist der Generierungsprozess für eine Signatur an einem Beispiel dargestellt. Der Quelltextauszug (3a) beschreibt eine Sicherheitlücke im Terminalprogramm *splitvt* [CVE10a], die eine Anhebung von Nutzerprivilegien ermöglicht. Im oberen Abschnitt der Abb. 3a ist der verwundbare Programmabschnitt dargestellt. Das letzte für ein typisches IDS wahrnehmbare Ereignis in diesem Abschnitt ist der Aufruf von *sprintf()*. Im unteren Abschnitt der Abb. 3a ist ein anderer, nicht-verwundbarer Programmabschnitt dargestellt, der ebenfalls einen Aufruf von *sprintf()* enthält. Die Kontrollflüsse, die zum ersten *sprintf()*-Aufruf führen, müssen eindeutig von den Kontrollflüssen, die zum zweiten *sprintf()*-Aufruf führen, unterschieden werden. Zu diesem Zweck werden zwei Mengen eingeführt. Die *Vulnerabilitätsmenge (*Vul*)* enthält alle zu den Verwundbarkeitsausnutzungspunkten führenden Kontrollflüsse. Die *Vergleichsmenge (*Vgl*)* enthält die Kontrollflüsse aus den nicht-verwundbaren Programmabschnitten, die bei einer Beobachtung durch einen IDS-Sensor von den verwundbaren Kontrollflüssen nicht zu unterscheiden sind. Beide Mengen werden mit den Basisblöcken der entsprechenden Kontrollflüsse initialisiert (Bsp. in Abb. 3c).

Der eigentliche Generierungsprozess läuft iterativ ab. Solange die Vergleichsmenge nicht leer ist, werden die Kontrollflüsse beider Mengen schrittweise verlängert, mit dem Ziel die Kontrollflusspfade der Vulnerabilitätsmenge von den Kontrollflusspfaden der Vergleichsmenge unterscheiden zu können. Dementsprechend werden die Elemente beider Mengen

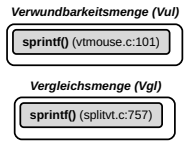
```
vtmouse.c:99-102, verwundbar:
if((strlen(...)+strlen(...)+1) > 512 )
  goto NoTitle;
sprintf(buffer, "xprop -id %s", title);
if((pipe=safe_popen(buffer, "r")) ) { ...

splitvt.c:755-757, nicht-verwundbar:
if((... malloc(strlen(...)+2)) == NULL)
  return(-1);
sprintf(temp, "-%s", args[0]);
```

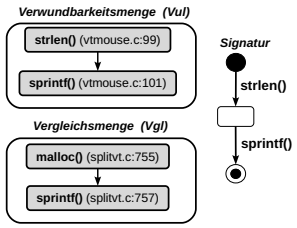
(a) Quellprogramm (Auszug)



(b) Kontrollflussgraphen (Auszug)



(c) Schritt 1 - Erzeugung der Startmengen



(d) Schritt 2 - Signaturgenerierung

Abbildung 3: Generierung der Signatur

als Untergraphen des globalen Kontrollflussgraph interpretiert. Da die Menge *Vul* mit den Verwundbarkeitsausnutzungspunkten initialisiert wurde, müssen die darin beschriebenen Kontrollflüsse entgegen der ursprünglichen Programmausführungsrichtung verlängert werden. Dementsprechend werden bei jeder Kontrollflussverlängerung für jeden Knoten der in diesen Mengen betrachteten Kontrollflussgraphausschnitte die zugehörigen Vorgängerknoten im Programmmodell ermittelt und in die Kontrollflussgraphausschnitte aufgenommen. Jeder Kontrollfluss wird somit um genau einen Basisblock erweitert. Die Abbildungen 3b und 3d zeigen die in der Vulnerabilitätsmenge und der Vergleichsmenge betrachteten Kontrollflussgraphaussschnitte (des globalen Kontrolflussgraphen) nach der ersten Verlängerung. Anschließend werden die Elemente beider Mengen paarweise miteinander verglichen. Falls zwei Kontrollflussgraphausschnitte mindestens einen identischen Kontrollfluss enthalten, werden sie als nicht unterscheidbar bewertet. Während des Vergleichsprozesses wird eine Statistik über identische und unterscheidbare Kontrollflusspfade geführt, die nach dem Abschluss des Generierungsprozesses für die Präzisionsabschätzung der Signatur genutzt wird. Aus der Vergleichsmenge werden alle Kontrollflussgraphausschnitte entfernt, die eindeutig von den Kontrollflussgraphausschnitten der Vulnerabilitätsmenge zu unterscheiden sind. Dieser Ablauf wird wiederholt, bis die Vergleichsmenge leer ist oder eine weitere Vergrößerung der Kontrollflussgraphausschnitte nicht mehr möglich ist. Im dargestellten Beispiel ist die Signaturgenerierung bereits nach der ersten Verlängerung abgeschlossen.

Im Anschluss an diesen Generierungsprozess wird die Vulnerabilitätsmenge in eine Signatur überführt, die typischerweise durch einen endlichen Zustandsautomaten beschrieben wird. Die Anweisungen in den Basisblöcken der Kontrollflussgraphausschnitte werden in Zustandsübergänge überführt und die Kanten der Kontrollflussgraphen werden in Zwischenzustände überführt. Anschließend wird ein Start- und Endzustand hinzugefügt. Die

jeweils erste Anweisung eines Kontrollflusses wird mit diesem Startzustand verbunden und die jeweils letzte Anweisung eines Kontrollflusses wird mit dem Endzustand verbunden. Im Beispiel der Abb. 3d ist eine Signatur abgebildet, die die Überführung von *splitvt* in den verwundbaren Programmzustand erkennen kann.

2.5 Präzisionsabschätzung

Nach dem Abschluss der Signaturgenerierung kann der Fall auftreten, dass der verwundbare Programmzustand nicht eindeutig zu identifizieren ist und die Vergleichsmenge nicht leer ist. Für diesen Fall wird eine Statistik geführt, die dem Signaturmodellierer die Abschätzung einer sinnvollen Kontrollflusspfadlänge und somit Signaturgröße ermöglicht. Die Präzisionsabschätzung basiert auf den Ergebnissen der Kontrollflusspfadvergleiche zwischen der Vulnerabilitätsmenge und der Vergleichsmenge. Diese Ergebnisse werden mittels der folgende Formel bewertet: $P = 1 - \#eq \,/\, (\#eq + \#diff)$.

In der dargestellten Formel entspricht *#eq* der Anzahl der in beiden Mengen identischen Kontrollflusspfade und *#diff* der Anzahl der zwischen beiden Mengen unterscheidbaren Kontrollflusspfade. Die Präzisionsabschätzung wird auf den Wertebereich [0,1] normalisiert. Bei einem Ergebnis von $P = 0$ ist die Signatur identisch zur Vergleichsmenge (geringste Präzision). Ein Ergebnis von $P = 1$ bedeutet, dass die Vulnerabilitätsmenge eindeutig von der Vergleichsmenge zu unterscheiden ist (höchste Präzision). In diesem Fall wird die resultierende Signatur keine Ungenauigkeiten bezüglich der Identifizierung des verwundbaren Programmzustandes aufweisen. Wenn die Analyse, bedingt durch die Programmstruktur, zu einem Ergebnis von $P < 1$ führt, kann der Signaturmodellierer einen Schwellwert d definieren, der eine maximal zulässige Abweichung festlegt.

Die Präzisionsbewertung beginnt nach der Erzeugung der beiden Mengen und wird nach jedem Signaturvergrößerungsschritt verfeinert, indem im Anschluss an den Mengenvergleich wieder die oben beschriebene Formel angewendet wird. Wenn die Präzisionsbewertung einen Wert von $P \geq 1 - d$ erreicht, wird die Vulnerabilitätsmenge gesichert. Diese Sicherung wird für die Generierung einer weniger präzisen Signatur genutzt, falls der Signaturgenerierungsprozess keine eindeutige Signatur generiert.

3 Anwendungsbeispiele

Um einen Eindruck über die Eignung des vorgestellten Verfahrens zu erhalten, wurde der gesamte Konstruktions- und Generierungsprozess in einem Werkzeug für C-Programme prototypisch umgesetzt und auf einem Testsystem (Quad-Core 2,83GHz) mehrfach exemplarisch evaluiert. Das implementierte Werkzeug automatisiert alle Schritte, von der Konstruktion des Programmmodells, über die Reduktion, bis hin zur Generierung der Signaturen und der Präzisionsabschätzung. Für die Evaluierung des Verfahrens wurden exemplarische Vertreter aus den Verwundbarkeitsberichten in [GLS10] ausgewählt. Dabei wurden unter anderem die folgenden zwei Beispiele evaluiert, die in diesen Berichten als kri-

tisch eingestuft wurden. Die aus der Evaluierung gewonnenen Ergebnisse sind vielversprechend.

3.1 Shadow – Anhebung von Nutzerprivilegien (CVE-2008-5394)

Im *login*-Programm (*shadow*-Version 4.0.18.1) von Linux, das für die Nutzerauthentifizierung und das Aufsetzen der Shell-Umgebung zuständig ist, existiert eine Verwundbarkeit, die das Überschreiben beliebiger Dateien erlaubt. Folgende Sicherheitswarnung wurde zum Zeitpunkt der Veröffentlichung der Verwundbarkeit herausgegeben: *„/bin/login in shadow 4.0.18.1 in Debian GNU/Linux, and probably other Linux distributions, allows local users in the utmp group to overwrite arbitrary files via a symlink attack on a temporary file referenced in a line (aka ut_line) field in a utmp entry.“* [CVE10c]

Bestimmung der Verwundbarkeitsausnutzung: Für die Ermittlung der Programmstellen, an denen eine Ausnutzung der Verwundbarkeit möglich ist, müssen lediglich die Hinweise in der Sicherheitswarnung ausgewertet werden. Die Sicherheitswarnung nennt konkret das verwundbare Programm (*/bin/login*), die Art des Angriffes (*symlink attack*) und ein Feld mit der Bezeichnung *ut_line*, das ein Bestandteil der utmp-Datenstruktur ist. Die Stelle der Verwundbarkeitsausnutzung wird nicht explizit erwähnt. Infolgedessen muss die Verwendung des *ut_line*-Feldes weiterverfolgt werden. Eine Durchsuchung der Programmquelltexte nach der Zeichenkette *ut_line* ermittelt in der Datei *login.c* den einzigen Verweis auf dieses Feld. Im *ut_line*-Feld der *utmp*-Datenstruktur wird der Pfad zur Gerätedatei des aktuell verwendeten Terminals hinterlegt. Die weitere Durchsuchung der Datei *login.c* nach der Verwendung dieses Feldes führt zu der Funktion *chown_tty()*, die die Zugriffsrechte des aktuell verwendeten Terminals ändert. Ein Angreifer kann über eine Manipulation dieses Feldes (und mit Hilfe einer symlink-Attacke) sicherstellen, dass die Zugriffsrechte einer beliebigen Datei auf die Zugriffsrechte der Gruppe *utmp* geändert werden. Der verwundbare Programmabschnitt endet mit dem Aufruf von *chown()* innerhalb der Funktion *chown_tty()*. Der Aufruf von *chown()* führt die zuletzt beschriebene Änderung der Zugriffsrechte aus und ist demzufolge mit der Ausnutzung der Verwundbarkeit gleichzusetzen.

Konstruktion und Reduktion des Programmmodells (Dauer: 1,8s): Das Programmmodell von *shadow* umfasst 305 Graphen mit insgesamt 548 Systemaufrufen und 3192 Bibliotheksaufrufen, die jeweils einzelne Programmfunktionen beschreiben. Durch die Integration und Reduktion dieser Funktionen wird der globale Kontrollflussgraph der *login*-Anwendung auf 111 Systemaufrufe und 539 Bibliotheksaufrufe reduziert.

Generierung des Signaturfragmentes und Präzisionsabschätzung: Obwohl eine einfache Zeichenkettensuche sechs Aufrufe von *chown()* innerhalb von *shadow* identifiziert, wird lediglich eine einelementige Signatur generiert: *chown()*. Die Code-Analyse identifiziert den *chown()*-Aufruf innerhalb der Datei *chowntty.c* als den einzigen Aufruf dieser Art in der *login*-Anwendung. Die anderen *chown()*-Aufrufe sind entweder ein Bestandteil anderer Hauptprogramme oder nicht erreichbar. Infolgedessen ermittelt der Prototyp bei der Präzisionsabschätzung einen Wert von $P = 1$, der gleichbedeutend mit einer eindeutigen Identifizierbarkeit des verwundbaren Programmabschnittes ist. Für die Erkennung des zu-

vor beschriebenen Angriffes ist die generierte Signatur jedoch nicht ausreichend, da der verwundbare Programmabschnitt ebenfalls bei legitimer Programmnutzung durchlaufen wird. Der Signaturmodellierer muss zusätzlich einen Vergleich des aktuell verwendeten Terminals mit dem im *ut_line*-Feld referenzierten Terminal ergänzen (bspw. über Analyse des *chown()*-Aufrufparameters).

3.2 ClamAV – DOS in rekursivem Aufrufpfad (CVE-2008-5314)

Die Antivirenlösung *ClamAV* enthält eine Verwundbarkeit, die in der folgenden Sicherheitswarnung beschrieben wurde: *„Stack consumption vulnerability in libclamav/special.c in ClamAV before 0.94.2 allows remote attackers to cause a denial of service (daemon crash) via a crafted JPEG file, related to the cli_check_jpeg_exploit, jpeg_check_photoshop, and jpeg_check_photoshop_8bim functions.“* [CVE10b]

Bestimmung der Verwundbarkeitsausnutzung: In der Meldung ist der vollständige rekursive Aufrufpfad aufgeführt, der zur Ausnutzung der Sicherheitslücke führt. Die Funktion *cli_check_jpeg_exploit* ruft zunächst die Funktion *cli_check_photoshop* auf. Im Anschluss erfolgt ein Aufruf der Funktion *cli_check_photoshop_8bim*, die wiederum die Funktion *cli_check_jpeg_exploit* aufruft (indirekte Rekursion). Der Zyklus beginnt demzufolge mit dem zuletzt genannten Aufruf innerhalb der Funktion *cli_check_photoshop_8bim* (Punkt der Verwundbarkeitsausnutzung). Unmittelbar vor diesem Aufruf erfolgt ein Betriebssystemaufruf von *lseek*, der von einem typischen IDS-Sensor protokolliert werden kann und infolgedessen als Ansatzpunkt für den Signaturgenerierungsprozess genutzt wird.

Konstruktion und Reduktion des Programmmodells (Dauer 42,3s): Das Programmmodell von *ClamAV* umfasst 931 Funktionen mit insgesamt 1335 Betriebssystemaufrufen und 4962 Bibliotheksaufrufen. Im Gegensatz zum ersten Beispiel hat der (reduzierte) globale Kontrollflussgraph des Virenscanners mit 3482 Betriebssystemaufrufen und 56448 Bibliotheksaufrufen einen gegenüber den summierten Einzelwerten größeren Umfang.

Generierung des Signaturfragmentes und Präzisionsabschätzung: Wegen der Programmstruktur von *ClamAV* sind die verwundbaren Programmabschnitte von den Vergleichsabschnitten kaum zu unterscheiden. Infolgedessen kann im *ClamAV*-Beispiel keine eindeutige Signatur generiert werden. Die Kontrollflusspfade werden im Signaturgenerierungsprozess bis zum Hauptprogramm verlängert, ohne ein unterscheidbares Ergebnis zu generieren. Der Signaturmodellierer muss in diesem Fall eine sinnvolle Signaturgröße wählen, die auf den Ergebnissen der Präzisionsabschätzung basiert. Die Tabelle 1 enthält einen Auszug aus den Ergebnissen dieser Präzisionsabschätzung. Im Vergleich zu der Anzahl der identischen Kontrollflüsse (*#eq*) wächst die Anzahl der eindeutig zu unterscheidenden Kontrollflüsse (*#cmp – #eq*) deutlich schneller. Für Kontrollflüsse mit zwei oder mehr Funktionsaufrufen konvergiert P bereits gegen die obere Grenze ($P = 1$). Wenn der Signaturmodellierer einen Schwellwert $d = 0.03$ für die Abweichung von der oberen Grenze festlegt, wird der in Abb. 4 dargestellte Untergraph generiert. Bei der praktischen Anwendung der aus diesem Graph generierten Signatur können Fehlalarme nicht gänzlich ausgeschlossen werden. Vom Signaturmodellierer sind keine weiteren Änderungen an der

Signatur notwendig, da die Identifizierung des rekursiven Aufrufpfades bereits den Angriff selbst identifiziert. Die Präzisionsabschätzung deutet an, dass die Fehlalarmrate der generierten Signatur als eher gering einzuschätzen ist.

Pfadlänge	1	2	3	4	5	6
Pfadvergleiche (*#cmp*)	1046	13541	151340	368896	1148018	2764263
Übereinstimmung (*#eq*)	1046	1222	13565	10881	12234	16946
Präzision (P)	0.000	0.901	0.910	0.971	0.989	0.994

Tabelle 1: Präzsionsabschätzung

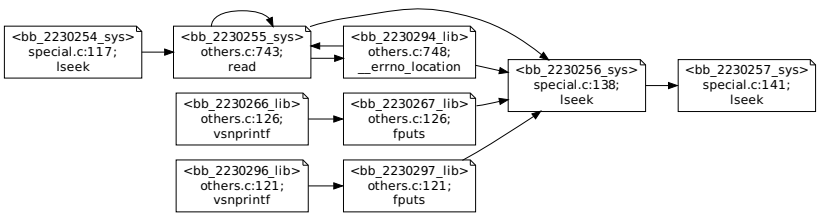


Abbildung 4: Kontrollflussgraphausschnitt der Pfadlänge 4

4 Zusammenfassung und Ausblick

Der in diesem Beitrag vorgestellte Signaturableitungsprozess generiert aus dem Quelltext verwundbarer Anwendungen automatisiert Signaturen für Host-basierte IDS. Diese Signaturen können die Überführung einer Anwendung in einen verwundbaren Programmzustand erkennen. Zu Beginn des Generierungsprozess werden lediglich wenige Informationen über verwundbare Programmstellen benötigt, die in fast allen Verwundbarkeitsberichten und Bug-Trackern verfügbar sind. Eine Evaluierung des Prozesses mittels realer Verwundbarkeiten hat zusätzlich gezeigt, dass die Präzisionsabschätzung eine Festlegung von sinnvollen Signaturgrößen ermöglicht, wenn Fehlalarme nicht gänzlich vermieden werden können. Die in diesem Beitrag exemplarisch evaluierten Verwundbarkeiten deuten desweiteren an, dass typischerweise kleine Signaturgrößen zu erwarten sind.

Eine Generierung von Signaturen, für die Fehlalarme völlig ausgeschlossen werden können, erfordert weiterführende Untersuchungen. Offen ist zum Beispiel, ob eine zusätzliche Datenflussanalyse im Programmquelltext die Genauigkeit der generierten Signaturen verbessert. In einem nächsten Schritt könnte ein IDS implementiert werden, das auf den Prinzipien schneller Sandbox-Verfahren wie z.B. *sydbox* [syd10] basiert. In Sandbox-basierten IDS können Angriffe unterbrochen werden, indem die Ausnutzung der Verwundbarkeit verhindert wird.

Literatur

[BNS+06] David Brumley, James Newsome, Dawn Song, Hao Wang, and Somesh Jha. Towards Automatic Generation of Vulnerability-Based Signatures. In *IEEE Symposium on Security and Privacy*, pages 2–16, CA, USA, 2006. IEEE Computer Society.

[CCZ+07] Manuel Costa, Miguel Castro, Lidong Zhou, Lintao Zhang, and Marcus Peinado. Bouncer: Securing Software by Blocking Bad Input. In *Proceedings of the 21st ACM Symposium on Operating systems principles*, pages 117–130, USA, 2007. ACM.

[CPWL07] Weidong Cui, Marcus Peinado, Helen J. Wang, and Michael E. Locasto. ShieldGen: Automatic Data Patch Generation for Unknown Vulnerabilities with Informed Probing. In *IEEE Symposium on Security and Privacy*, pages 252–266, Berkeley, CA, 2007. IEEE Computer Society.

[CVE10a] Common Vulnerabilities and Exposures – CVE-2008-0162. `http://cve.mitre.org/cgi-bin/cvename.cgi?name=2008-0162`, January 2010.

[CVE10b] Common Vulnerabilities and Exposures – CVE-2008-5314. `http://cve.mitre.org/cgi-bin/cvename.cgi?name=2008-5314`, January 2010.

[CVE10c] Common Vulnerabilities and Exposures – CVE-2008-5394. `http://cve.mitre.org/cgi-bin/cvename.cgi?name=2008-5394`, January 2010.

[FHSL96] Stephanie Forrest, Steven A. Hofmeyr, Anil Somayaji, and Thomas A. Longstaff. A Sense of Self for Unix Processes. In *IEEE Symposium on Security and Privacy*, pages 120–128, Oakland, CA, USA, 1996. IEEE Computer Society.

[Fla04] Halvar Flake. Structural Comparison of Executable Objects. In Ulrich Flegel and Michael Meier, editors, *DIMVA*, volume 46 of *LNI*, pages 161–173. GI, 2004.

[GJM04] Jonathon T. Giffin, Somesh Jha, and Barton P. Miller. Efficient Context-Sensitive Intrusion Detection. In *Proceedings of the Network and Distributed System Security Symposium*, San Diego, California, USA, 2004. The Internet Society.

[GLS10] Gentoo Linux Security Advisories. `http://www.gentoo.org/security/en/glsa/index.xml`, January 2010.

[MSK05] Michael Meier, Sebastian Schmerl, and Hartmut König. Improving the Efficiency of Misuse Detection. In *DIMVA 2005*, volume LNCS 3548, pages 188–205, Vienna, Austria, July 2005. Springer.

[NKS05] James Newsome, Brad Karp, and Dawn Song. Polygraph: Automatically generating signatures for polymorphic worms. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 226–241, CA, USA, 2005. IEEE Computer Society.

[SBD01] R. Sekar, M. Bendre, and D. Dhurjati. A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors. In *IEEE Symposium on Security and Privacy*, pages 144–155, Oakland, CA, USA, 2001. IEEE Computer Society.

[syd10] sydbox. `http://projects.0x90.dk/wiki/sydbox`, February 2010.

[WD01] David Wagner and Drew Dean. Intrusion Detection via Static Analysis. In *IEEE Symposium on Security and Privacy*, pages 156–168, Oakland, CA, USA, 2001. IEEE Computer Society.

[WFGPM09] T. Werner, C. Fuchs, E. Gerhards-Padilla, and P. Martini. Nebula - Generating Syntactical Network Intrusion Signatures. In *Proc. of the 4th International Conference on Malicious and Unwanted Software*, Montreal, Canada, October 13-14 2009.

Sichere Datenhaltung im Automobil am Beispiel eines Konzepts zur forensisch sicheren Datenspeicherung

Tobias Hoppe[1], Sönke Holthusen[2], Sven Tuchscheerer[1], Stefan Kiltz[1], Jana Dittmann[1]
[1] Arbeitsgruppe Multimedia and Security, Institut ITI, Fakultät für Informatik
[1,2] Otto-von-Guericke Universität Magdeburg
Universitätsplatz 2
39106 Magdeburg
{tobias.hoppe, sven.tuchscheerer, stefan.kiltz, jana.dittmann}@iti.cs.uni-magdeburg.de

Abstract: Seit mehreren Jahren wird in der Forschung auf potentielle Gefahren hingewiesen, die vorsätzliche Angriffe auf automotive IT-Systeme bergen [HKD08]. Aktuelle Arbeiten wie [Ko10] zeigen, dass Auswirkungen bis hin auf die leibliche Sicherheit (Safety) der Insassen zunehmend realistisch werden. Moderne automotive IT verarbeitet und speichert zunehmend auch personenbeziehbare Daten, so dass zusätzlich Beeinträchtigungen der Privatsphäre drohen. Folglich stellt die automotive IT-Sicherheit, d.h. der Schutz derartiger sensibler Daten vor unautorisiertem Ausspähen und Manipulationen ein wichtiges Forschungsziel dar. Diese Problematik adressierend, behandelt der vorliegende Beitrag Aspekte der sicheren Datenhaltung im Automobil. In diesem Beitrag wird ein kombiniertes Konzept zur sicheren Datenspeicherung vorgeschlagen und eine wissenschaftliche Diskussion angeregt. Es basiert auf dem bestehenden Konzept elektronischer Fahrzeugdatenschreiber, deren bisherige Lösungen jedoch meist primär zur Aufklärung *safety*-bezogener Ereignisse (insbesondere Verkehrsunfälle) konzipiert sind. Bedrohungen seitens der *Security* (z.B. Manipulationen von Daten im Gesamtfahrzeug) werden häufig nicht berücksichtigt. Der vorgestellte kombinierte Ansatz ist zusätzlich als strategische Vorbereitung für ggf. zukünftig folgende, IT-forensische Aufklärungen geeignet und adressiert somit auch Vorfälle, welche auf Verletzungen der Security zurückzuführen sind. Ziel ist, ein Konzept mit besonderem Fokus auf unterschiedliche Nutzer und deren variierende Sicherheitsanforderungen an die gesicherten Daten (wie Integrität, Authentizität und Vertraulichkeit) zu diskutieren.

1 Einführung / Motivation

Bei der Behandlung von Sicherheitsrisiken wird nicht nur im Automobilbereich zwischen zwei Ausprägungen der *Sicherheit* unterschieden. Ein sicherer Betrieb im Sinne der Funktionssicherheit (engl. *Safety*) berücksichtigt potentielle Komponentenausfälle oder auftretende Störungen und soll insbesondere die körperliche Unversehrtheit des Menschen bewahren. Die Informationssicherheit (IT *Security*) hat hingegen das Ziel, Vorkommnisse des unautorisierten Ausspähens und Manipulierens von Daten zu verhindern, erkennen bzw. behandeln zu können. Sie richtet sich damit gegen vorsätzliche Ereignisse und dient primär dem Schutz von Informationen.

Mit ihrem starken Einfluss auf die Unversehrtheit von Leib und Leben des Menschen sind moderne Automobile im Sinne der *Safety* bereits als sicherheitsrelevante Systeme

anerkannt. Die in Form vielzähliger Steuergeräte verbauten IT-Komponenten bieten bereits heute Möglichkeiten, ein Versagen sowie potentielle Fehlfunktionen verschiedenster automotiver (Teil-)Systeme erkennen zu können. Hierzu gehören beispielsweise digitale Fehlercodes (engl. DTC, Diagnostic Trouble Codes), die sich ein Steuergerät nach Detektion eines auffälligen Zustandes vermerkt und die anschließend (z.B. in der Werkstatt im Rahmen der regelmäßigen Inspektion) über die Diagnoseschnittstelle ausgelesen und zur Ursachenanalyse herangezogen werden können. Fehlercodes werden meist bei Ereignissen generiert, die im Vorfeld durch die Entwickler als potentielle Fehlzustände definiert wurden und sind damit hauptsächlich für ungewünschte Vorkommnisse konzipiert, die zufällig, sporadisch oder systematisch auftreten können.

Im Rahmen der seit jeher betriebenen vorsätzlichen Manipulationen an Fahrzeugsystemen werden zunehmend auch Veränderungen ihres Datenstandes betrieben. Derartige Verletzungen der *Security* können durch die bisherigen Einrichtungen zur Fehlzustandserkennung aktuell jedoch nicht bzw. nur sehr eingeschränkt erkannt werden (z.B. weil vorsätzliche Eingriffe nicht als relevante Ereignisse definiert wurden oder vorhandene Prüfbedingungen durch gezielte Manipulationen nicht verletzt werden). Gerade im Fall des safety-relevanten Automobils kann der Nachweis vorsätzlicher, unautorisierter Eingriffe von essentieller Bedeutung sein, wie auch Literatur aus dem Bereich der Unfallforschung und –rekonstruktion unterstreicht: *„Für den Sachverständigen ist es wichtig zu erkennen, ob an dem von ihm zu untersuchenden Fahrzeug bzw. Steuergerät Tuningmaßnahmen vorgenommen wurden."* ([BM09], S. 826). Während Tuning-Maßnahmen (vgl. auch [Bo08], Kapitel „Selbstbau und Tuning") im Automobilbereich eines der bekanntesten Felder teils unautorisierter Manipulationen darstellen, zeichnen sich weitere, modernere Ausprägungen IT-basierter Angriffe auf automotive Systeme ab. Einige Beispiele aus Forschung und Praxis sind das Vortäuschen der korrekten Funktion defekter oder entfernter Airbags [HKD08], das Senden gefälschter Verkehrsinformationen [BD07], unautorisierte (De-)Aktivierung der Bremsen [Ko10] oder das aus dem Internet initiierte, unberechtigte Lahmlegen einer gesamten Fahrzeugflotte per Funk [MFA10].

1.1 Forschungsziel und Stand der Technik

Ein wichtiges Forschungsziel stellt daher dar, in Zukunft auch die Aufklärung derartiger vorsätzlicher Eingriffe zu fördern. Als ein Beispiel finden sich in bestehenden Arbeiten bereits Untersuchungen zur Anpassung von Intrusion Detection Systemen (IDS) auf die automotive Domäne, um Angriffe auf automotive IT zunächst erkennen (vgl. [HKD09, MHD10]) und potentiell schwerwiegende Folgen einschränken zu können.

In Fokus dieses Beitrags steht die sichere Protokollierung / Speicherung fahrzeuginterner Daten um im Fall potentieller Vorfälle ggf. folgende IT-forensische Untersuchungen und damit deren Aufklärung erheblich unterstützen zu können. Derartige, vorab eingerichtete prophylaktische Vorkehrungen für IT-basierte Systeme (hier: das Fahrzeug) werden im Prozessmodell zur IT-Forensik nach [KHA09] auch als strategische Vorbereitung bezeichnet. Doch auch abseits der IT-Forensik gibt es sinnvolle Anwendungen für ein entsprechendes Datensicherungssystem. Ausgewählte Beispiele und jeweils relevante Anforderungen/Schutzziele an die Datenspeicherung werden in der Folge mit identifi-

ziert und einbezogen. Basierend auf Ergebnissen aus [Ho10] und ergänzt durch zusätzliche Betrachtungen wird in diesem Beitrag ein entsprechend erweitertes Konzept für ein solches System vorgeschlagen und diskutiert, das im weiteren Verlauf des Beitrags als „Forensischer Fahrzeugdatenschreiber“ (FFDS) bezeichnet wird.

Im Kontext einer prophylaktischen, automotiven Datenprotokollierung ist als Stand der Technik insbesondere auf das bestehende Konzept des Unfalldatenschreibers (UDS) zu verweisen. Für den speziellen Anwendungsfall der Unterstützung einer Unfallrekonstruktion speichert ein UDS ausgewählte Informationen (z.B. Geschwindigkeiten, Längs-/Querbeschleunigungen, Blinker-/Bremsvorgänge) i.d.R. nur für ein kurzes Zeitfenster; laut [BM09] genügen in der Regel z.B. ca. 45 Sekunden, um eine erhebliche Erleichterung bei der Unfallrekonstruktion zu erzielen. Entsprechende Produkte sind bereits seit mehreren Jahren verfügbar (z.B. Verweise in [Ha03]). Auch trotz Standardisierungsbemühungen in verschiedenen Ländern (z.B. bzgl. der Bereitstellung aufgezeichneter Daten [EDR06]) konnten sich UDS bisher nicht im breiten Einsatz durchsetzen.

Dieser Beitrag gliedert sich wie folgt: Im folgenden Abschnitt 2 werden fünf exemplarische Rollen von Nutzern eines entsprechenden Systems vorgestellt und ihre Anforderungen an die Datensicherung diskutiert. Ein Vorschlag für ein entsprechendes Konzept wird in Abschnitt 3 vorgestellt und in Abschnitt 4 anhand der prototypischen Umsetzung und ausgewählten Beispielen illustriert. Eine Diskussion zur Erfüllung der Anforderungen und ausgewählter Vor- und Nachteile folgt in Abschnitt 5, bevor eine Zusammenfassung und ein Ausblick in Abschnitt 6 diesen Beitrag schließt.

2 Ausgewählte Anwendungsfälle und ihre Anforderungen

Bzgl. der durch das FFDS vorgenommenen Datensicherung aus Fahrzeugen stehen im Kontext dieses Beitrags primär digitale Daten im Fokus, die intern über Feldbustechnologien wie CAN, LIN, MOST oder FlexRay [ZS08] ausgetauscht werden. Eine regelmäßige Protokollierung ausgewählter Daten kann zu unterschiedlichen Anwendungszwecken relevant sein. Dieser Abschnitt stellt fünf beispielhafte Rollen vor und diskutiert deren Anforderungen hinsichtlich des Schutzbedarfs der zu speichernden Logdaten.

2.1 Exemplarische Rollen

Rolle R1 — Fahrzeugbesitzer: Bereits für den Besitzer selbst, der oft der hauptsächliche Nutzer des Fahrzeuges ist, bieten sich sinnvolle Anwendungszwecke für ein entsprechendes Logging-System, etwa zur Erstellung eines digitalen Fahrtenbuchs oder Statistiken zu seiner Fahrweise, z.B. um Kosten und Schadstoffemissionen zu minimieren. Auch kann er die geloggten Daten als (entlastendes) Beweismaterial nutzen wollen – z.B. im Falle ihm vorgeworfener Verkehrsdelikte oder um (auch in anderweitigen Streitfällen) die (Nicht-)Nutzung des Fahrzeug zum fraglichen Zeitpunkt belegen zu können.

Rolle R2 — Versicherung: Versicherungen bieten zunehmend flexiblere Tarifmodelle, die Fahrgewohnheiten und –verhalten mit einbeziehen (z.B. bzgl. Wegstrecken, Ge-

schwindigkeiten, Fahrtzeiten oder Lokalitäten), wozu entsprechende Daten für die Versicherung zu protokollieren sind. Als eine frühe Realisierung dieses „Pay as you drive" Prinzips sieht z.B. das System MyRate [MR10] ein an der On-Board-Diagnose (OBD) Schnittstelle platziertes Gerät zur Protokollierung vor.

Rolle R3 — Unfallrekonstruktion: Die teils nach Verkehrsunfällen durchgeführte Unfallrekonstruktion hat das Ziel, u.a. anhand des Spurenbildes den Unfallablauf möglichst genau rekonstruieren zu können, insbesondere zur Klärung von Schuldfragen. Auch Daten aus den beteiligten Fahrzeugen können hierzu wertvoll sein. Zusätzlich zu den Leistungsmerkmalen eines UDS (Abschnitt 1.1) kann ein FFDS diese Aufgabe ebenso erfüllen und dem Sachverständigen noch zusätzlichen Nutzen bieten (z.B. Zugriff auf die Vorgeschichte des Fahrzeuges).

Rolle R4 — IT-Forensiker: Insbesondere auch angesichts zukünftiger Funkkommunikation zwischen Fahrzeugen sowie zur Infrastruktur könnten gezielte Angriffe auf automotive IT auch von externer Seite weiter zunehmen (vgl. auch [Ko10]), die teils einer nachträglichen Aufklärung nach Prinzipien der IT-Forensik [KH02] bedürfen. Eine erhebliche Unterstützung des IT-Forensikers kann hierbei insbesondere durch eine prophylaktische, forensik-konforme Protokollierung derartiger Daten geboten werden (vgl. Abschnitt 1.1), die auf das Eindringen eines Angreifers hinweisen können oder durch ihn potentiell ausgeführte Aktionen aufzeigen. Dies kann z.B. Zeit und Kontext etablierter Verbindungen (z.B. Diagnose oder Funk) oder die Ansteuerung ausgewählter Aktoren umfassen. Auch zur Laufzeit nicht näher spezifizierbare Anomalien können als potentielle Anzeichen security-relevanter Aktivitäten vorsorglich gesichert werden.

Rolle R5 — Werkstatt: Ergänzend zu bestehenden Fehlercodes (s. Abschnitt 1) können regelmäßig protokollierte Informationen und Anomalien auch Werkstätten bei der Ursachenanalyse rein safety-bezogener Störungen unterstützen. Während Fehlercodes oft nur Ort, Zeitpunkt und Art eines auffälligen Zustandes vermerken, kann in den Logdaten gezielt nach zeitlich und kausal korrelierenden Ereignissen gesucht werden. Dies kann die Behebung von Mängeln beschleunigen und so auch zu einer Kostenersparnis führen.

2.2 Sicherheitsanforderungen an das FFDS bzgl. der vorgestellten Anwendungsfälle

Die Diskussion der Sicherheitsanforderungen erfolgt in diesem Teilabschnitt exemplarisch für die Schutzziele Authentizität, Integrität und Vertraulichkeit. Die Betrachtungen erfolgen zusammenfassend für die internen (Besitzer/Nutzer des Fahrzeuges; R1) und externen Nutzer (dritte Parteien; R2-R5) des FFDS, da diese jeweils in ihren wesentlichen Anforderungen hinsichtlich des Schutzes der gesicherten Daten übereinstimmen.

Zunächst ist festzustellen, dass die Sicherheitseigenschaften des FFDS auch für den Fahrzeugbesitzer (bzw. die Nutzer des Fahrzeuges) von großer Bedeutung sind - z.B. gegenüber einem alternativ denkbaren ungeschützten Logging: Angesichts des großen Anteils personenbezogener und –beziehbarer Daten in aktuellen Automobilen kann als primäre Sicherheitsanforderung des Besitzers die Vertraulichkeit der gespeicherten Daten angesehen werden. Diese sollten daher gegen das Auslesen durch unberechtigte Personen geschützt sein. Als sekundäre Anforderungen seinerseits können die Integrität und

die Authentizität der Logdaten hinzu kommen, wenn er im Einzelfall die Möglichkeit haben will, für ihn gespeicherte Daten als Beweismaterial einsetzen zu können.

Für die externen Parteien sind die Integrität und Authentizität der Logdaten als primäre Anforderung zu sehen: Zum verlässlichen Durchführen ihrer Aufgabe (d.h. zur Ermittlung der Versicherungsbeträge, Rekonstruktion von Unfällen und IT-Vorfällen sowie zur Fehlersuche) dürfen diese Daten weder durch Nutzer oder Dritte verfälscht worden sein, noch sollten integre Daten aus einem anderen Fahrzeug/FFDS unerkannt eingebracht werden können. Im Falle des IT-Forensikers leiten sich beide Schutzziele direkt aus den forensischen Prinzipien ab [KH02]. Die Vertraulichkeit der Logdaten ist für die Aufgabe der externen Rollen i.d.R. nicht von Bedeutung; deren Interesse an einer Wahrung kann als maximal sekundär eingeschätzt werden (ggf. bestehen jedoch gesetzliche Vorgaben).

Nutzergruppe	Primäre Schutzziele	Sekundäre Schutzziele	Wesentliche Kategorien potentiell zu erwartender Angriffe / Angreifer
Intern (R1)	Vertraulichkeit	Integrität, Authentizität	Einsehen der Daten durch unberechtigte Dritte
Extern (R2-R5)	Integrität, Authentizität	Vertraulichkeit	R2-R3: Manipulation durch interne Nutzer R2-R5: Manipulation durch unberechtigte Dritte

Tabelle 1: Übersicht über relevante Beispiele für gestellte Schutzziele und erwartete Angreifer

Die zuvor erfolgten Abschätzungen zur Relevanz der betrachteten Schutzziele aus Sicht der vorgestellten Rollen werden in Tabelle 1 gegenübergestellt. Letztendlich sind die bei einer Realisierung des FFDS zu beachtenden Schutzziele jedoch im Kontext des Gesamtsystems zu sehen: Beispielsweise ist die Anforderung des Besitzers/R1 an die vertrauliche Speicherung auch auf Datensätze anzuwenden, die für Nutzer anderer Rollen (z.B. die Werkstatt/R5) gespeichert werden – auch wenn diese Anforderung aus deren Sicht nicht relevant ist. Folglich sollte die Vertraulichkeit, Integrität und Authentizität protokollierter Daten durch einen FFDS gleichermaßen durchgängig gesichert werden.

3 Konzept

Das vorgeschlagene Konzept für den FFDS, das den folgenden Betrachtungen zugrunde liegt, ist in der folgenden Abbildung 1 skizziert. Es ist so gestaltet, dass es bereits auf ein heutiges automotives System anwendbar ist, ohne dass dieses anwendungsspezifische Anforderungen erfüllen muss. Dazu wird dem Fahrzeug ein zusätzliches Gerät (mittlerer Block in Abbildung 1) hinzugefügt, das zwecks lesenden Zugriffs an dessen interne Busnetzwerke (linker Teil in Abbildung 1) angeschlossen wird. Busnachrichten aus allen relevanten Teilnetzwerken, die ggf. auch unterschiedliche Feldbustechnologien verwenden, werden von einem *BusListener* entgegengenommen und mit dem Zeitstempel des Eingangs versehen. Zur weiteren, parallelen Bearbeitung werden sie anschließend an zwei Komponenten weitergereicht: Der *Filter* hat die Aufgabe, nach einem festen Regelwerk (anhand rollenindividueller Filterregeln, siehe Abschnitt 4) die für jeden auf dem FFDS registrierten Nutzer relevanten Informationen zur Speicherung zu selektieren (bzw. irrelevante auszublenden). Der *Detektor* untersucht die aktuelle Kommunikation parallel auf Anomalien, die sich allein auf Basis der statischen Filterregeln nicht abdecken lassen (z.B. in Form eines automotiven IDS / siehe Abschnitt 1.1). Dies kann z.B.

die Detektion von Angriffsmustern (vgl. [HKD09]) sowie auch allgemeine Auffälligkeiten umfassen, die für einige Rollen ebenfalls relevant sein können – z.B. ein Aussetzen der Stromversorgung, das potentiell auf einen Versuch des Verschleierns einer Manipulation hinweist. In solchen Fällen kann zudem auch der aktuelle Loglevel erhöht werden.

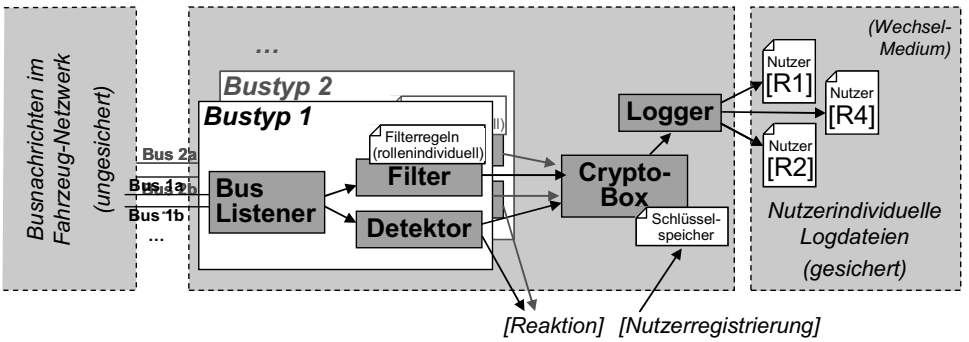


Abbildung 1: Vorgeschlagenes Konzept des forensischen Fahrzeugdatenschreibers (FFDS)

Die zur Speicherung selektierten Nachrichten und Ereignisse werden anschließend von der *CryptoBox* kryptographisch gesichert, um sie danach unter Einhaltung der geforderten Schutzziele (vgl. Abschnitt 2) durch den *Logger* in nutzerindividuellen Logdateien auf einem Wechseldatenträger (z.B. SD-Karte) ablegen zu können.

3.1 Das zugrundeliegende kryptographische Konzept

Zur Einhaltung der geforderten Sicherheitseigenschaften ist das kryptographische Konzept entscheidend, das seitens des FFDS in der CryptoBox zu implementieren ist.

Zur Sicherstellung der Vertraulichkeit der gesicherten Daten sieht das Konzept die Verschlüsselung der zu sichernden Einträge vor, um unbefugte Einsicht möglichst einzuschränken. Hierzu nutzt das Konzept die Vorteile asymmetrischer Verschlüsselungsverfahren: Bei der Registrierung eines neuen Nutzers wird sein Public Key im Schlüsselspeicher des FFDS hinterlegt, mit dem der FFDS jederzeit Daten verschlüsselt in der Logdatei des Nutzers ablegen kann. Um den Ressourcenbedarf gering zu halten (sowohl bzgl. Rechenleistung als auch Speicherplatz für die Chiffrate) kommt hierbei ein hybrides Verfahren zum Einsatz, bei der die Logeinträge über ein symmetrisches Verschlüsselungsverfahren chiffriert werden und lediglich die hierzu verwendeten, regelmäßig wechselnden Session-Keys asymmetrisch verschlüsselt mit abgelegt werden [Bu10].

Der FFDS hat die Authentizität und Integrität der Logdaten zusätzlich zu sichern, denn sonst könnte z.B. der Fahrzeugbesitzer mit dem Public Key der Versicherung Logdateien mit beliebigen verschlüsselten Inhalten auch außerhalb des FFDS selbst erzeugen. Die Versicherung muss daher überprüfen können, ob die z.B. jährlich bereitgestellten Logdateien wirklich durch den angemeldeten FFDS generiert wurden sowie unverfälscht sind. Dazu wird für jeden FFDS zusätzlich ein eigenes asymmetrisches Schlüsselpaar generiert. Der Private Key muss dazu sicher im Schlüsselspeicher innerhalb des FFDS verwahrt werden (dieser kann z.B. als Smartcard realisiert sein) und wird zum Signieren der

generierten Logeinträge verwendet (was ebenfalls blockweise erfolgt). Die Zugehörigkeit des entsprechenden Public Keys zu seinem FFDS wird dem Fahrzeugbesitzer durch den Hersteller des FFDS auf einem Zertifikat bescheinigt. Über dieses kann er z.B. der Versicherung, die er als Nutzer auf seinem FFDS registrieren möchte, den Public Key des FFDS nachweisbar für spätere Integritäts- und Authentizitätsprüfungen überlassen.

Soll der FFDS zusätzlich die Authentizität von Nutzern überprüfen können, die sich in einer unterstützten Rolle zum Aufzeichnen von Fahrzeugdaten anmelden, kann dies über eine vertrauenswürdige, dritte Instanz erfolgen. Als ein Beispiel möchte sich am FFDS ein neuer Nutzer in der Rolle ‚Versicherung' anmelden. Neben seinem Public Key muss er dann auch ein Zertifikat vorweisen, in der eine übergeordnete Instanz bescheinigt, dass es sich bei ihm um eine zugelassene Versicherung handelt. Der FFDS müsste für die Verifikation mit dem Public Key dieser Instanz ausgestattet sein, bei der es sich in einem einfachen Fall um den Hersteller des FFDS handeln kann. Äquivalent ließe sich über Sperrzertifikate auch ein Revocation-Management umsetzen (z.B. über zukünftige Car-to-Infrastructure Kommunikation).

Eine exemplarische Umsetzung des Konzeptes sowie eine detailliertere Diskussion bzgl. ausgewählter Vor- und Nachteile stehen im Fokus der folgenden Abschnitte 4 und 5.

4 Exemplarische Umsetzung und Illustration

Im Rahmen von [Ho10] wurde ein erster Prototyp als exemplarische Umsetzung des Konzeptes implementiert. Diese verwendet das Bussystem Controller Area Network (CAN), da es im Automobilbereich die zurzeit am häufigsten eingesetzte Feldbus-Technologie darstellt [ZS08]. Aus Anwendersicht besteht eine CAN-Botschaft im Wesentlichen aus einer numerischen ID (sog. CAN-ID), die gleichzeitig den Typ der Nachricht sowie ihre Priorität repräsentiert, sowie zwischen 0 und 8 Bytes an Nutzdaten. In den Nutzdaten werden i.d.R. mehrere Informationen, sog. Signale, gebündelt. Abbildung 2 illustriert den Aufbau der Nutzdaten einer fiktiven CAN-Botschaft (CAN-ID 0x3B7, 3 Nutzdatenbytes `A3 C1 53`), über die Signale zur Anzeige an das Kombiinstrument gesendet werden. In diesem Beispiel sind dies die aktuelle Geschwindigkeit (8 Bit) sowie je ein 1-Bit-Flag zum Setzen der Airbagwarnleuchte und zum Signalisieren eines eingehenden Anrufs. Ist die Rufnummer des Anrufers im Telefonbuch gespeichert, wird in letzterem Fall zusätzlich eine Kennung zur Anzeige seines Namens angegeben.

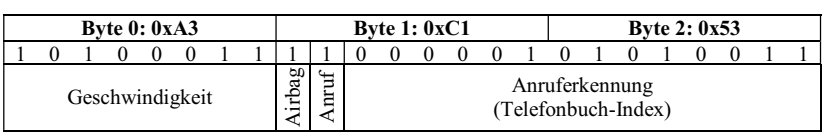


Abbildung 2: Aufteilung der Nutzdaten einer fiktiven CAN-Botschaft des Typs 0x3B7

Wie zuvor anmotiviert, benötigen verschiedene Rollen für ihre Zwecke nur einen Teil der kommunizierten Informationen und sollten nur entsprechende Zugriffe erhalten (Prinzip des notwendigen Wissens, vgl. [Eck03]). Am Beispiel der Busnachrichten o.g. Typs liefert Tabelle 2 eine exemplarische Zuordnung der enthaltenen Signale für die

Rollen Besitzer (R1), Werkstatt (R5) und Versicherung (R2, für die restlichen Rollen aus Abschnitt 2 ließe sich dies äquivalent vornehmen). Beispielsweise sollten sensible Daten wie Anruferkennungen (personenbeziehbare Daten auf Dritte) im Falle des Loggens nur einem begrenzten Personenkreis (im Wesentlichen dem Fahrzeugnutzer selbst) zugänglich sein. Während für Versicherungen z.B. bei Schadensfällen die Tatsache eines Anrufs als anonyme, binäre Information von Interesse sein kann, sollten für eine Werkstatt beide Angaben irrelevant sein. Dies wird durch den Filter (s.u. und Abb. 1) berücksichtigt. Damit der Vertraulichkeitsgewinn durch die rollenindividuelle Filterung für den Fahrzeugbesitzer transparent ist, muss der verwendete Regelsatz öffentlich sein.

	Besitzer (R1)	**Werkstatt (R5)**	**Versicherung (R2)**	**…**
Aktuelle Geschwindigkeit	X	X	X	…
Ereignis: Fehlfunktion Airbagsystem	X	X	-	…
Ereignis: eingehender Anruf	X	-	X	…
Anruferkennung (Telefonbuch-Index)	X	-	-	…

Tabelle 2: Beispiele für relevante Informationen einzelner Rollen (bzgl. CAN-Nachricht 0x3B7).

Aufgrund der bereits vorhandenen Unterstützung des CAN Protokolls erfolgt die Implementierung zunächst als C++ Programm unter Linux. Der ***BusListener*** für die gewählte Feldbustechnologie CAN verwendet die im Linux-Kernel zur Verfügung stehende SocketCAN Schnittstelle[1] und gibt die empfangenen Pakete samt Zeitstempel und Kennung des Eingangskanals zur weiteren Bearbeitung weiter. Der ***Filter*** wendet auf diese gemäß dem o.g. Prinzip rollenindividuelle Filterregeln an. Diese enthalten für jeden zu sichernden Nachrichtentyp eine Bitmaske, die alle bis auf die relevanten Bits ausblendet (Verknüpfung über logisches Und). Laut Tabelle 2 besteht z.B. für die Rolle Werkstatt (R5) z.B. eine Filterregel für die CAN-ID *0x3B7* mit der Filtermaske *FF 80 00* (Bits: *11111111 10000000 00000000*), die die Nutzbytes *A3 C1 53* der CAN-Nachricht aus Abbildung 2 zu *A3 80 00* filtert (Bits: *10100011 10000000 0000000*). Als zu sichernder Logeintrag wird jeweils eine Klartextzeile erstellt, die den Zeitstempel und Kanal des Eingangs, die CAN-ID und die gefilterten Inhalte umfasst (vgl. Abbildung 3). Bei der Implementierung der ***CryptoBox*** wurde die Bibliothek Botan[2] genutzt. Von den durch sie bereitgestellten kryptographischen Algorithmen wurden für die Umsetzung des Konzepts exemplarisch die aktuell als sicher eingestuften Verfahren AES-256 (symmetrisch), RSA-4096 (asymmetrisch) sowie SHA-256 (Hashfunktion) gewählt (vgl. auch [Bu10]). Die Verschlüsselung folgt dem im Abschnitt 3 vorgestellten Konzept. Da die (hier auf geloggten und gefilterten CAN-Nachrichten basierenden) Protokolleinträge blockorientiert sind, erfolgt die symmetrische AES-Verschlüsselung als Blockchiffre im CBC Modus (Cipher Block Chaining). Dies erschwert einerseits Offline-Angriffe auf die verschlüsselten Daten (im ECB Modus (Electronic Code Book) werden gleiche Klartextblöcke grundsätzlich als identische Chiffreblöcke abgelegt) und ist andererseits ein guter Kompromiss hinsichtlich der Performanz (im Vergleich zum OFB und CFB Modus, vgl. auch [Bu10]). Die von der CryptoBox gesicherten Logeinträge werden abschließend durch den ***Logger*** in den nutzerindividuellen Logdateien abgelegt. Wie Ab-

[1] siehe http://developer.berlios.de/projects/socketcan/
[2] siehe http://botan.randombit.net/

bildung 4 illustriert, wurde für diese ebenfalls ein Textformat gewählt, bei der zeilenweise ein Header den Typ des Eintrags anzeigt und die verschlüsselten Nutzdaten in Base-64-Kodierung[3] folgen. Entsprechend des in Abschnitt 3 beschriebenen Konzepts wird ein neuer, verschlüsselter Session-Key in einer `::key::` Zeile abgelegt. Nach dessen Entschlüsselung mit seinem Private Key kann der zugehörige Nutzer alle bis zum nächsten Schlüsselwechsel folgenden Logeinträge (`::mes::`) und Signaturen (Zeile `::sig::`) entschlüsseln. Äquivalent ermöglichen ihm die Signaturen, jeweils einen Block von Nachrichten auf Integrität und Authentizität zu überprüfen

```
1269618569.529082:can1:3B7:3:A38000
1269618569.529313:can2:420:8:00F273A3B42D0000
                          (…)
```

Abbildung 3: Schematischer Aufbau gefilterter CAN-Nachrichten als Logeintrag (z.B. für R5)

```
::key::5temeDmAUg6Rrq86sMyWViakoTnI40Jf2OVEc2hbpVKgRxRWel0iQ4Fq/dGgbNsR...
::mes::XCB2yafgPK0Tc2FeU2FcgZ67oHSm6XmtIOaYJnbGbBUEpSJz8+6aFgK+doHFQc83
::mes::YdjrOJJ0wvKQan9/4a7tBZ1c6o5Te+dRmjae+oBQSxt3XyXmJNsnnDVbtIdlcy0w
                                    (…)
::sig::kJD4ptmphyP6HCwTc9U8muEn0DEYUhYJhZSrRHSZ9Y3IwZ2mQEVHwj9GGYWh/DjD...
::key::(…)                          (…)
```

Abbildung 4: Schematischer Aufbau der gesicherten Logdateien (z.B. für einen Nutzer aus R5)

Da der Fokus auf die Filterung und Sicherung der Logdaten liegt, ist der ***Detektor*** im vorliegenden Prototyp lediglich rudimentär umgesetzt. Er kann in Zukunft jedoch z.B. um die in [HKD09] umgesetzten IDS-Konzepte ergänzt werden.

5 Diskussion des Konzeptes und der Umsetzung

In diesem Abschnitt werden abschließend ausgewählte Vor- und Nachteile diskutiert, die das vorgeschlagene Konzept (sowie die aktuelle Umsetzung) auszeichnen.

Schlüsselmanagement und Registrierung: Als eine zentrale Eigenschaft des Konzepts muss jeder Nutzer bereits im Vorfeld mit seinem Public Key auf dem FFDS registriert werden. Während die Versicherung (R2) und Stammwerkstatt (R5) im Vorfeld i.d.R. bekannt sind, kann dies im Fall anderer externer Rollen ggf. ein Problem darstellen. Da ein konkreter Unfallrekonstrukteur oder IT-Forensiker i.d.R. erst im Fall eines Vorkommnisses zugewiesen wird, können hier im Vorfeld keine individuellen Schlüssel hinterlegt werden. Eine Möglichkeit ist, für diese Rollen ein globales Schlüsselpaar einzusetzen, das seitens der zugehörigen Vereinigung verwaltet wird (potentiellen Vertraulichkeitsrisiken kann durch ausreichend restriktive Filterregeln für diese Rollen vorgebeugt werden). Als Alternative/Kompromiss kann der Besitzer vorab virtuelle Nutzer dieser Rollen anmelden, die Private Keys verwahren und beim Eintreten des jeweiligen Falles an den Ausführenden übergeben. Auch ist eine Erweiterung des Konzepts um Verfahren zum Widerrufen (z.B. wie in Abschnitt 3.1 kurz skizziert) und Updaten eingesetzter Schlüssel notwendig. Dies steht als Teil zukünftiger Arbeiten zur Diskussion.

[3] siehe RFC4648; http://tools.ietf.org/html/rfc4648

Berücksichtigte Schutzziele: Die Adressierung der *Vertraulichkeit* der gesicherten Daten ist ein wesentlicher Vorteil des Konzepts: Selbst wenn ein Angreifer an sämtliche im FFDS enthaltenen Informationen (inkl. kryptographischer Schlüssel) gelangt, ist es ihm nicht möglich, vorhandene (d.h. vor dem Einbruch generierte) Logeinträge zu entschlüsseln (allenfalls mit Ausnahme solcher Einträge, für die noch der aktuelle Session-Key gilt). Hierfür benötigt er die geheimen Schlüssel der Nutzer, die zu keiner Zeit auf dem System vorgehalten werden. Als Nachteil kann dagegen angesehen werden, dass der FFDS für die Signierungsvorgänge (Sicherung der *Authentizität und Integrität* der Logdaten) mit einem eigenen Private Key auszustatten ist: Gelangt ein Angreifer an diesen, kann er (zusammen mit den Public Keys der Nutzer) beliebige signierte Logdateien erzeugen (ein selektives Verfälschen bestehender Inhalte gestaltet sich mangels Möglichkeit der Entschlüsselung jedoch schwierig, s.o.). Dies macht den Einsatz von vergleichsweise teurerem, sicherem Speicher (z.B. einer Smartcard) erforderlich, um die benötigten Geheimnisse sicher im FFDS hinterlegen zu können. Im Falle einer sicheren Implementierung ist als wesentlicher Angriff auf die gesicherten Daten das Unbrauchbarmachen (z.B. Löschen/Überschreiben) der Logdateien zu nennen. Gezielte Angriffe auf die gesicherten Datenbestände sind problematisch: z.B. kann blockweises Löschen oder Vertauschen von Einträgen anhand der enthaltenen Zeitstempel (siehe Abb. 3) erkannt werden; eine Erweiterung um Sequenznummern ist zudem möglich.

Sicherheit kryptographischer Verfahren: Zukünftig könnten Angriffe auf die eingesetzten kryptographischen Verfahren effektiver werden. Gelangt ein Angreifer beispielsweise über Known-Plaintext-Angriffe [Bu10] an die symmetrischen Schlüssel oder durch Faktorisierung [Bu10] von RSA Public Keys an die Private Keys der registrierten Nutzer, könnten die geforderten Schutzziele nicht mehr gewährleistet werden. Während der Lebenszyklus kryptographischer Verfahren aktuell deutlich unter dem Lebenszyklus moderner Automobile liegt (oft 15-20 Jahre), kann sich z.B. ein nachträglicher Wechsel der kryptographischen Algorithmen aufgrund der eingeschränkten Systemressourcen schwierig gestalten. Potential bietet hier z.B. speziell auf eingebettete (automotive) Systeme zugeschnittene *light-weight* Kryptographie (vgl. www.ecrypt.eu.org/lightweight/).

Authentizität und Integrität der Eingangsdaten: Heutige automotive Busnetzwerke bieten noch keine Schutzmechanismen, die den Anforderungen der IT-Security genügen (z.B. für Authentizitäts- und verlässliche Integritätsprüfungen). Auf heutige Fahrzeuge angewandt, kann der FFDS die beschriebenen Schutzziele der zu protokollierenden Daten erst ab deren Eingang gewährleisten. Bereits an den Eingangsdaten erfolgte Manipulationen können daher heute noch nicht abgedeckt werden, d.h. ein Angreifer könnte das System nach physischem Umlegen der Eingangskanäle z.B. mit simulierten oder gezielt gefilterten Eingabedaten versorgen. Der Nachweis einer solchen Manipulation an einem heutigen System wäre symptomatisch anhand der gesicherten Logdaten zu führen (z.B. durch Auffinden von Widersprüchen in korrelierten Daten). Mittelfristig könnten Authentifizierungs-Mechanismen auf Geräteebene (z.B. durch Entwicklungen wie in [BZ08]) mit einbezogen werden, um die Echtheit der an der Kommunikation beteiligten Geräte festzustellen. Langfristig kann diese Lücke z.B. durch zukünftige Security-Erweiterungen automotiver Bussysteme effektiver geschlossen werden. Entsprechende Konzepte werden aktuell erforscht (z.B. [We09]), um potentiellen zukünftigen Bedrohungen wie den in [Ko10] demonstrierten begegnen zu können.

Erweiterungspotential: Das vorgeschlagene erste Konzept ist noch bzgl. weiterer Aspekte zu erweitern. Beispielsweise sollte zur beweiskräftigen Verwertung der Logdaten ein besonderes Augenmerk auf die Zeitsynchronität gesetzt werden. Ein weiterer Punkt ist, dass je nach Anzahl der registrierten Nutzer und Umfang der zu sichernden Daten sowie auch angesichts potentieller Denial of Service (DoS) Angriffe Kapazitätsengpässe seitens des Speichermediums auftreten können. Hier bietet es sich an, den Logger (siehe Abb. 1) mit robusten Verdrängungsstrategien auszustatten, wozu ein prioritätsbasierter Ansatz vorgeschlagen wird: Einzelne Rollen wie die Versicherung haben einen besonderen Anspruch auf die Vollständigkeit einiger Daten, während andere Daten bei Kapazitätsengpässen eher überschrieben werden können. Als eine Alternative zur nahe liegenden Zuordnung der gesicherten Einträge zu Prioritätsklassen kann die Erweiterung des kryptographischen Konzepts um homomorphe Algorithmen geprüft werden, die eine Suche auch auf den verschlüsselten Datenbeständen erlauben.

6 Zusammenfassung / Ausblick

In diesem Beitrag wurde ein Konzept vorgestellt, wie Informationen aus automotiven IT-Systemverbünden unter Anwendung IT-forensischer Prinzipien gesichert und dadurch für eine Vielzahl von Anwendungsmöglichkeiten nutzbar gemacht werden können. Hierzu wurden fünf exemplarische Rollen vorgestellt und ihre individuellen Anforderungen an die zu sichernden Daten diskutiert. Ein Konzept auf Basis eines hybriden kryptographischen Verfahrens wurde vorgestellt und zusammen mit der rollenindividuellen Datenfilterung anhand einer prototypischen Umsetzung illustriert. Abschließend wurden ausgewählte Vor- und Nachteile des Konzeptes diskutiert.

Auch weitere offene Punkte sollten in zukünftigen Arbeiten noch vertieft adressiert werden. Hinsichtlich des Prototyps ist der Ressourcenbedarf des FFDS (Rechenzeit, Speicherplatz) anhand anwendungsnaher Filterregeln und CAN-Kommunikation zu untersuchen. Insbesondere aufgrund des hohen Kostendrucks in der Automobilindustrie ist dies essentiell für die Rechtfertigung eines praktischen Einsatzes. Des Weiteren ist bezüglich des Konzeptes noch der Fall vertieft zu berücksichtigen, dass der Fahrzeugbesitzer (R1) zumindest zeitweilig nicht mit dem Fahrzeugführer übereinstimmen muss. Hinsichtlich von Datenschutzaspekten müsste hier noch feingranularer zwischen verschiedenen Fahrzeugführern unterschieden werden, was sich beispielsweise mit biometrischen Systemen zu Insassenauthentikation [BS07] kombinieren ließe. Auch adressiert das Konzept aktuell nur digitale Daten von den Fahrzeugbussen. Darüber hinaus kann es sinnvoll sein, weitere Informationen (ausgewählte Sensorwerte, interne Betriebsdaten von Steuergeräten, …) mit einzubeziehen. Trotz der in Abschnitt 5 skizzierten Angriffsszenarien sollten in den Prozessen (z.B. Wartung) auch zulässige Möglichkeiten vorgesehen werden, einen Wechsel des FFDS (z.B. bei Defekt) oder des Fahrzeugs (z.B. bei Neukauf) vornehmen zu können.

Danksagungen: In Teilen wurde diese Veröffentlichung durch die Europäische Union im Kontext des Verbundprojekts COmpetence in MObility (C-2007-5254) unterstützt. Teile der Erkenntnisse zu den Grundlagen der IT-Forensik dieser Arbeit entstanden aus der Bearbeitung eines Projektes des Bundesamts für Sicherheit in der Informationstechnologie (BSI). The work described in

this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

Literaturverzeichnis

[HKD08] Hoppe, T.; Kiltz, S.; Dittmann, J.: Security threats to automotive CAN networks – practical examples and selected short-term countermeasures. In: SAFECOMP 2008, Springer LNCS 5219, ISBN 978-3-540-87697-7, 2008; S. 235-248.

[Ko10] Koscher, K. et.al.: Experimental Security Analysis of a Modern Automobile. In: The IEEE Symposium on Security and Privacy, Oakland, CA, May 16-19, 2010.

[BM09] Burg, H.; Moser, A. (Hrsg.): Handbuch Verkehrsunfallrekonstruktion, 2., aktualisierte Auflage, Verlag Vieweg + Teubner, ISBN 978-3-8348-0546-1, 2009.

[Bo08] Borgeest, K.: Elektronik in der Fahrzeugtechnik - Hardware, Software, Systeme und Projektmanagement, 1. Auflage 2008, Vieweg Verlag, ISBN 978-3-8348-0207-1, 2008.

[BD07] Barisani, A.; Bianco, D.: Unusual Car Navigation Tricks: Injecting RDS-TMC Traffic Information Signals. In: Can Sec West, Vancouver, 2007.

[MFA10] MyFox Austin: Hacker Deactivates 100 Cars Via Internet, 17.3.2010, http://www.myfoxaustin.com/dpp/news/local/31710-Hacker-Deactivates-100-Cars-Via-Internet, Letzter Zugriff: 16.6.2010.

[HKD09] Hoppe, T.; Kiltz, S.; Dittmann, J.: Applying Intrusion Detection to Automotive IT – Early Insights and Remaining Challenges. In: Journal of Information Assurance and Security (JIAS), ISSN: 1554-1010, Vol. 4, Issue 6, 2009; S. 226-235.

[MHD10] Müter, M.; Hoppe, T.; Dittmann, J.: Decision Model for Automotive Intrusion Detection Systems. Erscheint in: Automotive - Safety & Security 2010, Ada Deutschland Tagung 21. - 23. Juni 2010, Stuttgart, Shaker Verlag, Aachen, 2010.

[KHA09] Kiltz, S.; Hildebrandt, M.; Altschaffel, R.; Dittmann, J.; Vielhauer, C.; Schulz, C.: Sicherstellung von gelöschtem Schadcode anhand von RAM- Analysen und Filecarving mit Hilfe eines forensischen Datenmodells. In: 11. Deutscher IT-Sicherheitskongress, Bonn 2009, SecuMedia Verlag Ingelheim, ISBN 978-3-922746-97-3, 2009; S. 473-488.

[Ho10] Holthusen, S.: Datenerhebung aus Fahrzeugnetzwerken nach IT-forensischen Prinzipien, Bachelorarbeit, Universität Magdeburg, 2010.

[Ha03] Harms, D.: Unfalldatenspeicher (UDS) als möglicher Beitrag zur Verkehrssicherheit im Meinungsbild Jugendlicher und Heranwachsender, Dissertation, Braunschweig, 2003.

[EDR06] National Highway Traffic Safety Administration (NHTSA), Department of Transportation (DOT): Event Data Recorders - Final Rule, Docket No. NHTSA-2006-25666, 2006.

[ZS08] Zimmermann, W.; Schmidgall, R.: Bussysteme in der Fahrzeugtechnik - Protokolle und Standards, 3., Auflage, Vieweg Verlag, ISBN 978-3834804471, 2008.

[MR10] MyRate Program, http://www.progressive.com/myrate/, Letzter Zugriff: 16.6.2010.

[KH02] Kruse, W.; Heiser, J.: Computer Forensics – Incident Response Essentials, Addison Wesley, ISBN 0201707195, 2002.

[Bu10] Buchmann, J.: Einführung in die Kryptographie, 5. Auflage, Springer-Verlag, ISBN 978-3-642-11185-3, 2010.

[Eck03] Eckert, C.: IT-Sicherheit, Oldenbourg, ISBN 3-486-27205-4, 2003.

[BZ08] Detlef Borchers, Peter-Michael Ziegler: Mit PKI gegen den Autoklau, Heise Newsticker 5.3.2008, http://www.heise.de/newsticker/meldung/104593, 2008.

[We09] Weyl, B. et. al.: Securing Vehicular On-Board IT Systems: The EVITA Project. In: 25. VDI/VW Gemeinschaftstagung - Automotive Security (CD-Rom), Düsseldorf, VDI Wissensforum GmbH, 2009.

[BS07] Büker, U.; Schmidt, R.: Biometrische Fahreridentifikation. In: 23. VDI/VW Gemeinschaftstagung - Automotive Security, VDI-Berichte Nr. 2016, VDI-Verlag, 2007.

Towards Secure Deletion On Smartphones

Michael Spreitzenbarth[1] Thorsten Holz[2]

[1]Laboratory for Dependable Distributed Systems
University of Mannheim, Germany
spreitzenbarth@informatik.uni-mannheim.de

[2]Horst Görtz Institute for IT-Security
Ruhr-University Bochum
thorsten.holz@rub.de

Abstract: Nowadays, smartphones constitute one of the most commonly used electronic devices. Today's smartphones combine a variety of different technologies: they offer in addition to excellent mobile availability and connectivity also high-speed data transfer for the user. Moreover, they are multimedia capable due to their integrated digital camera or music player, and offer a wide variety of communication services like e-mail, SMS or MMS. Consequently, they are used increasingly as a "mobile office".

In this paper, we outline the possibilities and obstacles of *secure deletion*, namely the problem of deleting sensitive data on a smartphone in such a way that this data cannot be restored during a later forensic investigation. In order to guarantee the complete deletion of data, it would be necessary to access the memory chip directly such that we can overwrite the address space of existing data with arbitrary data. However, this approach is not possible when dealing with smartphones due to several reasons. On the one hand, the user's activities are restricted on the device, which implies that far-reaching system interventions cannot be conducted easily. On the other hand, writing on a specific physical address is hindered due to the use of "wear leveling" algorithms on flash chips, which are intended to optimize durability. We discuss these problems in detail and introduce an approach to more securely delete data under certain constraints.

1 Introduction

A *smartphone* is a mobile handset that offers a wide variety of different technologies which reassemble typical functionality of a PC: besides offering excellent mobile availability and connectivity, also high-speed data transfer via universal mobile telecommunications system (UMTS) and wireless local area network (WLAN) are available to a user such that she is always connected to the Internet. Moreover, smartphones are multi-media capable due to an integrated digital camera or music player, and – not to forget – they serve the function of text-based communication via e-mail, multimedia messaging service (MMS), and short message service (SMS). Consequently, they are used increasingly as a "mobile office", thanks to the various office applications available for mobile phones. Another aspect which should not be neglected is the fact that current smartphones are more fre-

quently equipped with GPS modules, that enable the user to make use of the mobile phone for navigation purposes.

One of the drawbacks of this development is the fact that smartphones also store a lot of sensitive data like for example contact information (e.g., phone numbers or full address details), e-mail messages that might contain sensitive information, or private photos. Unfortunately, there is typically no easy way to delete this information from a given phone. Due to the fact that the second market (e.g. ebay) for smartphones is increasing the deletion of these data becomes more and more important.

There are two main obstacles that complicate *secure deletion*, i.e., the problem of deleting sensitive data on a smartphone in such a way that this data cannot be restored during a later forensic investigation: limited interactions with the device and *wear leveling*.

On the one hand, smartphones offer only a limited user-interface and the user's activities are often restricted to interaction with (pre-)installed tools on the device. This often prohibits system interventions which require privileged access to the device. On the other hand, there is a subtle physical peculiarity of smartphones that needs to be taken into account: mobile devices typically use flash chips to store data and these components use a technique called *wear leveling* to evenly distribute write access across the flash chip. When implementing secure deletion, we thus need to take this physical requirement into account. In this paper, we outline the possibilities and obstacles of secure deletion and introduce an approach to more securely delete data under certain constraints.

This paper is outlined as follows: In Section 2, we discuss related work in the area of secure deletion. We provide a brief background on wear leveling and the implications of this approach in Section 3. In Section 4, we introduce an approach to securely delete data under certain constraints, which we evaluate in Section 5. Finally, we conclude the paper in Section 6.

2 Related Work

We are not the first to study the problem of secure deletion and thus we provide an overview of related work on this topic. One of the first papers in the area of secure deletion was published by Gutmann, who analyzed how data from magnetic and solid-state memory can be deleted in a secure way [Gut96]. He introduced the 35-pass overwrite technique to address the implications of different encoding schemes used by hard drives. Furthermore, he noted that "A good scrubbing with random data will do about as well as can be expected." [Gut96] which is true for magnetic and solid-state memory, but some obstacles need to be addressed when dealing with flash chips as we explain in the next section.

Bauer and Priyantha proposed a technique to perform secure deletion at the filesystem level: they implemented an extension to the *ext2* filesystem that asynchronously deletes file data and meta-data by overwriting this data according to best known practices [BP01]. Joukov et al. extend this approach to also handle *ext3* filesystems and discuss in detail the problem of secure deletion [JPZ06]. We implemented a similar approach that can be used

on mobile phones. Leet et al. discuss a method that can handle NAND flash file systems and the authors provide a proof-of-concept implementation for YAFFS [LHC+08]. The main idea behind their approach is to encrypt files and forces all keys of a specific file to be stored in the same block, thus only a single erase operation is need to delete the key – as a result, the file can not be accessed anymore. Our approach is more light-weight and we overwrite the data according to best know practices.

Specific aspects of the secure deletion problem like secure deletion for a versioning filesystem [PBH+05] or a secure peer-to-peer backup system [BBST01] have been discussed in the literature. Typically, all these approaches implement different techniques to overwrite data and relevant meta-data with random data to impede forensic analysis. We perform a similar approach when dealing with flash chips.

3 Background: Wear leveling

In this section, we explain the different wear leveling techniques and the implied consequences for our approach. First, we outline general approaches of wear leveling followed by a technique which is used in more recent Nokia smartphones.

Flash chips have the drawback that their content can only be changed a limited number of times: at some point, after 100,000 or more cycles depending on the actual chip, so much voltage or time is required to either program or erase the cell (or both) that it becomes impractical to use it any further. The lifetime of that cell has ended at this point. To address this problem, the technique of wear leveling for a memory controller has been introduced: it provides a method to distribute the access the cells at times when it is detected that they are receiving significantly uneven use. The exact procedure of those operations is described in the following.

Figure 1 schematically depicts the memory operation techniques [LNTG08]. Here, the storage space is separated into single blocks whereby those blocks again consist of several blocks. If the user now sends data which are attached to a logical address, this logical address is converted with the help of an address translation table into a physical memory address, i.e., to a block within the bank.

This address translation table can be reprogrammed by a processing unit in order to change the physical address of the block in which data are intended to be stored. The process of reprogramming is used to lade the whole memory equally often with writing operations. In order to do such reprogramming, information on storage characteristics and storage occupancy are collected. Therefore, the number of writing operations on each storage block is stored. If it is the case that this number exceeds the average value of the other blocks by a certain amount, the wear leveling process is started.

When wear leveling is accomplished, two main events occur. First of all, data of the heavily used blocks are swapped with the data of those blocks which are least used. In a second step, the address translation table is modified such that the new physical addresses are connected to the old logical ones. Consequently, a data block having the same logical address as prior to wear leveling is now written on another physical address than beforehand.

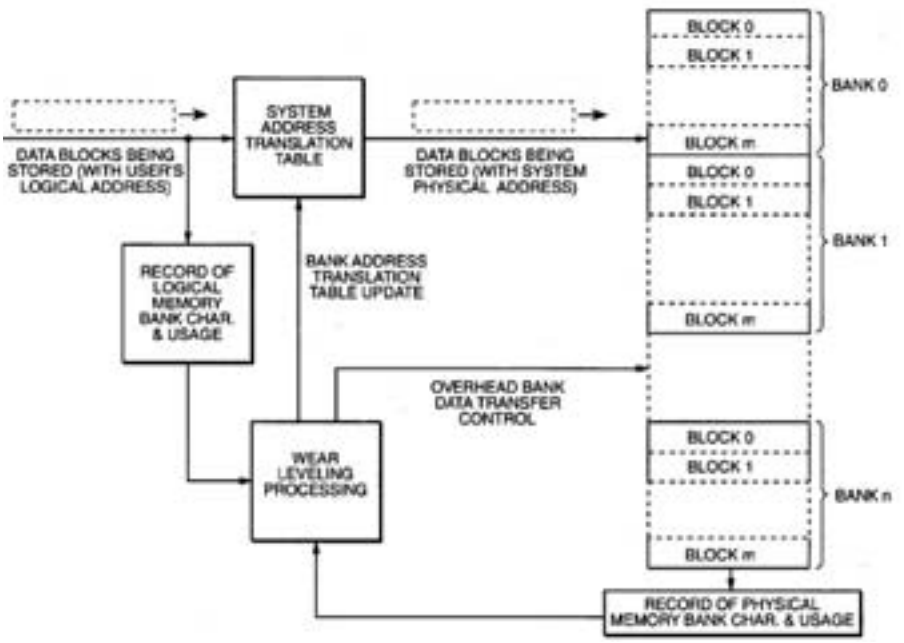


Figure 1: Schematic illustration of ways in which the solid state memory may be operated [LNTG08]

Figure 2 illustrates another approach of wear leveling by means of a process flow diagram [LNTG08]. In this diagram, the process of wear leveling is initialized by the system. This initialization can either be started by a system service or by a memory controller. The rest of the process is mostly identical to the already described one, except for the fact that in this case an additional *spare bank* is used. Data of the least used bank is written to the spare bank and data of the heavy used bank is now written to the least used bank, which in turn is now empty. In another step, the heavy used bank is converted to the new spare bank. The following process is now again identical to the previously described one.

Those two figures illustrate that there are various approaches for the execution of wear leveling. However, all approaches have the intention of even wear of the storage spaces. According to Symbian OS Internals [sym09] and Samsung [Fla07], Nokia uses in its current device series (e.g.: N-Series) Samsung OneNAND [sam09] storage with a modification of the already described wear leveling techniques. In this case, no swapping of data on the blocks is conducted due to time reasons; instead, a delete- and write-operation is saved. This happens by the storage of the erase-count of each block. If we would now like to write to a block having a higher count than one of the non-used blocks, the block which has not been used up to now is deleted and the write operation is done on that block. The real selected block is now marked as non-used and shifted to the so-called *garbage queue*, which is ordered descending according to the erase-count. If at a later point in time a block for writing is needed, its erase-count is compared to the block on first position of the queue, and the block with the lowest erase-count of those two is chosen for the writing

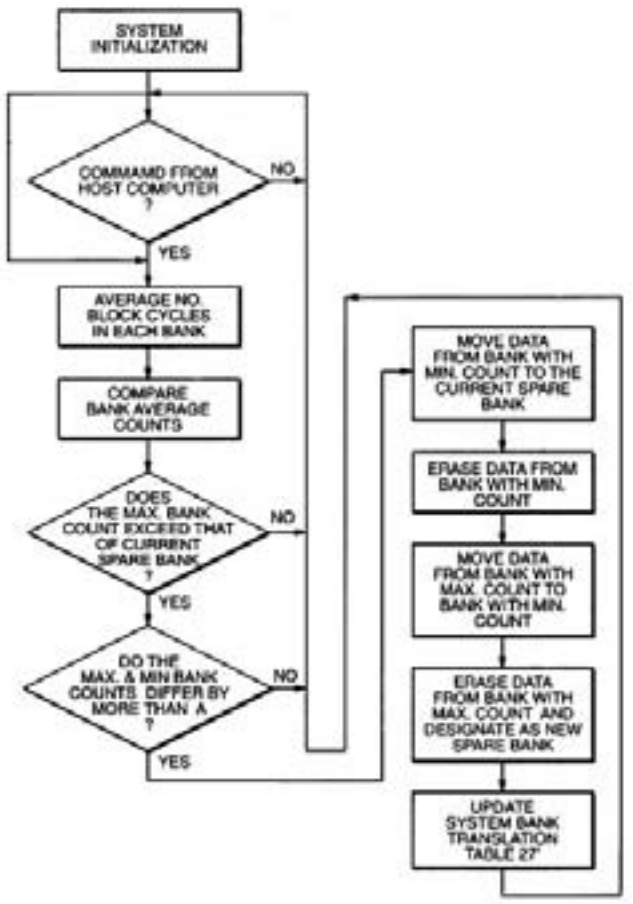


Figure 2: Flow diagram showing a preferred operation of the memory system [LNTG08]

operation. There are two methods of deleting data stored on blocks within the garbage queue: the first approach is to delete data when the block is inserted into the queue, while in the second approach data will be deleted as soon as the block leaves this queue and is then re-used for storing new data. When inquiring Samsung which approach is used in OneNAND chips, the company did unfortunately not answer our request.

4 Towards Secure Deletion on Symbian Phones

In order to develop a tool that helps to securely deleting data on a smartphone, we have resorted to the programming language Python. Concretely, we used Python for S60 [Nok08]

(also known as pyS60) available in Version 1.4.5. The Python for S60 platform simplifies application development and provides a scripting solution for the Symbian C++ APIs.

Within this development platform, we have developed a tool named *SecDel* which currently possesses the ability to delete SMS messages, telephone directories, as well as, calendar entries from a given phone. Moreover, it contains an update-function, which allows the user to load modules, which are developed in the future, via the Internet.

In the following, we discuss the secure deletion of telephone book entries exemplarily and in detail. When deleting calendar entries and SMS messages, we have only focused on specific aspects of these methods, as the remaining procedure is very similar in all cases. The contacts module offers an API to address book services allowing the creation of contact information databases. The contacts module represents a Symbian contact database as a dictionary-like *ContactDB* object, which contains *Contact* objects and which is indexed using the unique IDs of those objects [ST07].

As shown in the upper part of Listing 1, the *ContactDB* is opened and the unique ID and the entry title is read out of all entries with the help of a loop, both parts are then consolidated into one list element.

```
def searchContacts():
    appuifw.note(u"searching Contacts...", "info")
    db = contacts.open()
    list3 = []
    for db_id in db:
        id = str(db[db_id].id)
        title = db[db_id].title
        element = (id + " - " + title)
        list.append(element)
    selection = appuifw.multi_selection_list(list, 'checkbox', 1)
    for k in selection:
        contact_id = list[k].split(" ")[0]
        contact_id = int(contact_id)
        # change value
        kind = ['city', 'company_name', 'country', 'date', '
            dtmf_string', 'email_address', 'fax_number', '
            job_title', 'note', 'pager_number', 'phone_number', '
            po_box', 'postal_address', 'postal-code', 'state', '
            street_address', 'url', 'video_number', 'picture', '
            second_name', 'voip', 'sip_id', 'personal_ringtone', '
            share_view', 'prefix', 'suffix', 'push_to_talk', '
            locationid_indication', 'last_name', 'first_name', '
            mobile_number']
        for type in kind:
          for i in range(0,10):
            try:
                if db[contact_id].find(type)[i].value:
                    db[contact_id].find(type)[i].value = u"
                        XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
            except:
```

```
        continue
    # delete contact
    db.__delitem__(contact_id)
    appuifw.note(u"Contact deleted!", "conf")
```

Listing 1: Secure deletion of contacts used in SecDel

After such a list element has been created from each of the telephone book entries, it is saved in a list. Then a *Multi-Selection-Listbox* is created with the help of this list. This listbox (see Figure 3) enables the search for entries and marking of several entries in order to delete all these entries at once.

Figure 3: SecDel select contacts which should be deleted

In the next step, the script separates the ID of the selected list entries from the title in order to search the whole entries in the database again. If the corresponding entry is found, all existing objects within this entry, no matter how often they occur, are overwritten with a sign chain, consisting of 59 times "X". We have chosen the length of 59 due to the fact that this is the maximum allowed length of characters which is available for the longest of the available objects. We conduct this step in order to make sure that none of the characters of the initially entry is still available afterwards. In Figure 4, we depict an entry of the telephone's address book which has been overwritten in the above described way.

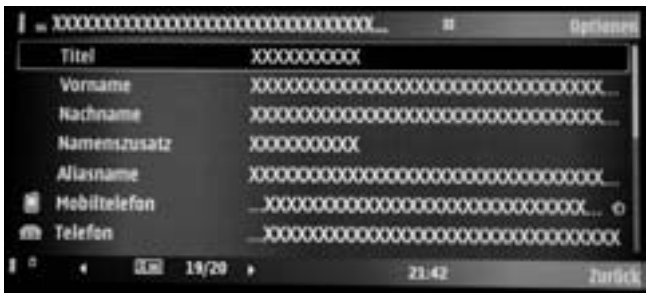


Figure 4: SecDel contact entry after renaming it

After successful overwriting, the objects of all previously chosen entries are deleted from the database. This is communicated to the user in another message.

A particularity exists when we deal with calendar entries or notes: here the stored dates and times as well as the point of time of reminders are either deleted completely or reset to the 01.01.1970, 00:00 o'clock by the code fragment shown in Listing 2. In addition, repeated events are converted to unique events before they are deleted from the database, such that this fact does not reveal any information.

```
db2[calendar_id].set_repeat(None)
if (db2[calendar_id].type == "note"):
        # clear date and time
        db2[calendar_id].set_time(None)
else:
        # set date to 01.01.1970
        db2[calendar_id].set_time(0, 0)
```

Listing 2: Code excerpt of SecDel which changes the date and time values

As a further module, we have included an update-function by which it is possible to load future modules and extensions via an Internet connection belatedly. As a second additional update, we have included a remote service function. It happens sometimes that one loses his smartphone, or that it is stolen. Typically a user can only hope that the important data stored on the phone are not used by the finder or thief. To address this problem, we have developed the following function: if SecDel is kept active in the background, it analyzes every SMS message in the inbox regarding a certain message sequence. As an authentication process, the user can specify a password which has to be in that message, too. If the phone now receives a SMS message with the sequence "run_secdel" in the first line and a password in the second line, the tool will validate the password and start with the immediate deletion of all data as described above. The validation of the password should prevent a denial-of-service attack. This process is done with the help of a callback function which triggers this behavior. The inbox object's *bind()* function binds a callback function to an event that is generated by an incoming message. In our case, the function *msg_rec()* is called up as soon as a new SMS is received on the smartphone. Apple's iPhone implements a similar protection technique that enables a user to remotely wipe the phone, which erases all personal data and restores the phone to the factory settings.

5 Evaluation

The operations we have discussed in the previous sections have been tested on a Nokia E90 phone with Symbian OS version 9.2 installed. For the verification of secure deletion on this generation of smartphones, currently only few technical possibilities exist. The reason lies in the fact that Symbian has introduced security restrictions (also called *Capabilities* [Symb]), which disable any foreign access to all important private files. In order to get to those storage spaces, we need an *All Files* capability by which our software agent becomes certified (compare [Syma]). The *All Files* capability is one of the Manufacturer-approved capabilities, which are highly sensitive capabilities that can only be obtained from the device manufacturer, even if it is just for experimenting on a phone under our

control. Getting these capabilities requires a user to pass a complex certification process, and to have an ACS Publisher ID from Verisign [ST07].

Due to the reasons stated above, we had to resort to a smartphone using an older Symbian version for the purpose of verification. When using older Symbian versions, the stated security restrictions had not been implemented yet. In this case, it is also not possible to create a memory dump with the help of *flasher-tools* like Twister Box. The single solution which is currently realizable, apart from de-soldering the memory element, is constituted by the solution with the help of a software agent. This agent is installed on the memory card of the mobile device and creates a dump of saved files from this location. Installing the agent on the memory card rather than on the smartphone itself provides the advantage that nearly no data will be changed during the installation process. This could be realized on condition of dumping the original inserted memory card on a bigger one and then installing the software agent on the new card. In Figure 5 we show the logical paradigm workflow of these tools.

Figure 5: Workflow of a software agent by means of Panoptes

For this purpose, we used a Nokia N70 phone and the software agents MIAT (of the University of Rome [MO07, DM08]) and Panoptes (which has been developed within a diploma thesis [Spr09]). After comprehensive tests, we received notice that these tools are not able to restore any deleted data. Within the data, which had been saved with the help of MIAT and Panoptes, we could neither find data which had been deleted by the Symbian interface, nor were data detectable which had been deleted by the tool SecDel. To the best of our knowledge, SecDel thus securely deletes data on the tested phone by overwriting the relevant data with garbage.

Without any doubt, the most perfect solution for the verification of the depicted operations and the throughout validation of the implemented wear leveling technique of the memory element, would be the de-soldering of the flash storage chip and the subsequent direct analysis of this element. Unfortunately, we were not given the possibility to do so which is why we resorted to the approaches presented beforehand. Another drawback of de-soldering is the fact that this approach is questionable from a forensic perspective since evidence might be destroyed during the process.

6 Conclusion and Future Work

In this paper, we presented the various influences which are related to secure deletion of data on mobile phones. In the beginning, we explained the different kinds of wear leveling used for flash storage. In general, wear leveling describes the possibility of extending the economic life-time of a flash chip by exchanging blocks on which data are used very frequently with those on which data are used less frequently. With the help of this process, the whole storage element is equally burdened with write operations in order to prevent deficiencies within single sections. In this context, various approaches exist to implement this concept. In most cases, those differ only in the handling of data which are stored on the blocks which should be swapped. When dealing with current Nokia smartphones, these blocks are deleted completely before data is written on these blocks again. As a result, the problem of reading out file fragments in the so-called *slack space* does not exist on these phones. Slack space is the unused space in a disk cluster: some file systems use fixed-size clusters and even if the actual data being stored requires less storage than the cluster size, an entire cluster is reserved for the file. Since always complete blocks are deleted, we can ignore this problem on mobile phones.

Subsequently, we presented a tool which tries to delete personal data on the smartphone in a secure way. This means that data is deleted in such a way that restoring them is not possible. We implemented this technique by overwriting the existing data with known garbage and thus erasing the information. Our evaluation suggests that this overwriting is successful and all relevant information is deleted. Unfortunately, we were not able to thoroughly verify the depicted deleting operations in a forensically correct way due to the fact that we did not possess the needed means to de-solder and examine the flash chip. We could only prove with the help of the tools MIAT [MO07, DM08] and Panoptes [Spr09] that deleted data are not contained in the files intended for storage any longer. Due to this reason, the tool can not guarantee “secure deletion”, but rather “more secure deletion”. The combination of overwriting of the data and the subsequent deletion of them on the operating system level on the one hand, as done by SecDel, and the wear leveling process on the memory level on the other hand, as done by the build in flash modules, significantly increase the difficulty to restore this data.

Our future work will focus on the development and improvement of forensically sound analysis tools for Symbian OS and other operating systems for mobile devices, e.g., Google Android or iPhone OS. At the same time, we will also work on improving the secure deletion tool which we have introduced in this paper, especially on improving the evaluation to verify that the deleted data can indeed not be restored.

References

[BBST01] Christopher Batten, Kenneth Barr, Arvind Saraf, and Stanley Trepetin. pStore: A Secure Peer-to-Peer Backup System. Technical report, MIT, 2001.

[BP01] Steven Bauer and Nissanka B. Priyantha. Secure data deletion for Linux file systems. In *USENIX Security Symposium*, 2001.

[DM08] Alessandro Distefano and Gianluigi Me. An overall assessment of Mobile Internal Acquisition Tool. *Digital Investigation*, 5:121–127, 2008.

[Fla07] Flash Software Group. XSR1.5 WEAR LEVELING. Technical report, Samsung Electronics Co., Ltd, 2007.

[Gut96] Peter Gutmann. Secure deletion of data from magnetic and solid-state memory. In *USENIX Security Symposium*, 1996.

[JPZ06] Nikolai Joukov, Harry Papaxenopoulos, and Erez Zadok. Secure deletion myths, issues, and solutions. In *Second ACM Workshop on Storage Security and Survivability*, 2006.

[LHC+08] Jaeheung Lee, Junyoung Heo, Yookun Cho, Jiman Hong, and Sung Y. Shin. Secure deletion for NAND flash file system. In *ACM Symposium on Applied Computing*, 2008.

[LNTG08] Karl MJ Lofgren, Robert D Norman, Gregory B Thelin, and Anil Gupta. Wear Leveling techniques for flash EEPROM systems. United States Patent, April 2008.

[MO07] Pontjho M Mokhonoana and Martin S Olivier. Acquisition of a Symbian Smart phone's Content with an On-Phone Forensic Tool. Technical report, Information and Computer Security Architectures Research Group, 2007.

[Nok08] Nokia. *PyS60 Library Reference*, 1.4.5 edition, Dezember 2008.

[PBH+05] Zachary N. J. Peterson, Randal Burns, Joe Herring, Adam Stubblefield, and Aviel D. Rubin. Secure deletion for a versioning file system. In *USENIX Conference on File and Storage Technologies (FAST)*, 2005.

[sam09] SAMSUNG OneNAND™, September 2009. `http://www.samsung.com/global/business/semiconductor/products/fusionmemory/Products_OneNAND.html`.

[Spr09] Michael Spreitzenbarth. Mobile Phone Forensics. Diploma Thesis, University of Mannheim, 2009.

[ST07] Jürgen Scheible and Ville Tuulos. *Mobile Python: Rapid Prototyping of Applications on the Mobile Plattform*, volume 1st. Wiley, October 2007.

[Syma] Symbian Press. Complete Guide to Symbian Signed. `http://developer.symbian.org/wiki/index.php/Complete_Guide_To_Symbian_Signed`.

[Symb] Symbian Press. Plattform Security. `http://developer.symbian.org/wiki/index.php/Platform_Security_(Fundamentals_of_Symbian_C%2B%2B)`.

[sym09] Symbian OS Internals, September 2009. `http://developer.symbian.org/wiki/index.php/Symbian_OS_Internals/`.

Amun: Automatic Capturing of Malicious Software

Jan Göbel
Laboratory for Dependable Distributed Systems, University of Mannheim, Germany
goebel@informatik.uni-mannheim.de

Abstract: This paper describes the low-interaction server honeypot Amun. Through the use of emulated vulnerabilities Amun aims at capturing malware in an automated fashion. The use of the scriping language Python, a modular design, and the possibility to write vulnerability modules in XML allow the honeypot to be easily maintained and extended to personal needs.

1 Introduction

Autonomously spreading malware is among the main threats in todays' Internet. Worms and bots constantly scan large network ranges worldwide for vulnerable machines to exploit. Compromised machines are then used to form large botnets for example to perform distributed denial of service attacks, send out masses of email spam, or to compromise even more machines.

With the help of honeypots we are able to capture such autonomously spreading malware in a fast and straightforward fashion. Especially low-interaction honeypots, i.e. honeypots which allow little to no interaction with the attacker, are very useful in this area of network security. Server based honeypots provide a number of emulated services to lure attackers, monitor the attack, and analyze the exploit code to get hold of the self-propagating malware binary.

Low-interaction honeypots provide a low risk method for capturing information on initial probes, as there is no full interaction with an attacker. Thus, these honeypots are easier to maintain and enable the collection of information in an automated manner. This property renders low-interaction honeypots excellent sensors for intrusion detection systems (IDS).

In this paper we introduce *Amun* a highly flexible and lightweight low-interaction honeypot, designed to capture malware that spreads by exploiting server based vulnerabilities. The use of a scripting language allows it to be easily extended and ported to different operating systems. Furthermore, Amun tries to actually "speak" the required protocol of an application an attacker is trying to exploit, making it more successful at fooling attackers.

2 Related Work

Several honeypot solutions have been developed in the past. In this section we introduce those honeypot solutions, that follow a similar approach as Amun does.

One of the most well known low-interaction honeypots is *Honeyd* [Pro04]. It is a Linux application that runs in the background, creates virtual hosts on a network and offers different vulnerable services. The virtual hosts can be configured to appear as a certain operating system, such as Microsoft Windows for example. Honeyd features a plug-in system for easy extension and some helpful tools like *Honeycomb* [KC03b, KC03a], that can automatically generate intrusion detection signatures from the captured data. Generated signatures are currently support by Bro [Pax98] and Snort [Koz03]. The focus of Honeyd is mainly on the collection of attack information rather than capturing malware binaries itself.

The following two low-interaction honeypots follow the same scheme as Amun does. The first is called *Nepenthes* [BKH+06]. The second is called *Omnivora* [Tri07]. Just like Amun, both honeypot solutions aim at capturing malware in an automated manner by emulating well-known vulnerabilities. Both solutions perform very well, but require good programming skills either in C++ or Delphi, in order to extend or modify the honeypots to personal needs.

In contrast to the latter two honeypots, Amun provides a wider range of vulnerability modules and is, due to its simpler structure, easier to deploy and maintain. The usage of a scripting language provides a straightforward way to extend Amun with new features without having to recompile the software every time. Additionally, Amun uses protocol specific replies instead of random byte replies to be more efficient in fooling attacks.

3 Amun Honeypot Architecture

Amun is written in Python[1], a small and simple scripting language. The honeypot is made up of different components that handle the individual parts needed for vulnerability simulation and shellcode detection, which will be described in this section in more detail. Following is a short list of the most important components of Amun:

- Amun Kernel (Section 3.1)
- Amun Request Handler (Section 3.2)
- Vulnerability Modules (Section 3.3)
- Shellcode Analyzer (Section 3.4)
- Download Modules (Section 3.5)
- Submission Modules (Section 3.6)
- Logging Modules (Section 3.7)

[1] http://www.python.org

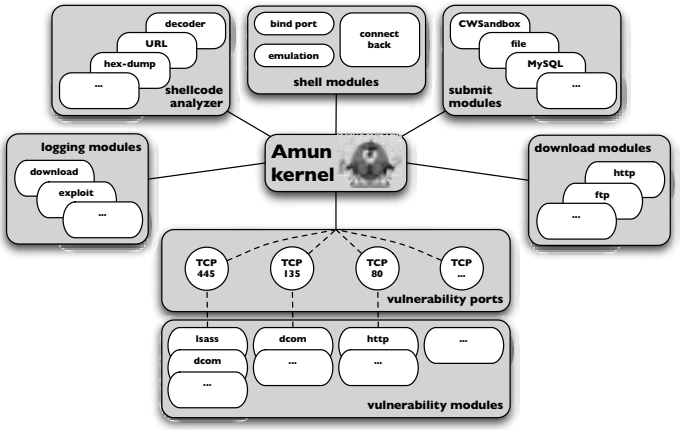


Figure 1: schematic setup of Amun

Figure 1 shows the schematic setup of Amun and the interaction of each part of the software with the kernel. Each of the above mentioned components is described in detail in the following sections.

3.1 Amun Kernel

The Amun Kernel is the core component of the honeypot. This part contains the startup and configuration routines, as well as, the main routine of the software. Amun is a single threaded application that uses the `select` operator to iterate over all open sockets. Besides these socket operations the Amun Kernel handles downloads, configuration reloads, shell spawning, and event logging in the main loop as well.

During the startup phase, the Amun Kernel initializes the regular expressions that are used for shellcode matching, reads the main configuration file creates the internal logging modules, and loads all external modules. External modules are the vulnerability modules, that are responsible for emulating single vulnerabilities, the logging modules, that log attack information to other services like databases, and the submission modules, that for example write downloaded binaries to hard disc.

For each loaded vulnerability module Amun starts a TCP server listening on the ports the module has registered for.

After all initial modules are loaded and the appropriate TCP servers are started, Amun Kernel enters the main loop. During this loop, it iterates over all connected sockets, triggers download events, transfers information to certain modules, and re-reads the main configuration file for changes. The latter option allows the reconfiguration of Amun during runtime, i.e. without the need to restart the honeypot.

3.2 Amun Request Handler

The Request Handler is responsible for all incoming and outgoing network traffic of the honeypot. For every connection request that reaches the Amun Kernel a Request Handler is created that handles the connection until it is closed. The Request Handler maintains the list of loaded vulnerability modules and delegates the incoming traffic to those modules that are registered for the current port.

Consider a connection to port 445: if it is a new connection the Request Handler loads all vulnerability modules registered for port 445. In the next step the incoming traffic is distributed to each of these modules. Each of the vulnerability modules checks if the incoming traffic matches the service that it emulates and either accepts or rejects the connection. As a result, the list of emulated vulnerabilities for a connection is thinned out with each incoming request of the attacker. In the worst case none of the registered modules matches the attack pattern and the connection is closed. Otherwise, there is exactly one module left, that successfully emulates all needed steps performed by the attacker and receives the final payload containing the download information of the malware.

Note that incoming network packets can be distributed to all registered vulnerability modules, but a reply packet can only be send by one. In the best case there should only be one module left to reply after the first packet is received, however, if there are more left, the reply of the first module in the list is chosen.

The Request Handler also receives the results of the vulnerability module that successfully emulated a service and obtained the exploit payload from the attacker. This payload is passed on to the Shellcode Analyzer to detect any known shellcode.

3.3 Vulnerability Modules

The vulnerability modules make up the emulated services that lure autonomous spreading malware to initiate an exploit. Each module represents a different vulnerable service, for example a FTP server. The services are emulated only to the degree that is needed to trigger a certain exploit. That means, the emulated services cannot be regularly used, i.e. they do not offer the full functionality of the original service.

Vulnerabilities are realized as finite state machines. They usually consist of several stages that lead through the emulated service. Figure 2 shows an example of a finite state machine describing a buffer overflow vulnerability in the ExchangePOP3 v5.0 software. Upon the first connection of an attacker Amun sends the banner of the emulated service and awaits the first command. In this case stage one waits for the command `Mail`, all other input leads to the dropping of the connection.

That way Amun assures that only requests that lead to the exploit of the emulated service are accepted. All data that leads to an undefined state is logged by the Request Handler. With this information it is possible to determine changes in exploit methods and add new stages or even built new vulnerability modules.

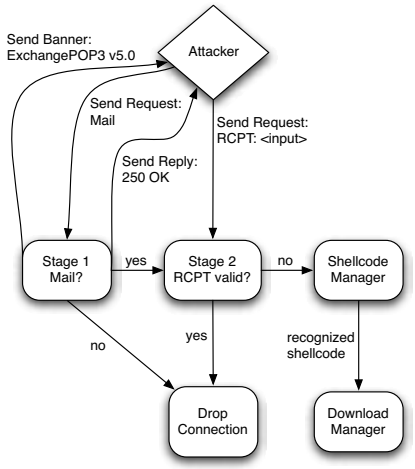


Figure 2: example of a finite state machine

To facilitate the process of writing new vulnerability modules, Amun supports XML to describe a module. This XML file is subsequently transformed to Python code by Amun and can then be used as a vulnerability module. This means, that for simple vulnerability modules there is no need to write Python code.

Figure 3 illustrates an example of a XML document representing the parameters necessary to create the Plug and Play (PNP) vulnerability. It shows the number of stages (`<Stage>`) needed to trigger the exploit and for each stage the expected number of bytes (`<ReadBytes>`) together with the according byte sequence (`<Request>`). After the sixth stage the module enters the shellcode collecting stage, i.e. at this point the exploit should already have taken place, and the attacker sends the shellcode. All data that is collected during the shellcode collection stage is subsequently passed to the Request Handler and then to the Shellcode Analyzer.

The final Python code of a vulnerability module consists of several different functions. The first function is for initialization of the module, here the name of the vulnerability, the starting stage, and a welcome message is defined. The welcome message is for example a banner displaying the service name and version upon the connection of an attacker. The main function of a vulnerability module is called `incoming`. This function receives the network packet, the number of bytes of this packet, the attacker IP address, a logging module, a previously created random reply, and the IP address of the honeypot.

To quickly analyze certain ports for incoming attacks, Amun has a so-called analyzer vulnerability module. This module simply registers for certain ports defined via the configuration file, collects all incoming data, and sends it to the Shellcode Analyzer. The purpose of this module is to quickly analyze traffic hitting a certain port and see if there are any unrecognized exploits in the wild.

Currently, Amun emulates 43 different vulnerable services on 53 different ports. An ex-

```
<Vulnerability>
 <Init>
  <Name>PNP</Name>
  <Stages>6</Stages>
  <WelcomeMess></WelcomeMess>
  <Ports>
   <Port>445</Port>
  </Ports>
  <DefaultReply>random</DefaultReply>
 </Init>
 <Stages>
  <Stage stage="1">
   <ReadBytes>137</ReadBytes>
   <Reply position="9">\x00</Reply>
   <Request>\x00\x00\x00\x85\xFF\x53\x4D\x42
            \x72\x00\x00\x00\x00\x18\x53\xC8
            [...]
            \x4C\x4D\x31\x2E\x32\x58\x30\x30
            \x32\x00\x02\x4C\x41\x4E\x4D\x41
            \x4E\x32\x2E\x31\x00\x02\x4E\x54
            \x20\x4C\x4D\x20\x30\x2E\x31\x32
            \x00</Request>
  </Stage>
  <Stage stage="2">
   <ReadBytes>168</ReadBytes>
   <Reply position="9">\x00</Reply>
   <Request> [...] </Request>
  </Stage>
       [...]
  <Stage stage="6">
   <ReadBytes>160</ReadBytes>
   <Reply position="9">\x00</Reply>
   <Request> [...] </Request>
  </Stage>
 </Stages>
</Vulnerability>
```

Figure 3: simple vulnerability module in XML

tract of the more well known vulnerabilities is displayed in Table A in the Appendices.

3.4 Shellcode Analyzer

If a vulnerability module successfully emulated a service to the point where the attacker sends exploit code, all incoming data is recorded and finally transferred to the Shellcode Analyzer. The Shellcode Analyzer is the backbone of Amun, as it is responsible for shellcode recognition and decoding. Shellcode is recognized using several regular expressions that match known parts of shellcode. In most cases this is the decoder part, a small loop that decodes the obfuscated shellcode back to its original.

Shellcode can be distinguished as being either clear text or encoded (obfuscated). Obfuscation of shellcode is often achieved with the XOR operator using a single byte (simple XOR) or four bytes (multibyte XOR) or by using an alphanumeric encoding.

Clear text shellcode does not provide any methods of hiding its content, thus it simply contains for example an URL like `http://192.168.0.1/x.exe`. Therefore, one of the

first steps of the Shellcode Analyzer is to check for unencoded URLs within the payload an attacker injected.

Simple XOR encoding means the shellcode is encoded using a single byte. The actual shellcode needs to be decoded prior to execution on the victim host, thus this kind of shellcode contains a so-called decoder part at the beginning. The decoder part is a loop performing a XOR operation with the appropriate byte(s) against the rest of the payload. The Shellcode Analyzer has several regular expression matching those decoder parts and extracting the needed XOR byte. In the next step the shellcode is decoded and the instructions are extracted. Instructions can again be a simple download URL, but also commands to open a certain port or connect back to the attacker and spawning a shell. The multi-byte XOR variant is very much the same, but utilizes more than one byte to encode the shellcode.

Alphanumeric shellcode encoding is a bit more different as its purpose is to use only alphanumeric characters for representation. The reason for using such prepared shellcode is that many new applications and intrusion detection mechanisms filter uncommon characters, thus using only characters like `0-9` and `A-Z` greatly reduces detection and improves the success rates of exploits.

If the analyzed payload is not recognized by any of the regular expressions, a file containing the data is written to hard disc.

3.5 Download Modules

As the goal of Amun is to capture autonomously spreading malware, we want to get hold of any advertised binary file, thus we need Amun to be able to handle different kinds of download methods. For each download method we can provide a module that is loaded upon the start of the honeypot. Amun currently provides four basic download modules, namely: HTTP, FTP, TFTP, and *direct download*. Following are examples for each of the different download methods. We use an URL like representation, as it is easier to read and display.

- `http://192.168.0.1/x.exe`
- `ftp://a:a@192.168.0.1:5554/32171\_up.exe`
- `tftp://192.168.0.1:69/teekids.exe`
- `cbackf://192.168.0.1/ftpupd.exe`

The first three methods are well known and need not be described any further. The direct download method (`cbackf`) does not involve a transfer protocol. Amun simply connects to the provided IP address at a specified port and receives in return the binary directly. In a few cases some kind of authentication is needed, that is included in the shellcode. After connecting, the honeypot needs to send a short authentication string prior to receiving any data. This kind of download method has been named connect back file transfer (`cbackf`).

Some shellcode does not contain download commands but require the honeypot to open a certain port or connect to a certain IP address and spawn a Windows command shell. Such commands are handled by the bindshell module. This module emulates a Windows XP or 2000 shell to the connected attacker and "understands" a few commands. That means a human attacker will notice directly that this is not a real shell, however automated attack tools simply drop their instructions to the shell and exit. These instructions are collected and again analyzed by the Shellcode Analyzer to extract the actual download command. Figure 4 shows an interesting example of commands send to an emulated shell of Amun.

```
Cmd /c
  md i &
  cd i &
  del *.* /f /q &
  echo open new.setheo.com > j &
  echo new >> j &
  echo 123 >> j &
  echo mget *.exe >> j &
  echo bye >> j &
  ftp -i -s:j &
  del j &&
  echo for %%i in (*.exe) do
    start %%i > D.bat &
    D.bat &
    del D.bat
```

Figure 4: command received at emulated shell

The commands instruct the victim system to create a new directory called `i`, change to it and delete all files in it, using the parameters for quiet mode (`/q`), and the parameter to enforce deletion of read-only files as well (`/f`). In the next step a new file is created containing FTP commands to download all files with the `.exe` extension. The attacker uses the `mget` command to retrieve multiple files. In the `FOR`-loop each downloaded binary is executed in its own separate window (`start`).

3.6 Submission Modules

Once a file has been downloaded using any of the above mentioned download modules it needs to be processed further. That means it can be stored to hard disc for example, or send to a remote service.

In the default configuration Amun only loads the *submit-md5* module. This module stores each downloaded binary to a certain folder on the hard disc. As a filename it uses the MD5 hash of the content of the file.

Other submission functions that are included in Amun are: *submit-cwsandbox*, *submit-anubis*, *submit-joebox*, and *submit-mysql*. These modules submit downloaded binaries to different sandbox services that execute malware and analyze its behavior. The resulting reports are then accessible either by web or email.

3.7 Logging Modules

The logging modules provide an easy way to generate different kinds of notifications whenever an exploit occurs. Currently Amun offers five modules: *log-syslog*, *log-mail*, *log-mysql*, *log-surfnet*, and *log-blastomat*. The last logging module belongs to an intrusion detection system (IDS) developed at the RWTH Aachen called Blast-o-Mat [Göb06]. The IDS uses honeypots as intrusion sensors to detect attacks in the network.

The log-syslog module sends all incoming attack information to the local syslog daemon. That way it is also possible to send attack information to remote machines, e.g., a central logging server. Another method is to use the log-mail module, that sends information about exploits to a predefined email address. Note, that according to the number of attacks, a lot of emails can be generated and flood the mail server. To prevent this, the block options of the configuration file can be used that handle reoccurring exploits.

The log-mysql module, allows the logging of attack information to a MySQL database. The layout for the database is stored in the configuration directory of Amun. This module is however still in development.

The log-surfnet module, allows the integration of Amun into the surfnet IDS, also called SURFids [Goz07]. SURFids is an open source Distributed Intrusion Detection System based on passive sensors, like honeypots. SURFids uses PostgreSQL as underlying database.

Logging modules support three main functions to log events: `initialConnection`, `incoming`, and `successfullSubmission`. The first function is triggered upon a connection request of a host to the honeypot. This request must not be malicious at this point in time. The second function is called as soon as an exploit is detected and some kind of download method has been offered. The last function is called whenever a binary was successfully downloaded, thus, this function receives the same parameters as the incoming function of the submission modules.

4 Limitations

Although low-interaction server honeypots are a great addition to todays' intrusion detection mechanism, they also have some limitations. The most obvious limitation with low-interaction honeypots in general is the lack of capturing zero day attacks. The reason is that only vulnerabilities can be emulated that we already know of, thus this approach is always one step behind. The same restriction applies to the use of shellcode. Amun can only recognize shellcode it already knows of.

Next is the fact that the vulnerable services are not fully simulated with every feature they offer, but only the parts needed to trigger an exploit. As a result, low-interaction honeypots will not fool any human attacker, but only automatically spreading malware, which does not verify the functionality of a service in the first place. Although such checks could be easily added, todays' malware is rather poorly written. There exist cases were not even the server reply is checked and the malware sends its shellcode regardless of the attacked

service being vulnerable or not.

5 Results

In this section we present some statistics resulting from data collected at a single Amun honeypot installation with a few thousand IP addresses assigned. The data was collected during the last twelve months, thus ranging from December 2008 till December 2009, with almost no downtime of the sensor.

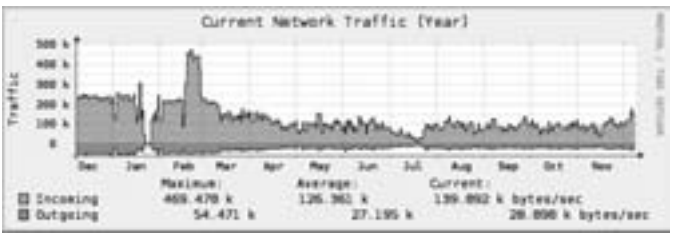


Figure 5: oberserved network traffic

Figure 5 shows the amount of network traffic observed at the honeypot. The data was generated in five minute intervals using RRDTool[2]. The average number of Kilobytes received is 126.36KB, whereas we saw a maximum during February 2009 with 469.48KB. Compared to the incoming traffic, the outgoing traffic is rather low, with an average of 27.2KB. The reason for this difference is, that the honeypot also receives the malware binaries, which usually make up the biggest amount of traffic.

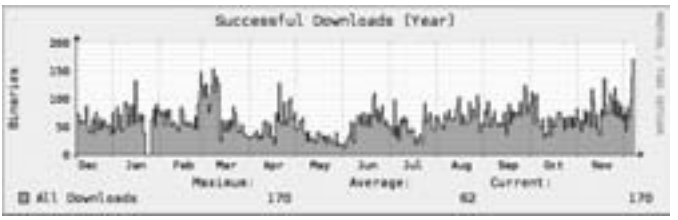


Figure 6: successfull downloads of malicious software

Figure 6 displays the number of successfully downloaded malware binaries seen over the last twelve months. For comparison, Figure 7 shows the number of successfully downloaded binaries for the last 24 hours. It shows that we have captured an average of 73 binaries every 5 minutes, with a maximum of 170 binaries, which is very similar to what we have seen over the last twelve months.

Figure 8 shows the increasing number of unqiue malware binaries that we have collected. Uniqueness is determined using the MD5 hash value of a binary. During the twelve months

[2]http://oss.oetiker.ch/rrdtool/

Figure 7: successfull downloads of malicious software in the last 24 hours

measurement period we have collected a total of 4790 malware binaries. Note that the used uniqueness criteria is not reliable, as polymorphic malware for example produces a different hash each time it is downloaded.

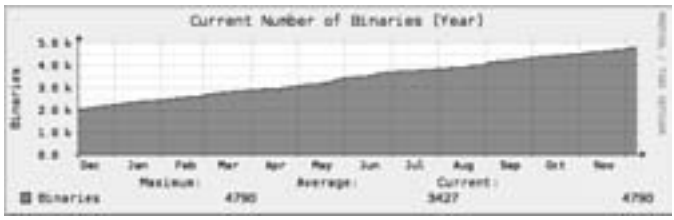


Figure 8: unique malware binaries captured

A more detailed analysis of honeypot data captured with low-interaction honeypots is presented at the DIMVA conference in 2007 [GWH07]. The complete analysis of all collected information during the twelve months period is left as future work.

6 Summary

In this paper we presented the low-interaction server based honeypot Amun. We gave a detailed description of the different parts of the software that are responsible to handle the emulation of vulnerabilities and download the propagating malware. We showed the individual modules for vulnerability emulation, logging, shellcoded analyses, and submission. All coordination between the different modules is handled by the Amun Kernel, which is the core part of Amun.

The main focus of Amun is to provide an easy platform for malware collection, like worms or bots. For this purpose Amun uses the simple scripting language Python and a XML based vulnerability module generation process to support an easy creation of new vulnerability modules. Thus, malware analysts are able to collect current malware in the wild and have the opportunity to extend the honeypot software without much programming knowledge.

Although low-interaction honeypots do have some limitation regarding zero-day attack

detection, they also make up a great addition to todays' network security systems. Considering well placed honeypots throughout a network. These passive sensors can detect any scanning machine and report it to a central IDS, without any falsely generated alarms. That means an alarm is only raised upon the exploitation of an emulated vulnerability.

Finally, honeypots are an important tool to study and learn about attackers and their procedures to compromise machines in the Internet.

Acknowledgements.

We would like to thank Philipp Trinius who provided valuable feedback on a previous version of this paper that substantially improved its presentation. Markus Engelberth for creating the schematic overview picture at the beginning of this report.

References

[BKH+06] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The Nepenthes Platform: An Efficient Approach to Collect Malware. In *RAID'06: 9th International Symposium On Recent Advances In Intrusion Detection*, 2006.

[Göb06] J. Göbel. Advanced Honeynet based Intrusion Detection. Master's thesis, RWTH Aachen University, July 2006.

[Goz07] R. Gozalbo. Honeypots aplicados a IDSs: Un caso practico. Master's thesis, University Jaume I., April 2007.

[GWH07] J. Göbel, C. Willems, and T. Holz. Measurement and Analysis of Autonomous Spreading Malware in a University Environment. In *DIMVA'07: Detection of Intrusions and Malware, and Vulnerability Assessment*, 2007.

[KC03a] C. Kreibich and J. Crowcroft. Automated NIDS Signature Generation using Honeypots. In *SIGCOMM'03: Special Interest Group on Data Communication*, 2003.

[KC03b] C. Kreibich and J. Crowcroft. Honeycomb - creating intrusion detection signatures using honeypots. In *HotNets'03: Second Workshop on Hot Topics in Networks*, 2003.

[Koz03] J. Koziol. *Intrusion Detection with Snort*. Sams, Indianapolis, IN, USA, 2003.

[Pax98] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. In *7th USENIX Security Symposium*, 1998.

[Pro04] N. Provos. A Virtual Honeypot Framework. In *13th USENIX Security Symposium*, 2004.

[Tri07] P. Trinius. Omnivora: Automatisiertes Sammeln von Malware unter Windows. Master's thesis, RWTH Aachen University, September 2007.

Appendices

A List of Emulated Vulnerabilities

CVE-ID	Description
CVE-2001-0876	Buffer Overflow MS Universal Plug and Play
CVE-2003-0352	Buffer Overrun Windows RPC - MS03-026
CVE-2003-0533	Buffer Overflow LSASS - MS04-011
CVE-2003-0818	Buffer Overflow Microsoft ASN.1 - MS04-007
CVE-2004-0206	Buffer Overflow Network Dynamic Data Exchange - MS04-031
CVE-2004-0567	Buffer Overflow Windows Internet Naming Service
CVE-2004-1172	Stack Overflow Veritas Backup Exec Agent
CVE-2005-0059	Buffer Overflow MS Message Queuing MS05-017
CVE-2005-0491	Knox Arkiea Server Backup Stack Overflow
CVE-2005-0582	Buffer Overflow Comp-Associates License Client
CVE-2005-0684	Buffer Overflow MaxDB MySQL Webtool
CVE-2005-1272	Buffer Overflow CA ARCserver Backup Agent
CVE-2005-1983	Stack Overflow MS Windows PNP - MS05-039
CVE-2005-2119	MSDTC Vulnerability - MS05-051
CVE-2005-4411	Buffer Overflow Mercury Mail
CVE-2006-2630	Symantec Remote Management Stack Buffer Overflow
CVE-2006-3439	Microsoft Windows Server Service Buffer Overflow - MS06-040
CVE-2006-4379	Stack Overflow Ipswitch Imail SMTP Daemon
CVE-2006-4691	Workstation Service Vulnerability - MS06-070
CVE-2006-6026	Heap Overflow Helix Server
CVE-2007-1675	Buffer Overflow Lotus Domino Mailserver
CVE-2007-1748	Windows DNS RPC Interface - MS07-029
CVE-2007-1868	Buffer Overflow IBM Tivoli Provisioning Manager
CVE-2007-4218	Buffer Overflows in ServerProtect service
CVE-2008-2438	HP OpenView Buffer Overflow
CVE-2008-4250	Microsoft Windows RPC Vulnerability - MS08-067
-	Axigen Mailserver Vulnerabilities
-	DameWare Mini Remote Control Buffer Overflow
-	Vulnerabilities in different FTP Server implementations
-	GoodTech Telnet Server Buffer Overflow
-	Buffer Overflow Password Parameter SLMail POP3 Service

Table 1: excerpt of Amun vulnerability modules

Towards Optimal Sensor Placement Strategies for Early Warning Systems

Jan Göbel Philipp Trinius

Laboratory for Dependable Distributed Systems, University of Mannheim, Germany
{goebel|trinius}@informatik.uni-mannheim.de

Abstract: A network early warning system consists of several distributed sensors to detect malicious network activity. The effectiveness of such early warning systems critically depends on the sensor deployment strategy used. We therefore analysed attack patterns of malicious software collected at sensors worldwide to determine an optimal deployment strategy. Our results show that due to the small numbers of attackers shared among networks, the benefit of large-scale sensor deployment is rather limited. However, there is some evidence that world-wide geographical distribution of sensors has some beneficial effect on the average early warning time.

1 Introduction

Autonomous Spreading Malware and Early Warning: Autonomous spreading malware (malicious software) like worms or bots are one of the most dangerous threats in the Internet today. Once infected, hosts start to scan large network ranges for more vulnerable machines that can be exploited. Acting in a coordinated fashion, bots can launch denial of service attacks or initiate high-volume spam campaigns. The extreme dimensions of botnets (for example with approximately 350.000 machines [SGE+09]) make them a hard opponent for security systems. It is therefore necessary to develop ways to detect emerging threats as early as possible.

Many different network and host based security solutions have been developed in the past to counter the threat of autonomous spreading malware. Among the most common detection methods for such attacks are *honeypots*. Honeypots offer vulnerable services and collect all kinds of information about hosts that probe and exploit these services. Today, honeypots act as efficient sensors for the detection of attack activity in the Internet. Such sensor data can be combined with other network data to form a partial view of the status of a network.

Given an attack phenomenon that occurs in location L_1 at time T_1, it should be expected that the same phenomenon happens later at some time $T_2 > T_1$ in a different location L_2. If $T_2 - T_1$ is sufficiently large, the hope is that the network at location L_2 can "prepare" for the attack in time. For example, if some corporate network in Germany suffers from attacks by a particular new worm, the network providers could warn other organizations of an upcoming threat. Systems that implement this idea are often called *early warning*

systems. The *Internet Malware Analysis System* (InMAS) [EFG+09, EFG+10], developed at the University of Mannheim, is a prototype for a German early warning system for malware activities. It consists of several different sensor systems, including honeypots, to cover the most common attack vectors of current malware.

Effectiveness of Early Warning: The idea of early warning is rather intuitive. However, it is still unclear whether an early warning system for malware activities can in fact be effective. How much can an organization at location L_2 profit from the data collected at location L_1? This obviously depends very much on the number of sensors deployed and the IP address ranges they cover. It also critically depends on a sufficiently large period $T_2 - T_1$.

There exist some detailed investigations of the attack patterns of autonomous spreading malware [GHW07, BHB+09, SGL04] that investigate (1) what vulnerabilities are frequently exploited, (2) where attackers originate from, (3) what kind of malware is spreading, and (4) what the effects are to the infected hosts. In this paper, we abstract from particular exploits and malware and focus on *correlations* between attacks in different networks. This has direct effect on the effectiveness of early warning systems on a national or even global scale. Therefore, the results from our research can help to develop sensor deployment strategies for these scenarios.

Further research in this area focused on recorded connection data collected at the border gateway [APT07] of an academic network and on data gathered from blackhole deployments [CBM+04], i.e. unused address space. In contrast, our observations are based on data that (1) is clearly identified as being exploits of known vulnerabilities and (2) was collected from used address space, mainly of academic nature. Our work can therefore further substantiate previous research.

Contributions: To study the effectiveness of early warning systems, we first looked at the problem from a *national* perspective, i.e., we collected attack data from different honeypot installations across Germany. We investigated the question whether attack patterns at one location also occur at another and much time lies between such events. We then turned to a more *international* perspective and studied correlations between the German data and data from two external locations, one in China and the other one in Italy. We therefore refine the investigations and explore how sensors located further away from the German sensors influence the effectiveness of a national early warning system.

Overall, we collected attack data from a four month period in 2009. The sensors in Germany covered a total of 15.167 IP addresses, with the majority residing in a single /16 network range. This network range also served as the basis for studying attacker behavior on several adjacent /24 networks.

The major observations from our analysis are:

- We show that, within individual /24 networks, attackers mainly target low-numbered IP addresses, i.e., those between .1 and .127. This is independent both of the particular /24 network and the time of day. We also show that attackers launch attacks from a wide range of IP addresses and that the majority of such IP addresses is used for a small number of attacks only (Section 3).

- Within the low-numbered IP addresses, the attack probability decreases almost linearly from .1 to .127. Thus low-numbered IP addresses are a good choice for sensor deployment (Section 4).
- We show that the number of *shared* attackers over all networks is very small, i.e., the attacks launched from the same IP address almost never occur in many different /24 networks. There is no indication that geographical distance reduces the probability that two sensors will see an attack from the same IP address (Section 5).
- We show that in almost all observed attacks the reaction time, i.e., the time to warn others or roll out patches, is far too short, due to the low number of recurring attackers, and the fast scanning mechanisms used. There is some indication of an increase in average detection time if international sensors are used (Section 6).
- We observed and classified the scanning patterns of attackers. Most attackers performed random scanning. This observation can (partly) be used to explain the above findings (Section 8).

Overall, the results provide valuable information on sensor placement, but also show that early warning systems cannot (always) guarantee timely attack prediction.

2 Measurement Setup

Honeypot Sensors: For data collection we used two similar low-interaction honeypots, namely Nepenthes [BKD+06] and Amun [Goe09]. Although the main focus of both applications is to collect the binaries of autonomous spreading malware, they also generate log files that were the basis for our experiments.

Being low-interaction honeypots, our sensors were not designed to trick a human attacker; they just emulate exploitable services and allow no real interaction. In total the honeypots emulated 43 different services that represent a large percentage of the attack vectors of autonomous spreading malware today (see Appendix [Goe10] for a complete list). As there exist other ways for malware to propagate, our findings are restricted to such an automated setting only.

Data Points and Definitions: The data consists of individual data points that contain the following information: (1) geographical location of sensor, (2) IP address of sensor, (3) timestamp, (4) attacked port/service, and (5) IP address of attacker

Every individual data point represents a *successful exploit* of a service emulated by a honeypot. We use the terms *attack* and *exploit* synonymously in this paper. Note that exploits targeting multiple emulated services of the same honeypot count as multiple attacks. We identify an *attacker* with the source IP address of the attack. Similarly, we identify a sensor with its IP address. Networks are described as /24 and range from .1 to .255 (class C). We therefore use the term /24 and class C synonymously.

Geographical Distribution: For our measurement study we received data points from honeypots located at five different cities: Dresden (Germany), Aachen (Germany), Mannheim

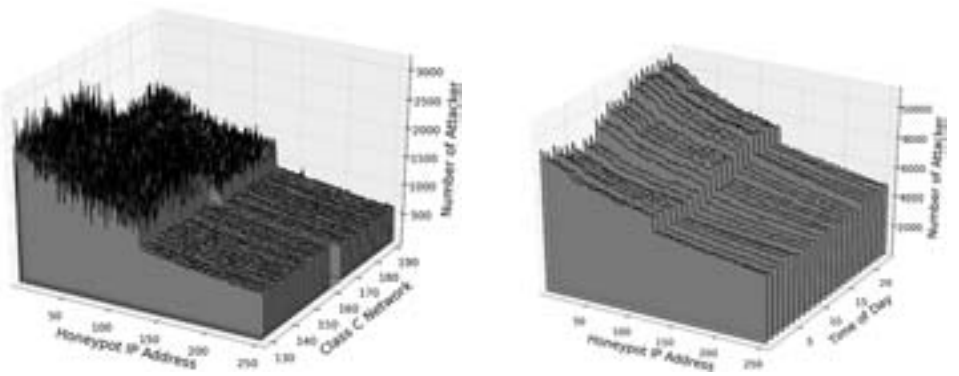
(Germany), Milano (Italy), and Macau (China). Germany is covered best with three honeypot installations that consist of a total of 15.167 sensor IP addresses, the majority of these residing in a single /16 network range at Aachen. China and Italy both only have a single sensor IP address to record attack data.

Measurement Periods: The time period of our measurement study is split into two ranges. The first period lasted from April, 29, 2009 until May, 14, 2009 and the second from June, 10, 2009 until September, 14, 2009. In total, we have almost four months (113 days) of data. The only exception is the honeypot running in Macau (China), here the measurement period lasted from July, 1, 2009 until October, 27, 2009. In total we have an overlap of 76 days between the Chinese sensor and all other sensors.

3 Positioning of Sensors

Typically, only two IP addresses of a /24 network are not used for productive systems. The first address (.0), and the last address (.255), which is reserved for broadcast communication. The second address (.1) is usually assigned to the gateway for the network. Therefore, a common assumption is that an attacker scans the complete range between .1 and .254 for vulnerable hosts. To detect such an attacker, it would suffice to randomly deploy sensors across the network range.

To verify or even falsify this predication we aggregate the number of successful exploits recorded at each IP address of all monitored /24 networks of the Aachen honeypot installation, see Figure 1a. The figure shows the number of attackers seen at the individual IP address of each of the monitored /24 networks. There is a clear preference of low IP addresses and an interesting drop in the number of attackers at the center of each of the IP address ranges. Additionally, the number of attackers decreases slightly with the higher number of /24 networks.



(a) Number of attackers per IP address in all /24 networks

(b) Number of attackers per IP address distributed over the day

Figure 1: Illustration of the number of attacker per IP address

As Figure 1a is an aggregate over the entire measurement period, we argued whether there is a dependence on the time of day. So we plotted the number of attackers again, this time for the different hours of a day (Figure 1b). Instead of generating a graph for each of the /24 networks, we merged them all into one and plotted them for each hour of a day. Figure 1b shows that there is an increase in the number of attackers for the late hours, however, the overall tendency we already showed remains.

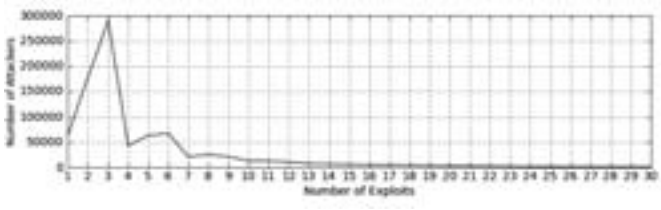


Figure 2: Number of exploits performed by attackers during the measurement period.

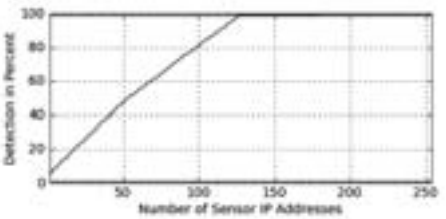
The reason for the uneven distribution of attacks within a network is that most attackers we monitored performed less than ten attacks during the complete measurement period. Figure 2 demonstrates the decrease of the number of exploits per attacker monitored during the complete measurement period for the Aachen honeypot installation. Note that for readability reasons the graph was cut at attackers performing more than thirty exploits. At the Aachen honeypots we monitored a total of 925.998 different attacking hosts and 83.64% have exploited the honeypot less than ten times. This phenomenon coincides with observations from previous work [GHW07].

Overall, using free IP addresses at the end of an address space as intrusion sensors makes less sense than placing sensors at specific points in between productive systems.

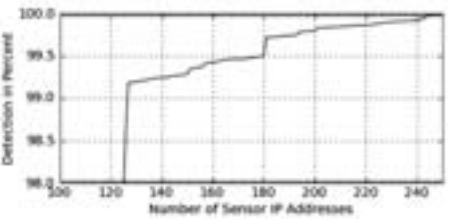
4 Number of Sensors

One way to measure the effectiveness of an early warning system is to count the number of attackers that are reported compared to the total number of attackers that targeted a network, i.e., the *detection ratio*. If we have the chance to place n sensors within a /24 network, clearly the most effective placement is to start with the IP address .1, then add .2, .3, up to $.n$. From the results of the previous section this strategy is even optimal.

Based on this observation we started with the lowest IP address as sensor and continuously added one address at a time (*linear increase*) until the complete network is covered with sensors. Figure 3a shows that a saturation of the detection ratio occurs in case 50% of the network is covered by sensors, starting at the lowest IP address of the network and increasing the number of sensors by one. Thus, if we deploy 127 sensors in the lower part of the /24 network, we achieve a detection ratio of 99.18%, i.e., our sensors detect 99.18% of all attackers that targeted this network during our measurement period.



(a) Relation between the number of sensors and the detection ratio in a single /24 network.

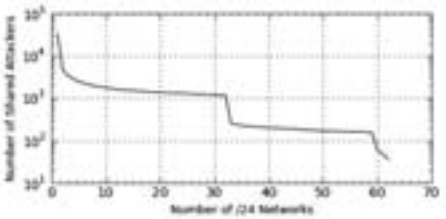


(b) Increase of detection ratio with more than half of the address space covered with sensors

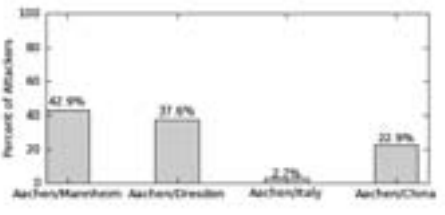
Figure 3: Relation between the position of sensors and the detection ratio.

Figure 3b displays the upper part of Figure 3a starting at 98% detection ratio. The Figure illustrates the increase of the detection ratio when deploying more than 127 sensors.

5 (Globally) Shared Adversaries



(a) Decrease in the number of shared attackers with an increase of monitored networks



(b) Shared attackers among geographically distant locations

Figure 4: Shared attackers according to adjacent networks and distant location

We investigated the number of shared attackers with the increase of monitored /24 networks. Figure 4a illustrates the results and shows the clear decrease in shared attackers among all the /24 networks of the Aachen honeypot installation. In total we monitored 925.998 different attackers and end up with only 37 attackers that exploited systems in all networks.

Thus for an early warning system it does not suffice to have few sensors located in a single /16 network range. In order to collect information about as many attackers as possible it is required to have sensors in (almost) every /24 network.

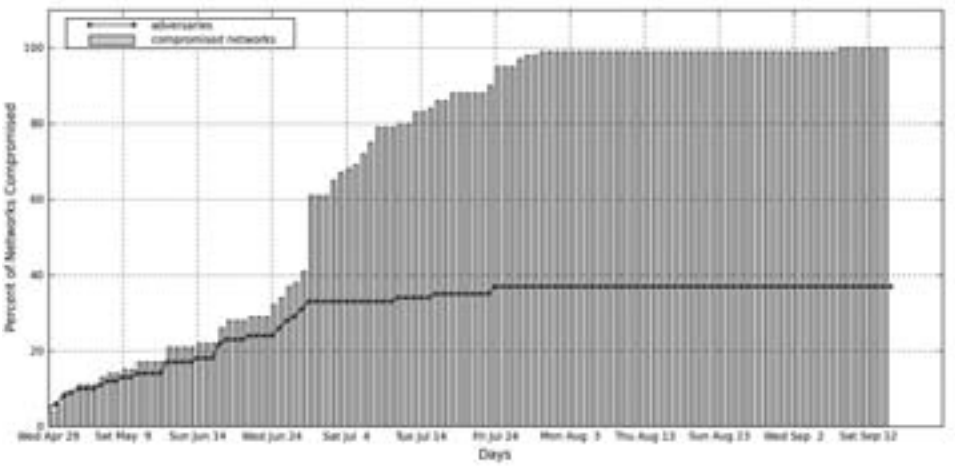
Figure 4b further supports this point. The graph shows the percent of attackers shared between geographically distant sensors. We compared attackers seen at all our honeypot locations with the complete sensor network maintained in Aachen. The total number of attackers monitored at Mannheim, Dresden, Italy, and China are 577, 234, 4.202, and

24.471, respectively.

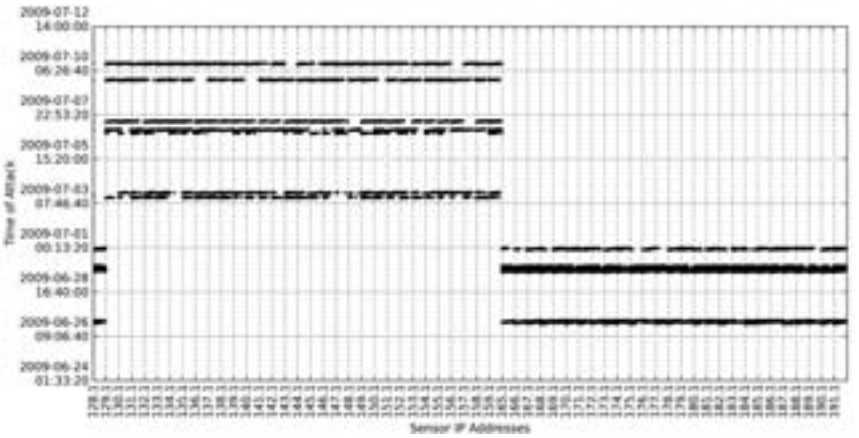
6 Attack Distribution and Detection Times

Another important factor with the detection of network incidents and the placement of sensors, is the time until the first detection of an attacker. The question is, how does the sensor deployment strategie affect the time of first detection for an attacker.

We first take a look at the 37 attackers that exploited honeypots in all /24 networks of the Aachen honeynet.



(a) Cumulative number of attacked /24 networks by shared attackers



(b) Individual attacker exploiting hosts in all networks

Figure 5

Figure 5a shows the time needed for all 37 shared attackers to cover the complete network range. As not all attackers started to attack at the beginning of the measurement period, we indicated in the figure the cumulative number of attackers (up to the maximum of 37) that were present at a given time. For example, five attackers that scanned the whole network appeared on the first day of the measurement period. All 37 attackers appeared half way through the measurement period.

The figure shows that it took almost the complete measurement period until all networks had been attacked by all attackers. Thus for some attackers it is possible to react upon first detection and protect further networks. However, when looking at other attackers the detection time can be rather short.

We investigated the behavior of selected attackers further. Figure 5b illustrates the timeline of exploits performed by a single attacker out of the 37 mentioned above during our measurement period. Almost 50% of all /24 networks of the Aachen honeynet were exploited by this attacker on a single day. In this case, reaction time is rather short for the first half of the networks, but sufficiently large for the rest of the networks. This kind of parallel exploit behavior is common for 30 of the 37 attackers.

The fastest attacker exploited the complete Aachen honeynet within 1:33:58 hours. Although the attacker sequentially exploited host by host (see also Section 8), the time to react is extremely short even for all /24 networks. Appendix B provides a complete list of the first and last attack time and the used scanning mechanisms for all of the 37 shared attackers.

Investigating the attack times further, we could not observe any patterns indicating a particular order in which attacks occur. The 37 attackers that exploited honeypots in all /24 networks of the Aachen honeynet did not attack networks in the same order (starting for example with the lowest or highest number). Instead we noticed 22 different /24 networks at which the attacks were started. Furthermore, these networks seem to be randomly chosen.

7 Convenience of Geographical Distribution

After looking at attack times within and inbetween the /24 networks, we studied the correlation with attackers monitored at sensors that were geographically distributed.

Out of the 248 shared adversaries between Aachen and Mannheim, 64 exploited sensors in Mannheim first. The lowest time difference between two attacks of a shared attacker are 14 seconds, whereas the highest difference is 135 days.

The results with Dresden are similar. A total of 37 out of the 88 shared adversaries exploited honeypots in Dresden first, with a minimal time difference of 44 seconds and maximum of 128 days.

Out of the 114 monitored shared adversaries of the Italian sensor, 17 attacked Aachen after exploiting the Italian sensor. The fastest of these attackers still needed 6 minutes before reaching the Aachen sensors. The slowest needed 137 days.

Finally, for the Chinese sensor we detected 546 attackers out of the 5.605 shared adversaries that first exploited the honeypot installation in China and afterwards in Aachen. The shortest time to react upon such an attacker was 14 seconds whereas the longest time was 179 days.

L_1 / L_2	# Adv.	first $T_2 - T_1$	avg. $T_2 - T_1$
Mannheim/Aachen	64	120 hrs.	18 days
Dresden/Aachen	37	25 hrs.	9 days
Italy/Aachen	17	430 hrs.	45 days
China/Aachen	546	27 hrs.	44 days

Table 1: Summary of shared attackers, reaction time $T_2 - T_1$ for first shared attacker hitting remote location L_1, together with average reaction time over all shared attackers.

Table 1 summarizes our findings on shared attackers among geographically distributed sensors. The results show that having more distant sensors deployed helps to increase the average time to react upon an incident inflicted by shared attackers by at least 25 hrs. Furthermore, all sensors increase the total number of new, previously unseen adversaries that are then detected. Interestingly, the average delay correlates with geographical distance.

From the findings in this section we can conclude, that there is no clear geographical correlation between national sensor data, but there seems to be a correlation between national and international sensor data in the sense that international sensors have a substantially larger average reaction time for shared attackers.

8 Scanning mechanisms

During our measurement study we monitored over 955.476 unique attacker IP addresses. With many of these IP addresses showing up at several sensors, we were able the determine the scanning mechanisms used. Overall we can distinguish between four scanning mechanisms:

(1) random scanning (attacker selects /24 networks at random), (2) parallel scanning (attacker targets sensors from many /24 networks in parallel), (3) local sequential scanning (attacker targets sensors within /24 networks sequentially), and (4) global sequential scanning (attacker targets /24 networks sequentially). Table 2 shows the distribution among those four scanning mechanisms for 3.380 attackers which we observed at ten /24 networks of the Aachen honeynet.

Random scanning is the most primitive and difficult to detect scanning mechanisms. As Figure 6a shows, it is impossible to reason where and at which point in time, the attacker is seen again. Note that we picture ten different /24 networks, but the attacker appears only at a few sensors. Thus considering the notation from the introduction, it is not possible to determine L_2 and T_2 upon the observation of L_1 and T_1.

Scan Method	Percent
Random Scanning	55.6%
Sequential Scanning (local)	36.7%
Parallel Scanning	7.4 %
Sequential Scanning (global)	0.3 %

Table 2: Distribution of Scanning mechanisms across 3.380 shared attackers.

Parallel scanning refers to the scanning mechanism show in Figure 6b. The malware attacks sensors out of all /24 networks in parallel. For every /24 network it attacks several sensors within a very short time period. Even if an attacker who is using a parallel scanning mechanism is very easy to identify, the parallelism eliminates the time to react to almost zero, i.e. $T_2 - T_1$ is to low to react.

Local sequential scanning describes a very frequent scanning mechanism. The malware attacks several /24 network in parallel, but within each network it runs a sequential scan (see Figure 6c). Most of these scans are performed in increasing sequential order according to the IP addresses of the sensors. This scanning mechanism is good for exploiting a lot of systems without being noticed. The attacker can easily extend the gap between two attacks to stay below the threshold for intrusion detection systems. Figure 6d shows an attacker performing a slow scan.

In *global sequential scanning*, attackers perform a sequential scan on both: (1) all the /24 networks and (2) the IP addresses within each network. As shown in Figure 7, this scan is very obvious and can easily be identified. Due to the global sequential behavior it offers a relatively long time to react. In this case it is fairly simple to determine both L_2 and based on the speed of the scanning also T_2 of an attack. A classical example for malware that shows this kind of scanning behavior is the blaster worm [Naz03].

For the shared attackers of all /24 networks of the Aachen honeynet, we exclusively observed local sequential and parallel scanning mechanisms (see Appendix B for details). Approximately one-third is performing a sequential scanning, the remaining two-thirds scanned all the /24 networks in parallel. As Figure 5b already indicates, most of the attackers split the scanning process into two pieces and first scanned the higher range of /24 networks (159 to 191). In average the attackers exploited hosts of all networks for around 67 days. The fastest attacker was only observed for a single day, whereas the most persistent attacker was observed 137 days.

9 Conclusions

In this paper we investigated four month of honeypot attack data in order to determine sensor deployment strategies to achieve optimal effectiveness for early warning systems with focus on autonomous spreading malware.

In our view, the most important findings are:

(1) Using free IP addresses at the end of an address space as intrusion sensors makes less sense than placing sensors at specific points in between productive systems.

(2) It does not suffice to have few sensors located in a single /16 network range to achieve optimal attacker detection. It is rather required to have sensors in (almost) every /24 network.

(3) Adding more sensors to an early warning system can increase the time of reaction for those attackers that are shared. Especially international sensors have a substantially larger average reaction time for shared attackers. Therefore, even national early warning systems must deploy sensors at several distant locations, to achieve usable reaction times.

Acknowledgments

We would like to thank the Center for Computing and Communications of RWTH Aachen University, especially Jens Hektor, for their work on maintaining one of the biggest honeynet installations in Germany. We would like to thank Jens Syckor from TU Dresden, Matteo Cantoni from Italy, and Eric Chio from China for sharing their honeypot data. We also would like to thank Andreas Dewald for proof-reading and commenting on an earlier version of this paper.

References

[APT07] Mark Allman, Vern Paxson, and Jeff Terrell. A Brief History of Scanning. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC)*, New York, NY, USA, October 2007. ACM.

[BHB+09] Ulrich Bayer, Imam Habibi, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. A view on current malware behavior. In *Proceedings of the 2nd Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, Boston, USA, April 2009.

[BKD+06] Paul Baecher, Markus Koetter, Maximillian Dornseif, Thorsten Holz, and Felix Freiling. The nepenthes platform: An efficient approach to collect malware. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 165–184. Springer, 2006.

[CBM+04] Evan Cooke, Michael Bailey, Z. Morley Mao, Danny Mcpherson, David Watson, and Farnam Jahanian. Toward understanding distributed blackhole placement. In *Proceedings of the 2004 ACM Workshop on Rapid Malcode (WORM)*, pages 54–64. ACM Press, 2004.

[EFG+09] Markus Engelberth, Felix C. Freiling, Jan Goebel, Christian Gorecki, Thorsten Holz, Philipp Trinius, and Carsten Willems. Frühe Warnung durch Beobachten und Verfolgen von bösartiger Software im Deutschen Internet: Das Internet-Malware-Analyse System (InMAS). In *11. Deutscher IT-Sicherheitskongress*, Bonn, Germany, May 2009.

[EFG+10] Markus Engelberth, Felix C. Freiling, Jan Göbel, Christian Gorecki, Thorsten Holz, Ralf Hund, Philipp Trinius, and Carsten Willems. The InMAS Approach. In *Proceedings of the 1st Workshop on Early Warning and Network Intelligence (EWNI)*, Hamburg, Germany, February 2010. https://eldorado.uni-dortmund.de/handle/2003/26689.

[GHW07] Jan Goebel, Thorsten Holz, and Carsten Willems. Measurement and Analysis of Autonomous Spreading Malware in a University Environment. In *Proceeding of 4th Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2007.

[Goe09] Jan Goebel. Amun: A Python Honeypot. Technical report, Laboratory for Dependable Distributed Systems, University of Mannheim, 2009.

[Goe10] Jan Goebel. Amun: Automatic Capturing of Malicious Software. In *Proceedings of Sicherheit*, 2010.

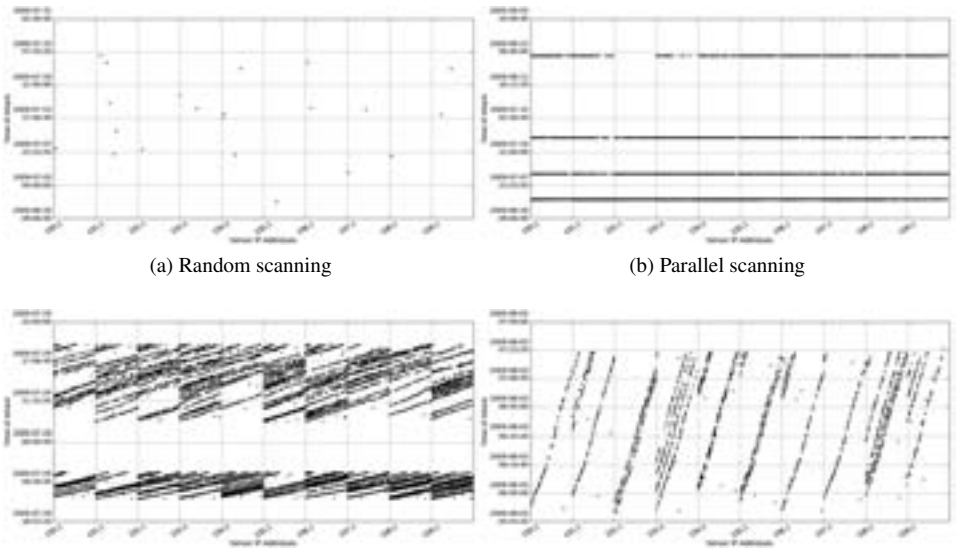
[Naz03] Jose Nazario. The Blaster Worm: The view from 10,000 feet, August 2003. http://www.nanog.org/mtg-0310/pdf/nazario.pdf.

[SGE+09] Ben Stock, Jan Goebel, Markus Engelberth, Felix Freiling, and Thorsten Holz. Walowdac: Analysis of a Peer-to-Peer Botnet. In *Proceedings of the 5th European Conference on Computer Network Defense*, Milan, Italy, November 2009.

[SGL04] Stefan Saroiu, Steven D. Gribble, and Henry M. Levy. Measurement and Analysis of Spyware in a University Environment. In *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI)*, pages 141–153, March 2004.

A Scanning Mechanisms

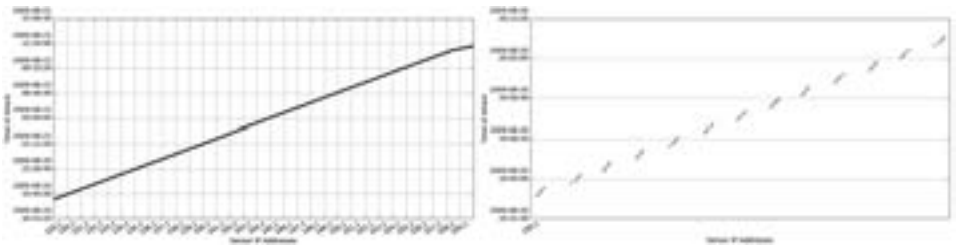
Figure 6 illustrates the four different types of scanning/exploiting mechanisms we detected during our investigation on the honeypot data. Figure 7 gives a more detailed view on the sequential scanning/exploiting.



(a) Random scanning (b) Parallel scanning

(c) Sequential scanning (local) (d) Slow sequential scanning (local)

Figure 6: Scanning mechanisms used during attack sequences.



(a) Sequential scanning (global) (b) Details on a global sequential scan

Figure 7: Gobal sequential scanning.

B Common Attackers

Table 3 shows for all common attackers of all /24 networks of the Aachen honeynet the points in time the attacker was first and last seen. Additionally, the number of days during this two dates and the monitored scanning behavior is listed. The star (*) marks the fastest attacker, with approximately 1.5 hours to exploit hosts in all /24 networks of the Aachen honeynet. Moreover, it was the first scanning the complete honeynet during the monitored period.

Attacker	First seen	Last seen	Days	Scan
xxx.xxx.69.42	Apr 29 00:00:15	Sep 9 23:02:13	134	parallel
xxx.xxx.246.190	Apr 29 12:04:58	Aug 9 07:28:33	103	parallel
xxx.xxx.191.229	Apr 29 13:37:50	Sep 12 16:21:51	137	parallel
xxx.xxx.215.199	Apr 29 18:20:26	Sep 5 12:24:36	130	parallel
xxx.xxx.251.85	Apr 29 18:43:20	Aug 21 20:45:34	115	parallel
xxx.xxx.163.2	Apr 30 15:28:19	Jul 8 22:00:11	70	sequential
xxx.xxx.122.82	May 1 12:25:51	Sep 14 12:11:17	137	parallel
xxx.xxx.93.188	May 1 17:20:56	Sep 11 22:10:36	134	parallel
xxx.xxx.123.5	May 2 18:10:19	Sep 14 16:30:18	136	parallel
xxx.xxx.146.12	May 3 20:35:52	Jul 24 00:49:59	83	parallel
xxx.xxx.122.162	May 6 16:34:01	Aug 28 20:28:42	115	sequential
xxx.xxx.6.161	May 7 00:41:38	Jul 30 12:12:37	85	parallel
xxx.xxx.140.96	May 9 03:46:09	Sep 14 00:11:24	129	parallel
xxx.xxx.110.66	May 11 00:54:14	Aug 27 16:04:28	109	parallel
xxx.xxx.242.175	Jun 10 00:00:00	Sep 14 23:59:54	97	sequential
xxx.xxx.220.245	Jun 10 01:13:59	Aug 1 03:35:26	53	parallel
xxx.xxx.196.182	Jun 10 16:25:04	Aug 21 23:01:56	73	parallel
xxx.xxx.246.63	Jun 14 15:40:26	Sep 12 19:15:35	91	parallel
xxx.xxx.1.47	Jun 17 11:54:48	Sep 9 10:25:31	85	parallel
xxx.xxx.123.7	Jun 17 15:02:57	Sep 14 16:30:18	90	parallel
xxx.xxx.123.6	Jun 17 15:03:27	Sep 14 16:30:20	90	parallel
xxx.xxx.123.4	Jun 17 15:03:18	Sep 14 16:32:19	90	parallel
xxx.xxx.49.97	Jun 18 03:13:36	Jul 11 20:34:22	24	parallel
xxx.xxx.210.182	Jun 21 11:22:40	Jul 8 01:16:23	18	sequential
xxx.xxx.175.65	Jun 25 13:16:20	Sep 14 14:17:50	82	parallel
xxx.xxx.145.111	Jun 25 22:42:53	Jun 29 17:31:55	5	sequential
xxx.xxx.247.43	Jun 26 08:35:10	Jun 29 08:22:11	4	sequential
xxx.xxx.226.238	Jun 26 15:03:36	Jul 5 17:57:23	10	sequential
xxx.xxx.204.112	Jun 27 00:18:13	Jul 10 15:37:40	14	parallel
xxx.xxx.193.222	Jun 28 22:57:23	Jun 29 06:51:11	2	sequential
xxx.xxx.53.76 (*)	Jun 28 23:18:25	Jun 29 00:52:13	2	sequential
xxx.xxx.136.107	Jun 29 02:33:06	Jul 1 23:58:25	3	sequential
xxx.xxx.246.215	Jun 29 08:20:20	Jun 30 11:38:25	2	sequential
xxx.xxx.232.86	Jul 4 19:46:23	Jul 31 21:42:07	28	parallel
xxx.xxx.101.211	Jul 16 05:06:15	Jul 24 13:13:52	9	parallel
xxx.xxx.122.216	Jul 24 06:41:50	Jul 24 11:22:19	1	parallel
xxx.xxx.187.187	Jul 24 02:50:56	Jul 24 17:34:54	1	sequential

Table 3: Attack dates and scanning mechanisms of common attackers

A Malware Instruction Set for Behavior-Based Analysis

Philipp Trinius[1,*], Carsten Willems[1,⋆], Thorsten Holz[1,2,†], and Konrad Rieck[3,‡]

[1] University of Mannheim, Germany
[2] Vienna University of Technology, Austria
[3] Berlin Institute of Technology, Germany

[*]trinius@uni-mannheim.de [⋆]cwillems@cwse.de
[†]tho@iseclab.org [‡]konrad.rieck@tu-berlin.de

Abstract: We introduce a new representation for monitored behavior of malicious software called *Malware Instruction Set* (MIST). The representation is optimized for effective and efficient analysis of behavior using data mining and machine learning techniques. It can be obtained automatically during analysis of malware with a behavior monitoring tool or by converting existing behavior reports. The representation is not restricted to a particular monitoring tool and thus can also be used as a meta language to unify behavior reports of different sources.

1 Introduction

The field of malicious software (*malware*) is one of the most active and also one of the most challenging areas of computer security. In recent years, we are observing a huge increase in the number of malware samples collected by anti-virus vendors [Mic09, Sym09]. Therefore, it is mandatory that we develop tools and techniques to analyze malware samples with no or very limited human interaction.

Typically, we distinguish between *static* and *dynamic* malware analysis as well as, in the field of the latter one, also between *code* and *behavior* analysis. Code analysis can be performed in a static way by using a disassembler or decompiler and also dynamically by the usage of a debugger. The main advantage of code analysis is that we can obtain a complete overview of what a given software does. However, code analysis is often obstructed by evasion techniques, such as binary packers, polymorphism and anti-debug techniques [LD03, MKK07, PDA07]. In behavior analysis, the malware is seen as a black box and only its effects to the system, *its behavior*, is analyzed. This can be achieved by several existing monitoring tools. Most of these tools monitor one specific group of operation, e.g. registry or filesystem accesses. But there are also comprehensive analysis suites, which perform an overall monitoring of all of the malware's operations [BMKK06, SBY$^+$08, WHF07]. These suites execute a malware sample in a controlled environment and record all system-level behavior by monitoring the performed system calls. As a result, an analysis report is created summarizing the observed behavior of the sample. In contrast to code analysis, behavior-based dynamic analysis suffers less from evasion and

obfuscation techniques, as the code is not examined at all. Of course, there are different techniques, which can be used to prevent or falsify behavior analysis [SLJL08].

To handle the increasing amount and diversity of malware, dynamic analysis can be combined with clustering and classification algorithms. A behavior-based clustering of malware helps to find new malware families. Tagging new families at an early stage is essential to effectually fight them. Unlike clustering, the classification of malware does not provide information about new families, but helps to assign unknown malware to known families. Thus, classification of malware filters unknown samples and thereby reduces the costs of analysis. This combination of dynamic analysis and machine learning techniques has been recently studied in different scenarios (e.g., [BOA+07, BCH+09, KM06, LM06, RHW+08]).

For effective and efficient analysis, however, algorithms and data representations need to be adjusted to each other, such that discriminative patterns in data are accessible to learning methods. In this paper, we introduce a new behavior representation—the *Malware Instruction Set* (MIST)—which is exclusively designed for efficacy of analysis using data mining and machine learning techniques. It can be obtained automatically during analysis of malware with a behavior monitoring analysis tool or by converting existing behavior reports. The representation is not restricted to a particular report layout and thus can also be used as a meta language to unify behavior reports of different sources. Empirically, we demonstrate the accurate representation of behavior realized by MIST, while significantly reducing the size of reports and stored instructions.

This paper is organized as follows: the Malware Instruction Set is introduced in Section 2. In Section 3 we present a short empirical evaluation of MIST and demonstrate its capabilities. Finally, we conclude this paper in Section 4.

2 The Malware Instruction Set

The majority of monitoring suites, such as Anubis [BMKK06] and CWSandbox [WHF07], employ textual or XML-based formats to store the monitored behavior of malware. While such formats are suitable for a human analyst, they are inappropriate for further automatic analysis. The structured and often aggregated reports hinder application of machine learning. On the one hand, the XML representations are often too rich, providing an overspecific view on behavior which is not appropriate for finding generic behavioral patterns. On the other hand, textual formats are too coarse due to aggregation and simplification, such that involved patterns of behavior are not visible. Moreover, the complexity of textual representations increases the size of reports and thus negatively impacts run-time of analysis.

To address this problem and optimize processing of reports, we propose a special representation of behavior denoted as *Malware Instruction Set* (*MIST*) inspired from instruction sets used in processor design. In contrast to textual and XML-based formats, the monitored behavior of a malware binary is described as a sequence of instructions, where individual execution flows of threads and processes are grouped in a single, sequential report. Each

instruction in this format encodes one monitored system call and its arguments using short numeric identifiers, such as `'03 05'` for the system call `'move_file'`. The system call arguments are arranged in different levels of blocks, reflecting behavior with different degree of granularity. We denote these levels as *MIST levels*. Moreover, variable-length arguments, such as file and mutex names, are represented by index numbers, where a global mapping table is used to translate between the original contents and the index numbers.

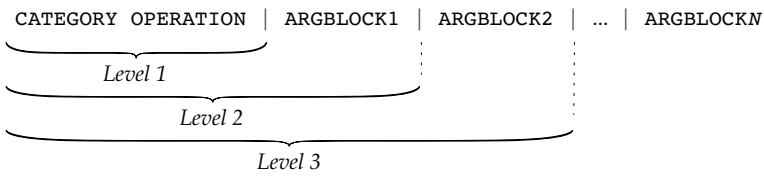


Figure 1: Schematic depiction of a MIST instruction. The field `CATEGORY` encodes the category of system calls where the field `OPERATION` reflects a particular system call. Arguments are represented as `ARGBLOCK`s.

Figure 1 shows the basic structure of a *MIST instruction*. The first level of the instructions corresponds to the category and name of a monitored system call. For example, `'03 05'` corresponds to the category `filesystem` (03) and the system call `'move_file'` (05). The following levels of the instruction contain different blocks of arguments, where the specificity of the blocks increases from left to right. The main idea underlying this rearrangement is to move "noisy" elements, such as temporary filenames, to the end of an instruction, whereas stable and discriminative patterns, such as directory and mutex names, are kept at the beginning. Thus, the granularity of behavior-based analysis can be adapted by considering instructions only up to a certain level. As a result, malware sharing similar behavior may be even discovered if minor parts of the instructions differ, for instance, if randomized file names are used.

After introducing the main concept of MIST, we describe the concrete representation for several selected system calls. Since MIST features 120 unique calls with their corresponding attributes, we can not describe all of them, but restrict ourselves to some representative examples to explain the overall design philosophy.

2.1 MIST Design

Every MIST report consists of several MIST instructions, which encode individual system calls monitored during run-time of a malware binary. While MIST can be obtained directly during dynamic analysis of a malware binary, we herein focus on translation of XML-based reports generated by the analysis tool CWSandbox [WHF07] to MIST. This conversion is order preserving, i.e., all contained MIST instructions occur in the same order as they were originally monitored and reported by CWSandbox.

As introduced previously, MIST instructions are composed of different fields: a `CATEGORY` field, an `OPERATION` field, and several `ARGBLOCK` fields. The field `CATEGORY` encodes the global class of the MIST instruction. As shown in Table 1, we distinguish between 20 different categories. Each category groups a set of related operations, e.g., the `'winsock_op'` category contains 13 MIST instructions, including the `'create_socket'`, `'connect_socket'`, and `'send_socket'` instructions. These instructions encode all system calls which are required to perform *Winsock* based network communication.

	Category	# syscalls		Category	# syscalls
01	Windows COM	4	*0B*	Windows Services	11
02	DLL Handling	3	*0C*	System	2
03	Filesystem	14	*0D*	Systeminfo	7
04	ICMP	1	*0E*	Thread	3
05	Inifile	5	*0F*	User	8
06	Internet Helper	5	*10*	Virtual Memory	5
07	Mutex	2	*11*	Window	5
08	Network	6	*12*	Winsock	13
09	Registry	9	*13*	Protected Storage	9
0A	Process	7	*14*	Windows Hooks	1

Table 1: MIST categories and encoding as well as the number of contained unique operations within each category

The amount and type of `ARGBLOCK`s for each MIST instruction depends on the particular system call. We will give some examples for those later in this text. We implement the concept of using *MIST levels* by dividing the ordered attribute blocks of each MIST operation into several levels, with higher level containing attributes with a higher *variability* and lower levels those which are more constant. The term variability is used with respect to different monitored behavior in multiple executions of one and the same sample, or executions of different variants from the same family. For example, if a monitored application creates a file in the Windows temp directory in one execution, it is highly likely that it will also create such a file in the very same folder in a second execution. In contrast to that, it is also likely that a different file name will be used, since temporary files are normally using random file names. Consequently, in a MIST operation we would encode the fact, that a file is created in level one, the target file path in level two, and the ultimate file name in level three.

The rationale underlying this rearragangement is that if we look at two executions of the similar programs on a lower level, e.g., level one, we observe identical behavior, whereas if we look on a higher level value, we are able to detect fine differences, e.g., in file names. This form of file name decomposition is strictly used in the MIST transformation: each file name is split up into the components `file type`, `file path`, `file name`, and `parameter`. In most cases the `file type` and `file path` are stored on a lower level than the rest, since these are more robust than other parts of file names.

2.2 Examples

load_dll. The `'load_dll'` system call is executed by every software during process initialization and run-time several times, since under Windows, dynamic-link libraries (DLLs) are used to implement the Windows subsystem and offer an interface to the operating system.

Figure 2 shows the original XML element of the CWSandbox representation and its corresponding MIST representation of one `'load_dll'` operation. Except for the attribute `filename_hash`, all attributes are transfused into the MIST representation. We sometimes discard some attributes, if they are not useful for later analysis steps. To achieve a case insensitive transformation, all attribute values are converted to lower case first. Then we apply a fast hash function, such as the standard ELF, and store the results in a lookup table. Finally, the resulting lookup index is used in our MIST representation as a hexadecimal number.

```
<load_dll filename="C:\WINDOWS\system32\kernel32.dll" successful="1"
 address="#7C800000" end_address="#7C908000" size="1081344"
 filename_hash="c88d57cc99f75cd928b47b6e444231f26670138f"/>
```

(a) CWSandbox representation

```
                    "c:\windows\system32\"                  parameter              successful
02 02 | 00006b2c 0c7d3f9c | 00108000 0c94b872 | 00000000 7C800000 7C908000 10
load_dll   "dll"              size   "kernel32"          address  end_address
```

(b) MIST representation

Figure 2: Feature representations of system call `'load_dll'`. The CWSandbox format represents the system call as an attributed XML element, while the malware instruction set (MIST) represents it as a structured string.

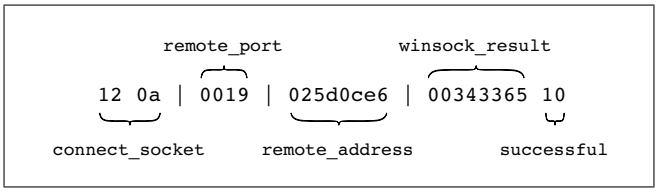
For the `'load_dll'` instruction we order the attributes as follows: the first attributes are the `file extension` and `file path` of the library. This information is quite constant and, therefore, is included in the second MIST level. Note that we order all attributes with respect to their *variability*. On level three we store the `file size` and the `file name` of the library. Both of these values may differ when two different variants of the same program are considered and, therefore, should only be stored on a lower—hence more detailed—MIST level.

It is evident from this example that the MIST instruction is more suitable for data mining and machine learning techniques than the traditional XML representation. The compact encoding ensures a proper comparability between all `'load_dll'` instructions and the ordering of all attributes permits the introduction of MIST levels, which finally enhances the quality of analysis.

connect_socket. If a malware binary initiates a TCP network communication via the Winsock library, it has to perform the `'connect_socket'` system call. Figure 3 shows a monitored system call in CWSandbox and MIST representation. The sandbox records five parameters for this system call, namely the used `socket number`, the `IP address` and `port` of the remote server, the winsock `result value` and the `successful` flag, which states if the connection could be established or not.

```
<connect_socket socket="1500" remote_addr="192.168.1.163"
 remote_port="25" successful="1" winsock_result="10035"/>
```

(a) CWSandbox representation



(b) MIST representation

Figure 3: Feature representations of system call `'connect_socket'`. The CWSandbox format represents the system call as an attributed XML element, while the malware instruction set (MIST) represents it as a structured string.

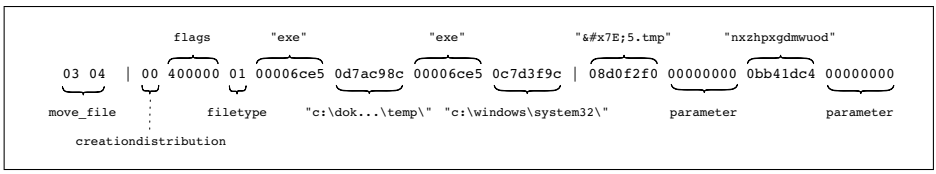
Except for the `socket number`, all attributes are converted into MIST representation (the socket number is a dynamic value that is created by the operating system and has no further semantic meaning). Furthermore, the attributes are ordered as show in Figure 3(b). Since the IP address of the remote server and the Winsock result value may vary quite often, this information is less interesting for our purpose and, therefore, moved into the third respectively fourth MIST level. Only the `remote port` remains in the second MIST level of this instruction.

move_file. The third example is one of the most complex conversions. As already shown, we decompose the file names and only use the file paths and extensions on the second MIST level, and delay the ultimate file names and possible parameters to a higher level. For the `'move_file'` call we have one filename-attribute that specifies the `source file`, and another one for the `destination file`. Thus, we have to split and arrange two file names. Figure 4 shows a monitored `'move_file'` system call in CWSandbox and MIST notation.

Again, the hash values are not converted into the MIST representation, because the malware binary may change, e.g., in discharge of the used packer. Therefore, only the ultimate file names and the parameters are those converted attributes which are not contained in the second MIST level, see Figure 4. In contrast to the file names and all prior discussed attributes, the values of the `filetype`, the `desiredaccess`, and the `flags` attributes

```
<move_file filetype="file" srcfile="C:\DOKU...\Temp\&#x7E;5.tmp.exe"
 srcfile_hash="hash_error"
 dstfile="C:\WINDOWS\system32\nxzhpxgdmwuod.exe"
 dstfile_hash="hash_error" desiredaccess="FILE_ANY_ACCESS"
 flags="MOVEFILE_REPLACE_EXISTING"/>
```

(a) CWSandbox representation



(b) MIST representation

Figure 4: Feature representations of system call `move_file`. The CWSandbox format represents the system call as an attributed XML element, while the malware instruction set (MIST) represents it as a structured string.

are not hashed while converting into MIST representation, but transferred directly into the MIST instruction. Since all possible values are known in advance, we use a fix mapping between the attribute values and the MIST values. Table 2 shows a fragment of the mapping table of the flags attribute. There are 26 predefined values which are freely combinable.

Value	MIST bit vectors
FILE_ATTRIBUTE_ARCHIVE	00000000000000000000000001
FILE_ATTRIBUTE_COMPRESSED	00000000000000000000000010
FILE_ATTRIBUTE_HIDDEN	00000000000000000000000100
FILE_ATTRIBUTE_NORMAL	00000000000000000000001000
FILE_ATTRIBUTE_OFFLINE	00000000000000000000010000
...	...
MOVEFILE_WRITE_THROUGH	00100000000000000000000000
MOVEFILE_CREATE_HARDLINK	01000000000000000000000000
MOVEFILE_FAIL_IF_NOT_TRACKABLE	10000000000000000000000000

Table 2: Mapping between CWSandbox and MIST representation for possible values of the flags attribute.

For attributes like `flags` or `desiredaccess`, we sum up the corresponding MIST bit vectors. The result is then interpreted as numeric value and transformed into hexadecimal encoding. This approach is robust against permutations between the single values and allows re-translating of the MIST encoding.

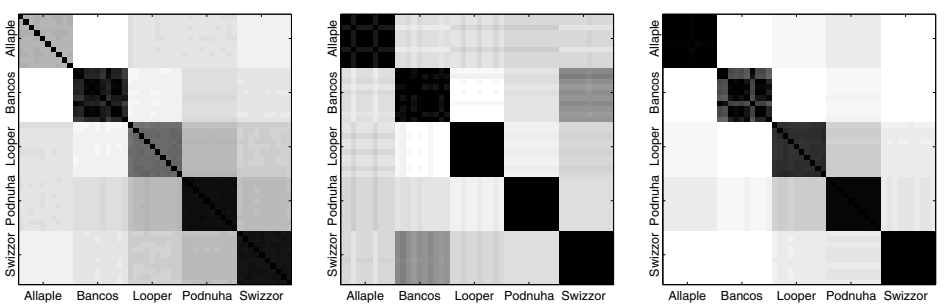
3 Empirical Evaluation

We now proceed to present an empirical evaluation of the MIST representation. For this evaluation, we consider a small sample of 500 malware binaries which have been obtained from the CWSandbox web site available at `http://cwsandbox.org`. The malware binaries have been collected over a period of more than two years from a variety of sources, such as honeypots, spam traps, anti-malware vendors, and security researchers. From the overall database, we select a subset of binaries which have been assigned to a known class of malware by the majority of six independent anti-virus products. Although anti-virus labels suffer from inconsistency [BOA+07], we expect the selection using different scanners to be reasonable consistent and accurate. The labeled malware binaries are then executed and monitored using CWSandbox, resulting in a total of 500 behavior reports of 5 common malware classes.

For each report we consider four different representations of behavior: First, the original XML format as generated by CWSandbox, second an extended version of the XML format proposed by [RHW+08] and, third and forth, the MIST representation of behavior for level 1 and level 2.

3.1 Behavior Representation

In this first experiment we compare the utility of MIST for representing malware behavior in a concise form. In particular, we apply the technique for embedding textual data of malware reports to a vector space introduced by [RHW+08]. That is, each report is represented by a vector, such that the similarity of behavior can be assessed in terms of geometric distances.



(a) XML (b) MIST level 1 (c) MIST level 2

Figure 5: Comparison of feature representations for malware behavior. Distance matrices for the behavior of five malware families are shown, each represented by ten reports. Dark shading indicates small distances and light shading large distances.

Figure 5 shows results for the representation using different formats. Distance matrices are shown for three behavior representations, where dark color indicates low distances and

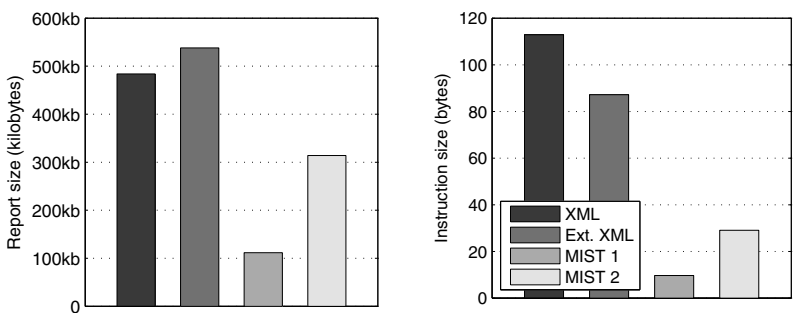
light color high distances. The extended XML format is omitted, as it only marginally differs from the regular XML representation. The representation provided by XML data is not sufficient to reflect all presented classes of malware. For the two malware families ALLAPLE and LOOPER the distance matrix for the XML data shows very light colored sections. Thus, the distance between the individual behavior reports is huge which precludes a good clustering or classification based on XML data. A threshold to classify all ALLAPLE members correctly would be too low for other malware families, for example, PODNUHA and SWIZZOR would also be classified as ALLAPLE.

The distance matrices for the MIST encoded data show a much darker coloration along the diagonal. The members of the individual malware families are much closer to each other, resulting in almost black colored sections for four malware families using MIST level 1 representation. Overall the distance matrix for MIST level 1 shows the closest match within the malware families, but in exchange also shows matches among different malware families, e.g., BANCOS and SWIZZOR. For MIST level 2 representation, we reduce the noise between the families at the cost of a sligthly less accurate separation between classes. This reduction of noise results from the more detailed information contained in MIST level2.

This experiment demonstrates the good representation of behavior realized by MIST. Although we have only studied five classes of malware, it is evident that MIST is more suitable for behavior-based analysis than XML, as samples of the same malware classes exhibiting similar behavior are close to each other in the vector space.

3.2 Data Reduction

In the second experiment we compare the size of instructions and reports between XML-based formats and MIST. Figure 6 shows a comparison for the four considered formats, namely the original XML format of CWSandbox, an extended representation used by [RHW+08], and MIST level 1 and 2.



(a) Average size of behavior reports

(b) Average size of instructions

Figure 6: Comparison of feature representations for malware behavior in terms of size.

The MIST representation significantly reduces both the length of the reports, as well as, the average length of instructions. While reports in XML format on average comprise more than 450 kilobytes, for MIST level 1 only 100 kilobytes and for MIST level 2 300 kilobytes are necessary to store the same behavior – though with a different granularity. Similarly, for the size of instructions MIST provides a more concise representation, where on average instruction requires less than 40 bytes. By contrast, for both XML formats the required size per instruction is twice as large, yielding over 80 bytes. Particularly with regard to data mining and machine learning methods this data reduction is quite important, since it dramatically reduces the run-time of analysis on large datasets.

This experiment demonstrates the advantages of representing behavior using the proposed MIST representation. The behavior is represented in a way which allows far better to discriminate classes, but at the same time storage size is significantly reduced, which ultimately provides the basis for effective and efficient further analysis of monitored malware behavior.

4 Conclusions

The *Malware Instruction Set* (MIST) is a meta language for monitored behavior, which can be used to make the results of different behavior monitoring systems more comparable. In addition, the MIST representation is optimized for analysis of software behavior using data mining and machine learning techniques. We restrict all instructions to significant attributes only and rearrange these attributes to achieve a well sorted ordering which allows the introduction of accuracy levels. Furthermore, we reduce the size of the reports by encoding each instruction and attribute with the help of hash tables. In summary, using the MIST representation as input enhances both the run-time and the results of data mining and machine learning analysis.

Acknowledgements

This work has been accomplished in cooperation with the German Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik (BSI)).

References

[BCH+09] U. Bayer, P.M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, Behavior-Based Malware Clustering. In *Proceedings of Symposium on Network and Distributed System Security (NDSS)*, 2009.

[BMKK06] U. Bayer, A. Moser, C. Kruegel, and E. Kirda. Dynamic Analysis of Malicious Code. *Journal in Computer Virology*, 2(1):67–77, 2006.

[BOA+07] Michael Bailey, Jon Oberheide, Jon Andersen, Z. Morley Mao, Farnam Jahanian, and Jose Nazario. Automated Classification and Analysis of Internet Malware. In *Proceedings of Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 178–197, 2007.

[KM06] J.Z. Kolter and M.A. Maloof. Learning to Detect and Classify Malicious Executables in the Wild. *Journal of Machine Learning Research*, 2006.

[LD03] Cullen Linn and Saumya Debray. Obfuscation of Executable Code to Improve Resistance to Static Disassembly. In *Proceedings of Conference on Computer and Communications Security (CCS)*, 2003.

[LM06] Tony Lee and Jigar J. Mody. Behavioral Classification. In *Proceedings of Annual Conference of the European Institute for Computer Antivirus Research (EICAR)*, April 2006.

[Mic09] Microsoft. Microsoft Security Intelligence Report (SIR). Volume 7 (January – June 2009), Microsoft Corporation, 2009.

[MKK07] A. Moser, C. Kruegel, and E. Kirda. Exploring multiple execution paths for malware analysis. In *Proceedings of IEEE Symposium on Security and Privacy*, 2007.

[PDA07] Igor V. Popov, Saumya K. Debray, and Gregory R. Andrews. Binary Obfuscation Using Signals. In *Proceedings of the 16th USENIX Security Symposium*, pages 275–290, 2007.

[RHW+08] Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Düssel, and Pavel Laskov. Learning and Classification of Malware Behavior. In *Proceedings of Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, pages 108–125, 2008.

[SBY+08] Dawn Song, David Brumley, Heng Yin, Juan Caballero, Ivan Jager, Min Gyung Kang, Zhenkai Liang, James Newsome, Pongsin Poosankam, and Prateek Saxena. BitBlaze: A New Approach to Computer Security via Binary Analysis. In *Proceedings of the 4th International Conference on Information Systems Security*, Hyderabad, India, December 2008.

[SLJL08] M. Sharif, A. Lanzi, G. Jonathon, and W. Lee. Impeding Malware Analysis Under Conditional Code Obfuscation. In *Proceedings of Symposium on Network and Distributed System Security (NDSS)*, 2008.

[Sym09] Symantec. Internet security threat report. Volume XIV (January – December 2008), Symantec Corporation, 2009.

[WHF07] Carsten Willems, Thorsten Holz, and Felix Freiling. CWSandbox: Towards Automated Dynamic Binary Analysis. *IEEE Security and Privacy*, 5(2), March 2007.

State-of-the-Art Kryptoverfahren für drahtlose Sensornetze – Eine Krypto-Bibliothek für MantisOS

Björn Stelte und Björn Saxe
Institut für Technische Informatik
Universität der Bundeswehr München
Neubiberg, Germany
{bjoern.stelte, bjoern.saxe}@unibw.de

Abstract: Gerade im Bereich der drahtlosen Sensornetze muss aufgrund der Ressourcen-Knappheit auf aufwendige Kryptoverfahren verzichtet werden. Dennoch ist gerade für den Einsatz von Sensornetzen in hoch-schutzbedürftigen Umgebungen eine vertrauenswürdige Nachrichtenkommunikation notwendig. Eine Verschlüsselung der Kommunikation unter den Sensorknoten wie auch zur Datensenke des Sensornetzes ist die Grundlage für eine sichere Authentifizierung und kann zur Lösung einiger bekannter Angriffe auf Sensornetze beitragen. Oftmals werden nur etablierte älterer Kryptoverfahren in Sensornetzen verwendet. In dieser Arbeit werden für den Einsatz in Sensornetzen vielversprechende neuere Verfahren untersucht und die Ergebnisse einer Untersuchung anhand einer beispielhaften Implementierung der Verfahren in MantisOS gezeigt. Die hieraus entstandene Software-Bibliothek von state-of-the-art Kryptoverfahren für MantisOS kann als Ausgangspunkt für zukünftige Sicherheitslösungen in drahtlosen Sensornetzen dienen.

1 Einleitung

Drahtlose Sensornetze bestehen aus hunderten oder tausenden von kleinen low-power Sensorknoten. Diese Knoten können mithilfe ihrer Sensorik Ereignisse in ihrer unmittelbaren Umgebung detektieren und per Radio-Module Nachrichten hierüber versenden. Die Nachrichten werden in einer Datensenke gesammelt und ausgewertet.

Viele Anwendungen im häuslichen Umfeld existieren bereits heute oder befinden sich in Planung. Die drahtlosen Sensornetze werden ebenfalls immer beliebter für Szenarien aus dem hochsicherheits Bereich, wie Objektschutz oder der Gebietsüberwachung. Auf dem Markt befindliche Sensorknoten sind jedoch bedingt durch ihre knappen Ressourcen (Rechenleistung, Speicherkapazität, Laufzeit, Batteriekapazität) nur bedingt für die genannten Szenarien aus dem hochsicherheits Bereich geeignet. Vor allem fehlen häufig effiziente Implementierungen von Kryptoverfahren, so dass in vielen Anwendungen Nachrichten nicht fälschungssicher übertragen bzw. kryptographische Signaturen keine Anwendung finden. Dieses Fehlen von geeigneten Kryptoverfahren ermöglicht potentiellen Angreifern Sensornetze mit wenig Aufwand ausser Gefecht zu setzen. So sind viele Angriffe aus der Kategorie 'man-in-the-middle' für drahtlose Sensornetze bereits bekannt, wie bspw. sniffing, data integrity, energy drain, black hole, hello flood, wormhole. Abgesehen von den in

jüngster Zeit stark beachteten Feld der Verschlüsselungverfahren mit elliptischen Kurven (ECC) hat sich auch in der klassischen Kryptographie einiges getan. Wir werden uns hier speziell mit symmetrischen Kryptoverfahren und den Hash-Verfahren auseinandersetzen.

Diese Arbeit ist wie folgt aufgebaut, in Kapitel 2 möchten wir Sicherheitsproblem und Angriffe auf die Datenübertragung des Sensornetze näher beleuchten. Eine exemplarische Implementierung von Kryptoverfahren für das Sensor-Betriebssystem MantisOS stellen wir in Kapitel 3 vor. Bekannte und neuere viel versprechende Verfahren werden kurz vorgestellt und die Verwendung der Bibliothek erläutert. In Kapitel 4 folgt der Vergleich und die Bewertung der implementierten Kryptoverfahren in Bezug auf ihrer Verwendbarkeit auf handelsüblichen Sensorknoten. Im letzten Kapitel fassen wir die Ergebnisse unserer Arbeit kurz zusammen und geben eines Ausblick auf mögliche Erweiterungen.

2 Sicherheitsproblem und Angriffe

Durch fehlenden Sicherheitsmechanismen ist die Datensicherheit gefährdet, so ist es ohne Verschlüsselung dem Angreifer möglich, den kompletten Netzverkehr abzuhören. Er kann somit Erkenntnisse über das Netz gewinnen und sie für geeignete Angriffe nutzen oder aus den Informationen neue gefälschte Informationen generieren und somit gezielt das Sensornetz zu stören.

Auch die Verfügbarkeit des Sensornetzes ist gefährdet. Ein Angreifer hat die Möglichkeit mit Denial-of-Service Angriffen, die Nutzung des ganzen oder auch von Teilen des Sensornetzes durch den Betreiber zu verhindern. [WS02] gibt einen guten Überblick über Denial-of-Service Angriffe auf drahtlose Sensornetze. So sind Denial-of-Service Angriffe prinzipielle auf jeder Schicht des Netzwerks möglich. Bezg. der Vermittlungsschicht könnte der Angreifer einzelne Sensorknoten durch manipulierte Knoten ersetzen, die eine Weiterleitung von Routing-Nachrichten unterbinden oder gezielt falsche Routing- Nachrichten verbreiten. Beispielsweise sind die Routingprotokolle, die nach Distanz-Vektor Verfahren funktionieren, anfällig für so genannte Black Hole Angriffe. In der Transportschicht gibt es hauptsächlich zwei Möglichkeiten: Flooding und Desynchronisation. Protokolle, die ihren Verbindungszustand speichern müssen, sind anfällig für Flooding-Angriffe. Die einfachste Variante ist das Senden von sehr vielen Verbindungsaufbauanforderungen an den Empfänger durch den Angreifer. Mit jeder dieser Anforderung muss ein kleiner Teil des Speichers für einen Verbindungskontext reserviert werden, auch wenn die Verbindung nie komplett aufgebaut wird. Mit genügend Anforderungen kann man so den Speicher des Empfängers überfüllen. Bei der Desynchronisation verschickt der Angreifer wiederholt Pakete mit manipulierten Sequenznummern oder vergleichbaren Kontrollschaltern an einen oder beide Kommunikationsenden. Die veränderten Kontrollinformationen veranlassen die Kommunikationspartner dazu, die vermeintlich verlogen gegangene Pakete neu zu übertragen. Mit geschicktem Timing kann der Angreifer so eine sinnvolle Kommunikation verhindern und Sender und Empfänger halten sich an zeit- und energieaufwändigen Synchronisationsmechanismen auf.

Mit Angriffen auf die Verfügbarkeit kann der Angreife eine Verhinderung und Verzögerung

von Meldungen erreichen. Da solche Angriffe zumindest auf physischer Ebene ohne grossen Aufwand durchzuführen sind, ist das Risiko für unser Szenario hoch. Die Sicherstellung der Datenintegrität und Authentizität unter den Netzwerkteilnehmern ist ebenfalls für die Gesamtsicherheit eines Netzes unerlässlich. Die reine Verschlüsselung von Nachrichten ohne Authentifizierung und Sicherstellung der Integrität hat sich schon mehrfach als verwundbar erwiesen [Bel96][BGW01][Kra01]. Ohne Integrität kann der Empfänger nicht feststellen, ob die vom Sender gesendete, ursprüngliche Nachricht verändert wurde. Schon durch Vertauschen einzelner Bits lassen sich gezielt vorhersagbare Änderungen im Klartext erzeugen [BGW01]. Ohne Authentifizierung können die Kommunikationspartner nicht sicherstellen, ob ihr Gegenüber auch derjenige ist, der er vorgibt zu sein. Dem Angreifer fällt es so deutlich leichter z.B. von ihm manipulierte Knoten in das Sensornetz einzuschleusen.

Angriffe auf die Integrität und Authentizität, die es dem Angreifer ermöglichen Nachrichten zu verfälschen, sind aufwändiger als Angriffe auf die Verfügbarkeit. Jedoch kann bei einem solchen Angriff der mögliche Schaden auch deutlich höher liegen, da der Angreifer unter Umständen die Kontrolle über das Netz erlangen und es für seine Zwecke Missbrauchen kann und dass im schlimmsten Fall dies noch nicht einmal bemerkt wird.

Die Kryptographie löst zwar nicht alle Probleme, jedoch ist der Einsatz von Kryptoverfahren unabdingbar, um Sensornetze in kritischen Infrastrukturen einsetzen zu können. So können bspw. Hash-Verfahren zur digitalen Signatur und Blockchiffren zur Nachrichtenverschlüsselung eingesetzt werden. Wir werden im nächsten Kapitel eine kryptographische Bibliothek für das Sensor-Betriebssystem MantisOS vorstellen. Kryptoverfahren aus dem Bereich der Blockchiffren und Hash-Verfahren wurden implementiert und als Bibliothek in MantisOS, einem bekannten Sensornetz-Betriebsystem, integriert.

3 Kryptographie Bibliothek für MantisOS

Wie für fast alle Arten von Rechnern, unabhängig von Größe und Einsatzgebiet, existieren auch für Sensorknoten Betriebssysteme, die für einen abstrahierten Zugriff auf Hardware der Sensorknoten und für Prozess- und Speicherverwaltung zuständig sind. Diese Arbeit beschäftigt sich mit MultimodAl system for NeTworks in In-situ wireless Sensors OS (MantisOS). MantisOS ist ein von der University of Colorado entwickeltes Betriebssystem für ein eingebettete Systeme. Es im Gegensatz zu anderen Vertretern seiner Art wie z.B. TinyOS von der University of California Los Angeles plattformübergreifend und multithreaded (TinyOS z.B. ist ereignisorientiert). Das Betriebssystem ist nach dem klassischen Schichtenmodell aufgebaut: Es besteht aus Kernel mit Scheduler, COMM-Layer, DEV-Layer und dem Netzwerk-Stack. Der Scheduler unterstützt Prioritätenvergabe und hat verschiedene Optimierungen, wie z.B. das Schlafen legen von Threads, um den Energieverbrauch zu minimieren. MantisOS enthält bis auf eine RC5 und CCITT CRC-16 Implementierung bislang keine eingebauten kryptographischen Funktionen.

3.1 Auswahl geeigneter Verfahren

Aus den Überlegungen zum vorherigen Kapitel ergeben sich zweierlei Arten von Anforderungen. Die rechen-betonten Anforderungen sind abhängig von der Art der Implementierung und den Fähigkeiten des Programmierers. Hier kommt es darauf an, dass ein Verfahren möglichst wenige, einfache Rechenoperationen durchführt, da diese massgeblich die Laufzeit und, im Falle von Sensorknoten den Energieverbrauch erheblich beeinflussen. Ein geringer Speicherverbrauch, sowohl während der Berechnung als auch für die Speicherung von eventuell benötigten Konstanten, ist ebenso wichtig. Ein Verfahren muss zur Laufzeit so wenig Speicher wie möglich verbrauchen, um für die eigentliche Anwendung genügend Speicher freizuhalten. Zugriff auf externen Speicher sollte möglichst vermieden werden, da mit jedem Zugriff der Energieverbrauch stark ansteigt. Interessant sind besonders Verfahren, deren Implementierungsaufwand gering ist oder hinsichtlich Ressourcen beschränkte Hardware wie z.B. Smart-Cards entwickelt wurden.

Des weiteren gibt es davon unabhängige kryptographische Anforderungen wie z.B. Schlüssel- und Hashlängen sowie bekannte Angriffe auf ein Verfahren und deren praktische Relevanz. Für die öffentliche und industrielle Verwendung von kryptographischen Techniken gibt es Empfehlungen von Regierungsorganisationen, so hat auch das Bundesamt für Sicherheit in der Informationstechnik (BSI) im Falle von Deutschland eine Technische Richtlinie [TR-] veröffentlicht, die Empfehlungen und Schlüssellängen für kryptographische Verfahren beinhaltet. Viele Verfahren werden im Rahmen von Auswahlverfahren für solche Empfehlungen ausführlich untersucht. Eines dieser Projekte ist das NESSIE-Projekt New European Schemes for Signatures, Integrity, and Encryption [dNP] der europäischen Kommission. Dort wurde bereits 2000 ein Aufruf zur Einreichung von kryptographischen Algorithmen gestartet, die zukünftigen Anforderungen an Sicherheit und Vorgaben an Hard- und Softwareumgebungen genügen sollten. Im Jahr 2003 wurden dann von ursprünglich 39 Algorithmen im Rahmen eines zweiphasigen Auswahlprozesses 17 Verfahren ausgewählt. Auch das amerikanische NIST National Institute of Standards and Technology benutzt Wettbewerbsverfahren zur Auswahl und Bestimmung neuer kryptographischer Standardverfahren. Momentan wird der Nachfolger des SHA Hash-Verfahrens in der Cryptographic Hash Algorithm Competition [dCHAC] bestimmt. Wir haben in unserer Bibliothek einige der bekannten und etablierten Verfahren neben neueren Verfahren aus dem CHAC Wettbewerb aufgenommen. Wir möchten hier nur einige der implementierten Verfahren kurz beschreiben, stellvertretend für die unserer Ansicht nach bedeutenden Verfahren für den Einsatz in Sensornetzen.

3.1.1 Hash-Verfahren

SHA-1 und SHA-256 ist eine vom NIST und der NSA zusammen entwickelte Hash-Funktion für den Digital Signature Standard und steht für Secure Hash Algorithm. In seiner ursprünglichen Version SHA-0 von 1994 hatte es einen Designfehler, den die 1995 erschienene Variante SHA-1 korrigierte. Die Funktion existiert in verschiedenen Varianten wie SHA-256 und SHA- 384, wobei die Zahl die Länge des von ihr berechneten Hash-Werts in Bits angibt. Ab SHA-224 spricht man auch von der SHA-2 Familie. 2005 und

2006 wurden mehrere Kollisionsangriffe gegen SHA-1 veröffentlicht [WYY05], die in naher Zukunft durchaus von praktischer Relevanz sein könnten. Seit der Veröffentlichung dieser Angriffe empfiehlt das NIST nur noch Verfahren der SHA-2 Familie zu verwenden. Auf längere Zeit gesehen sollen die Verfahren durch neue ersetzt werden und das NIST hat dazu wie schon beim Advanced Encryption Standard einen Wettbewerb ausgeschrieben. Die genaue Funktion von SHA ist im RFC 3174 beschrieben. Somit ist Verwendung von SHA-1 für nicht unbedingt sicherheitsrelevante Daten auf jeden Fall ausreichend, während SHA-256 als Mitglied der SHA-2 Familie auch für sicherheitskritische Daten verwendet werden kann.

BLAKE ist einer der aussichtsreichsten Kandidaten für SHA-3 aus der bereits erwähnten CHAC Ausschreibung. BLAKE hat bereits die erste Runde der Ausschreibung erfolgreich bestanden und ist nun mit 13 weiteren Kandidaten in der zweiten Runde. Es wurde im Auswahlprozesses ausführlich untersucht und hat bisher keine Schwächen gezeigt. Es sind insgesamt vier verschiedene Varianten verfügbar: BLAKE-28, 32, 48, 64. Die Zahl gibt dabei die Länge des Hashwert in Bytes an. Im Gegensatz zu einigen anderen SHA-3 Kandidaten besticht BLAKE mit seinem geringen Speicherbedarf und ist daher besonders für die Sensornetze von Interesse. Die Spezifikation von BLAKE haben die Autoren auf ihrer Website veröffentlicht [JPLWP09].

3.1.2 Symmetrische Verfahren

AES der Advanced Encryption Standard ist der Nachfolger von DES und wurde 2000 als neuer Standard bekannt gegeben. Vor seiner Ernennung hieß es nach den Namen seiner Entwickler Rijndael-Algorithmus. AES ist in den USA für Dokumente mit der höchsten Geheimhaltungsstufe zugelassen. Die Blockgröße ist festgelegt auf 128 Bit, mögliche Schlüssellängen sind 128, 192 und 256 Bit, die entsprechenden Verfahren heißen dann AES-128, AES-192 und AES-256. Es gibt bisher mehrere Angriffsansätze und -versuche (vgl. [BKN09]), doch keiner ist bisher von praktischer Bedeutung. Die offizielle Spezifikation von AES kann man im FIPS-197 [FIP] nachlesen. AES gehört zu den vom BSI empfohlenen Blockchiffren.

RC6 war einer der Kandidaten der AES Ausschreibung und hat es bis in die finale Runde geschafft. Er 1998 entwickelt als Nachfolger und auf der Grundlage von RC5. Wie auch RC5 hat RC6 variable Blockgrößen, Rundenzahlen (0-255) und Schlüssellängen (0-2048 Bit). Die bisher gefundenen möglichen Angriffe sind praktisch nicht durchführbar oder lassen sich durch entsprechende Wahl der Parameter verhindern [GHJ+01][BPVV99]. Auch RC6 ist patentiert, die komplette Spezifikation ist in [RRSY98] veröffentlicht.

Serpent ist von Ross Anderson, Eli Biham und Lars Knudsen ebenfalls als AES-Kandidat entwickelt worden landete auf dem zweiten Platz hinter Rijndael. Wie die meisten AES Kandidaten hat Serpent eine Blockgröße 128 Bit, aber variable Schlüssellänge von 0-256 Bit in 8 Bit Schritten. Die Rundenzahl ist höher als bei AES (10-14 vs 32) und hat somit

vermutlich eine höhere Sicherheit. Allerdings dürfte sich diese erhöhte Rundenzahl auch in der Geschwindigkeit niederschlagen. Serpent-128, -192 und -256 gehören zu den vom BSI empfohlenen Blockchiffren, eine Beschreibung zu Serpent findet sich [ABK].

3.2 Nutzung der Bibliothek

In diesem Abschnitt soll die allgemeine Vorgehensweise zur Einbindung eines kryptographischen Verfahrens in eine MantisOS Anwendung erläutert werden.

1. Header-Datei einbinden – Die Header-Dateien befinden sich im Verzeichnis SRC/-LIB/INCLUDE/CRYPTOLIB/.
2. Variablen deklarieren – für die Aufnahme des Hashwerts oder der Schlüssel bzw., wenn das Verfahren es erfordert, Kontextdatentypen.
3. Initialisierungen durchführen – nicht alle Verfahren benötigen diesen Schritt. Zur Initialisierung werden immer Schlüssel und Kontextvariable (und selten eventuell weitere Parameter, wie Rundenzahl) benötigt.
4. Berechnung des Hashwerts / Verschlüsselung – die eigentliche Berechnung erfolgt bei den Hashverfahren mit dem Aufruf der Hashfunktion. Übergeben wird die Nachricht und deren Länge in Bits. Bei den Blockchiffren wird die Verschlüsselung mit Chiffre-enc(), die Entschlüsselung Chiffre-dec() aufgerufen. Bei Verfahren, die mit Kontextvariablen arbeiten, übergibt man den Kontext und die Nachricht, ansonsten den Schlüssel und die Nachricht.
5. Freigabe von Ressourcen – ist Ver- oder Entschlüsselung erfolgt und wird ein Verfahren nicht länger benötigt, bietet es sich an den Speicherplatz, der für Kontextvariable reserviert wurde, wieder freizugeben.

4 Vergleich und Bewertung der Verfahren

Die Eingaben für die folgenden Messung (Schlüssel und Nutzdaten) bestanden aus Zufallsdaten. Alle Messungen wurden mit 100 unterschiedlichen Schlüsseln und Nutzdatensätzen durchgeführt und anschliessend stochastisch bewertet.

Die Messungen wurden mit dem Simulator Avrora (AVR Simulation and Analysis Framework Plattform) durchgeführt. Avrora wurde von der Compilers Group der University of California Los Angeles entwickelt [TT05][Web]. Mit dem Tool können Programme, die für AVR-Mikrokontroller und MICA2 Sensorknoten geschrieben wurden, simuliert und genau analysiert werden. Zusätzlich beinhaltet das Programm ein Java-Framework, mit dem der Simulator erweitert werden kann, etwa durch hinzufügen neuer Monitore. Der Simulator arbeitet auf Instruktionsebene auf den Taktzyklus genau. Die Simulation umfasst nicht nur den reinen AVR-Mikrokontroller, sondern auch externe Komponenten wie Sensoren und Netzfunktionalität (wie Chipcon CC1000 Transmitter).

4.1 Vergleich Taktzyklen

Der Avrora-Simulator kann über seinen energy-profile-Monitor die zur Abarbeitung eines Programms benötigten Taktzyklen anzeigen. Diese sind dann nach den ausgeführten Funktionen aufgeschlüsselt. Zur Bestimmung der von einem Verfahren benötigten Taktzyklen wurden die Werte der vom Verfahren verwendeten Funktionen aufaddiert. Die Werte für Ver- und Entschlüsselung bei Verfahren, die eine Initialisierungsfunktion vor ihrer Verwendung benötigen, wurden aus dem Gesamtwert für Initialisierung + Ver- bzw. Entschlüsselung ermittelt, d.h., vom Gesamtwert wurde der Wert für die Initialisierung und Freigabe wieder abgezogen.

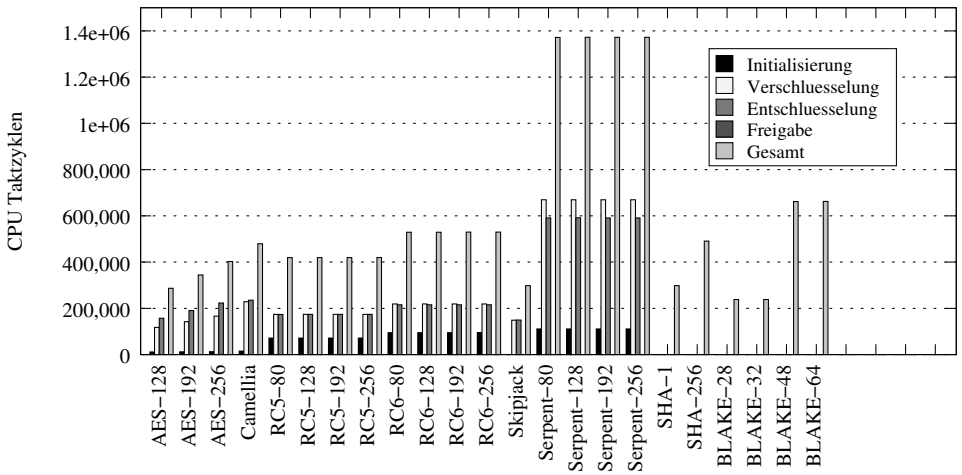


Abbildung 1: Taktzyklen Blockchiffren bzw. Berechnung des Hashwerts von 128 Byte Daten

Bei den Verfahren, die eine freie Wahl der Schlüssellänge anbieten, wirkt sich diese praktisch nicht auf die Zahl der benötigten Taktzyklen aus. Bei AES dagegen steigt sie mit der Schlüssellänge. Bei allen Verfahren bis auf AES sind auch die Werte für Ver- und Entschlüsselung entweder gleich oder minimal verschieden. Serpent benötigt mit Abstand am längsten für jeden Teilschritt, was an der hohen Anzahl der Runden (32, bei AES-128 z.B. nur 10) liegt. Lässt man die Zeiten für die eher selten ausgeführte Initialisierung weg, liegt die AES-128 Implementierung sogar auf gleicher Höhe mit dem eher einfachen und in die Jahre gekommenen Skipjack.

Die kleinen BLAKE-Funktionen (28/32) brauchen für die Berechnung eines 28 bzw. 32 Byte langen Hashwerts geraden einmal halb solange wie SHA-256, das ebenfalls einen 32 Byte lange Hashwert berechnet. Selbst SHA-1 mit einer Hashwertlänge von nur 20 Byte benötigt etwas länger als BLAKE-28/32. Die großen BLAKE-Funktionen benötigen entsprechend ihrer größeren Hashlänge noch mal deutlich länger. Die geringen Unterschiede zwischen BLAKE-28 und BLAKE-32 bzw. BLAKE-48 und BLAKE-64 ergeben sich dadurch, dass, vereinfacht gesagt, die Berechnung bei beiden gleich ist, der Hashwert aber

am Ende der Berechnung einfach abgeschnitten wird.

4.2 Vergleich Speicherverbrauch

Der Simulator bietet die Möglichkeit, alle während des Programmablaufs stattfindenden Lese- und Schreibzugriffe auf den Speicher zu protokollieren. Dieser Speichermonitor zeigt aber die Zugriffe nur in der Form “Adresse - Anzahl gelesene/geschriebene Bytes” an. Um daraus den Speicherverbrauch einzelner Funktionen abzulesen, fehlen einfach die nötigen Informationen. Auch die Ermittlung von groben Richtwerten anhand der Segmentgrößen der Data und BSS Segmente (Segmente für globale, statische Variablen) hätte bei den vorliegenden Implementierungen nicht zu brauchbaren Ergebnissen geführt (der statische Verbrauch ist fast überall 0 Bytes), da fast alle Variablen erst innerhalb von Funktionen deklariert werden. Zudem fehlt bei dieser Methode der durch Unterprogrammaufrufe wachsende Stack. Aus diesen Gründen wurde der Speichverbrauch manuell anhand des Quellcodes errechnet. Dazu wurde zunächst der Verbrauch der einzelnen Funktionen ermittelt, dann der tiefste Unterprogrammaufruf innerhalb eines abgeschlossene Schritts des jeweiligen Verfahrens (also z.B. Verschlüsseln eines Blocks) herausgesucht, um dann durch aufaddieren der Werte den maximalen Verbrauch zu bestimmen. Zu den lokalen Variablen innerhalb einer Funktionen wurden auch Rückgabewerte und die an die Funktionen übergebene Parameter mitgezählt. Auch bei dieser Methode ist die exakte Stackgröße schwierig zu ermitteln, insgesamt ist dieser Ansatz aber deutlich aussagekräftiger als nur den statischen Verbrauch der Segmente aufzuführen.

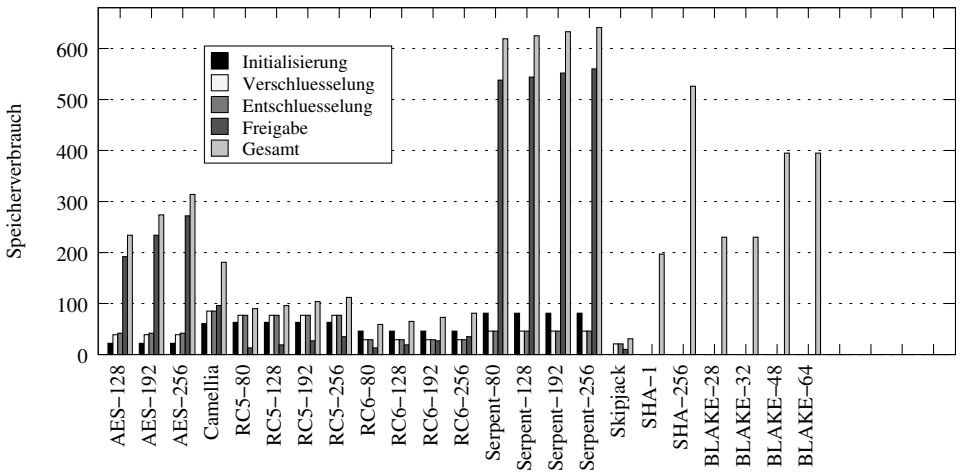


Abbildung 2: Speicherverbrauch RAM, alle Werte in Bytes

Zunächst fällt auf, dass sich bei den Verfahren, die variable Schlüssellänge unterstützen, die größere Schlüssellänge kaum im Speicherbedarf niederschlägt, nur der statische An-

teil des Schlüssels steigt zwangsläufig. RC5, RC6 befinden sich mit 100 Bytes ungefähr auf einem Niveau. AES und Serpent fallen durch einen großen statischen Anteil auf, der durch die Kontextvariablen zur Speicherung der Rundenschlüssel verursacht wird. Der Anteil von Serpent ist dabei durch die höhere Rundenzahl (32 vs 10-14) im Vergleich zu AES besonders groß. Serpent braucht somit auf einer üblichen 8-Bit Plattform mit 4 kByte RAM etwas mehr als ein Viertel des gesamten Speichers auf. Skipjack benötigt mit Abstand am wenigsten, es hat keine Kontextvariablen und auch der Verbrauch während Ver- und Entschlüsselung ist sehr niedrig.

Der Speicherverbrauch für SHA-256 und BLAKE-48/64 ist für einen sehr ressourcenschwachen Knoten wie MICA2 zu hoch. SHA-1 und die kleinen BLAKE-Funktionen liegen auf dem Niveau der meisten Blockchiffren.

4.3 Vergleich Codegröße

Die Größe des Programmcodes wurde anhand der Größe des text-Segments in Objektdateien ermittelt. Die Codegröße aller Verfahren liegt sehr platzsparend im unteren, einstelligen KByte Bereich. Nur Camellia reißt nach oben aus und stellt sich bei ohnehin vollem Flashspeicher als nicht geeignet heraus. Ähnlich sieht es bei den Hashfunktionen aus, SHA-1 ist etwas kleiner als der Rest, die BLAKE-48/64 Verfahren sind mehr als doppelt so groß.

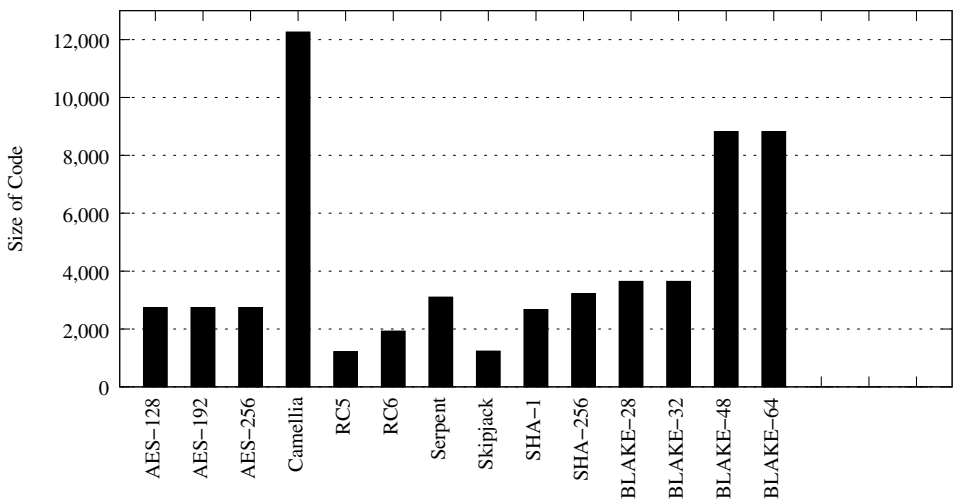


Abbildung 3: Codegröße im Flash, alle Werte in Bytes

4.4 Vergleich Flashspeicher

Zur Bestimmung der Anzahl der gelesenen Bytes aus dem Flash-Speicher wurden in den Quellcode Zähler eingefügt. Jedes mal nach dem Aufruf der speziellen Lesefunktionen (pgm_read_byte() etc.) wurde dieser entsprechend hochgezählt. Hierbei gilt es zu beachten, dass die Implementierungen von RC5, RC6, SHA-1 und SHA-256 keine Werte im Flash speichern.

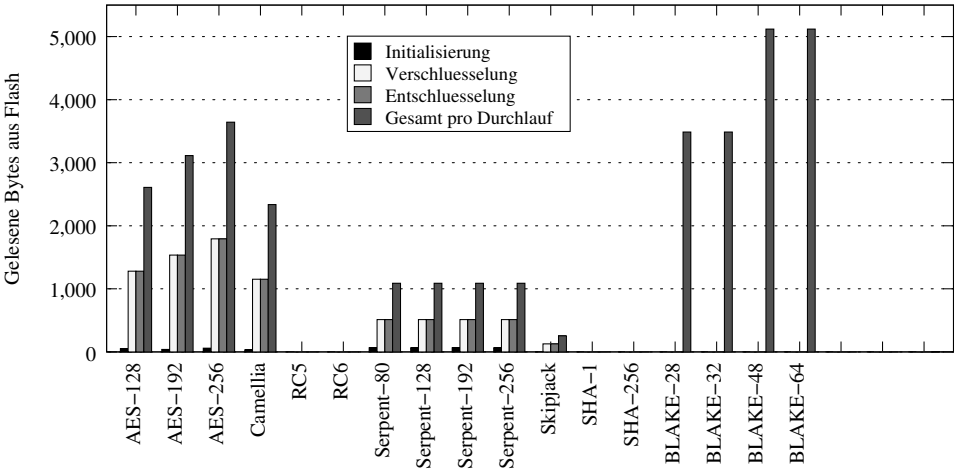


Abbildung 4: Gelesene Bytes aus dem Flash-Speicher

Die Anzahl der gelesen Bytes zur Initialisierung entspricht bei jedem Verfahren in etwa der Menge, die benötigt wird, um einen einzelnen Block zu verschlüsseln, und fällt somit nicht sonderlich ins Gewicht. Bei Serpent beeinflusst, wie auch beim RAM Verbrauch, die Schlüssellänge die Anzahl der Flashoperationen nicht, bei AES steigt die Anzahl der Flashzugriffe mit steigender Schlüssellänge. Wie beim RAM-Verbrauch, so auch bei den Flashoperationen, liest Skipjack am wenigsten aus dem Flashspeicher, abgesehen von RC5 und RC6, die überhaupt keine Werte im Flashspeicher ablegen. Bei den Hashfunktionen kommen SHA-1 und SHA-2 ganz ohne zusätzlichen Zugriffe auf den Flash aus. Sie erkaufen sich diesen Vorteil aber durch hohen Speicherverbrauch, da die für die Berechnung notwendigen Tabellen im RAM befinden.

4.5 Bewertung

In Tabelle 4.5 sind die implementierten Verfahren unter den untersuchten Gesichtspunkten gegenübergestellt. Je nach Anwendungsszenario bzw. Anforderungen, sowie zur Verfügung stehender Ressourcen, kann hierdurch ein passender Kandidat ermittelt werden.

Verfahren	Taktzyklen	Speicher RAM	Codegröße	Flash
AES-128	+	o	o	o
AES-256	o	o	o	–
Camellia	o	+	– –	o
RC5-256	o	+	++	++
RC6-256	o	++	+	++
Serpent-256	–	– –	o	+
Skipjack	+	++	++	++
SHA-1	+	++	++	++
SHA-256	o	– –	+	++
BLAKE-32	+	+	o	o
BLAKE-64	–	–	– –	–

Tabelle 1: Bewertungstabelle, Wertung von sehr gut ++ bis sehr schlecht – –, o entspricht dem Mittel

Bei der Bewertung der Verfahren haben wir versucht die Stärken und Schwächen der einzelnen Verfahren zu beachten und anhand der Kriterien Taktzyklen, Speicherverbrauch, Codelänge und Flashspeicher Beanspruchung eine objektive Empfehlung zu geben. Bzgl. der Taktzyklen bzw. der Ausführungsgeschwindigkeit ist das Verfahren AES-128 bei den Blockchiffren und die Verfahren Blake-32 und SHA-1 bei den Hash-Verfahren als besonders empfehlenswert zu nennen. Beim Punkt Speicherverbrauch führt die Empfehlung hingen zu RC6 und SHA-1, bzgl. der Codegröße ist RC5, Skipjack und SHA-1 im Vorteil. Die Verfahren RC5, RC6, Skipjack und die SHA Varianten benötigen keinen externen Flashspeicher und sind bei knappen Speichervorkommen zu verwenden.

5 Zusammenfassung und Ausblick

Zusammenfassend lässt sich folgendes sagen: Eine Implementierung von kryptographischen Verfahren ist auch für MantisOS ohne Probleme möglich und die hier verwendeten Implementierungen sind dabei auch entsprechend leistungsfähig. Für die hier untersuchten, konkreten Implementierungen kann man folgende, auf 8-Bit Plattformen anwendbare, Empfehlungen aussprechen. Auf Höheren Architekturen mit z.B. 32-Bit Prozessoren können die Ergebnisse (wegen höhere Wortbreite, mehr Befehlen, mehr Optimierungsmöglichkeiten durch den Compiler) anders ausfallen. Die AES-128 Implementierung hat sich als sehr effizientes Blockchiffre Verfahren erwiesen, nur ist der Speicherverbrauch problematisch. In diesem Fall eignet sich das langsamere aber platzsparende RC6 Verfahren. Bei den Hashverfahren geht die Empfehlung an die BLAKE-Variante BLAKE-32. Die Berechnung ist etwas schneller als bei SHA-1 mit minimal höherem Speicherverbrauch, die anderen Verfahren sind deutlich langsamer. Zieht man zusätzlich noch in Betracht, dass SHA-1 nicht mehr offiziell empfohlen wird, gibt es im Fall der hier untersuchten Bibliothek keinen Grund mehr SHA-1 noch zu verwenden. In naher Zukunft werden wir die Bibliothek um weitere bekannte Kryptoverfahren erweitern und öffentlich zugänglich ma-

chen. Durch eine erste experimentelle Portierung von TinyECC wurde die Erweiterbarkeit der Bibliothek hinsichtlich der public-key Verfahren bereits gezeigt.

Literatur

[ABK] Ross Anderson, Eli Biham und Lars Knudsen. Serpent: A Proposal for the Advanced Encryption Standard. In *First Advanced Encryption Standard (AES) Conference*.

[Bel96] Steven M. Bellovin. Problem Areas for the IP Security Protocols. In *Proceedings of the Sixth Usenix Unix Security Symposium*, Seiten 205–214, 1996.

[BGW01] Nikita Borisov, Ian Goldberg und David Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. Seiten 180–189, 2001.

[BKN09] Alex Biryukov, Dmitry Khovratovich und Ivica Nikolić. Distinguisher and Related-Key Attack on the Full AES-256. In *CRYPTO '09: Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology*, Seiten 231–249, Berlin, Heidelberg, 2009. Springer-Verlag.

[BPVV99] Johan Borst, Bart Preneel, Joos Vandewalle und Joos V. Linear Cryptanalysis of RC5 and RC6. In *Proceedings of Fast Software Encryption, Lecture Notes in Computer Science*, Seiten 16–30. Springer-Verlag, 1999.

[dCHAC] Homepage der Cryptographic Hash Algorithm Competition:. http://csrc.nist.gov/groups/ST/hash/sha-3/.

[dNP] Homepage des NESSIE-Projektes:. https://www.cosic.esat.kuleuven.be/nessie/.

[FIP] FIPS-197. Spezifikation von AES.

[GHJ+01] Henri Gilbert, Helena Handschuh, Antoine Joux, Serge Vaudenay und Gemplus Card International. A Statistical Attack on RC6, 2001.

[JPLWP09] Aumasson J.-P., Henzen L., Meier W. und Raphael C.-W. Phan. SHA-3 proposal BLAKE, 2009.

[Kra01] Hugo Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, Seiten 310–331, London, UK, 2001. Springer-Verlag.

[RRSY98] Ronald L. Rivest, M. J. B. Robshaw, R. Sidney und Y. L. Yin. The RC6 Block Cipher. In *First Advanced Encryption Standard (AES) Conference*, Seite 16, 1998.

[TR-] BSI TR-02102. Kryptographische Verfahren: Empfehlungen und Schlüssellängen.

[TT05] With Precise Timing und Ben L. Titzer. Avrora: Scalable Sensor Network Simulation. In *Proc. of the 4th Intl. Conf. on Information Processing in Sensor Networks (IPSN*, Seiten 477–482, 2005.

[Web] Avrora Website:. http://compilers.cs.ucla.edu/avrora/.

[WS02] Anthony D. Wood und John A. Stankovic. Denial of Service in Sensor Networks. *Computer*, 35(10):54–62, 2002.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin und Hongbo Yu. Finding Collisions in the Full SHA-1. In *Proceedings of Crypto*, Seiten 17–36. Springer, 2005.

On Security Protocols for Desktop Sharing

Ulrich Kühn*
DZ BANK AG, Germany
Ulrich.Kuehn@dzbank.de
ukuehn@acm.org

Abstract: In this paper we examine security protocols employed in a number of tools for desktop sharing. These tools allow one user to see and interact with the desktop of another user, i.e. transmitting the contents of one computer's logical display to another place, including user interaction. In contrast to remote sessions, with desktop sharing, the access to the machine is shared, e.g. for interactive user support or for supporting administrators by experts for certain tasks or application programs. A number of these tools use an external communication server as a relay to sidestep problems when both the user and the support agent are behind firewalls.

In this paper we identify design flaws in the security protocols employed by a number of such tools, most notably a problem which allows the provider of the communication server to compromise the security of the communication. Further, we examine the certificates of security that some of these tools bear in the light of our findings. Additionally, we analyse the security requirements for a relayed communication protocol, which seems to be missing so far, and make high-level suggestions for an instantiation.

1 Introduction

With computers being prevalent on office worker's desktops, efficient support functionality is required. Here desktop sharing tools come into play, where a support agent can see and directly interact with what the user is being displayed on the screen. The advantage of using such tools is that the support agent can help a user who is located in a different location, as it is common for many of today's companies. A similar scenario for desktop sharing tools is remote administration with an outsourced administrator or specialised application expert. Here, the use of desktop sharing tools allows to implement a segregation-of-duty policy while maintaining support efficiency by avoiding travel.

Thus, the important difference between desktop sharing tools and remote sessions is that with desktop sharing, the session is actually shared, i.e. duplicated, so that at least two users can view the display contents and potentially interact with the system, while a remote session simply allows to log into a remote machine and interact with it. A kind of extension to desktop sharing is the web conferencing scenario, where more than one recipient participates. However, this group communication scenario is beyond the scope of this paper.

*Any opinions, findings, conclusions or recommendations expressed in this paper are those of the author and do not necessarily reflect the views of DZ BANK AG.

The communication protocols employed by desktop sharing or remote administration tools fall into two classes: First, protocols that need a direct connection between the accessed and the accessing computer which do not work with restrictive firewall settings, and second, communication protocols that employ relay servers as rendez-vous point and thus allow both communicating computers being behind firewalls that may block incoming traffic.

In this paper we are concerned with this second class of protocols that allow firewall piercing. We identify common design problems in these protocols and illustrate them by a case-study on actual products that contain these design problems. Further, as a number of these products come with a certificate of security, we examine the certificates in the light of our findings. Finally, we analyse the security requirements for such relayed communication protocols when it comes to desktop sharing. Then we suggest a high-level instantiation based on the identified security requirements.

Despite the considerable number of desktop sharing tools available we found that a treatment of the security of communication protocols which use a rendez-vous point for firewall piercing seems to be largely missing in the literature. Likewise, an analysis of the requirements regarding security of these protocols seems also to be missing in the literature.

2 Communication Models for Desktop Sharing: Direct vs. Relayed

The communication protocols employed by desktop sharing or remote administration tools can be distinguished into two classes, based on the way the communication is handled. In this paper we are concerned with the second class of relayed communication protocols.

Direct connection. The first class is comprised of protocols where the initiator needs to be able to directly open a network connection to the responder. Here the responder must be directly "visible" from the initiator's point of network view. Typically, the responder must not be behind a firewall that blocks incoming connections, at least for some ports. This scenario is rather standard.

Relayed communication. The second class of protocols consists of protocols which allow that both initiator and responder can be located behind restrictive firewalls. Here, the logical connection is made by resorting to some freely accessible communication intermediary. Both endpoints open *outgoing* connections, which are allowed in one way or another by most firewall setups. Here, the intermediary acts as a relay by passing the communication back and forth.

2.1 Entities Participating in Relayed Communication

Communication protocols that support relayed communication make use of intermediate communication servers, *relays*, which logically connect two connections with the communicating machines to form a logical connection. We denote the involved entities by

- The *initiator* is the entity which starts a communication. In the setup of interest here, the initiator connects to the relay, and lets it set up a connection. At this point in time the logical connection is prepared, but not fully existing.
- The *responder* is the other communicating entity. The responder also connects to the relay, but after the initiator.
- The *relay* actually implements the logical connection between initiator and responder by relaying data back and forth.

It should be noted that in fact there are three communication connections involved here: The connection between initiator and relay, the connection between responder and relay, and the logical connection between initiator and responder. These three connections need to be taken into account when designing and analysing security functions for relayed communication protocols.

3 Design Flaws in Desktop Sharing Protocols

Here we give a case study on some desktop sharing tools that employ relayed communication. For the security of the employed protocols it is of critical importance who owns and controls the relay. First, we analyse the security of the protocols when employing the vendor-provided relay, and second, when one of the communicating parties owns and controls the relay.

3.1 Common Flaws

Before we give details on the security protocols in our case study we want to highlight the problem areas that we identified:

Vendor-provided Relay as Man in the Middle. The protocols described and analysed below share the property of protecting the two connections between initiator and relay resp. responder and relay, but effectively not protecting the logical connection between initiator and responder. It should be noted that this man-in-the-middle issue is present despite the use of session tokens that are passed from the initiator to the responder both for identifying the correct session and for authenticating the responder. As a consequence, the security of the protocols does critically depend on who operates and controls the relay. For the first case considered here, i.e. the relay is controlled by neither of the communicating parties, this gives rise to the possibility of the relay acting as a man in the middle. Later we see that if one of the communicating parties owns and operates the relay, this problem is resolved, at the expense of losing the firewall-piercing property of the protocols for one of the communication legs.

Data Integrity. Another common issue, although not present in all the protocols analysed below, is the integrity of the transmitted data. It is well-known in the cryptographic community that encryption itself does *not* provide data integrity, and even a number of constructions with redundancy have been shown to fail [BN00, AB01].

3.2 Case Study on Protocols with Vendor-provided Relay

We want to highlight that here only a small selection of tools is described and analysed. In fact it seems difficult to obtain a complete list of such tools, just because the number of tools on the market. However, often no or only a very brief description of the security functions is given, e.g. the name and the key length of the symmetric cipher used for data encryption, but no further detail on key management, entity authentication etc.

Given that the tools we examine here are offered commercially, we did not have source code available. Instead we base our analysis on available descriptions and white papers, and, for some tools, personal communication with the manufacturer.

3.2.1 WebEx

The WebEx tool uses a relay that is implemented as a distribution network run by WebEx. In [Web05], section "Transport Layer Security", it is stated that the actual contents is encrypted using AES. However, it is unclear which entities have the key and if the contents is decrypted resp. re-encrypted by the relay. Further, the connections between initiator / responder and the relay can optionally be protected by SSL, apparently in addition the mentioned AES encryption. WebEx communicates using TCP port 1270, but allows to pass firewalls using ports 80 (HTTP) or 443 (HTTP over SSL).

While [Web05] does give some details about session authentication using passwords, no key management details are mentioned. As the session parameters are determined by the relay, it has to be assumed that all keys used for contents encryption are controlled by the relay as well.

Relay as Man in the Middle. WebEx states in [Web05] that it does not retain any session information. Thus, from the wording of the statement we assume that the relay does have access to contents in clear. Thus, effectively the logical connection between initiator and responder is not secured against the service provider WebEx itself. We did not find any statement indicating the contrary. Further, despite of the possibility of session passwords, no indication is given that the password is used as a shared secret between initiator and responder for precluding a man-in-the-middle attack.

Data Integrity. While SSL does have integrity protection built in, the termination of the secured channel at the relay precludes using it for end-to-end integrity protection. Further it is not clear if integrity protection is present if SSL is not used for content transmission.

3.2.2 Netviewer

The tool Netviewer comes as a stand-alone, installation-free program in two versions for initiator and responder. Its security functionality is described in a white-paper [Net08]. The security protocol for connection establishment works as follows: There are two pairs of asymmetric keys, one for the relay $(\mathrm{PK}_M, \mathrm{SK}_M)$, and one for the clients $(\mathrm{PK}_C, \mathrm{SK}_C)$, which are generated at installation time of the relay and are built into the client programs. The asymmetric encryption scheme involved here is an elliptic curve-based scheme with a claimed key length of 160 bits. However, the transmitted data offers space for at least 258 bits, and a deeper examination shows the use of $\mathrm{GF}(2^{255})$ as the underlying finite field[1]. The symmetric encryption employs Blowfish with 128 bit keys.

The connection setup protocol between initiator and relay is comprised of mutual authentication between relay and initiator based on the asymmetric key pairs. Then, the initiator sends a random mask value m to the server, encrypted under PK_M. The relay generates a symmetric session key k_I and sends $k_I \oplus m$, encrypted under PK_C, to the initiator. An analogous authenticated key exchange takes place between responder and relay. The resulting encrypted signalling channels are used to transmit another symmetric key k to the clients which they use for communication with each other. According to [Net08] the relay is comprised of a connection server, and a group of communication servers, with the key k never being transmitted to the communication servers. In order to cope with restrictive firewalls, the protocol can use port 80 (HTTP) or port 443 (HTTP over SSL). With the latter, an additional layer of protection between initiator / responder and relay is used.

Relay as Man in the Middle. The relay does generate the symmetric keys that are used to secure the communication, both on the signalling channels, and between initiator and responder. While [Net08] states that the key is not transmitted to the "communication server"-part of the relay, it is nevertheless in the position for a man-in-the-middle attack by retaining the key k, decrypting and modifying traffic between initiator and responder.

Data Integrity. While the white papers etc. do not provide any indication whether the integrity of the transmitted data is protected by cryptographic means, [Net09] indicates that HMAC-SHA1 [Nat02] is used. However, because of the relay generating the key and its position as man in the middle, this does not provide end-to-end integrity.

3.2.3 TeamViewer

The tool TeamViewer is comprised of stand-alone, installation-free programs for initiator and responder. Its security functionality is described in [Tea09]. It works roughly as follows:

[1] As the employed finite field has composite degree, the Gaudry-Hess-Smart attack and its extensions are relevant here. In fact, [MMT02] shows that the attack complexity is about 2^{52} operations, indicating a rather limited cryptographic security of the protocol. However, this problem is cured when tunnelling over SSL is used, see below. Interestingly, the documentation for the Common Criteria certificate (see Section 3.4) doesn't mention elliptic curve cryptography at all, and instead fully relies on SSL.

Both initiator and responder contain the relay's RSA public key PK_M, and subsequently generate their own RSA key pairs which they pass on to the relay for later use. On connection initiation the initiator requests the responder's public key from the relay. This exchange is encrypted with the respective public keys, and the response is signed by the relay. The initiator generates a symmetric key K, which it encrypts for the responder and sends it, signed with its private key, to the responder via the relay. The responder in turn obtains the initiator's public key from the relay, verifies the signature and decrypts the symmetric key K. After this protocol is run, both initiator and responder have a symmetric key which they use for further communication. The RSA keys are 1024 bits long, the symmetric encryption is done with AES-256.

Relay as Man in the Middle. We note that the relay acts as a trusted third party, certifying the initiator's and responder's public keys, which are, however, transmitted unauthenticated to the relay[2]. As a consequence the relay can act as a man in the middle, replacing the initiator's and responder's public key when answering the respective requests, and manipulate the transmission of the encrypted and signed symmetric key. Nevertheless, [Tea09] claims that not even the relay can decrypt the communication between initiator and responder. This claim is refuted by our analysis.

Data Integrity. Integrity protection of transmitted data is not mentioned in [Tea09]. Thus, it is not clear if integrity of the data is protected at all.

3.2.4 FastViewer

The FastViewer family of tools offers remote administration, desktop sharing and web conferencing. Here, we are here interested in the tool for desktop sharing and support. The FastViewer protocol distinguishes between two phases [Fas07], one for the rendez-vous of initiator and responder, and another one for the actual data communication. The rendezvous phase always uses the relay. The data communication phase can use direct connection or the relay, depending on network reachability.

Assuming that the data communication phase is using the relay, the second phase starts by a key exchange. While [Fas07] does state that a 256-bit AES key is exchanged between initiator and responder, it does *not* state how this is actually done. In [Fas09] some more protocol details were given: First, the initiator and responder both obtain a symmetric AES from the relay by generating and transmitting an RSA key to the relay, which sends back the AES key encrypted under the respective RSA key. These replies are signed by the relay and checked by the clients with a built-in verification key. Using the AES key, the resulting encrypted channel, another RSA key is transmitted from responder to initiator, which is in turn used to transfer the symmetric AES session key. Apparently, this AES session key is the one mentioned in [Fas07] with a length of 256 bits.

[2] It seems difficult to add here more than a self-signature of the respective client, if one does not want to require some form of registration or a client certificate. This would vastly complicate the use of the tool.

Relay as Man in the Middle. As the relay generates the first AES key, it can replace the second RSA key during transfer in order to obtain the session key. This is a classical man-in-the-middle attack. Given that apparently only a session number of 5 digits for authentication is exchanged using an out-of-bounds channel (e.g. telephone) between initiator and responder, it seems unlikely that the designers did consider the possibility of man-in-the-middle attacks.

Data Integrity. Integrity of transmitted data is not considered in [Fas07], neither was it mentioned in [Fas09]. Thus, we have to assume that no integrity protections are in place for the FastViewer tools.

3.3 Owning a Relay as a Partial Cure

Some of the tools examined above are offered with the option to deploy and use one's own relay, i.e. Netviewer, TeamViewer, and FastViewer.

In this case one of the communicating parties does have full control over the relay. This avoids having a third party, namely the operator of the relay, with control over the traffic. The fact that one of the communicating parties is controlling the relay heals the flaw identified in Section 3.2 where the operator of the relay as a third party can act as a man-in-the-middle due to the protocol design. However, at the same time the advantage of the relayed communication protocols is partly removed. Here, the party operating the relay must make sure that the relay can be reached by the other party from the outside world.

Nevertheless this operating model is well-suited for an enterprise scenario: The relay inside the enterprise network works as a central communication point. This allows to configure the firewall such that communication from outside is permitted to the relay, but there is no need to open the firewall to permit communication to all computers that potentially need remote support. Further, the initiator still fully controls if and when a desktop sharing session is possible.

3.4 Certification

Some of the tools are advertised with their security functionality being certified. Here we examine these certificates in the light of the findings presented above:

- The tool Netviewer (see Section 3.2.2) is advertised with a number of certificates: First, a certificate issued by Fiducia IT AG [FID06] on version 3.2 of Netviewer one2one presented as a "Security Certificate". However, it only certifies that the product is safe to use on a certain client setup for banking branches with no implication regarding security.

  Second, at the time of submission of this paper (March 2010), the tool was offered with a certificate issued by Fraunhofer Institut Sichere Informationstechnologie SIT

[Fra04] for version 2.0 of Netviewer one2one. Here, no evaluation criteria are given, and no special requirements are given for the user to fulfill. This certificate is listed as expired by the issuer [Fra]. However, the certificate does not identify the man-in-the-middle issue described above. This must be considered a flaw in the evaluation criteria or a failure in the evaluation process.[3]

Third, a common criteria certificate has been granted by Germany's Federal Office for Information Security (BSI) on version 5.1 of Netviewer one2one [Bun10] with an assurance level of EAL2 and a specific security target [Net09]. The security target lists the requirement that the relay is installed in a trustworthy environment. This fits well to our analysis in section 3.2.2. Nevertheless, the assurance level of EAL2 is rather low[4]. Nevertheless, the manufacturer's web site [Net10] contains the claim "experience shows that Netviewer is extremely secure". Interestingly, "experience" is cited, not the certificate.

- The tool FastViewer (see Section 3.2.4) is advertised with a certificate issued by TÜV Süd (see [Fas]). The certificate lists ISO/IEC25051:2009 for functionality and PPP13011:2004 for data security as certification criteria. The former is a public document containing requirements for software quality for commercial off-the-shelf software, but the latter not publicly available, not even upon request [Wol09]. Likewise, no evaluation report is available. In fact, the man-in-the-middle issue identified in Section 3.2.4 is not mentioned at all.

The certificates examined here, except the Common Criteria certificate for Netviewer, share the common problem that neither the certification criteria nor the certification report are not available for the customer. Thus, from a customer's perspective, these certificates do *not* provide enough information under which conditions these tools provide security against which attacks or adversaries. Consequently, these certificates are not helpful to understand the security the tools provide, and might lead to a false sense of security.

The exception here is the Common Criteria certificate with all the necessary information available. However, the details of security target and the assurance given by the EAL2 certification have to be evaluated carefully if they are suitable for the intended usage scenario.

4 Designing a Relayed Security Protocol for Desktop Sharing

Here we state the basic security requirements for relayed communication protocols and give a high-level description of building blocks for instantiation.

[3] Another certificate by the same issuer on another desktop sharing tool (not included in the case study above for lack of details on the security protocol) state as condition "provided the [...] server component is located in a protected and trustworthy environment" [Fra06]. The inclusion of this condition hints to an update in the (unpublished) evaluation criteria.

[4] [Bun10] explains that EAL2 is "applicable [...] where [...] users require a low to moderate level of independently assured security in the absence of ready availability of the complete development record. Such a situation may arise when securing legacy systems [...]"

4.1 Requirement Analysis and Protocol Flow Proposal

As indicated in Section 3.1 with relayed communication there are not only two but effectively three connections involved which must be secured: First, Initiator ↔ Relay, second, Responder ↔ Relay, and last, Initiator ↔ Responder via the relay. In fact there is another – out-of-band – channel between initiator and responder over which the responder obtains information on how to make the relay connect the responder's to the initiator's session. In a remote support scenario this out-of-band channel would be provided by telephone.

The general setup protocol would flow as follows, where the first seven steps are in place to connect the initiator and responder via the relay, and the last step addresses the man-in-the-middle issue by securing the logical connection:

1. The initiator connects to the relay, which sets up an (unconnected) session s.
2. The relay provides some session token t_s to the initiator.
3. The initiator passes t_s to the responder using an out-of-band channel.
4. The responder connects to the relay which sets up an (unconnected) session s'.
5. The responder forwards t_s to the relay.
6. The relay uses t_s to identify the session s and logically connects sessions s and s'.
7. The relay signals both the initiator and responder that the sessions has been successfully connected.
8. Initiator and responder use the logical channel to set up the session between them.

Note that there is no need for a session token for s' if the responder identifies itself as a responder. In this protocol framework four steps, namely steps 2, 5, 6, and 8, give rise to security requirements:

R1. As t_s is an authenticator for connecting to the session s it must be delivered in a confidential way, first to the initiator in step 2, then to the responder using the out-of-band channel, and lastly to the relay in step 5.

R2. t_s must be constructed such that the relay can reliably determine its authenticity, its freshness and its connection to the session s in step 6.

R3. For transmitting out of band in step 3, the token t_s must be short and easy to communicate by humans.

R4. The logical connection between initiator and responder must be secure, providing both confidentiality and integrity / authenticity of transmitted data. To preclude man-in-the-middle attacks the keys and authentication information must not be solely based on information exchanged with or with the help of the relay.

R5. Depending on the business model of the entity providing the relay the initiator resp. the responder optionally must first successfully authenticate to the relay in step 1 resp. 4 to show that he or she is entitled to used the relay's services.

4.2 High-level Description of Protocol Instantiation

Here we provide a high-level sketch of how the protocol framework could be instantiated and how the security requirements could be fulfilled. For a secure instantiation of the framework protocol there are three building-blocks to be specified:

Securing the connections between initiator and relay resp. responder and relay. We propose using SSL/TLS with a server certificate for the relay. This provides secrecy and integrity of the transmitted data as well as authenticity of the relay for this control connection. As it is a point-to-point connection, the SSL/TLS standard is appropriate. Optionally, the initiator could authenticate against the relay by using user-ID / password or a certificate. This way a business model with pay-per-use could be set up for the relay. This choice addresses security requirements R1 and R5.

Designing a secure scheme for the session token. Given the requirements for the session token, we propose the following construction, based on the assumption that about 40 bits of information can be passed by human communication when properly encoded.

When setting up a session s with an initiator, the relay generates state information $I_s = (N, T, K)$ consisting of a session number N, a time T of establishing I_s, and a MAC key K. K is generated from a cryptographically secure random bit generator. The session number N is 20 bits long, allowing 2^{20} parallel open sessions, and is either generated randomly (excluding collisions with existing session numbers) or from a counter. The session number can theoretically be reused after the full connection is made.

The session token $t_s = (N, \tau)$ is computed by $\tau = [(\mathrm{MAC}_K(N, T)]_{20}$ where $[.]_x$ means truncation to x bits, $I_s = (N, T, K)$, and MAC is a secure MAC scheme, such as CBC-MAC [ISO99] based on AES or HMAC [Nat02]. We propose to use alpha-numerical characters for encoding t_s, i.e. 5 bits per character, yielding a string of 8 characters. Such a string can be easily transmitted by telephone.

Verifying a session token $t_s = (N, \tau)$ during step 5 of the protocol framework requires the relay to find session information $I' = (N', T', K')$ such that $N' = N$ and verifying that $\tau = [(\mathrm{MAC}_{K'}(N', T')]_{20}$. Further, it checks that T is not older than a short time-out. Here, the probability of success of an adversary is roughly $1/2^{-20}$ if it successfully identifies an open session. In our view this provides sufficient protection, given that the actual authentication between responder and initiator is done in step 8. Actually, an adversary passing the token verification can be seen as a residual risk of the adversary using the relay. Possibly the authenticator τ could be truncated even more. This construction addresses security requirements R2 and R3.

Securing the logical connection between initiator and responder. Any authenticated key exchange method could be used here. However, we propose to employ the out-of-band channel between initiator and responder also for this task, making sure that the entity whose machine needs to be accessed receives the authentication information:

First a key exchange, e.g. Diffie-Hellman, would run between the two entities. Second, an authenticator string would be computed using the computed shared secret from the entities public keys or all messages of the key exchange. The authenticator would then be transmitted by the accessing entity to the accessed entity using the out-of-band channel. Finally, access would be granted after successfull verification by the accessed entity. This provides a corroboration to the accessed entity that the right entity is on the other end of the connection, thereby addressing security requirement R4.

5 Conclusion

In this paper we have first analysed the security of desktop sharing and remote support tools that employ relayed communication. Such communication patterns allows parties to communicate even if one or both of them are behind firewalls.

It turned out that with this kind of indirect communication new security issues arise that are not addressed by usual secure communication protocols like SSL/TLS. We have examined a selection of tools for desktop sharing and remote support, where we identified common security issues. Furthermore, we found that in an enterprise environment, placing and controlling the relay inside the enterprise removes this fundamental issue and, additionally, allows a better-controlled configuration of firewalls.

In the second part of this paper we did an analysis of the security requirements for relayed communication and proposed a framework and possible high-level instantiations of the security building blocks.

To summarise, our analysis shows that designing a secure relayed communication protocol is not a trivial task and that trusting the relay without securing the logical connection between the communicating entities can allow the relay to act as a man in the middle, thus threatening the privacy and integrity of the communication.

Future research could be directed towards the related scenario of web conferencing, where more than one responder is present. Here relayed communication has to be combined with secure group communication protocols.

References

[AB01] Jee Hea An and Mihir Bellare. Does Encryption with Redundancy Provide Authenticity? In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT '2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 509–524, Innsbruck, Austria, 2001. Springer-Verlag, Berlin Germany.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT '2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545, Kyoto, Japan, 2000. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany.

[Bun10] Bundesamt für Sicherheit in der Informationstechnik (BSI). Zertifizierungsreport BSI-DSZ-CC-0524-2010, March 2010. https://www.bsi.bund.de/cae/servlet/contentblob/950786/publicationFile/61801/0524a_pdf.pdf.

[Fas] FastViewer. TÜV Süd Certificate (2009). http://www.fastviewer.com/awards.html.

[Fas07] FastViewer. Erklärung zum Verschlüsselungsverfahren und zur Datensicherheit beim Einsatz von FastViewer, October 2007. http://www.additive-net.de/ftp/win32/software/fastviewer/ADD_ES_FSV_Verbindungsaufbau_und_Sicherheit.pdf.

[Fas09] FastViewer. Personal Communication, 2009.

[FID06] FIDUCIA IT AG. Bestätigung Sicherheitstechnische Prüfung von Fremdsoftware – Netviewer one2one und one2meet, Juni 2006. http://www.genodata.de/support/download/zertifikate/Fiducia%20IT%20AG_Zertifikat%20one2one%20one2meet.pdf.

[Fra] Fraunhofer Institut Sichere Informationstechnologie. List of Certificates. http://testlab.sit.fraunhofer.de/content/output/certificates.php.

[Fra04] Ergebnis der Prüfung der Netviewer Software durch das Fraunhofer Institut für Sichere Telekooperation, April 2004. http://testlab.sit.fraunhofer.de/downloads/certificates/netviewer%20200404.pdf.

[Fra06] Certificate on pcvisit 4 certified security version 4.1.0.1476, July 2006. http://testlab.sit.fraunhofer.de/downloads/certificates/Urkunde%20pcvisit%20en%2006-104701.pdf.

[ISO99] ISO/IEC. IS 9797-1: Information Technology – Security techniques – Message authentication codes (MACs) – Part 1: Mechanisms using a block cipher, 1999.

[MMT02] Markus Maurer, Alfred Menezes, and Edlyn Teske. Analysis of the GHS Weil Descent Attack on the ECDLP over Characteristic Two Finite Fields of Composite Degree. *LMS J. Comput. Math*, 5:127–174, 2002.

[Nat02] National Institute of Standards and Technology (NIST). The Keyed-Hash Message Authentication Code. Federal Information Processing Standards Publication (FIPS PUB) 198, March 2002.

[Net08] Netviewer. White Paper Security: Netviewer Support, Version 1.6, September 2008. http://www.netviewer.de/fileadmin/PDF/whitepaper/Netviewer_Whitepaper_security_Netviewer_Support_EN.pdf.

[Net09] Netviewer. Sicherheitsvorgaben (Security Target) zu Netviewer one2one Version 5.1. https://www.bsi.bund.de/cae/servlet/contentblob/950784/publicationFile/61802/0524b_pdf.pdf, September 2009. BSI-Zertifizierungs-ID BSI-DSZ-CC-0524.

[Net10] Netviewer. Web site, May 2010. http://www.netviewer.com.

[Tea09] Teamviewer GmbH. TeamViewer Security Information, May 2009.

[Web05] WebEx Communications Inc. WebEx Security Overview, February 2005. http://www.webex.com/pdf/wp_security_overview.pdf.

[Wol09] Wolf-Rüdiger Heidemann (TÜV Süd). Personal Communication, July 2009.

Extended Lattice Reduction Experiments using the BKZ Algorithm

Michael Schneider Johannes Buchmann

Technische Universität Darmstadt
{mischnei,buchmann}@cdc.informatik.tu-darmstadt.de

Abstract: We present experimental results using lattice reduction algorithms. We choose the BKZ algorithm, that is the algorithm considered the strongest one in this area in practice. It is an important task to analyze the practical behaviour of lattice reduction algorithms, as the theoretical predictions are far from being practical. Our work helps choosing the right parameters for lattice reduction in practice. The experiments in this paper go beyond the results of Gama and Nguyen in their Eurocrypt 2008 paper. We give evidence of some facts stated in their work, concerning the runtime and the output quality of lattice reduction algorithms.

Keywords: Lattice Reduction, LLL, BKZ, Hermite-SVP

1 Introduction

Lattices have been known in number theory since the eighteenth century. They already appear when Lagrange, Gauss, and Hermite study quadratic forms. Former applications of lattice reduction are discrete optimization and integer programming, as well as factoring polynomials. Nowadays, lattices and hard problems in lattices are widely used in cryptography as the basis of promising cryptosystems.

Lattice reduction is also a useful tool in cryptanalysis. Various cryptosystems are broken using lattice reduction, e.g. knapsack systems [LO85, CJL+92] as well as RSA in special settings [May10]. Further on, factoring composite numbers and computing discrete logarithms is possible using lattice reduction [Sch91, May10]. The security of lattice based cryptosystems is based on the hardness of lattice problems that are solved using lattice reduction. Therefore it is necessary to analyze the strength of lattice reduction algorithms.

A lattice is an additive subgroup of the Euclidean vector space $\mathbb{R}^n$, generated by a lattice basis. Its elements can be considered to be vectors in a vector space. Lattice reduction is basically the search for vectors that are short and nearly orthogonal. There are two basic notions of lattice reduction, namely *LLL-reduction* and *BKZ-reduction*. LLL-reduction is a special case of the stronger BKZ-reduction.

The most famous algorithm for lattice reduction is the LLL algorithm by Lenstra, Lenstra, and Lovász [LLL82], which outputs an LLL-reduced basis. Theoretically, the best algorithm to find short vectors is the *slide reduction* algorithm [GN08a]. In practice, the most

promising algorithm is the BKZ algorithm by Schnorr and Euchner [SE91], that outputs a BKZ-reduced basis. A practical comparison of lattice reduction algorithms can be found in [NS06, GN08b, BLR08].

One major problem with lattice reduction algorithms is the fact that in practice, the algorithms like LLL and BKZ behave better than the theoretical worst-case analysis predicts. Therefore it is necessary to examine their practical, average-case behaviour. In 2008, Gama and Nguyen published a comprehensive summary of their experiments [GN08b], that helps analyzing the practical behaviour of the LLL and BKZ algorithm as well as the deep insertion variant of LLL. The results of this work are used widely when facts about the strength of lattice reductions algorithm is required, it became kind of a standard work.

Our Contribution. In this paper we present experimental results using lattice reduction algorithms that we collected during the last years. The focus of our work is on the BKZ algorithm, which is the algorithm most widely used in practice. We analyze the runtime of BKZ using high blocksizes, give details about the output quality of BKZ-reduced bases, and based on our observations present a strategy for lattice reduction in high lattice dimensions. With our work, we present heuristics that give further evidence to some arguments stated unproven by Gama and Nguyen in [GN08b]. Therefore, we extend the state-of-the-art in practical lattice reduction.

Organization of the Paper. Firstly, we present the necessary facts on lattices and lattice reduction in Section 2. Secondly, we present three main questions of this paper and answer them successively in Section 3. Finally, we give a conclusion and present open questions in the area in Section 4.

2 Preliminaries

A lattice L is an additive subgroup of the euclidean space $\mathbb{R}^d$. Let $n, d \in \mathbb{N}$, $n \leq d$, and let $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^d$ be linearly independent vectors. Then $L(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$ is the lattice spanned by the column matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$. $L(\mathbf{B})$ has dimension n, the matrix $\mathbf{B}$ is called a basis of the lattice. Such a basis is not unique, lattices are invariant under unimodular transformations. Therefore, every lattice in dimension $n > 1$ possesses infinitely many bases. We write L instead of $L(\mathbf{B})$ if it is clear which basis is concerned. The first successive minimum $\lambda_1(L)$ is the length of a shortest vector of a lattice. The lattice determinant $\det(L(\mathbf{B}))$ is defined as $\sqrt{\det(\mathbf{B}\mathbf{B}^t)}$. It is invariant under basis changes. For full-dimensional lattices ($n = d$) there is $\det(L(\mathbf{B})) = |\det(\mathbf{B})|$ for every basis $\mathbf{B}$. Throughout this paper, vectors and matrices are written in bold face, e.g., $\mathbf{x}$ and $\mathbf{M}$. We write $\|\mathbf{x}\|$ for the Euclidean norm of $\mathbf{x}$.

The lattices that arise in cryptography have a more special structure. Let $q \in \mathbb{N}$, $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. We define an m-dimensional lattice $\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m : \mathbf{A}\,\mathbf{v} \equiv \mathbf{0} \pmod q\}$. Those lattices are called *modular* or *q-ary* lattices. For efficiency reason, so-called *ideal lattices* are often used in cryptography. An ideal lattice of dimension m is the set of all

points that belong to an ideal in the ring $R = \mathbb{Z}[x]/\langle f \rangle$, for a polynomial f of degree n. For special f, it allows to exhibit the lattice from only m representatives in $\mathbb{Z}_q$, instead of mn in the q-ary lattice case, where m typically is in $\mathcal{O}(n \log n)$.

As mentioned above, each lattice has infinitely many bases. Creating a basis consisting of short and nearly orthogonal vectors is the goal of lattice reduction.

Hard lattice problems. There are several problems on lattices that are supposed to be or proven to be hard [MG02]. The most famous problem is the shortest vector problem (SVP). The goal of γ-SVP is to find an (approximate) shortest non-zero vector in the lattice, namely a vector $\mathbf{v} \in L \setminus \{\mathbf{0}\}$ with $\|\mathbf{v}\| \leq \gamma \lambda_1(L)$, where $\gamma \geq 1$ is the approximation factor. It is possible to formulate the problem in every norm, the most usual norm is the euclidean norm, that we are using throughout this paper.

As the length of the shortest vector $\lambda_1(L)$ might not be known, it is hard to control the approximation factor of SVP in practice. Therefore it is common practice to use the Hermite-SVP variant: given a $\gamma \geq 1$, find a non-zero vector $\mathbf{v} \in L \setminus \{\mathbf{0}\}$ with $\|\mathbf{v}\| \leq \gamma \cdot (\det L)^{1/n}$. Recall that the determinant of a lattice is always known, as it can be computed from every basis. Having reduced a basis $\mathbf{B}$ one can easily calculate the reached Hermite factor using $\gamma_{\text{Hermite}} = \|\mathbf{b}_{\text{min}}\| / (\det L)^{1/n}$. The main computational problem in q-ary lattices $\Lambda_q^{\perp}(\mathbf{A})$ is the short integer solution problem (SIS): given $n, m, q, \mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and a norm bound ν, find $\mathbf{v} \in \Lambda_q^{\perp}(\mathbf{A})$ with $\|\mathbf{v}\|_2 \leq \nu$. The SIS was first introduced and analyzed by Ajtai [Ajt96]. Numerous improvements to the analysis where made in, e.g., [MR07, GPV08]. For information about further lattice problems we refer the reader to [MG02] and [MR08].

Algorithms. The γ-SVP was solved by Lenstra, Lenstra, and Lovász in [LLL82] for factors γ exponential in the lattice dimension n. Their LLL algorithm has runtime polynomial in the lattice dimension. It is parameterized by a δ with $\frac{1}{4} < \delta \leq 1$, and outputs a basis whose first vector has length $\|\mathbf{b}_1\| \leq (\delta - \frac{1}{4})^{(1-n)/4} \cdot \det(L)^{1/n}$ [LLL82]. In other words, LLL provably reaches a Hermite factor of $(\delta - \frac{1}{4})^{(1-n)/4}$. An overview about applications and facts about LLL can be found in [NV10]. A typical choice for δ in practice is $\delta = 0.99$.

In [SE91] the authors introduce the BKZ algorithm, that is a blockwise variant of the LLL algorithm. BKZ is today's best algorithm for lattice reduction in practice. A BKZ instance using blocksize parameter β is called BKZ-β. It finds a lattice vector with length $\|\mathbf{b}_1\| \leq (\gamma_\beta)^{\frac{n-1}{\beta-1}} \cdot \lambda_1$, where γ_β is the Hermite constant in dimension β [Sch94]. The runtime cannot be proven to be polynomial in the lattice dimension, but practical experiments point out that the runtime of BKZ is polynomial in the lattice dimension. In fact, the runtime is exponential in the blocksize parameter β. This parameter allows a trade-off between runtime and output quality. Bigger blocksize causes shorter vectors at the expense of increased runtime, and vice versa.

Theoretically, the most promising algorithm for approximating shortest vectors is the algorithm of [GN08a]. In [SE91] the authors also proposed, besides BKZ, the deep insertion variant of LLL.

There are various algorithms that find a shortest lattice vector, not only an approximation. The algorithms of Kannan [Kan83], Fincke and Pohst [FP83], and Schnorr and Euchner [SE91] perform an exhaustive search for the shortest lattice vector. The algorithm used in practice today is the variant of Schnorr and Euchner, called ENUM. A second approach for solving the exact SVP are probabilistic sieving algorithms like the one of [AKS01], analyzed in [NV08], or the improved algorithm of [MV10b] and [PS10b]. The most practical sieving variant is presented in [MV10b], called Gauss-Sieve. In [MV10a] a deterministic, single exponential time algorithm based on Voronoi cell computations is presented.

Concerning implementations, there are two libraries available for lattice reduction: the NTL library of Shoup [Sho] and the fpLLL library of Stehlé et al. [CPS]. The fpLLL library does not offer BKZ. Therefore, for our experiments we use the NTL library, as was done in [NS06, GN08b, BLR08]. An improved LLL algorithm was presented as the L^2 algorithm by Nguyen and Stehlé [NS05]. It is implemented in the fpLLL library.

Practical behaviour. In practice however, lattice reduction algorithms behave much better than expected from theory. In the average case they find much shorter vectors than theoretical worst case bounds suggest. In [GN08b] Gama and Nguyen give a practical analysis of random lattices using the NTL library [Sho]. The authors state that a Hermite factor of 1.01^n and an approximation factor of 1.02^n in high lattice dimension (e.g. dimension 500) is within reach today, but a Hermite factor of 1.005^n in dimension around 500 is totally out of reach. The authors of [NS06] state similar facts for the LLL algorithm. In [BLR08] and [RS10], the runtime of different lattice reduction algorithms on q-ary lattices is analyzed. Those are the lattices that arise in lattice based cryptography, and their practical behaviour is not well studied to date.

3 Experiments

This section shows the experiments that were performed during the last two years. We provide answers to the following three questions:

- Which Hermite factor is reachable with a given blocksize?
- Is it possible in practice to run BKZ with big blocksizes?
- Is running BKZ with increasing blocksize a good strategy for lattice reduction?

All experiments are performed on random, full-rank lattices in the sense of Goldstein and Mayer [GM03], with bit size of the entries in the order of magnitude $10n$. For LLL-reduction we used a parameter $\delta = 0.99$ throughout all our experiments. We use the NTL library [Sho] in version 5.4.2. The experiments were performed on an AMD Opteron (2.3GHz) quad core processor, using one single core for the computation. For BKZ, we use the quad precision (QP1) variant of NTL.

3.1 Which Hermite Factor is Reachable With a Given Blocksize

It is well known that the blocksize parameter β allows a trade-off between runtime and reduction quality. Here we analyze the quality, in the shape of the Hermite factor. We provide a prediction for the Hermite factor that is reachable with a given blocksize β. This directly predicts the length of the shortest vector after BKZ-reduction.

For each dimension $n \in \{25, 50, \ldots, 250\}$ we generated 5 different, randomized bases for each of 5 random lattices. The entries of the input bases where bounded by 2^{10n}. These bases were reduced using BKZ with each blocksize $\beta \in \{5, 6, \ldots, 23\}$, independently. The reduced lattices were then used to calculate the Hermite factor reached in each case, i.e.,

$$\gamma_{\text{Hermite}} = \sqrt[n]{\|\mathbf{b}_1\| \cdot \det(L)^{1/n}}\,.$$

The mean values of all 25 bases in each dimension are depicted in Figure 1. We only show every fourth value of β, for clarity reason.

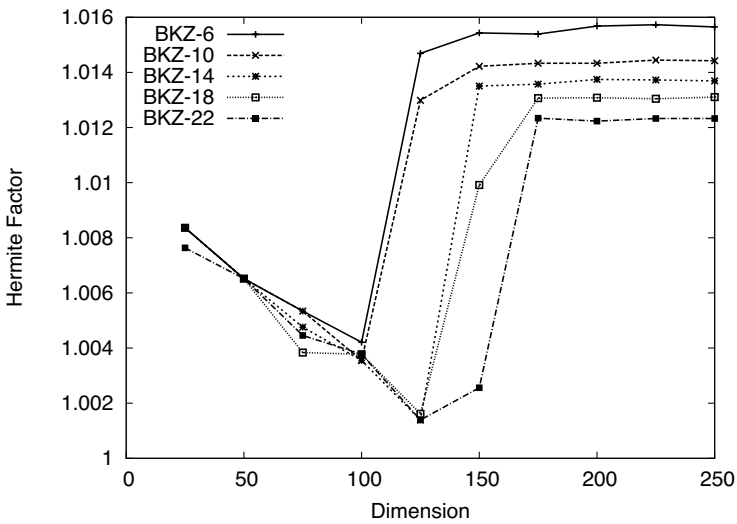


Figure 1: Hermite factor in dimension 25 to 250. The figure shows the experimental results, i.e., the average Hermite factors γ_{Hermite} reached by BKZ with different blocksize β.

One notices that in low dimensions, say less than $n = 175$, BKZ reaches very small Hermite factors. In higher dimension of $n \geq 200$, the Hermite factor for each blocksize stabilizes at a certain constant value (as was already stated in [GN08b]). In practice one is interested in higher dimensions $n \geq 200$, We extract these constants and, using least-squares fitting, we gain a fitting function (cf. Figure 2). For a given blocksize β, this function predicts the Hermite factor that can be reached with BKZ using the specified blocksize, in higher lattice dimensions. The resulting function is $f(\beta) = 1.01655 - 0.000196185 \cdot \beta$. Proposition 1 summarizes the result.

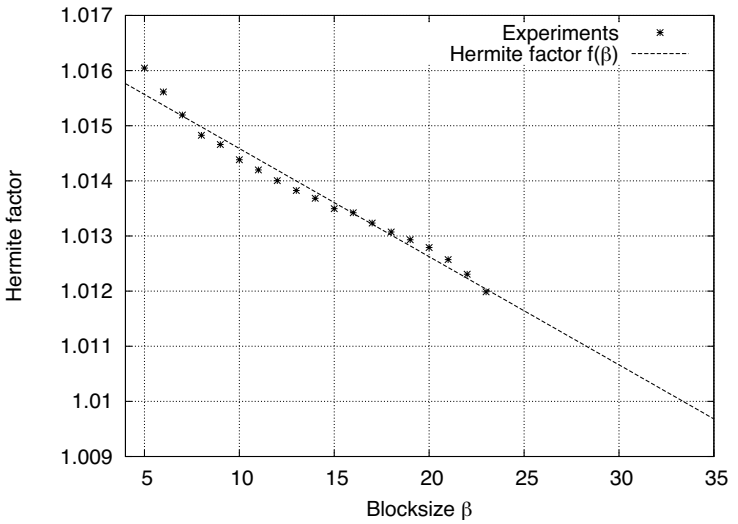


Figure 2: Mean values of the reached Hermite factors γ_{Hermite} and the fitting function $f(\beta)$. For high lattice dimensions (say $n \geq 200$), the function $f(\beta)$ predicts the Hermite factor that is reachable with a chosen blocksize β.

Proposition 1 *When BKZ is applied to a random, full-rank lattice L of dimension n using blocksize β, the output length of the shortest vector after BKZ reduction is*

$$\|\mathbf{b}_1\| \approx f(\beta)^n \cdot \det(L)^{1/n} \quad where \quad f(\beta) = 1.01655 - 0.000196185 \cdot \beta\,.$$

This analysis implies that, for example, for reaching a hermite factor of 1.005, one has to run BKZ with blocksize $\beta = 59$. Running BKZ with blocksize 50 it is possible to reach a Hermite factor $\gamma = 1.0067$. Our result is similar to that of [GN08b]. For $\beta = 20$ we compute $\gamma = 1.0126$ compared to 1.0128 ([GN08b]) and for $\beta = 28$ we get $\gamma = 1.0111$ compared to 1.0109 ([GN08b]). Therefore we extend the analysis of Gama and Nguyen when presenting our function for extrapolation of the possible Hermite factor for a given block size parameter β.

This result is very useful when attacking lattice based cryptosystems. Assume that our random lattices represent the average case of lattice reduction for ideal and q-ary lattices, that arise in lattice based cryptography. The security of most signature and encryption schemes is guaranteed as long as the SIS problem is hard for q-ary or ideal lattices, respectively [MR08, RS10]. As the lattice determinant is always known, it is easy to map a Hermite-SVP solution to a SIS solution. More precisely, when a vector $\mathbf{v} \in \Lambda_q^{\perp}$ with $\|\mathbf{v}\| \leq \nu$ is required, this corresponds to a Hermite factor $\gamma = \sqrt[n]{\nu/(\det \Lambda_q^{\perp})^{1/n}}$ needed to render a crypto system insecure. Using our analysis, we can predict the blocksize parameter required for this. The prediction of the runtime of the chosen blocksize is the concern of our second question.

3.2 Runtime of BKZ Using Bigger Blocksizes

In [GN08b], the authors state that BKZ reduction is only practical in blocksizes β up to 30 (they present results with blocksizes up to 40). With higher blocksizes, the runtime of BKZ rises too fast to be practical, at least in high lattice dimensions. In this section, we give evidence to this argument using runtime experiments with BKZ in high blocksizes.

We run BKZ with blocksize $\beta = 40$, 45, and 50 on five random lattices of each dimension $n \in \{30, 35, \ldots, 90\}$. As stated before, higher lattice dimensions are intractable. Again, the bit size of the entries of the input bases are in the order of magnitude $10n$. The logarithmic plot of Figure 3 shows the results, including least squares fitting lines $t_\beta(x)$ that help guessing the runtime of BKZ-β in bigger lattice dimensions.

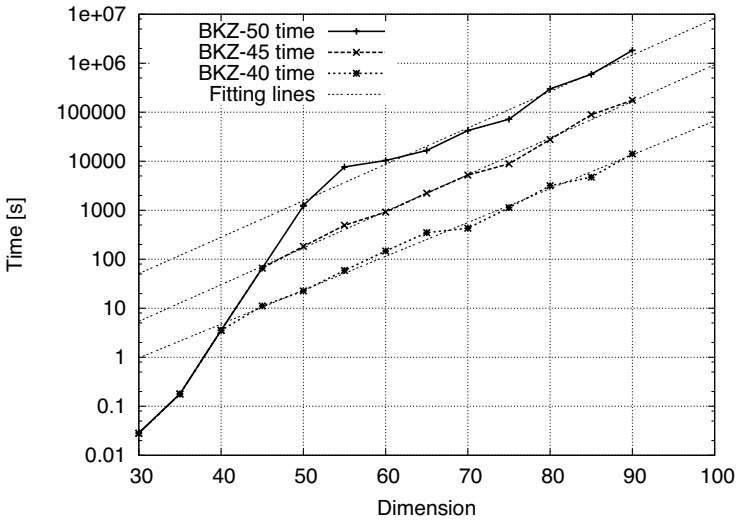


Figure 3: Runtime of BKZ-β on random lattices, for $\beta \in \{40, 45, 50\}$. The fitting lines are the linear functions $t_\beta(x)$ that can be used to predict the runtime of BKZ-β in higher dimension > 90.

For lattice dimensions $n \leq \beta$, BKZ-β behaves like BKZ-n. This can be observed in Figure 3, where the lines of all three blocksizes are the same in small dimensions. Therefore, lower dimension $n \leq \beta$ are excluded when computing the fitting functions. The resulting functions that we compute from our experiments are

$$t_{40}(x) = 10^{0.0690798 \cdot x - 2.08621} \text{ seconds},$$
$$t_{45}(x) = 10^{0.0748307 \cdot x - 1.51248} \text{ seconds},$$
$$t_{50}(x) = 10^{0.0743183 \cdot x - 0.523057} \text{ seconds}.$$

Using this we can for example predict the runtime (on state-of-the-art hardware comparable to ours) of NTL's BKZ-50 for a 108-dimension lattice to be around 1 year and for a 150-dimensional lattice around 1300 years. This concretizes the statements of Gama and Nguyen that BKZ in higher blocksizes is intractable in high lattice dimensions.

In addition, our experiments show that, when running BKZ with blocksize $\beta \geq 40$, the enumeration part takes more than 99% of the reduction time. In low blocksizes, the enumeration usually takes less then 10% of the reduction time, stated for example in [SE91, GN08b]. This is an important fact, since, during the last years, there are numerous improvements to enumeration algorithms, see [HSB+10, GNR10, DS10, PS10a], for example. These improvements will speed up BKZ in high blocksizes best, BKZ with low blocksizes will be enhanced only slightly.

3.3 Running BKZ with Increasing Block Size

It is common practice to run BKZ with increasing blocksize, that means starting with low blocksize (say, $\beta \approx 3$), fully BKZ-β reduce the basis, and increase the blocksize, until a vector of desired length is found. The question that we want to answer in this section is, if this approach is faster than directly starting with high blocksize, and if the reduction quality is concerned by the chosen strategy. Is it faster to run BKZ in increasing block size, using the pre-reduction of lower blocksizes to speed up the reduction in higher blocksize?

Again, we run BKZ on $5 \cdot 5$ random bases of dimension $n \in \{200, 300\}$, once with increasing blocksize with β from 3 up to 15, once directly running BKZ-15. We measure the total reduction time and the Hermite factor reached by the shortest output vector. The results are presented in Figure 4. The graphs show average, maximum, and minimum values of the runtime and the Hermite factor, respectively.

One notices that, concerning the runtime, starting BKZ directly with high blocksize β results in lower runtime in total. On average, the reduction finishes faster than in the increasing blocksize case. Concerning output quality, we also notice a minor advantage for the direct BKZ-15 case. This behaviour is surprising, since in both cases, the output basis is BKZ-15 reduced, and with increasing blocksize, more time was spend reducing the basis.

However, in practice it was unclear which blocksize will be necessary (and sufficient) to find a vector of suitable size. Therefore it was good practice to run BKZ with increasing blocksize, as a short vector might be reached much faster than when starting BKZ with high blocksize. With this technique, Gama and Nguyen could find the secret key of NTRU-107 lattices. The new strategy that one might apply is to use Proposition 1 to predict the blocksize needed for a desired Hermite factor. Combining the results of both sections in this paper leads to a new strategy, that will outperform the strategy with increasing blocksize.

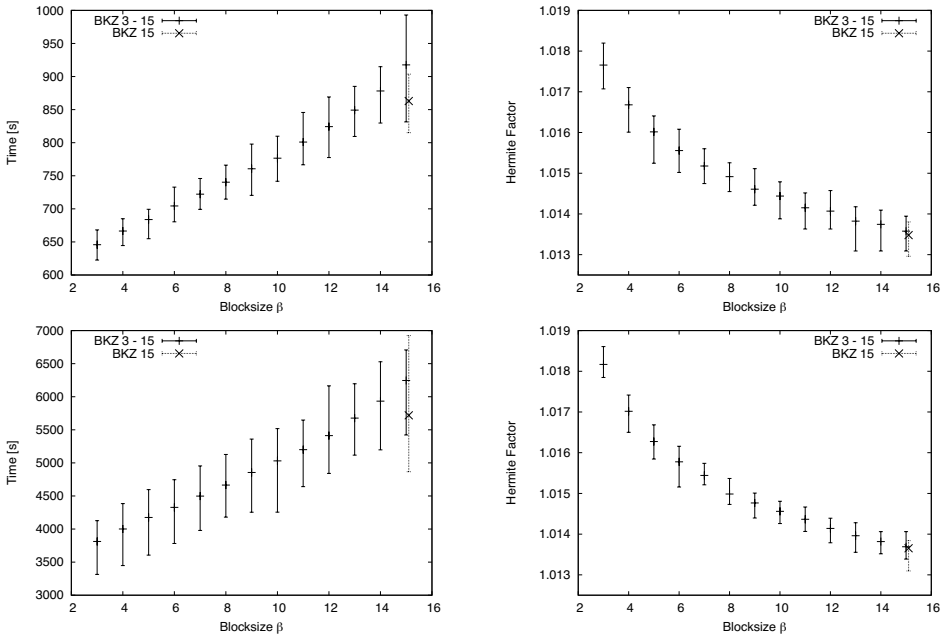


Figure 4: Running BKZ on random lattices with increasing block size β. The left pictures show the runtime, the right figures present the Hermite factor γ, both in dimensions 200 (top) and 300 (bottom). The solid data points were gained with increasing block size, the results depicted with dotted lines were started directly with blocksize $\beta = 15$. The graphs show average, maximum, and minimum values.

4 Conclusion

In this paper we have presented a prediction of the Hermite factor reachable with a given blocksize. We have shown how the choice of big blocksize effects the BKZ algorithm. These two results yield a new strategy for BKZ reduction, that replaces the strategy of increasing blocksize.

It is an open problem if lattice reduction for ideal lattices is as hard as it is for random lattices. So far, there is no algorithm that makes use of the special structure of ideal lattices, and people believe that lattice reduction in the ideal lattice case is as hard as in the regular case. In addition, it is unclear if q-ary lattices, which are more important for cryptography than random lattices, behave different to random lattices. There is need for exhaustive testing of the existing lattice reduction algorithms on q-ary and ideal lattices.

Unfortunately the NTL library is the only implementation of BKZ that is publicly available. It is evident that this version is no more up to date. Numerous improvements to SVP solvers were made during the last years. Therefore, it is necessary to test BKZ including these new enumeration and sieving variants to show its real strength today.

The *theoretical* analysis of BKZ is still in the early stages. Basic facts like worst-case runtime are still not known. There is a lot of work left to do in this area.

References

[Ajt96] Miklós Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *STOC 1996*, LNCS, pages 99–108. ACM, 1996.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC 2001*, pages 601–610. ACM, 2001.

[BBD08] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors. *Post-Quantum Cryptography*. Springer, 2008.

[BLR08] Johannes Buchmann, Richard Lindner, and Markus Rückert. Explicit hard instances of the shortest vector problem. In *PQCrypto 2008*, volume 5299 of *LNCS*, pages 79–94. Springer, 2008.

[CJL+92] Matthijs J. Coster, Antoine Joux, Brian A. LaMacchia, Andrew M. Odlyzko, Claus-Peter Schnorr, and Jacques Stern. Improved Low-Density Subset Sum Algorithms. *Computational Complexity*, 2:111–128, 1992.

[CPS] David Cadé, Xavier Pujol, and Damien Stehlé. fpLLL - A floating point LLL implementation. Available at Damien Stehlé's homepage at école normale supérieure de Lyon, `http://perso.ens-lyon.fr/damien.stehle/index.html`.

[DS10] Özgür Dagdelen and Michael Schneider. Parallel Enumeration of Shortest Lattice Vectors, 2010. To appear in Euro-Par 2010.

[FP83] U. Fincke and Michael Pohst. A procedure for determining algebraic integers of given norm. In *European Computer Algebra Conference 1983*, volume 162 of *LNCS*, pages 194–202. Springer, 1983.

[GM03] Daniel Goldstein and Andrew Mayer. On the equidistribution of Hecke points. *Forum Mathematicum 2003, 15:2*, pages 165–189, 2003.

[GN08a] Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *STOC 2008*, pages 207–216. ACM, 2008.

[GN08b] Nicolas Gama and Phong Q. Nguyen. Predicting Lattice Reduction. In *Eurocrypt 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, 2008.

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice Enumeration using Extreme Pruning. In *Eurocrypt 2010*, volume 6110 of *LNCS*, pages 257–278. Springer, 2010.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC 2008*, pages 197–206. ACM, 2008.

[HSB+10] Jens Hermans, Michael Schneider, Johannes Buchmann, Frederik Vercauteren, and Bart Preneel. Parallel Shortest Lattice Vector Enumeration on Graphics Cards. In *Africacrypt 2010*, volume 6055 of *LNCS*, pages 52–68. Springer, 2010.

[Kan83] Ravi Kannan. Improved Algorithms for Integer Programming and Related Lattice Problems. In *STOC 1983*, pages 193–206. ACM, 1983.

[LLL82] Arjen Lenstra, Hendrik Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LO85] J. C. Lagarias and Andrew M. Odlyzko. Solving Low-Density Subset Sum Problems. *Journal of the ACM*, 32(1):229–246, 1985.

[May10] Alexander May. Using LLL-Reduction for Solving RSA and Factorization Problems. In Nguyen and Vallée [NV10], pages 315–348.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 2002.

[MR07] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

[MR08] Daniele Micciancio and Oded Regev. Lattice-based Cryptography. In Bernstein et al. [BBD08], pages 147–191.

[MV10a] Daniele Micciancio and Panagiotis Voulgaris. A Deterministic Single Exponential Time Algorithm for Most Lattice Problems based on Voronoi Cell Computations. In *STOC 2010*. ACM, 2010.

[MV10b] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *SODA 2010*, pages 1468–1480. ACM/SIAM, 2010.

[NS05] Phong Q. Nguyen and Damien Stehlé. Floating-Point LLL Revisited. In *Eurocrypt 2005*, volume 3494 of *LNCS*, pages 215–233. Springer, 2005.

[NS06] Phong Q. Nguyen and Damien Stehlé. LLL on the Average. In *ANTS 2006*, volume 4076 of *LNCS*, pages 238–256. Springer, 2006.

[NV08] Phong Q. Nguyen and Thomas Vidick. Sieve Algorithms for the Shortest Vector Problem are Practical. *J. of Mathematical Cryptology*, 2(2), 2008.

[NV10] Phong Q. Nguyen and Brigitte Vallée, editors. *The LLL Algorithm - Survey and Applications*. Springer, 2010.

[PS10a] Xavier Pujol and Damien Stehlé. Accelerating Lattice Reduction with FPGAs, 2010. To appear in Latincrypt 2010.

[PS10b] Xavier Pujol and Damien Stehlé. Solving the Shortest Lattice Vector Problem in Time $2^{2.465n}$, 2010. submitted.

[RS10] Markus Rückert and Michael Schneider. Selecting Secure Parameters for Lattice-based Cryptography. Cryptology ePrint Archive, Report 2010/137, 2010. `http://eprint.iacr.org/`.

[Sch91] Claus-Peter Schnorr. Factoring Integers and Computing Discrete Logarithms via Diophantine Approximations. In *Eurocrypt 1991*, volume 547 of *LNCS*, pages 281–293. Springer, 1991.

[Sch94] Claus-Peter Schnorr. Block Reduced Lattice Bases and Successive Minima. *Combinatorics, Probability & Computing*, 3:507–522, 1994.

[SE91] Claus-Peter Schnorr and M. Euchner. Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. In *FCT 1991*, pages 68–85. Springer, 1991.

[Sho] Victor Shoup. Number Theory Library (NTL) for C++. `http://www.shoup.net/ntl/`.

Bedrohungsmodellierung (Threat Modeling) in der Softwareentwicklung

Fabian Schwab, B.Sc.; Alexander Findeisen;
Peter Sakal, B.Sc.; Prof. Dr. Hartmut Pohl

Informationssicherheit, Fachbereich Informatik
Hochschule Bonn-Rhein-Sieg
Grantham-Allee 20
53757 St. Augustin
Fabian.Schwab@h-brs.de
Alexander.Findeisen@smail.inf.h-brs.de
Peter.Sakal@h-brs.de
Hartmut.Pohl@h-brs.de

Abstract: Threat Modeling ermöglicht als heuristisches Verfahren die methodische Überprüfung eines Systementwurfs oder einer Softwarearchitektur, um Sicherheitslücken kostengünstig und frühzeitig - idealerweise in der Design Phase - im Software-Entwicklungsprozess zu identifizieren, einzugrenzen und zu beheben. Threat Modeling lässt sich aber auch noch erfolgreich in der Verifikationsphase oder noch später - nach dem Release - zur Auditierung der Software einsetzen.

Durch die Früherkennung von Sicherheitslücken können die Kosten zur Behebung bis auf ein Hundertstel reduziert werden. Die auf dem Markt verfügbaren Threat Modeling Tools werden identifiziert, analysiert und hinsichtlich Ihrer Eignung zur Erstellung komplexer, vollständiger Threat Models mit entwickelten Bewertungsparametern einem einfachen Bewertungsverfahren unterworfen.[1]

[1] Förderung des Projektes SoftSCheck durch das Bundesministerium für Bildung und Forschung (Förderkennzeichen 01 IS 09030)

1 Einführung Threat Modeling

Das heuristische Verfahren Threat Modeling unterstützt die Identifizierung von Sicherheitslücken. Durch Bewertung und Kategorisierung von Sicherheitslücken können Schutzmechanismen und Gegenmaßnahmen bestimmt werden, um ein Sicherheitsrisiko zu mindern, zu minimieren, zu beheben oder auch zu akzeptieren [HL06]. Allerdings können grundsätzlich nicht alle Sicherheitslücken identifiziert werden [Tu36, Di72].

Die Erstellung eines Threat Models für einen Systementwurf erfolgt idealerweise bereits in der Design Phase, da dort die Kosten zur Beseitigung von Sicherheitslücken minimal sind [SP10]. Werden Sicherheitslücken dagegen erst nach dem Release der Software an die Kunden entdeckt, so müssen Patches erstellt und ausgeliefert werden – ein deutlicher Mehraufwand gegenüber der Entdeckung in der Design Phase. Nach vollständiger Identifizierung schützenswerter Komponenten (Assets) sowie zugehöriger Bedrohungen und Sicherheitslücken ist ihre Bewertung erforderlich. Identifizierung und Bewertung der Bedrohungen und Sicherheitslücken kann z.B. durch Attack Trees erfolgen [Sc99]. Auf Grundlage dieser Bewertung kann eine Minderung (Mitigation) der Bedrohungen und eine Behebung der Sicherheitslücken erfolgen. Neben den unterschiedlichen, in den Threat Modeling Tools implementierten, Maßnahmen (z.B. Redesign, Standard Mitigation, Custom Mitigation oder Accept Risk) ist eine individuelle Behandlung einzelner Bedrohungen und Sicherheitslücken sowie die Kontrolle der implementierten Verfahren erforderlich. Die Entstehung neuer Bedrohungen für ein bereits durch Threat Modeling spezifiziertes System ist nach [Ow09] unwahrscheinlich. Für den systematischen Ablauf des Verfahrens Threat Modeling vgl. Abbildung 1: Threat Modeling Ablauf.

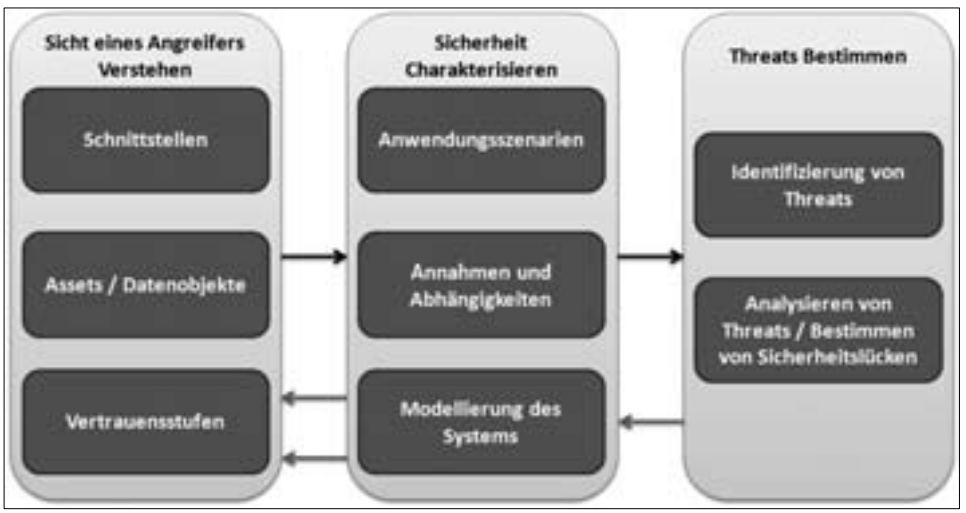


Abbildung 1: Threat Modeling Ablauf [nach: SS04]

Durch den Einsatz von Threat Modeling bereits in der Design Phase ist eine vertrauenswürdige Implementierung von Software bereits vor erstmaliger Veröffentlichung (Release) möglich. Die Kosten zur Behebung einer Sicherheitslücke können durch die Forcierung einer proaktiven Früherkennung von Sicherheitslücken auf ein Hundertstel reduziert werden (vgl. Kosten zur Behebung von Sicherheitslücken im Verlauf der Softwareentwicklung). Threat Modeling ist ein kosteneffektives und vollständiges Verfahren zur Identifizierung, Vermeidung und Verminderung von Sicherheitslücken und Bedrohungen [My05, MLY05]. Durch Threat Modeling gewonnene Erkenntnisse können ebenso zur Entwicklung strategischer Penetration Tests genutzt werden. Hierbei wird auf Erkenntnisse über Angriffspunkte und Eingabeschnittstellen zurückgegriffen, welche aus der modellierten Systemarchitektur gewonnenen werden können.

Neben der Anwendung in der Design Phase von Software ist es auch möglich ein Threat Model für eine bereits implementierte Systemarchitektur zu erstellen. Hierbei wurden in den durchgeführten Untersuchungen große Erfolge erzielt. Durch herkömmliche Verfahren nicht identifizierte Bedrohungen und Sicherheitslücken konnten durch die Anwendung von Threat Modeling, am Beispiel einer Individualsoftware, identifiziert und behoben werden.

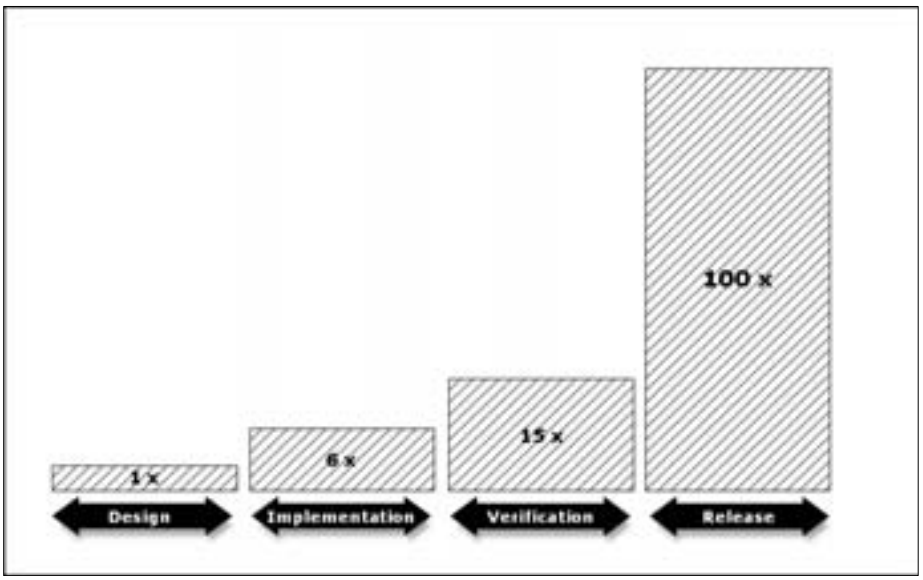


Abbildung 2: Kosten zur Behebung von Sicherheitslücken im Verlauf der Softwareentwicklung [nach: Ni02]

Die auf dem Markt verfügbaren Threat Modeling Tools werden identifiziert, analysiert und hinsichtlich Ihrer Eignung zur Erstellung komplexer, vollständiger Threat Models bewertet.

2 Bewertungsverfahren für Threat Modeling Tools

Um eine einheitliche Klassifizierung der zu untersuchenden Threat Modeling Tools zu gewährleisten, ist die Entwicklung und Begründung von passenden Bewertungsparametern unabdingbar. Diese müssen unter Berücksichtigung der im Projekt festgelegten Ziele neben einer eindeutigen Produktzuordnung die anfallenden Kosten, den Funktionsumfang, die Entwicklungsmöglichkeiten, den Installationsaufwand, die Benutzerfreundlichkeit und die Qualität der Dokumentation berücksichtigen. Die Bewertungsparameter sind in unterschiedliche Kategorien unterteilt. Kategorien beinhalten jeweils Parameter einer Funktionsgruppe. Parameter sind entsprechend ihrer Relevanz gewichtet und fließen in die Berechnung der jeweiligen Kategorie ein. Der Wert eines numerischen Parameters wird anhand der folgenden Formel berechnet:

$$\text{Wert des Parameters} = \frac{\sum \text{Punkte}}{\text{Mögliche Punkte}} \times \text{Parametergewichtung}$$

Die möglichen Punkte werden anhand der Bewertungsmethode bestimmt. Sofern eine Existenzprüfung (E) vorgenommen wird, wird lediglich zwischen Existenz (1 Punkt) und Nicht-Existenz (0 Punkte) unterschieden. Die maximale Anzahl der möglichen Punkte beträgt somit 1. Im Falle einer qualitativen Bewertung (Q) des Parameters beträgt die maximale Anzahl der möglichen Punkte 3. Der Bewertungsmaßstab ist für die einzelnen Parameter individuell definiert.

Einzelne Kategorien werden ebenso auf der Grundlage Ihrer Relevanz unterschiedlich gewichtet. Die Berechnung erfolgt anhand der folgenden Formel:

$$\text{Bewertung der Kategorie} = \frac{\sum \text{Werte der Parameter}}{\sum \text{Gewichtungen d.Parameter}} \times \text{Kategoriengewichtung}$$

Das Endergebnis des Bewertungsparameterkatalogs erfolgt anhand der folgenden Formel und liefert einen für Vergleiche relevanten Wert:

$$\text{Gesamtbewertung} = \frac{\sum \text{Bewertung der Kategorien}}{\sum \text{Kategoriengewichtungen}} \times 100$$

Die Gewichtungen der Kategorien und Parameter sowie die verwendete Bewertungsmethode können Abbildung 3 entnommen werden.

Die Produkte werden anhand der folgenden Kategorien bewertet:

- Erfassung der Modellierungsmöglichkeiten hinsichtlich Assets, Threats, Threat Mitigation, Vulnerabilities und dem eingesetzten System bzw. der Systemumgebung.
- Folgende Visualisierungsmöglichkeiten werden erfasst: Datenflussdiagramm, Bedrohungsbaum, Anwendungsdiagramm sowie sonstige Möglichkeiten.
- Berichtswesen (Reporting) erfasst Möglichkeiten des grafischen und textuellen Exports sowie den Reportumfang.

- Folgende Entwicklungsmöglichkeiten werden berücksichtigt:
  - Vorhandener Quellcode
  - Entwicklungsschnittstellen: Möglichkeit zur Anbindung eigener Erweiterungen und/oder zusätzlicher Module)
  - Communityprojekt: Entwicklung und/oder Support durch eine Community
  - Dokumentation der Entwicklungstools
- Beim Installations- und Nutzungsaufwand wird unterschieden nach der Grundinstallation, speziellen Voraussetzungen (z.B. erforderliche Software), Einarbeitungszeit, Usability und Anwendung (vollständige Modellierung eines Testszenarios).

Für eine vollständige Übersicht der Bewertungsparameter inkl. Gewichtung vgl. Abbildung 3: Gewichtung der Bewertungsparameter.

Nr.	Bewertungsparameter	Prüfung	Gewichtung	%
1	**Modellierungsmöglichkeiten**		**1**	**34%**
1.1	Assets	Q	1	29%
1.2	Threats	Q	1	29%
1.3	Threat Mitigation	E	0,5	14%
1.4	Vulnerabilities	Q	0,5	14%
1.5	System/Umgebung	Q	0,5	14%
2	**Visualisierung**		**0,3**	**10%**
2.1	Datenflussdiagramm	E	0,2	18%
2.2	Bedrohungsbaum	E	0,2	18%
2.3	Anwendungsdiagramm	E	0,2	18%
2.4	Sonstiges	Q	0,5	45%
3	**Reporting**		**0,8**	**28%**
3.1	Grafischer Export	Q	0,5	20%
3.2	Textueller Export	Q	1	40%

3.3	Umfang Report	Q	1	40%
4	**Entwicklungsmöglichkeiten**		**0,3**	**10%**
4.1	Quellcode vorhanden	E	0,8	36%
4.2	Entwicklungsschnittstellen	E	0,8	36%
4.3	Communityprojekt	E	0,3	14%
4.4	Dokumentation der Entw.-Tools	E	0,3	14%
5	**Installations-/Nutzungsaufwand**		**0,5**	**17%**
5.1	Grundinstallation	Q	0,8	20%
5.2	Spezielle Voraussetzungen	Q	0,8	20%
5.3	Einarbeitungszeit	Q	0,8	20%
5.4	Usability	Q	0,8	20%
5.5	Anwendung	Q	0,8	20%
Gesamt			**2,9**	**100%**

Abbildung 3: Gewichtung der Bewertungsparameter

3 Marktanalyse und Bewertung

Zum Zeitpunkt dieser Untersuchung waren sechs Software-Produkte (entgeltfrei oder entgeltpflichtig) zur Erstellung von Threat Models verfügbar. Die Produkte können durch folgende Kategorien beschrieben werden:

- Universal: Das Produkt eignet sich zur Erstellung eines Threat Models ohne spezifische Ausrichtung.
- Netzwerk: Das Produkt eignet sich zur Erstellung eines Threat Models für ein Netzwerkszenario. Es werden insbesondere Elemente eines Netzwerks betrachtet.
- Software Design: Das Produkt eignet sich zur Erstellung eines Threat Models im Rahmen des Software Designs und berücksichtigt spezifische Elemente der Software Entwicklung. Die Integration in den Software-Entwicklungsprozess von Microsoft - dem Security Development Lifecycle (SDL) - wird unterstützt.

Zu den untersuchten Produkten sowie deren Einsatzkategorie vgl. Abbildung 4: Erhältliche Threat Modeling Tools.

Name	Hersteller	Kategorie	Version
The CORAS Method	CORAS	Universal	20060714
Microsoft SDL Threat Modeling Tool	Microsoft	Software Design	3.1.3.1
Microsoft Threat Analysis & Modeling	Microsoft	Software Design	3.0
Practical Threat Analysis	PTA Technologies	Universal	1.6 Build 1212
Skybox Secure	Skybox Security	Netzwerk	4.5
Trike	Dymaxion	Universal	1.12a

Abbildung 4: Erhältliche Threat Modeling Tools

Um die Interessen von Herstellern zu wahren werden die Ergebnisse im weiteren Verlauf anonymisiert dargestellt. Im Rahmen der durchgeführten Untersuchungen wurden große Diskrepanzen beim Funktionsumfang - explizit bei Modellierungs- und Visualisierungsmöglichkeiten - festgestellt. Fünf der untersuchten sechs Produkte bieten Funktionen zur Modellierung elementarer Bestandteile eines Threat Models - Assets, Threats und Vulnerabilities [Sc06] - in unterschiedlichem Umfang. Mit dem weiteren Produkt (Tool C) ist ausschließlich die Modellierung von Assets möglich - bei dem Produkt handelt es sich um eine Sammlung von Symbolen zur Darstellung eines Threat Models. Lediglich zwei Produkte (Tool A und Tool E) bieten umfangreiche Visualisierungsmöglichkeiten in Form von Datenflussdiagrammen, Bedrohungsbäumen und Anwendungsdiagrammen. Die anderen Produkte bieten ausschließlich eigene Visualisierungsformen in unterschiedlichem Umfang. In den weiteren Kategorien werden Stärken und Schwächen der Produkte deutlich - extreme Diskrepanzen existieren jedoch nicht. Die Bewertung erfolgte auf Grundlage des entwickelten Bewertungsverfahrens. Die Ergebnisse der einzelnen Parameter, Kategorien und der Gesamtbewertung werden in Prozent der möglichen Punkte dargestellt. Lediglich drei Produkte erzielten in der Gesamtbewertung mehr als 70% der möglichen Punkte (Tool A, Tool B und Tool E).

Die weiteren Produkte weisen große Defizite (weniger als 10% der möglichen Punkte) in mindestens einer Kategorie auf und erzielen in der Gesamtbewertung weniger als 70% der möglichen Punkte. Die Erstellung komplexer, vollständiger Threat Models ist mit drei Produkten möglich. Für eine detaillierte Darstellung der Ergebnisse vgl. Abbildung 5: Ergebnisse des Bewertungsverfahrens.

Nr	Bewertungs-parameter	Tool A	Tool B	Tool C	Tool D	Tool E	Tool F
1	**Modellierungs-möglichkeiten**	**++++**	**++++**	**+**	**+++**	**++++**	**+++**
1.1	Assets	++	+++++	++	+++	+++++	++
1.2	Threats	+++++	+++++	-	+++	+++	+++
1.3	Threat Mitigation	+++++	+++++	-	+++++	+++++	+++++
1.4	Vulnerabilities	+++++	+++++	-	++	+++	++
1.5	System/Umgebung	+++++	+++++	-	++	+++++	++
2	**Visualisierung**	**+++++**	**+**	**-**	**+**	**+++++**	**+**
2.1	Datenflussdia-gramm	+++++	-	-	-	+++++	-
2.2	Bedrohungsbaum	+++++	-	-	-	+++++	-
2.3	Anwendungs-diagramm	+++++	-	-	-	+++++	-
2.4	Sonstiges	+++++	++	-	++	+++++	++
3	**Reporting**	**++++**	**++++**	**+**	**++**	**++++**	**+**
3.1	Graphischer Export	++	++	-	++	++	+++
3.2	Textueller Export	+++++	+++++	++	+++	+++	-
3.3	Umfang Report	+++++	+++++	++	++	+++++	++
4	**Entwicklungs-möglichkeiten**	**+**	**+**	**-**	**-**	**+++**	**-**
4.1	Quellcode vorhan-den	-	-	-	-	-	-
4.2	Entwicklungs-schnittstellen	-	-	-	-	+++++	-
4.3	Communityprojekt	+++++	-	-	-	+++++	-

4.4	Dokumentation der Entw.-Tools	-	+++++	-	-	-	-
5	**Installations-/Nutzungsaufwand**	**++++**	**++++**	**+++**	**+++++**	**++**	**++++**
5.1	Grundinstallation	++	+++++	+++++	+++++	-	+++++
5.2	Spezielle Voraussetzungen	+++	+++++	+++++	+++++	++	+++++
5.3	Einarbeitungszeit	+++++	-	+++++	+++++	+++	++
5.4	Usability	+++++	+++++	-	+++++	+++	++
5.5	Anwendung	+++++	+++++	++	+++++	+++	++
Gesamt		**++++**	**++++**	**+**	**+++**	**++++**	**++**

Abbildung 5: Ergebnisse des Bewertungsverfahrens

Die Ergebnisse in Abbildung 5 werden prozentual ausgedrückt, wobei +++++ dem Maximalwert von 100% entspricht. Die durch die Produkte erzielten Punkte in den einzelnen Kategorien sind in Abbildung 6 grafisch dargestellt. Die Stärken und Schwächen sowie die Gesamtbewertung und der Funktionsumfang der einzelnen Produkte werden verdeutlicht.

Tool A zeichnet sich durch gute Modellierungsmöglichkeiten sowie durch eine sehr gute Visualisierung aus; das Reporting ist gut ausgebildet; der Installations- und Nutzungsaufwand ist gut und Entwicklungsmöglichkeiten sind befriedigend - es handelt sich um ein Communityprojekt.

Tool B zeichnet sich durch gute Modellierungsmöglichkeiten und ein gutes Reporting aus; Entwicklungsmöglichkeiten sind in geringen Umfang vorhanden und gut dokumentiert; die Visualisierung ist ausreichend (nur eigene Visualisierungsformen) und der Installations- und Nutzungsaufwand ist gut.

Tool C bietet unzureichende Modellierungsmöglichkeiten und Berichtsfunktionen; Funktionen zur Visualisierung und Entwicklungsmöglichkeiten sind nicht vorhanden; der Installations- und Nutzungsaufwand ist befriedigend; es handelt sich um eine Sammlung von Symbolen zur Darstellung eine Threat Models.

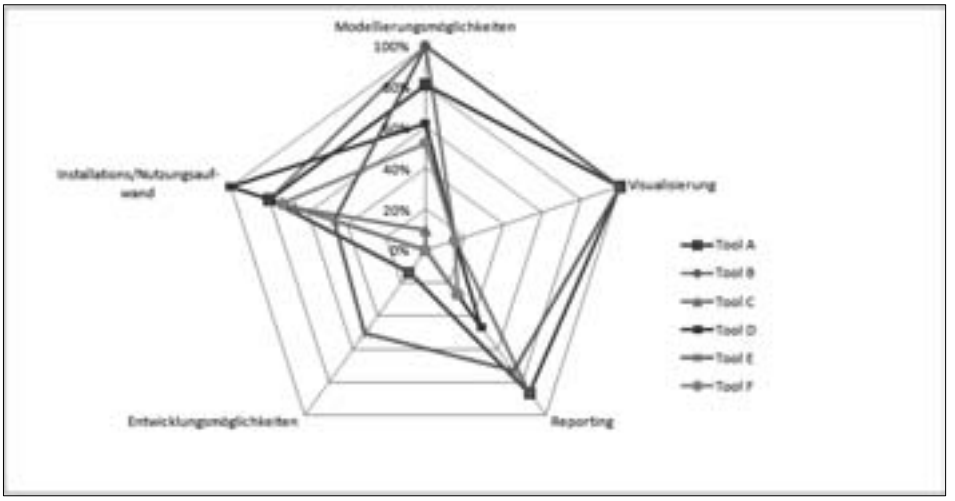


Abbildung 6: Ergebnisse: Grafische Ergebnisdarstellung des Bewertungsverfahrens

Tool D bietet befriedigende Modellierungsmöglichkeiten und ein ausreichendes Reporting; Visualisierung ist mangelhaft und Entwicklungsmöglichkeiten sind nicht vorhanden; der Installations- und Nutzungsaufwand ist sehr gering.

Tool E zeichnet sich durch gute Modellierungsmöglichkeiten und eine sehr gute Visualisierung aus; das Reporting ist gut ausgebildet; die Entwicklungsmöglichkeiten sind gut - es handelt sich um ein Communityprojekt und es sind Entwicklungsschnittstellen vorhanden; der Installations- und Nutzungsaufwand ist ausreichend gering.

Tool F bietet befriedigende Modellierungsmöglichkeiten und einen guten Installations- und Nutzungsaufwand; Visualisierung und Reporting sind mangelhaft; Entwicklungsschnittstellen sind nicht vorhanden.

Zusammenfassend ergibt sich eine Spitzengruppe gebildet aus den Tools A, B und E. Die Tools C, D und F zeigen - bei Anwendung der hier zugrunde gelegten Bewertungsparameter - nur mittlere oder sogar unzureichende Leistungen - vgl. Abb. 7: Vergleich der untersuchten Produkte.

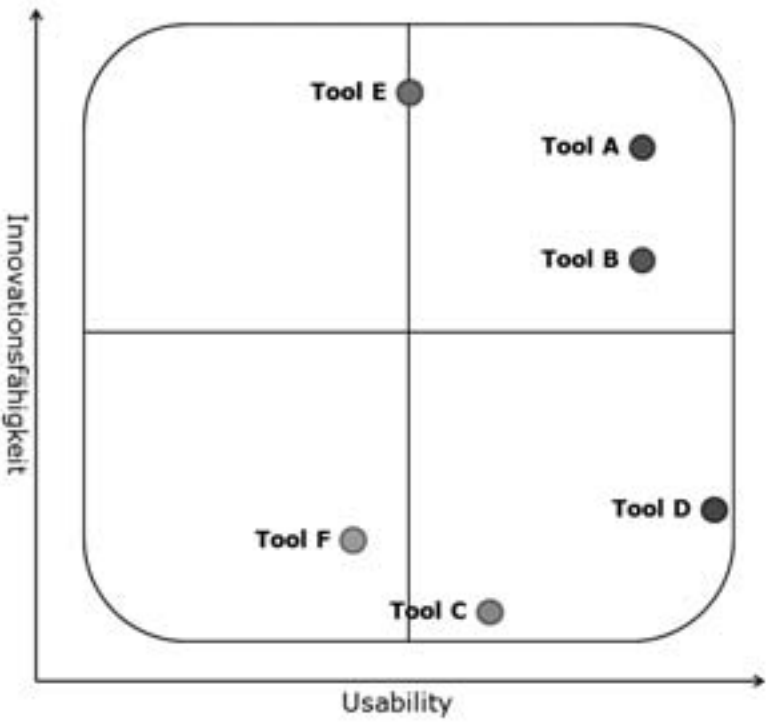


Abbildung 7: Vergleich der untersuchten Produkte

4 Fazit

Die Erstellung eines Threat Models kann mit entsprechenden Tools unterstützt werden. Die erhältlichen Produkte wurden anhand eines entwickelten Verfahrens bewertet. Lediglich drei der untersuchten Produkte erzielten zufriedenstellende Ergebnisse und können zur Erstellung komplexer, vollständiger Threat Models eingesetzt werden. Alle Tools wurden auf Basis des durchgeführten Bewertungsverfahrens zusammenfassend im Hinblick auf ihren Innovationsgrad bewertet – vgl. Abbildung 7: Vergleich der untersuchten Produkte**.**

Das best-bewertete Tool wurde bereits mehr als 20-mal zur Identifizierung von jeweils etwa 50 aus dem Internet ausnutzbaren Sicherheitslücken in Standard- und Individualsoftware erfolgreich eingesetzt.

Es ist daher zu erwarten, dass sich Threat Modeling zur Identifizierung von Sicherheitslücken durchsetzt - neben Fuzzing, mit dem nach der Implementierung weitere Softwarefehler und Sicherheitslücken erfolgreich identifiziert werden können.

Literaturverzeichnis

[Di72] Dijkstra, E. W.: Chapter I: Notes on structured programming Structured programming" (pp. 1-82). Eindhofen 1972

[HL06] Howard, M.; Lipner, S.: The Security Development Lifecycle. SDL: A Process for Developing Demonstrably More Secure Software. Microsoft Press, Redmond 2006.

[My05] Myagmar S.: Threat Modeling networked and data-centric systems, Urbana 2005, http://www.projects.ncassr.org/threatmodeling/myagmar-msthesis.pdf

[MLY05] Myagmar, S.; Lee, A. J..; Yurcik, W.: Threat Modeling as a Basis for Security Requirements. Symposium on Requirements Engineering for Information Security. Paris 2005. http://www.suvda.com/papers/threat_sreis05.pdf.

[Ni02] National Institute of Standards and Technology (NIST): The Economic Impacts of Inadequate Infrastructure for Software Testing. Gaithersburg 2002. http://www.nist.gov/director/prog-ofc/report02-3.pdf

[Ow09] OWASP Foundation: Threat Risk Modeling. Columbia 2009. http://www.owasp.org/index.php/Threat_Risk_Modeling

[Sc06] Schumacher, M. et. al.: Security Patterns. Integrating Security and Systems Engineering. Wiley, Hoboken 2006

[Tu36] Turing, A.: On computable numbers, with an application to the "Entscheidungsproblem", Proceedings of the London Mathematical Society, Series 2, 42 pp 230–265, London 1936

[Sc99] Schneier, B.: Attack Trees: Modeling Security Threats. In: Dr. Dobb's Journal, v. 24, n.12. San Francisco 1999. http://www.schneier.com/paper-attacktrees-ddj-ft.html

[SP10] Sakal, P.; Pohl,H.: Entwicklungshelfer und Stresstester -Toolgestützte Identifizierung von Sicherheitslücken in verschiedenen Stadien des Softwarelebenszyklus. In: KES - Fachzeitschrift für Informationssicherheit 63 - 66, 2, Gau-Algesheim 2010. http://softscheck.com/publications/Pohl_Sakal_Kostenguenstige_Identifizierung_von_Sicherheitsluecken_final.pdf

[SS04] Swiderski, F.; Snyder, W.: Threat Modeling. Microsoft Press, Redmond 2004.

Why Showing one TLS Certificate is not enough? — Towards a Browser Feedback for Multiple TLS Certificate Verifications

Henrich C. Pöhls*
University of Passau, Institute of IT-Security and Security Law
hp@sec.uni-passau.de

Abstract: Content reuse on the Web 2.0 is a common "phenomenon". However, it has now reached critical and sensitive areas, as for example online shopping's submission forms for credit card data. Browsers lack the ability to show anything else than the outer most's TLS certificate verification to the user. We show that there is a trend to embed security critical content from other site's into a website. We will use VISA's credit card submission form embedded in an `<iframe>` as example. We give detailed examples of existing tentatives to solve the problem. After analyzing them, we argue that a solution can only be at the web browser's core. Finally, we postulate five steps to be taken into consideration for to evaluate and structure future solutions.

1 Introduction

Since, "Web 2.0" Internet users are aware that the content they see in their web browser's window no longer comes from one single source. It can be created by others, re-distributed through several services, re-combined by one or more web application, before finally being displayed in the web browser to be viewed and used by the user. The combination of content from different sources to provide new or improved services for the typical Internet content is routinely done in "Mashups" [Pro07]. Just recently this phenomenon, especially that of embedded content from other locations, has begone to be applied to critical areas like banking or shopping. VISA's "Verified by Visa" is the most prominent example, it is also known by the name of MasterCard SecureCode or 3-D Secure [Vis06, Vis]. "Verified by Visa" is about to reach German customers in spring 2010 when more and more banks will introduce it [com10a].

Therefore, we took this prominent example to explore and analyze the before mentioned concerns. The screenshot in Figure 1 shows how a Verified by Visa enhanced web page looks like in the Safari web browser. As the content in section (a) of Figure 1 provides input fields for sensitive data, the question "Where does this content originate from?" is now paired with "Where do my credentials end up?". Of course, a sophisticated user would use browser functions to open the embedded content in a new window.

*is funded by BMBF (FKZ:13N10966) and ANR as part of the ReSCUe IT project

Figure 1: Credit card checkout of a shop: (a) shows the "Verified by Visa" content included via `<iframe>` (b) some logos (c) social network links (d) ads; Equal output for `https` and `http`

Figure 2: The "Verified by Visa" content's `<iframe>` loaded into a window and the certificate information dialog of the Safari web browser

Indeed, the content is embedded as an <iframe>, and once the <iframe>'s content was opened in a separate browser window the result of the browser's validation result can be seen as Figure 2 shows. The browser confirms that section (a) indeed came through an encrypted and authenticated Transport Layer Security (TLS) tunnel using HTTPS as the protocol [DR06].

However, the main goal of this work is not to criticize Visa's approach, Ross Andersen and Steven J. Murdoch have already given an exhaustive analysis of the system's flaws in [MA10]. Though, we will use "Verified by Visa" as an example to demonstrate a more general shortcoming and the resulting weaknesses: Today's browsers only provide the user with the certificate verification result of the outer most TLS certificate. In more general terms, every time the user is left unaware of the true origins or has to take additional and uncomfortable measures to verify the content's validity he is not able to make an informed decision. One strategy to judge the credibility of information is to find and verify its author [FSD+03]. A user can only do this for the TLS secured tunnel that the regular and outer HTML container was delivered. The embedded content, such as pictures, JavaScript and <iframe>s, can be dynamically, still securely, fetched from different sources via their own TLS secured tunnels. Though the regarding verification results are not available for the user of the website. This absence has been neglected too long. Furthermore we discuss the issues that arise from using the embedding techniques in sensitive applications like the input of credit card data. The above described shortcoming is still not widely known and discussed, but causes security problems when applied to sensitive applications.

The paper is organized as follows, we will start with existing approaches and then classify them. We then reason about the security of the existing approaches. Based on the most promising one, we will finally close with suggestions for future browser development.

2 Security Goal: GUI-Feedback of Authenticity and Confidentiality

First of all we will define the security goals that we would like to reach when showing content from mixed sources securely in one page. The first property is origin authenticity. Following the definition from Gollmann [Gol05], *data origin authentication* will "provide the means to verify the source and integrity of a message". Hence, the content's integrity is also protected. We will further distinguish two notions for authenticity of origin: *hop-to-hop* or *end-to-end*. *End-to-end authenticity* allows the origin of the content to be verified even if it has been copied, moved, or stored elsewhere, iff the content is not modified. Thus, it is persistant. *Hop-to-hop authenticity* allows to verify the origin of the sender of the content in an ongoing transmission. So for the point-to-point transmission of content, for example from the web server to a web browser, the sending party as the source can be verified. TLS offers exactly hop-to-hop authenticity, the authentication of origin is not preserved after the connection has ended, so it can not provide protection for the content after its transfer [Kah06].

Confidentiality is the second property. A usable confidentiality protection only makes sense if the communicating partners that keep their messages confidential can be authenti-

cated (before mentioned hop-to-hop authenticity). In the web confidentiality can be guaranteed for already downloaded content, but is also important when considering future sending of data. As in the example of "Verified by Visa" 'downloaded web content can contain a HTML `form`, the filled in form fields are submitted to a target which is totally unrelated to the source. The `forms` target, as specified in the `forms`'s `action`, could be a different domain or a different protocol (`https` or `http`) . Hence, we include confidentiality assurances for future actions because they are usual in the user web browser web server interactions. So in the following we will distinguish between *downstream confidentiality* of received data and *upstream confidentiality* protecting data that is send upstream in future transmissions.

As motivated before, we need a way to inform the user about: Confidentiality protection (up- or downstream) including the hop-to-hop authenticity verification result of the communicating endpoint (up- or downstream), and the authenticity and integrity (end-to-end) verification results for protected content. This includes information about failed verifications. The information must be conveyed to the user in such a way that they are hard to spoof by a maliciously crafted web page.

3 Discussion of Existing Approaches

Turning from the security goals to existing implementations or specifications, digital signatures and related technologies are in widespread use to securely mark a content's origin and allow for its verification. The idea of signing web content outside the scope of TLS secured communication tunnels was already mentioned in 2002 in a mailing list discussion [RM02]. In the same year the work of Chi and Wu discussed signed web content for caching [CW02]. More recently, new services were created[1] that allow authors to "claim" web content, thus providing the user with means to "verify" the origin. Our own analysis [BP08], and a recent review of likewise services have shown that while these services can provide end-to-end authenticity they do not provide any confidentiality protection, neither downstream nor upstream. Thus, the existing mechanisms included in TLS secured tunnels, for hop-to-hop authenticity and down- and upstream confidentiality seem a good starting point to look for a solution. We found four existing approaches that can be used to tackle the problem of verification of content from various trusted sources within one web browser's page. We will shortly describe each of them in the following and which of our security goals they reach.

3.1 Approach I: Why SSL is not enough

In their work "Why SSL is not enough", Matthias Quasthoff, Harald Sack, and Christoph Meinel [QSM07] content signed by XML signatures was embedded into XHTML web

[1] i.e. MicroID.org, ClaimID.com, FindMeOn.com, RegisteredCommons.org, Numly.com, or DulyNoted.co.uk

pages. For verification they used an extension to the Firefox web browser using an extension that shows the signed and verified content in a separate window as shown in Figure 3. This solution circumvents attacks from a malicious web page as the information inside the extension's window is not controllable by the web content. Therefore, it provides the correct information in case the malicious web page tries to change the rendering of signed content after the verification or visually highlights unsigned content as signed. In other words, the approach is suited to verify end-to-end authenticity. However, even with the use of XML encryption, which was not considered by the authors, a satisfying up- and downstream confidentiality protection would require a previously established session key or a previous established public key, under which sensitive content could be encrypted.

3.2 Approach II: ConCert

Our own solution named ConCert, as presented for example in [PÖ8], also signs content before it is embedded into HTML web pages. Rather than XML, ConCert uses Microformats [Mic06] to mark signed content and to embed a X.509 [HPFS02] compatible content certificate. Our choice for verification also was to extended the Firefox web browser. Differently from approach I the extension displays the signed and verified content in a sidebar. The sidebar, as is the separate window of approach I, is solely controlled by the extensions or the core browser and therefore out of control of regular and potentially malicious web content. On top of this feature set, which is comparable to the approach I, it allows to highlight the verified content inside the original rendered web page. Like approach I, our own solution offers end-to-end authenticity of content and additionally allow policy-conforming removal of signed content. It was not designed to offer confidentiality protection. If one would introduce confidentiality by including encryption, again keying material is needed.

I)  II) 

Figure 3: I) Infobox [QSM07]; II) Content in sidebar and CSS highlighting [PÖ8]; both Firefox.

3.3 Approach III: Comodo's Content Verification Certificates

The comercial TLS certificate issuer Comodo offers the service to register parts of certain web pages using so called Content Verification Certificates [Com10b]. We installed the external application for testing which communicates with the browsers using a plugin. Tests were run under Firefox and Internet Explorer. If a verifiable content is under the mouse pointer the verification application draws a green border around the whole screen, as shown in the screenshot in Figure 4. This is paired, by the possibility to right click on the object and see the certificate as issued by Comodo. The user guide [Com05] especially names trademarks, logos, and login boxes as verifiable content and they further claim that thy would detect what they called "Any overlap of content (even by a single pixel)" [Com10b]. This approach, again visually highlights the verification results, though does not re-render the content, but only display a green border around the screen. This is done in addition to the browser based TLS certificate verification and the browser's graphical user interface (GUI), and it also works for `http`, so non TLS secured, content. Only on Comodo's demo page[2] we were able to see that one could right click on a verified part, i.e. the logo, and were able to view the X509 certificate that Comodo was assigning this verifiable content. Generally speaking, approach III provides additional feedback on the authenticity for certain content by visual feedback. Further detailed hints of the source are available using Internet Explorer's windows dialog for showing the X509 certificate. The browser still signals only the outer most TLS connection to the user. The solution has two major drawbacks, first technical details are not well documented and exclusively works with Comodo's special certificates. Second, it tries to solve the problem to: "show the difference between a real website and a spoofed website, keeping a website verifiably authentic"[3]. We assume it ties content to domains, so moving a logo from one domain, where certified, to another should invalidate the certificate. With the little documentation at hand, this seems to offer hop-to-hop authenticity.

3.4 Approach IV: Local Browser User Scripts

The last approach to the problem is prototype we build ourselves. It uses the greasemonkey Firefox extension [gre10], which basically allows the user to run additional scripts on page load to modify a web page's content. With it, we wrote a proof-of-concept prototype. All `iframe`s with a `https` target automatically open new tabs with the `iframe`'s URL in the new tab. Hence, these windows have a full URL and browser bar, allowing the user to verify the certificates in the usual manner, as shown in Figure 2. This approach breaks the `iframe`'s embedding and thus allows the visualization of the otherwise hidden TLS certificate verification results. However, opening new windows is not in line with Visa's best practices for deployment of the "Verified by VISA" content: Visa recommends not to use popups [Vis05]. It has some usability issues, i.e. it is unclear, whether all web applications and shopping sites that embed content via an `iframe` are still usable

[2] www.contentverification.com
[3] www.comodo.com/e-commerce/ssl-certificates/content-verification.php

Figure 4: Visual of approach III: Border when mouse is over login form [Com10b]

when data is entered into the detached browser window instead of the `iframes` or not. However, this approach is appealing for several reasons: First, it is generic and works with each and every page that uses TLS-secured content in an `iframe`. Second, it works without depending on additional signatures. Third, it allows the user to clearly separate which content came from which site, thus offering hop-to-hop authenticity and up- and downstream confidentiality. Thus, approach IV solves the problem. Last but not least, it is secure as long as Firefox's TLS verification works securely, which is assumed. The drawback: It radically breaks the unity that probably was the reason for embedding the contents in the first place.

4 Further Classification and Discussion of Results

In the following we will shortly introduce the classification properties that we developed to compare and evaluate the existing approaches. We would like those properties to become the basis of future discussions and requirement for developing future solutions. We will shortly introduce the properties and then discuss how and to which extend each approach fulfills it, wrapping it up in a discussion also relating it to other work.

4.1 Classification Properties and Classification Results

4.1.1 Method to Distinguishably Mark the Verified (or Confidentiality Protected) Content within HTML (or HTTP)

A single web page as displayed in the browser can contain several embedded secured contents. The different contents have to be marked either within the HTML code or by ways of the HTTP protocol. This must be done in such a way that the verifier can avoid ambiguities. The approaches I and II discuss or use solutions that allow them to clearly state which content is signed and which signature is attached to which content. In the case of approach I this is done through XML signatures around the XHTML contents and in approach II we described how to mark signed content and embed X.509 [HPFS02] compliant signatures using Microformats. These inline approaches can be seen as steps in the direction of the Tim Berners-Lee's idea of a "Semantic Web" [BLHL01], but the marking can also be build on standards like RDFa [W3C06]. Comodo's approach (III) is not well documented, all we found is that the verification tool "will extract the credentials of the site from the IdAuthority service" [Com10b]. How the certified parts are distinguished from uncertified ones is not documented. Finally, approach IV uses `<iframe` and `src=''https://` as markers.

4.1.2 Way of Visual Highlighting the Verification Outcome

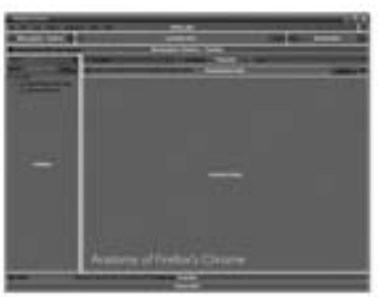
a)  b) 

Figure 5: a) Zones for visual feedback [Faa07a]; b) Markers to relate and access machine-readable content (design study) [Gla07]; both Firefox.

Browsers start to understand more and more semantic annotations that are embedded in XHTML (cp. Firefox 3.0 [Faa07b, Kap07]). Figure 5a taken from [Faa07a] shows the different zones one could possibly place GUI elements within the Firefox browser. This can be transferred to other browsers as well. Now we have seen that the approaches I and II add a copy of the content, to either the sidebar or some other part of the browser. These areas, unlike the Content Area, are not controllable by potentially malicious content. The approach IV also duplicates the content in the new window and leaves the existing certificate verification result notification as it is and thus outside the Content Area. Only approach III, implemented by Comodo, uses the mouse over event to interactively show

the user that the element under his mouse pointer has been verified by Comodo.

4.1.3 Browser Extension or Standalone

All four approaches need to extend the Browser using extensions or plug-ins, and most of them will need some additional components. While approach I uses Java, the second approach needed openssl [Ope06], and the third approach needed Comodo's Verification Engine to run in the background. While these approaches needed external applications, the last approach needed the greasemonkey browser extension. We assume in all cases that all the browser's extensions are trusted, thus an additional trusted extensions output can not be manipulated by the web site displayed. The same holds true for the browser in general and also assume this for the standalone application.

4.1.4 Academic or Commercial

The last classification distinguishes between commercial approaches and academic ones. To start with approach III, Comodo's is the only commercial approach. This is also the only approach that requires a content owner to pay for an additional content verification certificate from Comodo. However, it is the only one with a working interface (at least for Windows and Internet Explorer) which is not beta or research prototype, and has some certified real world content. For example, the Google logo on Google's index page or certain banks' logos such as the PayPal logo. However, it is unclear how many user's are actually using the verification tool and how many contents are verified today. The demoed pages do not contain secured content from multiple sources.

4.2 Discussion and Related Work

We showed that the idea of certified, signed or confidential content from different sources embedded into and alongside one and another is not a new concept. It has been discussed in one or another direction for the last 5 to 8 years. However, we pointed out that today's browsers lack support to securely use `https` to verifiably secure `iframes`. Without the help of additional tools or extensions their certificates are not accessible to the regular user of those web sites.

Apart from the four approaches presented in this paper in detail, it is important to mention two streams of work done in the field of web browsers. We will shortly look at one other extensions for the browser that deal with visually highlight privacy policy verification outcome. PrivacyBird [CMU02] is an existing extension for Internet Explorer. Secondly, we will discuss two design studies [Faa07b, Gla07] for Firefox. This helps to better understand the property, as they deal with the same problem of visual highlighting. The PrivacyBird [CMU02] extension for example. PrivacyBird automatically parses and then judges a web site's privacy policy. The policy must be expressed using the machine-readable P3P language [W3C02], a W3C standard to describe privacy policies. It is a security relevant

extension that adds a pictogram of a bird into the Status Bar (of Internet Explorer). If it displays a green bird with an extra red exclamation point this indicates that the site generally matches a users preferences but contains embedded content that does not match or does not have a privacy policy. A user study [CGA06] showed that this constant feedback is considered helpful and that users liked to be always informed. This kind of visual information bears one basic problem, the above mentioned "green bird with exclamation" does not need to be linked to a specific content. In other words, PrivacyBird's verification result is for a server or service, so its fine that the visual covers the actual domain the browser is showing. The problems start if parts of the shown web site come from different servers and have different privacy policies, PrivacyBird does not cater for this. Providing a link between verification outcome and verified content must be done most unobtrusively, but still securely and hard to spoof. This exact problem, is still left unresolved by browser vendors when regarding the security error messages related to non TLS content. For example, the browser displays an error message when a website, which is delivered through a TLS tunnel, tries to embed content, such as an image, from a non-https-secured source. This error message is not content specific, but generic. It does not tell the user which content and thus which part of the website is touched by the decision at hand.

The second example discusses several design studies that show how data, for example street adresses, shall be highlighted so that the user could activate other applications using the detected data. Kaply's Firefox extension Operator [Kap07] gives a good starting point, while Alex Faaborg's design studies [Faa07b], as mentioned above, show how hard visual highlighting (of not security critical) information already is. Another concept is given by Dimitri Glazkov [Gla07] is shown in Figure 5b. He provides markers on the left side of the browser right next to the Content Area, at the same hight as the regarding content. It should be pointed out, that both ideas are not considering security issues, like spoofing. This is rather concerning, but might change if use cases like the "Verified by Visa" are used broadly and highlight the need for a secure and usable solution. We postulate that such a security relevant functionality must be offered as a core browser functionality. We have learned from our own Firefox extension prototyping for other research projects that it is otherwise too cumbersome or even impossible to implement functions like certificate checking or certificate trust management in an extension. So, even though management is already done in the browser for the X509 certificates used to authenticate also the "hidden" TLS tunnels, additional components can not easily use them. We hope that concepts like the Operator Extension can be integrated and extended to the needed security functionality.

5 Conclusion and Future Work

Soon content reuse, already common for non-security sensitive content on the Web 2.0, will reach out into critical and sensitive areas, like credit card details during online shopping. Browsers still lack the ability to show the user anything else than the outer most's TLS certificate verification result. We argue that, based on the analysis of the detailed examples and existing tentatives to solve the problem, this must be done at the web browser's core. No solutions yet exist within the browsers themselves. We postulate that each

TLS session's certificate and its verification result shall be made differentiable and visible within the browser to allow users to verify the content that originated from that TLS session. Following our classification, we propose the following steps to reach the goal:
- Standardize the marking for content parts that are integer and authentic
- Provide visuals that one or several secured contents are within the actual content area.
- Highlight secured content visually within the content area; must be hard to spoof and visually unobtrusive.
- Integrate into the browser: no extensions, not external standalone.
- Provide an open solution: allowing public screening and academic or comercial development of new services.

This article hopes to further raise the awareness for this problem in web browser security and to stimulate the future integration of the above mentioned points into the development of web browsers and web content creation and management systems.

References

[BLHL01] T. Berners-Lee, J. Handler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[BP08] Bastian Braun and Henrich C. Pöhls. Authenticity: The missing link in the social semantic web. In *Digitale Soziale Netze - GI 2008*, 2008.

[CGA06] Lorrie Faith Cranor, Praveen Guduru, and Manjula Arjula. User interfaces for privacy agents. *ACM Trans. Comput.-Hum. Interact.*, 13(2):135–178, 2006.

[CMU02] CMU Usable Privacy and Security Laboratory at Carnegie Mellon University. Privacy Bird. `www.privacybird.org/`, 2002.

[Com05] Comodo Inc. COMODO Verification Engine User Guide. `www.vengine.com/pdfs/userguide.pdf`, 2005.

[com10a] comdirect. Ab Ende April kaufen Sie noch sicherer online ein. `http://www.comdirect.de/pbl/cms/cms/services/pages/s2/sicherheit/was_leisten/verified_by_visa/5663_se_verified_by_visa.html`, Mar. 2010.

[Com10b] Comodo Inc. What are CVCs? - Content Verification Certificates. `www.instantssl.com/ssl-certificate-products/content-verification.html`, 2010.

[CW02] C.-H. Chi and Y. Wu. An XML-Based Data Integrity Service Model for Web Intermediaries. In *Workshop on Web Content Caching and Distribution (WCW)*, 2002.

[DR06] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), April 2006. Updated by RFC 4366.

[Faa07a] Alex Faaborg. Anatomy of Firefox's Chrome. `http://people.mozilla.com/~faaborg/files/20070204-detectionUI/anatomyChrome.jpg_large.jpg`, 2007.

[Faa07b] Alex Faaborg. The User Interface of Microformat Detection. `blog.mozilla.com/faaborg/2007/02/04/`, February 2007.

[FSD+03] B. J. Fogg, Cathy Soohoo, David R. Danielson, Leslie Marable, Julianne Stanford, and Ellen R. Tauber. How do users evaluate the credibility of Web sites?: a study with over 2,500 participants. In *DUX '03: Proceedings of the 2003 conference on Designing for user experiences*, pages 1–15, New York, NY, USA, 2003. ACM Press.

[Gla07] Dimitri Glazkov. Margin Marks UI Concept. `http://glazkov.com/2007/09/04/margin-marks/`, Sept. 2007.

[Gol05] D. Gollmann. *Computer Security 2e*. John Wiley & Sons, 2005.

[gre10] Greasemonkey Firefox Extension. `www.greasespot.net`, Feb. 2010.

[HPFS02] R. Housley, W. Polk, W. Ford, and D. Solo. RFC 3280 - Internet X.509 PKI Certificate and Certificate Revocation List (CRL) Profile, Apr. 2002.

[Kah06] Usability and document authentication issues. `www.w3.org/2005/Security/usability-ws/papers/12-kahan-usability-and-doc-auth/`, March 2006.

[Kap07] Mike Kaply. Operator. `www.kaply.com/weblog/operator/`, Oct. 2007.

[MA10] Steven J. Murdoch and Ross Anderson. Verifed by Visa and MasterCard SecureCode: or, How Not to Design Authentication. In *Financial Cryptography and Data Security '10*, Jan. 2010.

[Mic06] Microformats. website. `www.microformats.org`, Aug. 2006.

[Ope06] OpenSSL Project. *OpenSSL*. OpenSSL, Dec 2006.

[PÖ8] H. C. Pöhls. ConCert: Content Revocation using Certificates. In *Sicherheit 2008*, volume 128 of *GI-Edition Lecture Notes in Informatics (LNI)*, pages 149–162, Saarbrücken, Germany, April 2008. GI.

[Pro07] ProgrammableWeb.com. Mashup Matrix. `programmableweb.com/matrix`, Aug 2007.

[QSM07] M. Quasthoff, H. Sack, and Ch. Meinel. Why HTTPS is Not Enough – A Signature-Based Architecture for Trusted Content on the Social Web. In *IEEE / WIC /ACM Int. Conf. on Web Intelligence*, 2007.

[RM02] Doug Ransom and Peter V. Mikhalenko. Discussion on "XHTML adoption curve". `lists.xml.org/archives/xml-dev/200204/msg00559.html`, Apr. 2002.

[Vis] Visa USA. Verified by Visa. `https://usa.visa.com/personal/security/vbv/index.html`.

[Vis05] Visa EUROPE. Verified by Visa: Merchant Deployment Best Practices Factsheet. `www.visaeurope.com/documents/vbv/verifiedbyvisa_merchantdeploymentbestpractices.pdf`, 2005.

[Vis06] Visa USA. Verified by Visa System Overview External Version 1.0.2. `https://partnernetwork.visa.com/vpn/global/retrieve_document.do?documentRetrievalId=119`, Dec. 2006.

[W3C02] W3C. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. `www.w3.org/TR/P3P/`, Apr. 2002.

[W3C06] W3C. RDFa Primer. `www.w3.org/TR/xhtml-rdfa-primer`, May 2006.

WatchCop – Safer Software Execution through Hardware/Software Co-Design

Christian Ristig, René Fritzsche, Christian Siemers

Department of Computer Science, Clausthal University of Technology,
Julius-Albert-Straße 4, D-38678 Clausthal-Zellerfeld, Germany
christian.ristig@tu-clausthal.de
rene.fritzsche@tu-clausthal.de
christian.siemers@tu-clausthal.de

Abstract: This paper introduces a novel approach to support runtime execution monitoring with nearly no negative impact on runtime. The monitoring is configurable to application-specific requirements and supports real-time behaviour during development and runtime. An extension of the basic approach for monitoring contains watchdog capabilities for a fine-grained observation of runtime activities and a two-level support of recovery from program failures. It is shown how these capabilities may be used to ensure safe software execution.

1. INTRODUCTION

Safe software execution is a major requirement in many application areas. As nearly all applications in automation technology are software-based, and the software is executed through a von-Neumann-microprocessor, guaranteeing safe execution becomes a major issue also inside this application class.

To face this issue, most approaches address the development process. This resulted in models for the software process like the V-model and its extension V-model XT, in equivalent test models as part of the software development process and in model-based development. The motivation for model-based approaches is mostly based on complexity handling and on handling safety issues, as software-generated source code might follow coding rules and is said to be much more reliable than its hand-coded counterpart.

Nevertheless software-based systems remain complex. Beside their algorithmic complexity, where functions, methods and data structures interfere, time complexity is even harder to manage, as language support is still missing. Usually timing issues are mapped to the operating system, which is a second level of programmability often called “programming in the large”. The operating system will be able to manage tasks in a coarse-grained way, but the drawbacks are that single task behaviour is not influenced in a fine-grained manner, and that monitoring of task behaviour might be time consuming. As the operating system is just software being executed in most cases on the same processor, time consuming monitoring will negatively influence runtime behaviour.

In summary, safe software execution faces at least two dimensions: timing and algorithmic. For the algorithmic dimension, several approaches like lock-step execution of at least two processors or triple module redundancy (TMR) are well-known. They are quite expensive in using computational resources and energy, but required fault coverage is obtained by them.

1.1 The Organisation of This Paper

The approach in this paper addresses runtime issues. A coprocessor is used to monitor and/or control the runtime behaviour of a set of tasks. This coprocessor is configured by the main processor at initialisation time and later fed by runtime information also provided by the main processor, but the build-in control algorithms work independently from main processor after configuration. As the coprocessor is kept small and simple, the additional overhead in silicon area and energy consumption remains small.

Therefore the coprocessor is used for monitoring and timing issues instead of an operating system. The coprocessor consists in its basic architecture of a set of registers, a monitoring memory of arbitrary size, limited algorithmic capacities and a control unit. Section 2.1 gives a more detailed description of the underlying hardware architecture.

The purpose of the basic architecture is to monitor the runtime behaviour of the program. The monitored data give an exact view of the runtime and may also be used for worst-case-execution-time analysis. This basic approach can be extended by some configurable comparing units to automatically monitor execution during runtime and to signal the processor critical situations in a fine-grained manner. This extension works like an advanced multi-watchdog and is responsible for the name of this approach: WatchCop, a *watch*dog *cop*rocessor. Section 2.2 shows more details of this extension.

Section 3 discusses the hardware/software interface between processor and coprocessor. This interface is kept quite simple and uses only a small set of instructions to couple both units. More important as the number of coprocessor instructions is the usage of these instructions in the monitored program. Once the coprocessor is initialised, the instructions for monitoring are very rare in the binary code. Every label selected to monitor the program flow is translated into one cop2-instruction with specific parameter resulting in one execution cycle inside a RISC-based architecture.

Section 3.2 discusses the use of the coprocessor instructions for monitoring software execution in detail. It should be mentioned that this approach is not non-invasive therefore negative runtime impact can be minimised but won't be zero. We followed the clearly structured coprocessor approach as a good balance between minimum impact on runtime and necessary changes in the microarchitecture.

Section 4 discusses some applications of the WatchCop for safe software execution, including some classical approaches like challenge/response, and finally section 5 gives a summary and an outlook for future work.

1.2 Related Work

The observation of program execution is an issue since several decades. The authors in [Ma88] give a very good survey of approaches in the 1980s to take care of program execution. These early works are focussed on memory access errors – now a part of a memory protection system – and on illegal opcodes – inside modern architectures a task of the exception system. Most of the surveyed approaches in [Ma88] have found their way into the microprocessor and closely coupled units.

The approach in [Na05] on the other side addresses the actual requirements for reliability and real-time execution. The authors use a formal method to extract models from source code written in Ada for verification. The information obtained by verification are then used to arm a specifically designed chip called SafetyChip. Real-time monitoring is performed by observing the communication between application and run-time kernel, and any deviance from the predefined behaviour is signalled to the system.

The authors in [Sa90] use an approach to implement control functionality within a coprocessor for any program part, not only specific parts. The main constraint of this approach is to minimise impact on program execution time. To ensure this, the approach in [Sa90] as well as other approaches ([Ra05], [Fa08] and [Ir06]) use checksums and trace functionality to ensure or monitor the correct program execution in the sense of arithmetically correct program flows but not in timely manner.

In contrast to the briefly discussed approaches, WatchCop is used to monitor and control program execution concerning its timing behaviour first to ensure real-time functionality and secondly to ensure correct program execution at all. The main constraints are to minimise the impact on runtime – specifically by reducing the amount of additional instructions in the program flow – and the impact on processor microarchitecture – specifically by avoiding any impact on the architecture inside the execution pipeline.

Consequently the WatchCop approach shows most similarity to the work published in [Na05]. In contrast to that, WatchCop may observe any program flow, even if a run-time kernel is not available, and a formal execution model is not required to enhance the design with monitoring capacity. Nevertheless such a model is very useful, and concerning WatchCop, we are following the way to enhance a language model and a compiler to generate the monitoring information semi-automatically.

Compared to all previously discussed approaches, WatchCop shows the following differences:

1. WatchCop monitors and controls the timing behaviour, not just the control flow behaviour of the program and couples runtime and real-time.

2. While maintaining a minimum of negative impact on runtime behaviour as well as the original processor architecture, WatchCop may monitor all kinds of program flows with no restriction.

2. Hardware Architecture

2.1 Basic Architecture

Figure 1 shows the basic architecture of WatchCop. A similar approach to [Sa90] is used in the sense that all functionality is implemented within a coprocessor with a minimized impact on program execution time.

The architecture shows two major parts, containing the basic part and the extended part. The basic part carries all necessary parts for monitoring applications including an interface to obtain the monitored values. This addresses operating system issues in the sense that actual runtime data are available for use in scheduling decisions.

The extended part on the other side contains direct control mechanisms and all necessary interfaces. This was designed to directly control runtime behaviour and to recover from exceptional operating conditions like deadlocks. This may be used to control program flow as well as to support operating system capabilities.

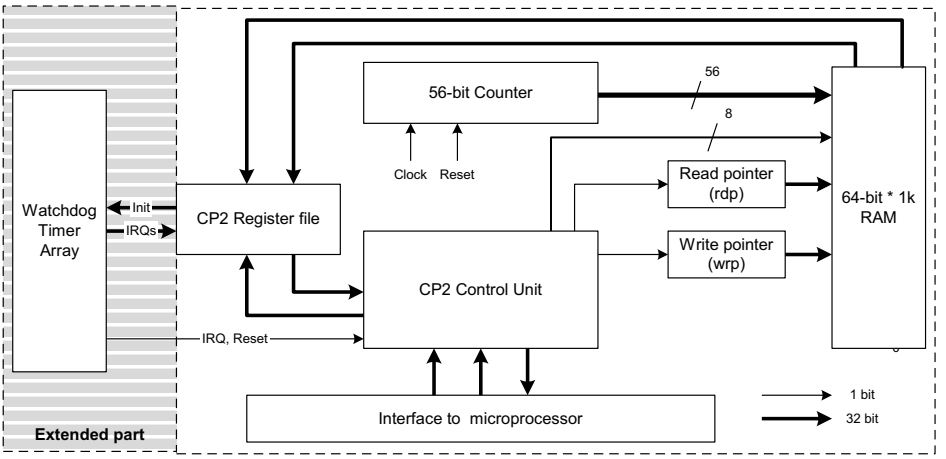


Figure 1. Basic/extended (left) microarchitecture of WatchCop

The shown implementation in figure 1 (without the shaded extended part) contains a free running counter of arbitrary width (here 56 bit), a set of 4 coprocessor registers, a read and a write pointer to manage the ring buffer for storage of monitoring events, the monitoring memory of arbitrary size (here 1K depth with 64 bit width) and the control unit. While the arbitrary values were chosen for long-running counter implementation without overflow, maintaining the other 8 bit for label identification and for easy interfacing the internal 64 bit with a 32-bit microprocessor, these values may change frequently.

The basic approach works as a monitoring system. If the processor fetches a cop2-instruction with according arguments, this instruction is directed to the coprocessor and executed there. In detail, 8 bit of the argument are decoded as label number and stored together with the actual 56-bit counter value inside the RAM. This implements the monitoring functionality, while the read and write pointers manage the access using specific cop2 instructions.

2.2 Extended Architecture to implement Watchdog Functionality

The monitoring capacities of the basic architecture are extended by some watchdog functionality (see fig. 1 including the extended part). For this purpose, a set of configurable counters is integrated in the architecture. Each counter may be configured independently to a start value and to a label number. During program execution, each cop2-instruction containing this label (ref. section 3) as argument resets the corresponding counter to the start value.

On the other side, the counter is continuously decremented each clock cycle, and if an underflow occurs, a signal to the processor is set to active, because a defined runtime condition was not met. This is well-known watchdog functionality with the extension that several points in program flow might be used to monitor the runtime behaviour.

Different to known watchdog implementations, the severity level of this watchdog alert will be configurable and may change between two successive events. WatchCop uses at least two signalling lines to the processor, one for interrupt request, the other for reset. Therefore, a watchdog timer underflow can initialize an interrupt, e.g. if this is the first time, and may reset the processor, if this happens again. Applications of this feature are discussed in section 4.

3 Hardware/Software Interface

The interface between WatchCop and the application software contains three parts (see also figure 2): The coprocessor instructions for configuration during initialisation phase and for monitoring, the signalling part addressing interrupt service resources of the processor, and the data exchange for initialization and monitoring evaluation. This reflects to the four main parts of software interface between main program and monitoring: Initialization, monitoring during program execution, reaction on event signalling and monitoring evaluation.

During initialisation, the main processor may set several register to arbitrary values for configuring the coprocessor. After this period, the coprocessor normally works autonomously without executing an instruction flow provided by the main processor. Only few instructions are inserted into normal program flow to synchronise the coprocessor with instruction flow of the main processor. This interface was designed to reduce the impact on program execution as much as possible while maintaining the coprocessor approach to preserve the main processor from redesign, as mentioned before.

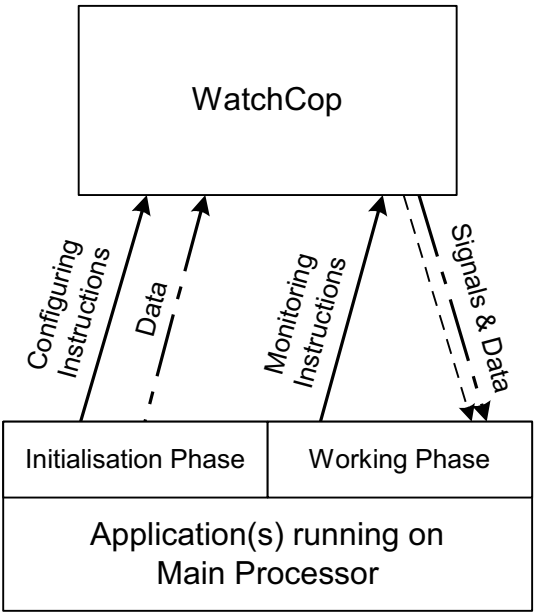


Figure 2. Interfaces between WatchCop and main processor

3.1 Hardware Interface between WatchCop and Main Processor

The hardware interface to/from coprocessor and main processor consists of a bus system for instructions of arbitrary size, e.g. 32 bit width, optionally of a data bus system of arbitrary size, and of few signalling lines. While the instruction bus system is mandatory, the same lines could be used for transmitting additional data, if they are capable of bidirectional data transfer, when data transfer from WatchCop to main processor is desired. The latter could be required for transmitting monitored values from WatchCop to main processor for further evaluation.

The signalling part was designed to immediately inform the main processor about events like watchdog timer underflow. During first project evaluation it appeared to be desirable to implement a signalling system with more than one level, and a two-level system was chosen. The 1st and 2nd level signalling is mapped on interrupt requests of arbitrary priority. In most cases, the 2nd level underflow of one watchdog timer will generate a highest priority interrupt request, in many cases of non-maskable type.

3.2 Instruction Set

The chosen instructions were picked from the standard instruction-set of MIPS-based microprocessors in order to be able to use unmodified development tools and compilers. Due to this fact language-checkers and optimizing strategies may still be applicable after inserting our monitor-instructions to the code. Three commands are necessary for using all WatchCop functionality:

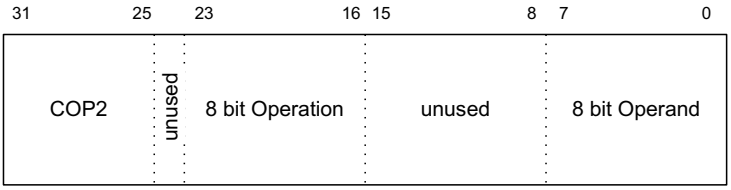


Figure 3. Binary format of COP2-instructions in WatchCop

- The mtc2-instruction (move to coprocessor 2 register) copies data from processor register to a specified coprocessor register. This may be used for initialization of some functions, e.g. timeout values for the watchdog timer or all kinds of configuration setup.

- The mfc2-instruction (move from coprocessor 2 register) copies data from coprocessor register to processor register. This will be frequently used, e.g. for reading status register or accessing monitored values in the event list.

- The cop2-instruction (coprocessor 2) is used as general purpose format for all other instructions. This format normally holds some bits for free use, in our implementation 25. Figure 3 shows the usage of the bits: Bits 23 through 16 contain a function number, bits 7 through 0 a label number.

Operation	Operand	Descrpition
No_Operation	None	Do nothing
Reset_Monitoring	None	Monitoring is stopped, all pointer are reset
Start_Monitoring	None	Monitoring is (re-)started, from this point on every Store_Event is recorded
Stop_Monitoring	None	Monitoring is stopped, but data are maintained
Set_Read_Pointer	None	Sets the read pointer to first/last value
Get_value_from_list	None	Gets the value from the list in RAM pointed to by the read pointer.
Increment/decrement read_pointer	None	The read pointer is incremented/decremented (in circular way)
Store_Event	Label (8 bit)	The actual clock counter is stored in RAM including the label
Set_Timeout_Value	Watchdog Timer (8 bit)	The 56-bit value in CP2 R2/R3 is copied into the according watchdog timer

Table 1. Used subformats for some cop2-instruction

The list of actually implemented functions for cop2-instruction is shown in table 1. Actually this may be extended for additional functions if required, because the instruction set space is not densely used in the current implementation..

4 Applications with Use of WatchCop

4.1 Applications Using the Basic Architecture

The first major application that is supported by WatchCop is the development, specifically the test of real-time applications. For this purpose, the critical paths or parts to be tested will be instrumented by cop2-instructions with monitoring. Each time the program runs through this point a label containing the 56-bit counter value is written into the private RAM of the WatchCop, and may afterwards be read and analyzed for real-time behaviour.

As 256 different labels may be inserted into assembler code, some interference like “if the program runs through label xx, timing constraint on label yy is not met” between paths through the program may also be observed. This leads to a post-run analysis with exact timing labels with very low impact on the runtime (by the additional instructions). Using this basic feature during runtime is also possible and a good monitor to prove real-time behaviour, even after problems have occurred, or as information base for operating system decisions.

This approach was successfully tested during testing real-time applications using several threads to monitor specifically inter-thread communications. These communications, either implemented in blocking (waiting) or non-blocking fashion, show runtime-specific and even data-specific behaviour. Therefore the observation of realistic behaviour might be essential to consider modifications to the design, and it is hard work to realise these realistic constraints only by simulation.

4.2 Applications Using the Extended Architecture

The full power of WatchCop is provided by the extended features. This is due to the fact that WatchCop has now a direct feedback channel by using signalling lines for interrupt request or reset, and therefore watchdog, challenge/response-functionality and extended support for scheduling may be included into the application.

4.2.1 n-Level-Reaction Scheme for Watchdog Events

First, the watchdog part inside can be used for a classical watchdog functionality. Up to 8 timer in the actual implementation can be configured as watchdog timer and may be started. Specific cop2-instructions will reset the timer to its initial value, if the program executes this instruction. Therefore, WatchCop supports 8 different watchdog activities during runtime.

We implemented a configurable 3-level reaction scheme. The first underflow of any watchdog timer results in an interrupt request, a highly prioritized interrupt/trap or in a reset. The level of reaction is configurable, and therefore the 'classical' functionality of a watchdog might be configured. After requesting the interrupt, an additional time is loaded into the watchdog timer register. This time must be explicitly configured during initialization, otherwise it is set to '0', and the next level of reaction is instantaneously invoked.

If the processor reacts during that additional reaction time, the system appears to work properly, and the watchdog is reloaded with the original value. If not, the second reaction level is started and results in a trap (non-maskable interrupt) or a reset (if additional time is configured to 0). The third level is always a reset, but in other implementations even more level might be integrated. In this case the processor has run out of control, and the application must be restarted completely.

This may be used for additional functionality in the following way. If the processor receives the 1st-level interrupt request, the reaction inside the interrupt service routine (ISR) might be the reset of the watchdog timer. Nevertheless it is still not sure that the processor does not stay in a deadlock inside the main-program, because the reaction is performed inside the ISR.

To avoid this misinterpretation and to introduce a challenge/response-function, it is proposed to include the reaction not into the ISR but into normal program operation. The interrupt initialised by the coprocessor results into a software event which in turn must be handled inside main program (see fig. 4), and algorithmic capacities may also be included. In this case, the interrupt is interpreted as challenge, and the software of the processor responses.

```
void interrupt vISR()
{
   ...
   switch( cp2IRQStatus )
   {
      case WATCHDOG_LEVEL_1:
         setEvent( EVENT_WATCHDOG_L1 );
         break;
   ...
   }
}
```

```
void main()
{
   int tempEvent;
   ...
   while( 1 )
   {
      getEvent( &tempEvent )
      switch( tempEvent )
      {
         case EVENT_WATCHDOG_L1:
            resetWatchdogTimer();
            break;
   ...
   }
}
```

Figure 4. Code fragment to integrate challenge/response-functionality

4.2.2 Scheduling Support by WatchCop

Scheduling is basically supported by measuring time differences inside WatchCop. If the microprocessor system does not use an operating system, the possibility of using a cooperative approach for threads with an application-own scheduling might be used. If all cooperative threads work perfectly well and do not block, everything is okay, and the system works with high efficiency. All overhead from an operating system is omitted, but multithreading is still supported.

If the application is not as perfect as described, WatchCop can be used to obtain a good compromise between cooperative and preemptive scheduling. The basic principle can be still cooperative, but all threads are individually controlled by the n-level watchdog mechanism. In this case the planned reaction upon a watchdog timer underflow could be the cancelling of the blocking thread (which is definitely a serious issue) and continuing work with next event or next thread.

This scheduling was successfully implemented, and we call it forced-cooperative due to the fact that preemptive scheduling only occurs when exceptional operating conditions are observed and threads block. This is comfortably supported by WatchCop, even if the forced scheduling times vary from thread to thread.

Furthermore, the TaskCombining approach as described in [Si05] or the TaskPair approach in [Ge01], both are supported. In this case a twofold reaction scheme for real-time applications is proposed. If reaction time is tide, at least one of a set of tasks is not executed but reacts with an emergency value. This reaction system is known as precise time, imprecise logic.

As shown in [Si05], the presence for timer support is essential for efficient implementation. In this case, the WatchCop may support with the built-in timer, because they can support the processor with according interrupts. One watchdog timer is used for each task inside the task-combining-system and is set to a value close to the deadline of this task with a reaction value of “trap”. If this value is reached without reset before, then the task is not able to react precisely, and the imprecise value (which must be computed earlier as discussed in [Si05]) is used. As long as this task has not started to perform computation and all others are ready or close to readiness, no time is wasted.

5 Summary and Outlook

We presented a system consisting of a coprocessor specialized to monitor execution time as well as a rudimentary software interface to use this system. The purpose is to implement a reaction system for deadlocks and other software errors to keep the software-based system operating.

This WatchCop is implemented as softcore using a MIPS32-compatible processor with coprocessor interface. The implementation of the coprocessor uses roughly 20% of the processor, with most of the silicon area is used for storing time/label values in the RAM. These values are the base for detecting time failures or weaknesses and for correction. As simple example, an operating system providing preemptive scheduling using the WatchDog capabilities could be implemented very easy and fast.

The next step will be to integrate the coprocessor into high level languages and to establish high-level language support. The roadmap to do this is to use normal C and to enhance this with special comments that are interpreted by a pre-compiler. The pre-compiler extracts the timing constraint from source code and generates a user constraint file, which is then interpreted for code generation.

This approach is first published in [Fr08] and [Fr10], and WatchCop is the ideal hardware architecture to execute the necessary monitoring and measurement for inter-thread scheduling. The formal models in [Na05], which are essential for defining the correct functionality of the proposed Safety Chip, are here replaced by language constructs.

While the integration of high-level language support and WatchCop into a development tool and framework is the actual next step, the plan is to automatically generate a multithreading system from such enhanced C-code including WatchDog capabilities to support multithreading and monitor the complete system.

References

[Fa08] N. Farazmand, M. Fazeli, S.G. Miremadi, “FEDC: Control Flow Error Detection and Correction for embedded systems without program interruption”. *ares, pp.33-38, 2008 Third International Conference on Availability, Reliability and Security, 2008.*

[Fr08] R. Fritzsche, G.Kemnitz, C. Siemers, “TEC: Time-Enhanced C – Erweiterung einer imperativen Programmiersprache um Zeitvorgaben”. *Tagungsband Embedded Software Engineering, S. 427-430, Sindelfingen, Germany, Dezember 2008 (in German language).*

[Fr10] R. Fritzsche, C. Siemers, “Scheduling of Tme-Enhanced C (TEC)”. *Accepted for publication in Proceedings of World Automation Conference 2010 (WAC 2010), Kobe, Japan, September 2010.*

[Ge01] M. Gergeleit, “A Monitoring-based Approach to Object-Oriented Real-Time Computing,” *Otto-von-Guericke-Universität Magdeburg, Universitätsbibliothek, 2001, http://diglib.uni-magdeburg.de/Dissertationen/2001/margergeleit.pdf*

[Ir06] K. Irrgang, J. Braunes, R.G. Spallek, S. Weisse, T. Gröger, “A new Concept for Efficient Use of Complex On-Chip Debug Solutions in SOC based Systems”. *Embedded World 2006 Conference, pp. 215-223 (2006). Franzis Verlag, Poing, 2006, ISBN 3-7723-0143-6.*

[Ma88] A. Mahmood, E.J. McCluskey, “Concurrent Error DetectionUsing Watchdog Processors – A Survey”. *IEEE Transactions on Computers 37(2), pp. 160-174 (1988).*

[Na05] G. Naeser, L. Asplund, J. Furunäs, “Safety Chip – A Time Monitoring and Policing Device”. *SIGAda 05, pp. 63–68 (2005).*

[Ra04] A. Rajabzadeh, M. Mohandespour, G. Miremadi, "Error Detection Enhancement in COTS Superscalar Processors with Event Monitoring Features". *prdc, pp.49-54, 10th Pacific Rim International Symposium on Dependable Computing (PRDC'04), 2004*

[Ra05] A. Rajabzadeh, S.G. Miremadi, "A Hardware Approach to Concurrent Error Detection Capability Enhancement in COTS Processors," *prdc, pp.83-90, 11th Pacific Rim International Symposium on Dependable Computing (PRDC'05), 2005*

[Sa90] N.R. Saxena, E.J. McCluskey, “Control-Flow Checking Using Watchdog Assists and Extended-Precision Checksums”. *IEEE Transactions on Computers 39(4), pp. 554-559 (1990).*

[Si05] C. Siemers, R. Falsett, R. Seyer, K. Ecker, “Reliable Event-Triggered Systems for Mechatronic Applications,” ”. *The Journal of Systems and Software 77*, *Elsevier, 2005, pp. 17–26.*

A Fuzzy Model for IT Security Investments

Guido Schryen

schryen@winfor.rwth-aachen.de

Abstract: This paper presents a fuzzy set based decision support model for taking uncertainty into account when making security investment decisions for distributed systems. The proposed model is complementary to probabilistic approaches and useful in situations where probabilistic information is either unavailable or not appropriate to reliably predict future conditions. We first present the specification of a formal security language that allows to specify under which conditions a distributed system is protected against security violations. We show that each term of the security language can be transformed into an equivalent propositional logic term. Then we use propositional logic terms to define a fuzzy set based decision model. This optimization model incorporates uncertainty with regard to the impact of investments on the achieved security levels of components of the distributed system. The model also accounts for budget and security constraints, in order to be applicable in practice.

1 Introduction

Emerging digital environments and infrastructures have rapidly generated new ways and services of communication, information sharing, and resource utilization for individuals, organizations, and societies in past years. For example, it has become common for individuals to use security services, such as *I2P Anonymous Network* and *TOR*. Organizations have started to explore the opportunities of web services, including storage services (e.g., *Amazon Simple Storage Service*) and computing services (e.g., Microsoft's Azure Services Platform and Google App Engine). While the aforementioned services are realized with cloud computing, services can also be requested from multiple administrative domains (*grid computing*). Even whole societies are involved in scenarios with shared information and transaction processing, as political elections with electronic voting systems show.

What all these services have in common is that some kind of distributed information processing and/or information sharing occurs, across private, organizational, or national boundaries. Often, consumers of these services have no control over their data, and they need to trust service providers not to violate their security policies. For example, scientific computation results can be modified or provided to third parties. In some cases, organizational, legal, and/or technical countermeasures have been taken in order to prevent or to mitigate the consequences of data abuse. For example, in Internet voting the separation of duties is quite common in order to realize the separation of voter's identity and his/her vote. In such cases, the abuse of data by a single party (insider abuse) and the compromise of systems by attackers (outsider abuse) do not disclose confidential information.

However, what happens when multiple parties maliciously cooperate and join their information, or when multiple system components are compromised jointly by attackers? This leads to scenarios where a voter's ID can be assigned to his/her vote, where the identity of a user is disclosed through the cooperation of parties of an anonymity mix net, etc. Consequently, when security investments in distributed systems are planned, the questions arise of (1) how important the security of particular system components is, and (2) how much should be invested in which component to increase the overall security of the distributed system. Thereby, we focus on the ex ante security assessment of distributed systems, and the support of security investment decision makers.

Beyond the challenge to address the aforementioned interdependencies between system components, decision makers also face budget constraints and various sources of uncertainty. Unfortunately, uncertainty is often not probabilistic so that the application of probabilistic approaches is of limited effectiveness. We thus draw on fuzzy set theory, which is a valuable uncertainty theory in the absence of probabilities and in the presence of subjective assessments.

The main purpose of this paper is to present a novel fuzzy set based decision support model for security investment decision makers. From the methodological perspective, we formally derive the decision model by proposing a formal security language and by applying propositional logic, decision theory, and fuzzy set theory. We further draw on computational complexity theory to analyze the complexity of the model.

The remainder of this paper is structured as follows: Section 2 presents related work. In Section 3, we describe our research framework. Section 4 proposes the formal security language, demonstrates its applicability, and shows how resilience terms of the security language can be mapped on propositional logic terms. Section 5 provides a brief introduction into uncertainty modeling and fuzzy set theory. In Section 6, the fuzzy decision support model is proposed and analyzed. Section 7 discusses implications and shows opportunities for further research.

2 Related work

The economics of information security investments has been analyzed in the literature at both the ex post level and the ex ante level (decision making). An example of the former perspective is the NIST *Performance Measurement Guide for Information Security* [NIS08], which focuses on ex post security measures. As the focus of this paper lies on decision making, we concentrate our overview on papers on the ex ante analysis of information security investments.

In their survey of economic approaches for security metrics, [BN08] analyze the literature through a methodological lens and identify two main areas of research, where one has its roots in investment and decision theory and is mainly pursued in the field of information technology-oriented business administration, and the other area of research has ancestors in micro-economics and deals with market concepts to gather security-relevant information. We adopt a different perspective and focus on theoretical approaches used to

address uncertainty in security investment decision making. Unsurprisingly, the literature is very much focused on probabilistic approaches and often adopts the risk-based perspective. [GL02] present an economic model that determines the optimal amount to invest to protect a given set of information and that uses probabilities that attacks are successful. [GJC09] propose metrics for measuring the price of uncertainty due to the departure from the payoff-optimal security outcomes under complete information, and assume that agents face randomly drawn probabilities of being subject to direct attacks. [GCC08] apply game theory to study how economic agents invest into security in different economic environments, and they assume that attacks arrive with a probability that remains constant over time. [CRY08] consider the decision-making problem of a firm when attack probabilities are externally given. In their approach to derive implications for security investment strategies based on attackers' decisions, [CN06] draw on the probability of (attackers') success given an amount of effort put into attacking a given target. [HHB06] propose an economic model that considers simultaneous attacks from multiple external agents with distinct characteristics, and derive optimal investments based on the principle of benefit maximization. In their model they draw on security breach probabilities.

However, there are also dissenting voices, which doubt the appropriateness of using probabilistic approaches. For example, [WCR05] argue that risk-driven decision models are limited due to the difficulty of reliably estimating the potential losses from security breaches and the probability of these breaches. [HN10] find that risk assessment methods found in the literature tend to underestimate the risks associated with large-impact, hard-to-predict, and rare events.

We found two papers that suggest to apply fuzzy sets in the context of security investment decisions. [Lee03] presents a simple model that uses linguistic variables to represent criteria upon which investment decisions are made. [KSST09] suggest to use fuzzy sets in the context of evaluation processes. In their paper, fuzzy sets are used to express the extent with which security measures are implemented in an organization.

3 Research framework

Our approach (see Figure 1) assumes that the structure of a distributed system is known. We draw on this structure to derive a formal resilience term, which specifies which components and/or groups of components need to be secure with regard to a particular security requirement r (e.g. confidentiality, anonymity) so that the overall distributed system is secure with regard to r. The specification of r is important, because different security requirements can lead to different resilience terms. For example, in a system that implements a mixnet that routes messages sequentially through a set N of n anonymizing nodes, each node must be secure with regard to availability (n out of N), while only one node needs to be secure with regard to achieving anonymity (1 out of N). The formal security language that we propose in this paper draws on [HKS00], who use secret shares [BK05] and the concept that k out of n entities are required for revealing a secrecy. We adopt and adapt this concept, and we say: *"k out of N entities must be secure"*. In contrast to the aforementioned papers, which regard entities/components as homogeneous, we account

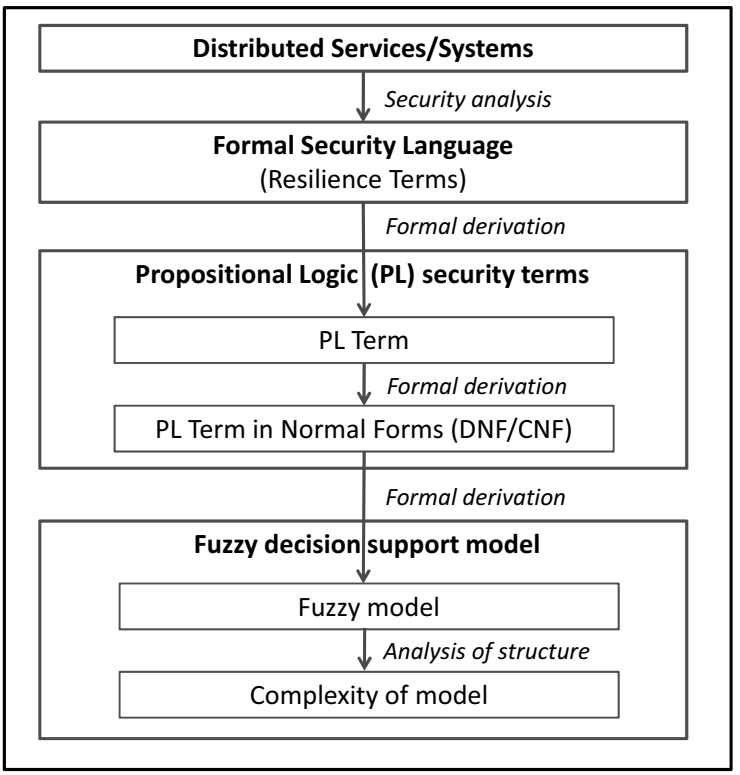


Figure 1: Research framework

for heterogeneity of entities by explicitly itemizing them in the set N.

While resilience terms are a useful representation of required security properties of a distributed system, they are less appropriate for analyzing systems with regard to weak points and strong points, and for making security investment decisions. We show that each resilience term can be mapped on a propositional logic term such that both terms are semantically equivalent, and we show that converting propositional logic terms into normal forms, such as the *conjunctive normal form* (CNF), is a useful way to identify such weak and strong points.

Accounting for the fact that security investment decision makers need to consider various sources of uncertainty and different types of constraints, we suggest a fuzzy decision support model. The goal function of this model is derived from the CNF representation of the particular resilience term, which links the introductory, theoretical parts of this paper with the proposed decision model. We finally analyze the structure of the decision model, and we discuss types of required data.

4 Formal security language and propositional logic terms

4.1 Formal security language

As our formal security language describes required security properties of distributed systems, we first define distributed systems: A distributed system is either an "atomic system" or is composed of other (sub)systems. We define a system as "atomic" if it contains only (atomic) components that are not being split any further. These components can be persons, computers, or even organizational units.

The definition of the security language (resilience terms) in terms of syntax and semantics follows the inductive definition of systems and is provided by definitions 4.1-4.4. In order to keep definitions short, we introduce the abbreviation "*wrts. r*" (with regard to security requirement r).

Let S be an atomic system with the set of atomic components $A = \{A_i\}_{i=1}^{n}$.

Definition 4.1 *A system S is (k out of N)-resilient, $k \in \{1, \ldots, |N|\}$, $N \subseteq A$, wrts. r*

$:\Leftrightarrow$ *At least k components out of N need to be secure wrts. r in order to make S meet r.*

In order to get more flexible representations of requirements on atomic systems, we define the following resilience terms:

Definition 4.2 *A system S is a) $((k_1 \oslash \ldots \oslash k_m)$ out of $(N_1, \ldots, N_m))$-resilient, b) $((k_1 \otimes \ldots \otimes k_m)$ out of $(N_1, \ldots, N_m))$-resilient, $k_i \in \{1, \ldots, |N_i|\}$, $N_i \subseteq A$ $\forall i$, wrts. r*

$:\Leftrightarrow$ *For a) each, b) any $i \in \{1, \ldots, m\}$, at least k_i components out of N_i need to be secure wrts. r so that S meets requirement r.*

With regard to non-atomic systems, we define resilience terms similarly: Let $\{S_i\}_{i=1}^{n}$ be (sub)systems of a system S, and let system S_i be l_i-resilient for all $i \in \{1, \ldots, n\}$.

Definition 4.3 *A system S is (k out of $\{l_{i_1}, \ldots, l_{i_m}\}$)-resilient, $k \in \{1, \ldots, m\}$, $\{i_1, \ldots, i_m\} \subseteq \{1, \ldots, n\}$, wrts. r*

$:\Leftrightarrow$ *At least k systems out of $\{S_{i_1}, \ldots, S_{i_m}\}$ need to be secure wrts. r so that S meets requirement r.*

Definition 4.4 *A system S is a) $((k_1 \oslash \ldots \oslash k_m)$ out of $(N_1, \ldots, N_m))$-resilient, b) $((k_1 \otimes \ldots \otimes k_m)$ out of $(N_1, \ldots, N_m))$-resilient, $k_i \in \{1, \ldots, |N_i|\}$, $N_i \subseteq \{l_1, \ldots, l_n\}$ $\forall i$, wrts. r*

$:\Leftrightarrow$ *For a) each, b) any $i \in \{1, \ldots, m\}$, at least k_i systems out of the set of systems for which N_i contains resilience terms need to be secure wrts. r so that S meets requirement r.*

We now illustrate the security analysis and the determination of resilience terms with an example.

Example 4.1 *We use a web service scenario, in which a retailer uses three web services in order to identify customers' behavior. Service A offers data mining capabilities and stores sales data, including customer IDs. Service B is offered by a financial service provider, who provides credit ratings of customers. Service C provides storage capacities and stores master data on customers, including their customer IDs and identities. In this example, we consider secrecy with regard to information on which customer has bought what under which financial conditions. Secrecy is kept if one of the providers A and B is secure, or if one of B and C is secure. With regard to provider A, we assume that this provider accounts for secrecy by storing data on two components (A_3 and A_4) and implementing a secret share mechanism [BK05]. Components A_1 and A_2 are responsible for distributed computation in terms of data mining; both components get data from A_3 and A_4. With regard to financial service provider B, customer IDs generated by B (they differ from customer IDs stored at A) are stored on B_1 and B_2 together with financial data by implementing a secret share mechanism. Components B_3 and B_4 store names of customers and customer IDs (generated by B) redundantly. Analogous to A and B, storage provider C implements a secret share mechanism when storing customer data. Figure 2 shows the overall system S. Applying definitions 4.1, 4.2a, 4.2b, and 4.4b, we yield the following resilience terms:*

- A *is* $\underbrace{((2 \oslash 1) \textit{ out of } (\{A_1, A_2\}, \{A_3, A_4\}))}_{l_1}$*-resilient wrts. r. (def. 4.2a)*
- B *is* $\underbrace{((1 \otimes 2) \textit{ out of } (\{B_1, B_2\}, \{B_3, B_4\}))}_{l_2}$*-resilient wrts. r. (def. 4.2b)*
- C *is* $\underbrace{(1 \textit{ out of } \{C_1, C_2\})}_{l_3}$*-resilient wrts. r. (def. 4.1)*
- S *is* $((1 \otimes 1) \textit{ out of } (\{l_1, l_2\}, \{l_2, l_3\}))$*-resilient wrts. r. (def. 4.4b)*

4.2 Propositional logic terms

As example 4.1 shows, resilience terms can become complex, even for small systems. In order to yield representations that are comfortable to interpret for persons and appropriate for the computation of the uncertainty with which a system does not fulfill a specific requirement r, we transform resilience terms into propositional logic formulas. Particularly useful is the subsequent transformation of formulas into semantically equivalent formulas in normal form, such as the disjunctive normal form (DNF) or the conjunctive normal form (CNF). These normal forms show different strengths: while the CNF allows to determine "weak points", such as single points of failure, the DNF is useful for identifying "strong points", such as components or subsystems where security results in the security of the

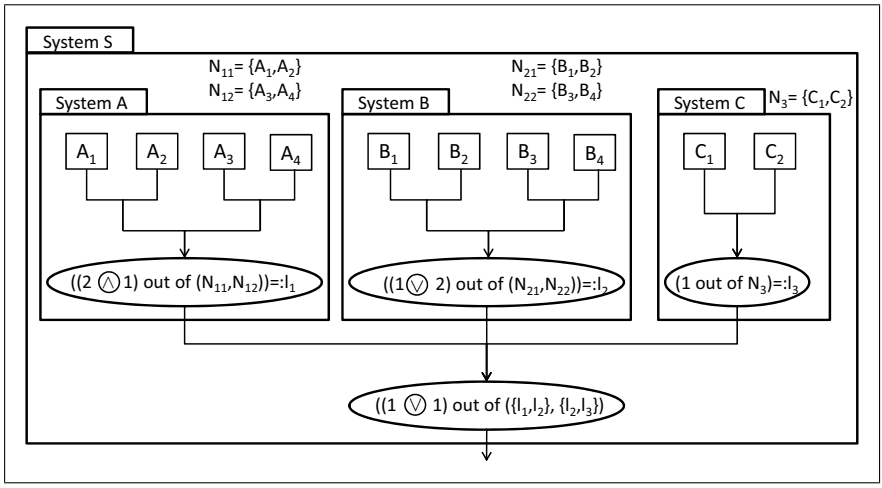


Figure 2: System structure and resilience values of Example 4.1

overall system, regardless of the security (levels) of other components and subsystems. Thus, both normal forms should be applied complementarily.

Theorem 4.1 *Let system S consist of basic components $A = \{A_1, \ldots, A_n\}$, and let $\{X_{A_1}, \ldots, X_{A_n}\}$ be literals with X_{A_i} =true $\forall i$, iff A_i is secure. Then, the resilience term l of S can be mapped on a propositional logic formula $f(l)$ such that S is secure iff $f(l)$ is true.*

Due to limitations of space, we provide a sketch of proof only: The principal idea of the proof is that we reformulate the expression "k out of a set L" by explicitly considering all combinations of elements of L, where L can be a set of basic components or of resilience terms of subsystems. The provision of such a mapping f (of resilience terms on propositional logic terms) proves the theorem.

We use the example shown in Figure 2 to illustrate how to determine the propositional logic formula of a particular resilience term.

Example 4.2

- *resilience term* $l_1 = ((2 \oslash 1)$ *out of* $(\{A_1, A_2\}, \{A_3, A_4\}))$

$$\begin{aligned} \Rightarrow f(l_1) &= (f((2 \text{ out of } \{A_1, A_2\})) \wedge (f((1 \text{ out of } \{A_3, A_4\}))) \\ &= ((A_1 \wedge A_2)) \wedge ((A_3) \vee (A_4)) = A_1 \wedge A_2 \wedge (A_3 \vee A_4) =: f_A \end{aligned}$$

- *resilience term* $l_2 = ((1 \otimes 2)$ *out of* $(\{B_1, B_2\}, \{B_3, B_4\}))$

$$\begin{aligned} \Rightarrow f(l_2) &= (f((1 \text{ out of } \{B_1, B_2\})) \vee (f((2 \text{ out of } \{B_3, B_4\}))) \\ &= ((B_1 \vee B_2)) \vee ((B_3) \wedge B_4)) = B_1 \vee B_2 \vee (B_3 \wedge B_4) =: f_B \end{aligned}$$

- *resilience term* $l_3 = (1 \textit{ out of } \{C_1, C_2\})$

$$\Rightarrow f(l_3) \quad = \quad (C_1) \vee (C_2) = C_1 \vee C_2 =: f_C$$

- *resilience term* $l = ((1 \Circledv 1) \textit{ out of } (\{l_1, l_2\}, \{l_2, l_3\}))$

$$\begin{aligned} \Rightarrow f(l) &= (f((1 \textit{ out of } \{l_1, l_2\})) \vee (f((2 \textit{ out of } \{l_2, l_3\})) \\ &= (((f(l_1))) \vee ((f(l_2)))) \vee (((f(l_2))) \vee ((f(l_3)))) \\ &= (f(l_1)) \vee (f(l_2)) \vee (f(l_3)) = (f_A) \vee (f_B) \vee (f_C) \\ &= (A_1 \wedge A_2 \wedge (A_3 \vee A_4)) \vee (B_1 \vee B_2 \vee (B_3 \wedge B_4)) \vee (C_1 \vee C_2) \end{aligned} \tag{1}$$

We now convert the resulting propositional logic term into DNF and CNF. The DNF representation can be easily derived from (1) and is given by

$$(A_1 \wedge A_2 \wedge A_3) \vee (A_1 \wedge A_2 \wedge A_4) \vee B_1 \vee B_2 \vee (B_3 \wedge B_4) \vee C_1 \vee C_2 \tag{2}$$

Having available the DNF representation, we can easily derive the CNF representation, which is given by

$$\bigwedge_{\substack{X \in \{A_1, A_2, A_3\} \\ Y \in \{A_1, A_2, A_4\} \\ Z \in \{B_3, B_4\}}} (X \vee Y \vee B_1 \vee B_2 \vee Z \vee C_1 \vee C_2) \tag{3}$$

While the DNF representation in 2 shows that each of the components B_1, B_2, C_1, C_2 is a strong point, the CNF representation reveals that there is (fortunately) no single point of failure.

5 Fuzzy set theory

Fuzzy set theory goes back to Lotfi Zadeh [Zad65], who proposed fuzzy sets as means for dealing with non-probabilistic uncertainty. As it is far beyond the scope of this paper to provide an overview of this field, we briefly introduce the very basic ideas of fuzzy set theory [Zim96, BE02]. The key idea of fuzzy set theory is the extension of the (crisp) membership concept in traditional set theory by providing for a degree with which an element belongs to a set. The degree is specified by a membership function.

Definition 5.1 *Let Ω be some set. Then we define a fuzzy set A as follows:*

$$A := \{(x, A(x)) | x \in \Omega\}, \textit{ with } A(x) := \mu_A(x) : \Omega \rightarrow [0, 1] \textit{ being the membership function.} \tag{4}$$

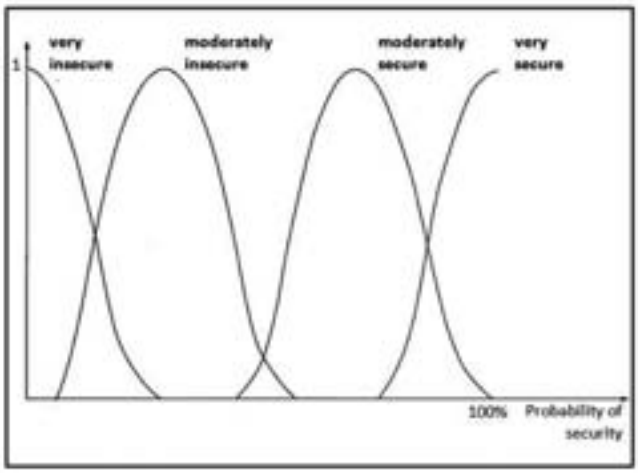


Figure 3: Linguistic variable *security* [Zim96, p. 132]

A particular type of fuzzy set is a *fuzzy number*:

Definition 5.2 *A fuzzy number is a fuzzy set A over $\Omega = \mathbb{R}$ such that $\mu_A(x)$ is piecewise continuous and it exists exactly one interval $[a, b]$ with $\mu_A(x) = 1\ \forall a \leq x \leq b$.*

For example, a fuzzy set can represent an *integer number close to 10*, where

$$A = \{(x, \mu_A(x)) | \mu_A(x) = (1 + (x - 10)^2)^{-1},\ x \in \mathbb{R}\}$$

Set-theoretic operations with fuzzy sets are pointwise defined over their membership functions [Zim96, BE02]. For example, the membership function $\mu_C(x)$ of the intersection $C = A \cap B$ can be defined by $\mu_C(x) = \min\{\mu_A(x), \mu_B(x)\}$, $x \in \Omega$. Further set-theoretic operators, and relational operators and arithmetic operators for fuzzy numbers are presented in [Zim96, BE02].

Another powerful concept in the field of fuzzy set theory turned out to be linguistic variables [Zad73]. We present a formal definition provided by [Zim96, p. 131]:

Definition 5.3 *A linguistic variable is a quintuple $(x, T(x), \Omega, G, \overline{M})$, where (1) x is the name of the variable (e.g., security), (2) $T(x) = T$ denotes the terms of the variable (e.g., $\{$very insecure, moderately insecure, moderately secure, very secure$\}$), (3) Ω is some set, (4) G is a syntactic rule for generating terms, and (5) $\overline{M}(T)$ assigns a fuzzy set to term T.*

Example 5.1 *Figure 3 shows an example of the linguistic variable security.*

6 Fuzzy decision support model

6.1 The model

In order to keep the model simple, we do not consider more than one security requirement at the same time, thus receiving a model that contains only one goal function. If we need to address several security requirements contemporaneously, which differ with regard to their resilience terms, we get a multi-criteria decision model, which contains one goal function for each security requirement. We assume that the security description of a distributed system, which contains the set of components A, is given by the propositional logic formula (in CNF)

$$\begin{aligned} A &= (A_{11} \vee \ldots \vee A_{1n_1}) \wedge \ldots \wedge (A_{m1} \vee \ldots \vee A_{mn_m}) \\ &= \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{n_i} A_{ij} \right), \; A_{ij} \in A \; \forall i,j, \; A_{ij} \text{ not necessarily different} \end{aligned} \tag{5}$$

In accordance with our assumption that security levels and security investment expenses can be appropriately represented by terms of linguistic variables and by fuzzy numbers, respectively, we suggest the following fuzzy decision model, which includes a fuzzy objective function, fuzzy variables, fuzzy parameters, and crisp constraints:

$$\max \; ((X_{11} \cup \ldots \cup X_{1n_1}) \cap \ldots \cap (X_{m1} \cup \ldots \cup X_{mn_m})) = \bigcap_{i=1}^{m} \left(\bigcup_{j=1}^{n_i} X_{ij} \right) \tag{6}$$

$$\text{s. t.} \quad X_{ij} \geq B_{ij}^0 \quad \forall i,j | A_{ij} \in A \tag{7}$$

$$X_{ij} \geq B_{ij}^* \quad \forall i,j | A_{ij} \in A \tag{8}$$

$$\sum_{i,j|A_{ij} \in A} c_{ij}(X_{ij}, B_{ij}^0) \leq b \tag{9}$$

$$\sum_{i,j|A_{ij} \in A^{(t)} \subset A} c_{ij}(X_{ij}, B_{ij}^0) \leq b_t \, \forall t \tag{10}$$

$$c_{ij}(X_{ij}, B_{ij}^0) \leq b_{ij} \quad \forall i,j | A_{ij} \in A \tag{11}$$

$$X_{ij}, B_{ij}^0, B_{ij}^* \in \overline{M}(T(\text{security})) \quad \forall i,j | A_{ij} \in A \tag{12}$$

$$b, b_{ij}, c_{ij}(X_{ij}, B_{ij}^0) \in \overline{\mathbb{R}} \text{ (fuzzy numbers)}, \quad \forall i,j | A_{ij} \in A \tag{13}$$

The fuzzy decision model aims at maximizing the overall security of the system described by A under security constraints (regarding single system components) (7-8) and budget constraints (9-11). The decision variables X_{ij} are fuzzy variables and can be assigned fuzzy sets of terms of the linguistic variable *security* (12) – $\overline{M}$ maps linguistic terms on fuzzy sets. For example, $T(\text{security})$ could consist of the terms *very insecure*, *moderately insecure*, *moderately secure*, *very secure*. The fuzzy goal function (6) combines all decision variables (security expressions) according to the logic-based security description A. In (6), the fuzzy set operators $\cup$ and $\cap$ are union and intersection operators, respectively,

and they correspond to the logical operators $\vee$ and $\wedge$. Here we can see how the binary differentiation between *secure* and *insecure* components is fuzzified. It should be noted that, in contrast to crisp decision models, it still needs to be specified how fuzzy sets are ordered in order to maximize the objective value.

The optimization underlies two types of security constraints: for each component A_{ij}, the security level X_{ij} after the investment $c_{ij}(X_{ij}, B^0_{ij})$ must be larger than or equal to its original level B^0_{ij} (7) and must also be larger than an exogenously given aspiration level B^*_{ij} (8), where B^0_{ij}, B^*_{ij} are fuzzy sets of linguistic terms (12). The optimization also underlies three types of budget constraints: the overall expenses to increase the security levels from B^0_{ij} to X_{ij} must be not larger than an overall fuzzy budget constraint b (9). Similarly, there may be a budget constraint for a single component (11) or for a group of components (10). It should be noted that the expenses for increasing the security level of a component depend on the current security level and the future security level (11). We model expenses as fuzzy numbers.

It should also be noted that while all constraints contain fuzzy sets, the decision of whether a constraint is met is sharp (in our model). This is due to the fact that (in our model) $\leq$ and $\geq$ define dichotomous relations between two fuzzy sets. Alternatively, we could define fuzzy relations. In this case, we would have to specify how constraints are combined. In this paper, we do not follow this path. We now present a simple example instance of our fuzzy decision model.

Example 6.1 *Let us assume that a distributed anonymizing system connects a source node S and a destination node D through two different paths (due to availability concerns), each of which contains two nodes (see Figure 4a). We use the linguistic variable security, which contains the terms T(security)=$\{$very insecure, moderately insecure, moderately secure, very secure$\}$. The security conditions of the nodes are shown in Figure 4a. Figure 4b shows graphically the membership functions of the fuzzy sets of the terms. Figure 4c provides the expenses for increasing the security level. For the purpose of simplicity, the fuzzy numbers given in the table refer to all components. We also require component A_{11} to become at least moderately insecure. The overall budget of all security investments is approximately 70,000 USD, and the security investment in component X_{12} should not exceed 25,000 USD.*

The resilience term of the system is $((1 \oslash 1)$ out of $(\{A_{11}, A_{12}\}, \{A_{21}, A_{22}\}))$, which results to the propositional logic formula (in CNF): $((A_{11} \vee A_{12}) \wedge (A_{21} \vee A_{22}))$. We get the following fuzzy decision model (the numbers in the model refer to the numbers of the

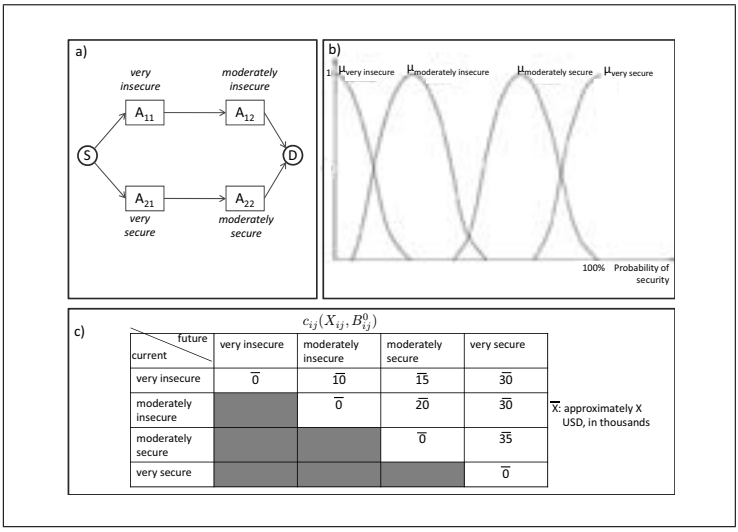


Figure 4: Data of example 6.1

generic fuzzy decision model presented above):

$$
\begin{aligned}
&\textit{max} && ((X_{11} \cup X_{12}) \cap (X_{21} \cup X_{22})) && (6)\\
&\textit{s. t.} && X_{11} \geq \overline{M}(\textit{very insecure}) && (7)\\
& && X_{12} \geq \overline{M}(\textit{moderately insecure}) && (7)\\
& && X_{21} \geq \overline{M}(\textit{very secure}) && (7)\\
& && X_{22} \geq \overline{M}(\textit{moderately secure}) && (7)\\
& && X_{11} \geq \overline{M}(\textit{moderately insecure}) && (8)\\
& && c_{11}(X_{11}, \overline{M}(\textit{very insecure})) + c_{12}(X_{12}, \overline{M}(\textit{moderately insecure}))+ && \\
& && c_{21}(X_{21}, \overline{M}(\textit{very secure})) + c_{22}(X_{22}, \overline{M}(\textit{moderately secure})) \leq \overline{70} && (9)\\
& && c_{12}(X_{12}, \overline{M}(\textit{moderately insecure})) \leq \overline{25} && (11)\\
& && X_{ij} \in \overline{M}(T(\textit{security})) \; \forall i, j | A_{ij} \in A && (12)
\end{aligned}
$$

6.2 Solving model instances

We now discuss which data and which specifications are necessary to solve an instance of the model.

1. The model requires knowledge of the security structure A. This structure can be derived by determining the resilience term and its representation as propositional logic formula in CNF.

2. In order to maximize the goal function, possible results need to be ordered, i.e. the decision maker needs to specify how fuzzy sets are ordered. There is not one single

solution of this problem; for example, we can draw on the relation of fuzzy sets as used in constraints (7) and (8). Alternatively, we can first defuzzify both fuzzy sets and then compare the resulting crisp values.

3. As the model uses a preference relation of two fuzzy sets ((7)-(11)), the decision maker needs to specify a concrete preference relations s/he applies (see Subsection 5).

4. The decision maker needs to know budget constraints. While the overall budget is often known (at least approximately), approximate budgets for components and groups of components are not always given. In case budget constraints of the latter type are not desirable, the respective constraints can be removed from the model.

5. The linguistic variable *security* needs to be defined, including the definition of the terms T(security) (e.g., *very insecure*) and their membership functions. Figure 4b provides such a linguistic variable.

6. For each component, the expenses to increase the level of security need to be specified in terms of a fuzzy number. For example, the decision maker may find that it cost about 50,000 $ to make a *very insecure* component *moderately secure*. Figure 4c shows an example of expenses for increasing the security of a component.

Each instance is a discrete optimization problem, where

- the number of decision variables n_V is the number of components ($n_V \leq \sum_{i=1}^{m} n_i$)[1],
- the solution space contains $n_S = |T(security)|^{n_V}$ elements, i.e. the size of the solution space increases polynomially in the number of linguistic terms, but it increases exponentially in the number of components,
- the number of constraints n_C is $n_V(7) + n_V(8) + 1(9) + n_V(11) = 3 \cdot n_V + 1$. However, we can easily reduce the number of constraints to $2 \cdot n_V + 1$, when we merge constraints (7) and (8) and substitute these with $X_{ij} \geq \max\{B^0_{ij}, B^*_{ij}\}$.

As the size of the solution space increases exponentially in the number of components –four security levels and ten components lead to a solution space that consists of $n_S = 4^{10} \approx 10^6$ elements –, solving a problem instance through enumeration becomes computationally infeasible. Even worse, the decision model turns out to be NP-hard so that the application of heuristic procedures becomes necessary for large instances. Again, due to space limitation we cannot provide details of the formal proof of NP-hardness in this paper. The guiding idea of the proof is the demonstration how the satisfiability problem 1-3-SAT, which is NP-complete [KL99, p. 59], is reducible to the fuzzy decision problem in polynomial time.

[1] A_{ij} do not need to be pairwise different.

7 Discussion

We now discuss some limitations and drawbacks of our approach and show how they can be addressed in further research.

While a key advantage of using a fuzzy decision support model lies in the dispensability of historic, probabilistic data, which are often unavailable, the solution of an instance of our (generic) fuzzy model requires to specify membership functions of fuzzy sets, terms of linguistic variables, and fuzzy operators in such a way that the model mirrors the attitudes and assumptions of the decision maker with regard to security investments. More precisely, what needs to be specified is the order of fuzzy sets with regard to the objective function and with regard to the constraints, the terms and membership functions of the linguistic variable *security*, upper budget bounds and lower security bounds, the current security level of components, budget constraints, and the function c, which maps a pair of (current) security level and (future) security level on an amount of investment. Empirical work would need to identify these attitudes and assumptions of decision makers. In addition, one could also draw on security standards, such as the Common Criteria [ISO09], to specify under which conditions a component is how secure and to determine terms of the linguistic variable *security*.

A further assumption of our decision model is that the structure of the distributed system is known. This is not always the case; for example, in several anonymizing networks the participating components are determined during process execution.

Due to the NP-hardness of the decision model, solving large instances optimally becomes computationally infeasible. Consequently, further research needs to develop heuristic algorithms.

If the decision maker needs or wants to distinguish between different security requirements, s/he would have to use one (fuzzy) goal function per requirements. The resulting model is a fuzzy multi-criteria problem, which can be solved with methods proposed in the literature (e.g., [Zim96, p. 303ff]).

Despite the aforementioned challenges with regard to the application of the fuzzy decision support model, we argue that a fuzzy set based perspective on security investment situations is a valuable means for practitioners, who need to deal with uncertainty in the absence of (reliable) probabilities.

References

[BE02] James J. Buckley and Esfandiar Eslami. *An Introduction to Fuzzy Logic and Fuzzy Sets*. Advances in Soft Computing. Physica-Verlag, 2002.

[BK05] Robert Blakley and Gregory Kabatiansky. *Encyclopedia of Cryptography and Security*, chapter Secret Sharing Schemes, pages 544–545. Springer, 2005.

[BN08] Rainer Böhme and Thomas Nowey. Economic Security Metrics. In Irene Eusgeld, Felix C. Freiling, and Ralf Reussner, editors, *Dependability Metrics*, volume 4909 of

Lecture Notes in Computer Science, pages 176–187, 2008.

[CN06] Marco Cremonini and Dmitri Nizovtsev. Understanding and Influencing Attackers' Decisions: Implications for Security Investment Strategies. In *Workshop on the Economics of Information Security*, 2006.

[CRY08] H. Cavusoglu, S. Raghunathan, and W. Yue. Decision-theoretic and game-theoretic approaches to IT security investment. *Journal of Management Information Systems*, 25(2):281–304, 2008.

[GCC08] Jens Grossklags, Nicolas Christin, and John Chuang. Security investment (failures) in five economic environments: A comparison of homogeneous and heterogeneous user agent. In *Workshop on the Economics of Information Security*, 2008.

[GJC09] Jens Grossklags, Benjamin Johnson, and Nicolas Christin. The Price of Uncertainty in Security Games. In *Workshop on the Economics of Information Security*, June 2009.

[GL02] Lawrence A. Gordon and Martin P. Loeb. The Economics of Information Security Investment. *ACM Transactions on Information and System Security*, 5(4):438–457, 2002.

[HHB06] C. Derrick Huang, Qing Hu, and Ravi S. Behara. Economics of Information Security Investment in the Case of Simultaneous Attacks. In *Workshop on the Economics of Information Security*, 2006.

[HKS00] T. Hofmeister, M. Krause, and H.U. Simon. Optimal k out of n secret sharing schemes in visual cryptography. *Theoretical Computer Science*, 240:471–485, 2000.

[HN10] Kjell Jorgen Hole and Lars-Helge Netland. Toward Risk Assessment of Large-Impact and Rare Events. *IEEE Security and Privacy*, forthcoming, 2010.

[ISO09] ISO/IEC. Common Criteria for Information Technology Security Evaluation, July 2009.

[KL99] Hans Kleine Büning and Theodor Lettmann. *Propositional Logic: Deduction and Algorithms*. Cambridge University Press, 1999.

[KSST09] Philipp Klempt, Hannes Schmidpeter, Sebastian Sowa, and Lampros Tsinas. Business Oriented Information Security Management A Layered Approach. In *Proceedings of the OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS*, volume 4804/2009 of *Lecture Notes in Computer Science*, pages 1835–1852, 2009.

[Lee03] Vincent C.S. Lee. A Fuzzy Multi-criteria Decision Model for Information System Security Investment. In *Proceedings of the Intelligent Data Engineering and Automated Learning*, volume 2690/2003 of *Lecture Notes in Computer Science*, pages 436–441, 2003.

[NIS08] NIST. Performance Measurement Guide for Information Security, May 2008. NIST Special Publication 800-55 Revision 1.

[WCR05] J. Wang, A. Chaudhury, and H.R. Rao. An extreme value approach to information technology security investment. In *Proceedings of the International Conference on Information Systems*, Las Vegas, NV, 2005.

[Zad65] L. Zadeh. Fuzzy sets. *Information Control*, (8):338–353, 1965.

[Zad73] L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning. Memorandum ERLM 411, Berkeley, Calififfornia, 1973.

[Zim96] H.-J. Zimmermann. *Fuzzy set theory - and its applications*. Kluwer, Boston, Dordrecht, London, 3rd edition, 1996.

Quantifying the Attack Surface of a Web Application

Thomas Heumann, Jörg Keller
University of Hagen
Dept. of Mathematics and Computer Science
58084 Hagen, Germany
joerg.keller@fernuni-hagen.de

Sven Türpe
Fraunhofer SIT
Rheinstraße 75
64295 Darmstadt, Germany
sven.tuerpe@sit.fraunhofer.de

Abstract: The attack surface of a system represents the exposure of application objects to attackers and is affected primarily by architecture and design decisions. Given otherwise consistent conditions, reducing the attack surface of a system or an application is expected to reduce its overall vulnerability. So far, only systems have been considered but not single applications. As web applications provide a large set of applications built upon a common set of concepts and technologies, we choose them as an example, and provide qualitative and quantitative indicators. We propose a multi-dimensional metric for the attack surface of web applications, and discuss the rationale behind. Our metric is easy to use. It comprises both a scalar numeric indicator for easy comparison and a more detailed vector representation for deeper analysis. The metric can be used to guide security testing and development. We validate the applicability and suitability of the metric with popular web applications, of which knowledge about their vulnerability already exists.

1 Introduction

Measuring security properties is challenging yet necessary. The need to make informed decisions implies necessity, while the complex and sometimes counter-intuitive nature of security makes measuring it a challenge. Various security metrics for various purposes have been proposed, but meaningful quantification of security remains an open problem [Ver09].

We contribute to this ongoing endeavour a metric for the attack surface of web applications, capturing a set of exposure factors in the architecture and design of such applications. Our attack surface metric differs from existing approaches in two important aspects: the metric is tailored to a class of applications, and it does not require access to the application's source code. For a more in-depth discussion see [Heu10]. Security practitioners often use the term *attack surface* colloquially to refer to the amount of code, functionality, and interfaces of a system exposed to attackers [How04]. The idea behind this notion is that attackers can exploit vulnerabilities only in those parts of a system that the system exposes to adversaries. Comparing two otherwise equal systems, exposed to equal populations of adversaries, the one with the smaller attack surface is statistically expected to be less vulnerable. Since other factors—e.g. code quality and defect distribution, security mechanisms, or user behaviour—independently influence the security of a system, this ex-

pected correlation may not always be observed in real-world settings where such factors cannot be controlled.

So far, there have been few attempts to formalize the intuitive notion of an attack surface. A notable exception is the seminal work by Manadhata [Man08], Manadhata and Wing [MW05, MW10] and Howard, Pincus and Wing [HPW05]. They consider the attack surface from a system's perspective and establish a generic formal notion based on the dimensions *targets and enablers*, *channels and protocols*, and *access rights*. Their apparent starting point are threats in conjunction with common rules of thumb [Pei08], such as the principle of least privilege and the recommendation to disable unnecessary network services. Consequently, their attack surface definition and metric is generic though perhaps slightly tailored to network hosts with network services. They provide a formal justification and underpinning for measures reducing the attack surface of a system.

Web applications more and more become the standard type of application software. They are built upon a common set of concepts and technologies: browsers as generic clients; HTTP as the communication protocol between browsers and web servers; server-side and client-side runtime environments for code execution; and XML as a universal technology for data representation. Their common platform brings about common security considerations and common security problems of web applications, as illustrated by the OWASP Top Ten Most Critical Web Application Security Risks [OWA10]. Securing a web application is therefore different from securing a web server host, and it is a task primarily for application developers rather than system administrators. To this end the developer has to accomplish a variety of tasks: properly employ security mechanisms and functions, avoid certain classes of software defects, and follow design principles and best practices, and reduce the attack surface of the application, to name just a few. An attack surface metric helps with the latter task. It makes effects measurable and applications or versions comparable. We present a metric to measure the attack surface of a web application and discuss its rationale. We validate its applicability with known web applications.

The remainder of this paper is organized as follows. Section 2 briefly introduces web applications. In Sect. 3 we describe our approach to quantifying the attack surface of such applications. The parameters used to estimate attack surfaces are described in Sect. 4. Section 5 demonstrates how the metric is applied in an experimental evaluation. We conclude in Sect. 6.

2 Web Applications and Their Attack Surface

Web Applications A typical web application comprises three subsystems: the browser, the server, and back-end systems (see Fig. 1). The *browser* constitutes the platform for the user interface. It renders HTML code, images, and other data visible, handles user input, and provides the runtime environment for client-side code. Helper applications, such as a PDF reader, and plug-in components, e.g. for Java or Flash, may extend the capabilities of the browser. A browser process at a point in time, and often also the entire browser installation, is associated with a particular user as a personal software tool. The *web server*

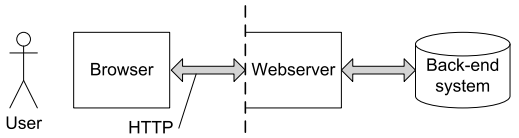


Figure 1: High-level architecture of a typical web application

waits for browsers to connect and answers their HTTP requests. Each request comprises a URL pointing to some resource and possibly additional parameters. The server responds either by serving a static resource, or by executing a program and returning the output of this program to the requesting browser. In the course of fulfilling a request the server, or a program executed by the server, may access *back-end systems* such as databases. Contrary to the impression that Fig. 1 might give, multiple instances may exist for each of the subsystems, with complicated relationships between the instances. In particular, a browser may access multiple web servers simultaneously.

Attack Surface of a Web Application The purpose of our attack surface metric is to estimate the amount of functionality and code that a web application exposes to outside attackers. Web applications comprise a natural security boundary, indicated by the dashed line in Fig. 1: users and attackers alike normally are not granted access to the web server and back-end systems other than through the HTTP interface of the web server. We consider any attack that requires such elevated privileges as a prerequisite an insider threat and beyond the scope of this paper. We further exclude from our consideration attacks that primarily target the users of an application, such as phishing attacks to get their passwords. Accessible to the outside attacker—who may or may not also be a user of the application—are therefore the HTTP interface(s) of the web server(s), any server-side or back-end functionality accessible through the server, and any data displayed and code executed on the browser side, including the browser itself. Finally we exclude vulnerabilities of the underlying client or server (operating) system.

A simplistic definition of the attack surface would consider only the HTTP interface of the web server and the server-side functionality accessible through this interface. However, many typical risks [OWA10] involve important client-side aspects. In a Cross-Site Scripting (XSS) attack, for instance, the attacker may exploit server-side software defects to inject JavaScript code into the client-side portion of the application, and exploit the capabilities thus gained to copy session cookies that give access to additional server-side functionality. An attack surface metric should therefore also consider the way the application uses the client subsystem.

3 Approach

Requirements and Assumptions An application attack surface metric is a tool for developers and testers. Developers use it, for instance, to assess the impact of design and

implementation decisions, whereas security testers may base on the metric estimates of required testing effort or of the expected number of defects. Our metric is designed with practical applicability for testers in mind, which entails a number of further requirements. We are aiming for these properties:

Attack surface approximation. The metric should represent the attack surface of a web application as outlined in Sect. 2. It should yield low numeric values for a plain old web server with only static content, and growing values as one adds application-level exposures such as interactive functionality.

Universal applicability. The attack surface metric should be applicable to any web site or application regardless of its purpose, functionality, or implementation platform.

Grey-box measurability. There should be practical methods to determine the metric for a given application. Approximate measurements should remain possible if all or some of the application source code is unavailable. This is important for testers as grey-box security testing is common.

Support qualitative comparison. Besides an overall numeric score, the metric should also support comparison on a semi-aggregated level, similar to CVSS [MSR07]. This makes it easier for users of the metric to spot qualitative differences that the overall score would obscure.

Relevant measurements. The metric should use measurements and parameters that commonly vary between applications or versions of an application.

In addition to these requirements we make two assumptions. First, we limit our considerations to external attacks, excluding insider threats from our consideration. Second, we focus on attack surface elements with immediate exposure to external attackers. This means we will not attempt to distinguish applications, for instance, by their use of back-end subsystems or interaction with the operating system of the server host.

Metric Construction We construct our metric using a real vector space of weighted parameters. Every possible attack surface is represented by an *attack surface vector* AS. Raw measurements are mapped into this space according to a set of rules and principles. The Euclidean norm provides us with a straightforward definition for the *attack surface indicator* $ASI = \|AS\|$ as an easy-to-compare numeric value. In addition we can create intermediate representations of the attack surface that preserve some qualitative information by defining several subspaces, projecting AS into each of them, and using the norms of the projections.

Raw measurements can be Boolean values representing presence or absence of a feature, enumerations representing multiple-choice measurements, or non-negative integer values as the result of counting. We represent a Boolean measurement by 0 for *false* or 1 for *true*; an n-ary enumeration by the consecutive sequence of natural numbers from 0 through $n-1$; and integer values by themselves. If possible, we define the measurement such that a smaller contribution to the attack surface implies a lower raw value.

Raw measurements are mapped to components of AS by means of functions. Most components are simply the measured parameter multiplied with a weight. For some dimensions of the vector space the mapping is a different function of a single or of multiple raw

Value	0	1	2	3	4	5
Raw count	0	1−2	3−5	6−9	10−14	15−20
Value		6	7	8	9	10
Raw count		21−27	28−35	36−44	45−54	55−∞

Table 1: Mapping of unbounded counts to a fixed range

measurements. We use this method with several groups of measurements as described in Sect. 5. We also use this method to map unbounded counts to the bounded range $[0, 10]$ according to Table 1. This is particularly useful where we cannot formally justify any upper boundary, yet expect the vast majority of measurements to stay below a certain value, such as with the number of different cookies or domains used by an application.

Attack Surface Vector The attack surface AS of a web application defined according to our construction is a vector consisting of the following components: degree of distribution $ddist$, page creation method dyn, security mechanisms $\langle security \rangle$, input vectors $\langle input \rangle$, active content $\langle active \rangle$, cookies $cookie$, user roles $role$ and access rights $rights$:

$$AS = \langle ddist; dyn; \langle security \rangle; \langle input \rangle; \langle active \rangle; cookie; role; rights \rangle .$$

Angle brackets indicate groups of components; the overall number of dimensions is 22. With measurements adjusted such that each component of the vector takes its maximum value, the attack surface vector looks like this:

$$AS_{max} = \langle 34; 1; \langle 1; 10; 10 \rangle ; \langle 1; 1; 1; 4; 1; 8 \rangle ; \langle 5; 7; 8; 6; 8; 10; 10; 4 \rangle ; 40; 10; 10 \rangle .$$

4 Attack Surface Parameters

For the first version of our metric, we gather candidates from sources such as the OWASP Top Ten, cf. [Heu10] for a detailed discussion. Among those, we select according to measurability. Each of the raw measurements required can be made or approximated by observing an application at runtime. Weights are chosen such that the impact of each parameter on the overall score ASI conforms to our estimation of the relative importance of this parameter. Future versions of the metric should use empirically validated weights. Table 2 shows an overview of all components of the attack surface vector, which we explain in detail below.

Degree of Distribution The degree of distribution $ddist$ summarizes how an application spans multiple domains. Cross-domain issues are a common source of vulnerability. Distribution makes such issues more likely as it requires the application to work around the same-origin policy enforced by web browsers to separate resources of different origins. Distribution also increases the number of potential attack vectors. To determine the degree of distribution we count the number of subdomains under a common 2nd– or lower-level

Parameter Family	Short Name	Parameters	Range
Degree of Distribution	$ddist$	Subdomains $sdom_{wa}$	$[0, 10]$
		Domains dom_{wa}	$[0, 10]$
		Foreign Domains dom_{ext}	$[0, 10]$
Dynamic Creation	dyn	Dynamic Creation	$\{0, 1\}$
Security Features	$security$	TLS $crypt$	$\{0, 1\}$
		Partial TLS $crypto\text{-}mix$	$\{0, 10\}$
		Validate $validate$	$\{0, 10\}$
Input Vectors	$input$	URL Parameters $urlparam$	$\{0, 1\}$
		Forms $forms$	$\{0, 1\}$
		Hidden Fields $hidden$	$\{0, 1\}$
		Authentication Methods $auth$	$\{0, 1\}$
		Search $search$	$\{0, 2, 4\}$
		File Upload $files$	$\{0, 8\}$
Active Content	$active$	Client-side Scripting (own) js	$\{0, 5\}$
		Client-side Scripting (foreign) js_{ext}	$\{0, 7\}$
		Server-Side Scripting sss	$\{0, 8\}$
		AJAX $ajax$	$\{0, 6\}$
		Java $java$	$\{0, 8\}$
		RIA (own) $flash$	$\{0, 10\}$
		RIA (foreign) $flash_{ext}$	$\{0, 10\}$
		Feeds (foreign) $feeds$	$\{0, 4\}$
Cookies	$cookie$	Own Cookies c_{wa}	$[0, 10]$
		Foreign Cookies c_{ext}	$[0, 10]$
Access Control	$role$	Role	$\{0, 5, 10\}$
	$rights$	Privileges	$\{0, 5, 10\}$

Table 2: Attack surface parameters

domain used by the application; the number of 2nd-level domains used; and the number of foreign domain names accessed when the application is being used. Then we determine the value of each parameter $sdom$ of own subdomains, dom of own domains, and dom_{ext} of foreign domains and subdomains involved according to Table 1, e.g. if there are five foreign domains involved then $dom_{ext} = 2$.

From these measurements we calculate the distribution subscore $ddist = \frac{1}{2} \cdot sdom + dom + 2 \cdot dom_{ext} - 1$. Subdomains of a common parent are sometimes considered equal under the same-origin policy whereas foreign domains indicate that external sites or applications are invoked, hence the weights. We subtract 1 as at least one domain is necessary and used per se.

Page Creation Method The parameter dyn represents the binary distinction between applications that dynamically create pages on the server side and those that do not. If an application uses server-side technologies such as PHP, ASP or JSP the value is 1 ($true$), otherwise 0. Server-side application code is a source of vulnerability.

Security Mechanisms The $security$ group of components of the attack surface vector represents the use of common security mechanisms. Security mechanisms are different from other factors in that their presence *reduces* the attack surface. We consider two mechanisms, Transport Layer Security (TLS) and input validation. In addition to their presence we evaluate how TLS is being used by the application. The $security$ group thus comprises three parameters based on Boolean measurements. Presence of TLS is indicated by $crypt \in \{0, 1\}$, the value 0 expressing that HTTPS is in use somewhere in the application. If pages exist that mix content accessed over TLS with content accessed over plain HTTP, we set $crypto\text{-}mix = 10$, otherwise to 0. The third parameter $validate \in \{0, 10\}$ takes on the value 10 if there is any indication, e.g. from testing, that server-side input validation is not in place or broken.

Input Vectors The HTTP interface between client and server supports a number of input vectors. The more vectors an application uses, the more complex it is. We consider URL parameters $urlparam$, HTML forms $forms$, hidden form fields $hidden$ and HTTP authentication mechanisms $auth$ as Boolean parameters with weight 1; file uploads $files \in \{0, 8\}$ with a higher weight; and the presence of search functions as a ternary value $search \in \{0, 2, 4\}$. The parameter $search$ is 0 if no site search is present, 2 if a locally-implemented mechanism is there, and 4 if the application uses an Internet search engine such as Google or Bing. Note that generally foreign features and embedded code from external sources outweigh internal counterparts, because they are beyond control. File uploads get a higher weight than other input vectors because they are potentially more powerful in the hands of attackers and their handling within applications is often different from the handling of other inputs.

Active Content Active content increases the attack surface on the client side, requiring the user to have plug-ins or helper applications in place and adding client-side code and processes to the application. We capture the use of client-side active content and related technologies as a group of Boolean components with different weights: $js \in \{0, 5\}$ if the application contains JavaScript code; $js_{ext} \in \{0, 7\}$ if it loads such code from a different site; $sss \in \{0, 8\}$ for server-side scripting; $ajax \in \{0, 6\}$ if the use of AJAX is observed; $java \in \{0, 8\}$ representing the use of Java applets; $flash \in \{0, 10\}$ and $flash_{ext} \in \{0, 10\}$ for the use of Adobe Flash or similar Rich Internet Application (RIA) technology; and $feeds \in \{0, 4\}$ if the application integrates, or supports the integration of RSS feeds from other sites. Note that the use of a sandbox by a technology does not imply a lower weight. Client-side issues may still affect the application; the sandbox may be optional, as, for instance, with signed Java applets; or the client-side runtime environment may be vulnerable. The weights represent a rough assessment of relative risk.

Cookies Cookies have various implications for the attack surface: cookies constitute another input channel into the application; they are often used to implement user authentication and session management; cookies can be used to track users; and cookies leave easy-to-observe traces in the browser. For the attack surface metric we create a compound parameter, $cookie$, from two measurements. We count the maximum number

of cookies and the number of foreign cookies from other sites that the application sets during a user session. The number of cookies is transformed into an intermediate score $c \in [0, 10]$, and the number of foreign cookies into $c_{ext} \in [0, 10]$ according to Table 1, e.g. if there are six foreign cookies then $c_{ext} = 3$. From these intermediate values we calculate $cookie = c + 3 \cdot c_{ext}$ as a component of the attack surface vector.

Access Control The parameter $role$ reflects the user status and can assume one of three values: unauthenticated (0), i.e. a user is not logged in (and possibly anonymous[1]), authenticated (5) or root (10), which means a) the user is logged-in and linked to an identity, b) certain access rights are associated with his identity. A user can have the following access rights, represented by the parameter $rights$: none (0), limited (5) or root (10).

5 Experimental Evaluation

We validate our attack surface metric by applying it to several web sites. This demonstrates how to use the metric, whether its application is practical in real-world scenarios, and whether the results are in line with our expectations. We expect our metric to yield higher values for ASI if the target is large and complicated, and similar values for similar applications. We also explore the range of values that we get from real applications and compare it to the theoretical extremes.

Measuring Parameters The preferable way of measuring attack surfaces would be by using automated tools. Since the metric is designed such that most components of the attack surface vector can be derived from client-side observations, implementing such a tool as an extension of either a web browser or a testing platform should be straightforward. However, such a tool does not yet exist. We demonstrate the metric using manual measurements.

Reliable measurement requires that the application is being used to a sufficient extent. Without delving into a discussion of possible coverage criteria, we require that every accessible function in the user interface of the application is touched during the measurement process. The measurement process starts with a clean client without any cached content or stale cookies and in a configuration that does not restrict the application or its client-side components in any way.

For the purposes of this paper we used a web browser, Firefox 3.5.8, in a particular configuration. We enabled all pertinent warnings and user permission requests, particularly those related to TLS and to cookies; permitted all third-party interactions; disabled automated features such as the password manager and input auto completion as well as phishing and malware warnings; and configured the browser to clear all private data on exit. This configuration ensures that we start with a clean environment each time we restart the browser, receive popup notifications for some of the information we are interested in (e.g. use of

[1] The anonymity may be repealed by persistent cookies, even if the user is not logged in.

Site or Application	Attack Surface Vector	ASI
min	$\langle 0;0;\langle 0;0;0\rangle;\langle 0;0;0;0;0;0\rangle;\langle 0;0;0;0;0;0;0;0\rangle;0;0;0\rangle$	0
gaos.org	$\langle 0;1;\langle 1;0;0\rangle;\langle 1;1;1;1;2;8\rangle;\langle 5;0;0;8;0;0;0;0\rangle;1;5;5\rangle$	14.63
testlab.sit.fraunhofer.de	$\langle 2.5;1;\langle 1;0;0\rangle;\langle 1;0;0;0;0;0\rangle;\langle 5;7;0;8;0;10;0;0\rangle;6;0;0\rangle$	16.83
BSCW	$\langle 0.5;1;\langle 0;0;0\rangle;\langle 1;1;1;1;2;8\rangle;\langle 5;0;0;8;0;0;0;0\rangle;1;10;5\rangle$	16.98
cFolders	$\langle 0.5;1;\langle 0;0;0\rangle;\langle 1;1;1;1;2;8\rangle;\langle 5;0;0;8;0;0;0;0\rangle;1;10;5\rangle$	16.98
Livelink	$\langle 0.5;1;\langle 0;0;10\rangle;\langle 1;1;1;1;2;8\rangle;\langle 5;0;0;8;8;0;0;0\rangle;2;10;5\rangle$	21.34
last.fm	$\langle 13;1;\langle 0;10;0\rangle;\langle 1;1;0;1;2;0\rangle;\langle 5;0;6;8;0;10;10;0\rangle;25;5;5\rangle$	35.74
max	$\langle 34;1;\langle 1;10;10\rangle;\langle 1;1;1;4;1;8\rangle;\langle 5;7;8;6;8;10;10;4\rangle;40;10;10\rangle$	60.79

Table 3: Attack surfaces of sample web sites and applications

TLS, cookies), and that the browser does not unnecessarily interfere with the application or our measurement process. In addition we use an interactive HTTP(S) proxy, such as WebScarab or Paros, to observe HTTP traffic.

5.1 Sample Web Applications

We chose our set of evaluation samples to cover a range of different web sites and applications, but with subsets of multiple similar applications. *Similar* is defined loosely here, meaning just that two applications serve more or less the same purpose and exhibit comparable core functionality. Our samples include:

- Simple web sites primarily serving pages with information. We use `gaos.org` and `testlab.sit.fraunhofer.de` as samples. One is the site of an open source software advocacy group and built using a wiki engine. The other is the web site of the research group of one of the authors. It serves mostly static pages but uses a limited amount of PHP code on the server side and JavaScript on the client side without providing much interactive functionality.
- Several web-based document management systems: Livelink 9.2, BSCW 4.4.5 and cFolders 4.0.0. All three are multi-user applications allowing their users to store and share files. They have been developed independently by different companies using different programming languages.
- The site `last.fm` as a representative of a typical web 2.0 site. This site offers a social network platform for music lovers, comprising, for instance, music streaming functions, profiles of users' listening habits, user-to-user communication features, and advertising.

5.2 Results

Quantitative Comparison Table 3 depicts the attack surfaces measured for the sample applications. The indicator values in the rightmost column exhibit several interesting properties. First, they cover only about one third of the theoretical range. This does not come unexpected: the lower and the upper end of the range represent extreme cases that

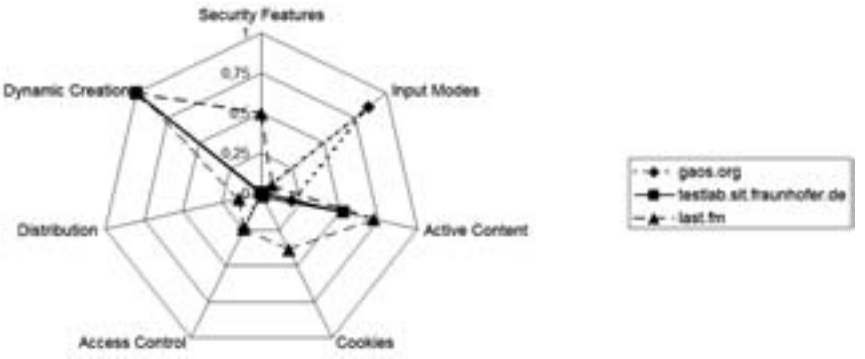


Figure 2: Qualitative comparison of the attack surfaces of web sites

real-world sites or applications are unlikely to reach. Second, and more surprisingly, the two apparently simple web sites reach indicator values close to those of the document management applications considered. This illustrates what the attack surface metric really measures. The two web sites have a high indicator value because they use similar technologies and concepts as web applications. With few exceptions our metric does not consider differences in complexity or size, but rather the mere presence or absence of features. Third, indicator values behave as expected for similar applications. The two simple web sites yield similar values, as do the three document management applications. The higher indicator value of Livelink is largely due to client-side Java applets and known deficiencies in input validation in the (outdated) version considered here. The indicator value clearly separates `last.fm` from the other sites and applications, which seems appropriate considering its nature, purpose, and features.

Analysis of Results Comparing attack surface indicators is a quick way of comparing applications, but the indicator is a lossy compression. Different vectors may lead to similar values. For a detailed analysis and further insight we need to compare the vectors rather than just their norms. One way is to look at the difference of two attack surface vectors, which has non-zero components in just those dimensions in which the two attack surfaces differ. Note that the difference of two attack surface vectors is not always a valid attack surface vector itself.

Due to the number of dimensions the vector difference of two attack surfaces may be difficult to interpret, but we can construct an intermediate view that preserves some of the qualitative information. To this end we calculate intermediate square sums for the seven parameter groups introduced in Sect. 3. Considering their varying ranges imposed by our weighting of parameters, we scale the ranges for each group to the interval $[0, 1]$. Figure 2 shows the resulting radar chart for the three web sites, and Fig. 3 the chart for the three applications. We can immediately see, for instance, that the larger attack surface of `last.fm` is due to the presence of several factors not found in the other samples, whereas for the simpler web sites only some of the factors have an effect at all. We can also see that the attack surfaces of the document management applications are similar.

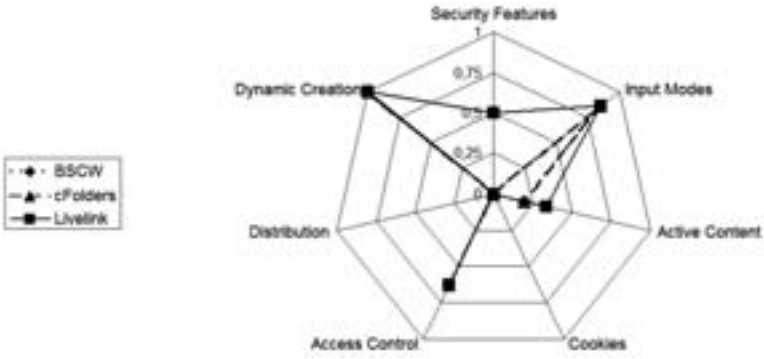


Figure 3: Qualitative comparison of the attack surfaces of document management applications

6 Conclusion

The attack surface metric defined in this paper is a first attempt to measure vulnerability expectations for the class of web applications under grey-box testing conditions. A first evaluation with existing applications indicates a good relation of our proposal with expectations.

In future work, we would like to compare our approach to other metrics, e.g. the traditional source lines of code (SLOC) [Ros97]. Also, we would like to refine our parameters, considering e.g. session management or sensitivity to the type of browser. Finally, it might be worthwhile to investigate thresholds for comparisons of attack surfaces in situations where only a subset of the parameters varies between applications. A common implementation platform, for instance, may fix some of the parameters.

Our attack surface metric is work in progress. The version described in this paper can be improved and refined in various ways:

- Consider further variables. The current version of the metric does not always clearly distinguish simpler web sites from more complex web applications. Considering further features may improve the precision.
- Replace n-ary parameters with counts or percentages where feasible. This, too, may improve the precision of the metric.
- Reconsider weights. We used weights to adjust the impact of individual parameters on the attack surface indicator. If, however, the metric is primarily used for vector comparisons, weights may be less important than we thought.
- Refine the semantics of some of the parameters. Authorization and access control, in particular, are too complex to be properly represented in a few simple parameters.
- Understand the impact of correlations. Not all parameters used in the metric are independent. For instance, each foreign cookie implies involvement of a foreign web site, but not vice versa.
- Properly handle replicated mechanisms. The Flash Player, for instance, has its own scripting language and cookie store. Such alternative mechanisms, when used by an

application, should have just the same impact on the metric as comparable functionality implemented using the primary mechanisms.
- Consider the effects of volatile properties of a web site. The attack surface determined for `last.fm`, for instance, was increased by a banner ad that initiated many connections to foreign sites.

Further work in these directions may ultimately lead to some kind of security metric construction kit providing building blocks and composition rules for the construction of domain– and purpose-specific metrics. The vector approach seems promising as a basis for such work.

Acknowledgement This work was supported by CASED, www.cased.de.

References

[Heu10] Thomas Heumann. Bestimmung der Angriffsfläche von Web-Anwendungen. Master's thesis, Dept. of Mathematics and Computer Science, University of Hagen, 58084 Hagen, `http://publica.fraunhofer.de/documents/N-134399.html`, March 2010.

[How04] Michael Howard. Attack Surface – Mitigate Security Risks by Minimizing the Code You Expose to Untrusted Users. `http://msdn.microsoft.com/en-us/magazine/cc163882.aspx`, November 2004.

[HPW05] Michael Howard, Jon Pincus, and Jeannette M. Wing. *Computer Security in the 21st Century*, chapter Measuring Relative Attack Surfaces, pages 109–137. Springer, 2005.

[Man08] Pratyusa Manadhata. *An Attack Surface Metric*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, November 2008. CMU-CS-08-152.

[MSR07] Peter Mell, Karen Scarfone, and Sasha Romanosky. A Complete Guide to the Common Vulnerability Scoring System. `http://www.first.org/cvss/cvss-guide.pdf`, June 2007.

[MW05] Pratyusa Manadhata and Jeannette M. Wing. An Attack Surface Metric. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, July 2005. CMU-CS-05-155.

[MW10] Pratyusa K. Manadhata and Jeannette M. Wing. An Attack Surface Metric. *Software Engineering, IEEE Transactions on*, Preprint, June 2010.

[OWA10] OWASP Top 10 – 2010. available online, `http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project`, 2010.

[Pei08] Holger Peine. Rules of Thumb for Developing Secure Software: Analyzing and Consolidating Two Proposed Sets of Rules. In *ARES '08: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security*, pages 1204–1209, Washington, DC, USA, 2008. IEEE Computer Society.

[Ros97] Jarrett Rosenberg. Some Misconceptions About Lines of Code. In *Fourth International Software Metrics Symposium (METRICS'97)*, volume 0, pages 137–142, Los Alamitos, CA, USA, November 1997. IEEE Computer Society.

[Ver09] Vilhelm Verendel. Quantified security is a weak hypothesis: a critical survey of results and assumptions. In *NSPW '09: Proceedings of the 2009 New security paradigms workshop*, pages 37–50, New York, NY, USA, 2009. ACM.

Social Lending aus der Perspektive des Datenschutzes

Rainer Böhme und Stefanie Pötzsch
{rainer.boehme|stefanie.poetzsch}@tu-dresden.de

Abstract: Als *Social Lending* bezeichnet man die Kreditvergabe zwischen Privatpersonen. Beim *Online Social Lending* veröffentlichen potenzielle Kreditnehmer ihre Gesuche auf entsprechenden Webseiten im Internet, um private Investoren zu finden. Dieser Beitrag weist auf einen Konflikt zwischen wirtschaftlichen Interessen und Datenschutzzielen hin und analysiert ihn mit empirischen Daten der größten deutschen Social-Lending-Plattform *Smava.de*. Der Analyse zufolge lohnt es sich gegenwärtig nicht, mehr personenbezogene Daten zu veröffentlichen als unbedingt notwendig.

1 Online Social Lending: Kreditvergabe durch Crowdsourcing

Die erste kommerzielle Plattform für Online Social Lending *Zopa.com* wurde 2005 in England gegründet. Derzeit existieren mehrere Anbieter von Online-Marktplätzen für unbesicherte Konsumentenkredite, deren Businessplan am ehesten mit den aus Entwicklungsländern bekannten Mikrokrediten vergleichbar ist: Potenzielle Kreditnehmer können im Internet ihren Finanzierungswunsch vorstellen und treffen auf private Investoren, die einzeln entscheiden, in welche der angebotenen Kreditvorhaben sie investieren möchten. Bei erfolgreicher Finanzierung wird das Ausfallrisiko auf mehrere Investoren verteilt, die jeweils einen Teil der Gesamtsumme eines Projekts finanzieren und dafür später neben ihrem ursprünglich investierten Betrag auch Zinsen vom Kreditnehmer erhalten. Die Betreiber solcher Plattformen erheben vorher festgelegte Gebühren, die unabhängig vom Kreditrisiko sind. Die Plattformen unterscheiden sich in der konkreten Ausgestaltung ihrer Marktmechanismen [BG09, CGL09].

Insgesamt ist Online Social Lending eine Alternative zum klassischen Bankkredit und umgeht weitgehend das Bankensystem als Finanzintermediär bei der Kreditvergabe. Diese technologiebedingten Entwicklungen auf dem Markt für Konsumentenkredite führen zu einer Reihe von ökonomischen und sozialen Veränderungen [FS08]. Der vorliegende Beitrag beschäftigt sich nur mit einem kleinen Ausschnitt, nämlich der (Un-)Vereinbarkeit von Social Lending und Datenschutzzielen. Mit ihren Kreditgesuchen veröffentlichen Kreditnehmer im Internet eine Vielzahl personenenbezogener Daten über sich selbst, aber auch über ihnen nahe stehende Personen. Im folgenden Abschnitt werden deshalb zunächst die Bedeutung von (personenbezogenen) Informationen für Kreditmärkte zusammengefasst und sich daraus ergebende Probleme aus Sicht des Datenschutzes aufgezeigt. Ergänzt wird dieser theoretische Teil im weiteren Verlauf des Beitrags mit einer empirischen Untersuchung. Mit Hilfe realer Daten der größten deutschen Social-Lending-Plattform *Smava.de* wird analysiert, inwiefern Einschnitte aus Sicht des Datenschutzes tatsächlich zu finanziellen Vorteilen, d. h. besseren Kreditkonditionen, führen. Der vorliegende Beitrag unter-

scheidet sich von veröffentlichten vorläufigen Ergebnissen durch einen erweiterten Untersuchungszeitraum und eine differenziertere Kontrolle von Drittvariablen [BP10].

2 Theorie

Wissenschaftler aller Disziplinen müssen einräumen, dass ein übergreifendes Verständnis der Konsequenzen von technologiebedingt immer einfacherer Erhebung, Übermittlung und Verarbeitung personenbezogener Daten fehlt. Es ist weithin unklar, wie sich die Verteilung von Informationen über Individuen in einer Gesellschaft auf Kenngrößen wie soziale Wohlfahrt oder Verteilungsgerechtigkeit auswirkt. Abgesehen von normativen Leitsätzen und der Kritik offensichtlichen Fehlverhaltens ist es deshalb schwierig, sachliche Empfehlungen zur Datenschutzregulierung zu unterbreiten. Es ist auch nicht zu erwarten, dass die Problematik hinreichend verstanden wird, bevor reale Entwicklungen Eingriffe notwendig machen. Also liegt es nahe, zunächst beherrschbarere Teilprobleme zu untersuchen.

Kreditmärkte stellen so ein Teilproblem dar, das sich aus mehreren Gründen besonders für eine Analyse eignet: Kreditmärkte nehmen eine zentrale Stellung in einem marktwirtschaftlich organisierten System ein. Weiterhin ist die Rolle von Informationen – oft personenbezogene Daten – auf Kreditmärkten in den Wirtschaftswissenschaften seit den 1970er Jahren untersucht und oft detailliert modelliert worden. Mit Online Social Lending führen technologische Innovationen außerdem zu strukturellen Veränderungen, die als exemplarisch für andere Bereiche der Gesellschaft angesehen werden können. Nicht zuletzt ist mit der Verfügbarkeit von Daten und einer klaren abhängigen Variable (Kreditkonditionen) auch eine Voraussetzung für praktische Untersuchungen erfüllt, die empirische Forschung zu Datenschutzthemen in anderen Bereichen erschwert.

2.1 Bedeutung personenbezogener Daten für Kreditmärkte

Zu den klassischen Aufgaben der Finanzintermediäre gehört die effiziente Transformation von Losgrößen, Risiken und Fristen im Kreditgeschäft. Nach der heute vorherrschenden Schule der Informationsökonomik ist zudem die Verfügbarkeit von Information entscheidend für Existenz und Effizienz von Kreditmärkten [Sti81]. Solche Informationen enthalten dabei fast immer auch personenbezogene Daten der Kreditnehmer. Während Datenschützer bereits den Umgang mit diesen Daten bei gemeinhin als vertrauenswürdig angesehenen Institutionen wie Banken oder Kreditauskunfteien kritisieren [Jen07], dürften sich diese Bedenken verschärfen, wenn beim Online Social Lending kreditrelevante personenbezogene Daten im Internet für alle potenziellen Investoren einsehbar sind.

Der Einfluss personenbezogener Daten von Kreditnehmern auf Kreditentscheidungen lässt sich theoretisch mit einer Reihe von Mechanismen erklären: *Asymmetrische Information* zum Zeitpunkt der Kreditvergabe verhindert, dass Investoren zwischen „guten“ und „schlechten“ Risiken unterscheiden können [Ake70]. Bessere Information über Kreditnehmer und deren geplante Vorhaben erlauben eine genauere Schätzung der Ausfallwahrscheinlichkeit und beeinflussen somit die Kreditkonditionen. Personenbezogene Daten der Kreditnehmer erleichtern weiterhin die Überwachung laufender Kreditverträge durch Investoren sowie den Einsatz von Rechtsmitteln im Falle eines Zahlungsausfalls. Allein die-

se Möglichkeit kann bereits Fehlverhalten (engl. *Moral Hazard*) der Kreditnehmer unterbinden. Eine ähnliche Präventivwirkung wird durch die (implizite) Mithaftung von dem Kreditnehmer nahe stehenden Personen und damit verbunden möglichen *sozialen Sanktionen* erreicht [BC95]. Das soziale Umfeld eines Kreditnehmers, beispielsweise Familienmitglieder, kann einen gewissen Druck auf den Kreditnehmer ausüben, die Raten pünktlich zurückzuzahlen, da die Reputation der gesamten Gruppe auf dem Spiel steht. Solche komplexen sozialen Mechanismen setzen voraus, dass der Kreditnehmer zuvor nicht nur Informationen über sich selbst, sondern auch personenbezogene Daten über Personen aus seinem Umfeld offenbart hat. Außerdem gilt es, indirekte Wirkungen der Kommunikation sensibler Informationen zu berücksichtigen. Der Austausch personenbezogener Daten kann stets auch als sozialer Hinweis verstanden werden und fördert den *Vertrauensaufbau* zwischen Akteuren [Kol00, PB10]. Schließlich nutzen Kreditnehmer verschiedene *Argumentationstechniken*, um potenzielle Investoren von ihrem Projekt zu überzeugen. Aussagen, die beispielsweise die Plausibilität des Vorhabens unterstreichen sollen oder an das Hilfeverhalten der Investoren appellieren, enthalten nebenbei oft personenbezogene Daten.

Obwohl diese Aufzählung unvollständig sein mag, verdeutlicht sie bereits das Spannungsfeld zwischen wirtschaftlichen Interessen und Datenschutz beim Online Social Lending. Nahe liegende Forschungsfragen betreffen daher die Analyse des Verhaltens von Akteuren, die diesem Zielkonflikt ausgesetzt sind, sowie den Entwurf von technischen und organisatorischen Maßnahmen, um ihn zu entschärfen ohne dabei den Kreditmarkt zu zerstören.

2.2 Stand der Forschung

Seit seiner Entstehung wird Online Social Lending vor allem von wirtschaftswissenschaftlicher Forschung begleitet. Datenquelle ist dabei meistens die US-amerikanische Plattform *Prosper.com*. Allerdings sind den Verfassern weder im Bezug auf *Prosper.com* noch generell andere Forschungsvorhaben bekannt, die Datenschutzaspekte bei Online Social Lending untersuchen.

In [Rav07] und [PS08] wurde die Diskriminierung bestimmter Gruppen bei der Kreditvergabe untersucht. Die Autoren finden einen Einfluss von Hautfarbe, Alter, Geschlecht und Gewicht auf Kreditkonditionen, wenn auch nicht immer statistisch signifikant. In einer weiteren Untersuchung, die der hier betrachteten Datenschutzproblematik einen Schritt näher kommt, haben Wissenschaftler den Detailgrad verschiedener Angaben in Projektbeschreibungen anhand einer dreistufigen Skala gemessen [HADL08]. Sie berichten, dass neben grundsätzlichen Parametern wie Kreditbetrag, Zinssatz und Dauer bis zur Finanzierung auch der Detailgrad zusätzlich angegebener Informationen den Finanzierungserfolg eines Projekts beeinflusst. Die Autoren unterschieden zwischen demographischen Fakten, Finanzinformationen (z. B. Kreditscore) und allgemeinen Aufwandsmaßen (z. B. Länge der Projektbeschreibung). Detaillierte Informationen der beiden letzten Kategorien haben einen nachweisbaren Einfluss auf den Erfolg eines Kreditgesuchs.

Theoretische und empirische Aspekte des Datenschutzes bei der herkömmlichen Kreditvergabe durch Geschäftsbanken werden aus Platzgründen nicht vertieft [KW06, KW08]. Für eine Zusammenfassung weiterer Arbeiten zu Online Social Lending unter Berücksichtigung von sozialen Netzwerken zwischen Akteuren sei auf [BP10] und [PB10] verwiesen.

2.3 Hypothese

Aus Perspektive des Datenschutzes ist eine der grundlegenden Forschungsfragen zu Online Social Lending die Rolle von personenbezogenen Daten in Projektbeschreibungen und deren Auswirkungen auf Kreditkonditionen. Vor dem Hintergrund der unter 2.1 genannten Mechanismen lautet die in diesem Beitrag zu untersuchende Hypothese wie folgt:

> Kreditnehmer, die mehr personenbezogene Daten veröffentlichen, erhalten ihren Kredit zu einem günstigeren Zinssatz.

Die Hypothese basiert auf der Annahme, dass freiwillig überwiegend personenbezogene Daten angegeben werden, die den Kreditnehmer in einem positiven Licht erscheinen lassen [Sti81]. Die Überprüfung der Hypothese ist mit einer Reihe von Schwierigkeiten verbunden. Eine davon ist – sowohl konzeptionell als auch praktisch – die Quantifizierung von personenbezogenen Daten in Freitexten und Bildern. Außerdem stellen personenbezogene Daten in Projektbeschreibungen „weiche“ Informationen dar, und es ist zu erwarten, dass diese im Vergleich zu „harten“ Fakten wie Kreditscore und Laufzeit einen schwächeren Einfluss haben [Pet04]. Letztere müssen also ebenfalls erfasst und beim Hypothesentest kontrolliert werden. Um den Effekt personenbezogener Daten in seiner Größenordnung einschätzen zu können, ist zusätzlich ein Vergleich mit der Wirkung anderer „weicher“ Informationen erforderlich. Dazu werden in der vorliegenden Untersuchung auch die in den Projektbeschreibungen verwendeten Argumentationstechniken berücksichtigt.

3 Datenerhebung und Auswertung

Der für die Analyse zur Verfügung stehende Datensatz besteht aus 1530 Kreditprojekten, die zwischen 01. November 2008 und 31. Oktober 2009 auf *Smava.de*, der größten deutschen Plattform für Online Social Lending, veröffentlicht wurden. Die Projekte repräsentieren ein Kreditvolumen von 13 Millionen Euro. Das entspricht etwa 50 % des Gesamtkreditvolumens, welches über die Plattform seit ihrem Start im März 2007 bis März 2010 vermittelt wurde. Der Zeitraum wurde gewählt, um einen möglichst dichten und homogenen Datensatz zu erhalten und gleichzeitig Anomalien innerhalb der relativ langen Startphase vom *Smava.de* zu vermeiden. Außerdem wurde so der kritische Bereich um den Zusammenbruch der US-Investmentbank Lehman Brothers im September 2008 von der Analyse ausgeschlossen, der sich auch auf deutsche Kreditmärkte auswirkte.

Smava.de bietet Kreditsuchenden die Möglichkeit, ihr Kreditprojekt vorzustellen und selbst einen Vorschlag zu unterbreiten, zu welchen Konditionen (Zinssatz, Laufzeit, Betrag) sie einen Kredit aufnehmen möchten. Die Betreiber der Plattform prüfen die Identität des potenziellen Kreditnehmers und ergänzen anschließend verifizierte demographische Angaben (Alter, Geschlecht, beruflicher Status) sowie den Kreditscore der SCHUFA und eine Abschätzung der Kapitaldienstfähigkeit (KDF), die sich aus dem erwarteten Verhältnis der Tilgungsrate zum Nettoeinkommen errechnet. Diese Fakten werden zusammen mit der Projektbeschreibung des Kreditnehmers und optional bereitgestellten Bildern auf der Internet-Plattform veröffentlicht. Potenzielle Investoren können alle auf der Plattform vorhandenen Kreditprojekte inklusive darin genannter Detailinformationen einsehen und zu

deren Finanzierung durch Gebote in Schritten von 250 Euro beitragen. Sobald ein Projekt vollständig finanziert ist oder nach Ablauf einer Frist von 14 Tagen wird der (Teil-)Kredit durch eine Bank vergeben, die mit *Smava.de* kooperiert. Das Einbeziehen der Bank an dieser Stelle ist aufgrund der in Deutschland geltenden gesetzlichen Bestimmungen der Bankenaufsicht notwendig. Nach Bewilligung durch ausreichend viele Investoren hat die Bank angeblich keinen Einfluss auf die Kreditentscheidung oder Konditionen. Als weiterer Partner von *Smava.de* stellt die SCHUFA für jeden potenziellen Kreditnehmer den entsprechenden Kreditscore bereit. Zur Eintreibung säumiger Tilgungsraten kooperiert *Smava.de* im Interesse der Investoren außerdem mit einem Inkasso-Unternehmen.

Sollte ein eingestelltes Kreditprojekt zunächst nur auf geringeres Interesse bei Investoren stoßen, haben Kreditnehmer die Möglichkeit, den angebotenen Zinssatz nachträglich zu erhöhen, um das Angebot attraktiver zu gestalten. Sowohl Kreditnehmer als auch Investoren sind auf der Plattform unter selbst gewählten Benutzernamen aktiv, ihre tatsächliche Identität ist den Betreibern von *Smava.de* jedoch bekannt und durch ein Postident-Verfahren im Vorfeld verifiziert.

Von den ursprünglich 1530 Kreditprojekten im Untersuchungszeitraum wurden 79 (5 %) nicht vollständig finanziert. Vor der weiteren Analyse wurden diese aus dem Datensatz entfernt, um eine bessere Vergleichbarkeit des Zinssatzes als abhängige Variable in der vorliegenden Untersuchung zu erreichen. Darüber hinaus wurden weitere 228 Projekte herausgefiltert, die innerhalb der ersten zwei Minuten nach Einstellen des Kreditgesuchs vollständig finanziert waren. Damit sollen alle so genannten „Sofortkredite“ von der Untersuchung ausgeschlossen werden. Diese Art von Krediten wird automatisiert bewilligt, d. h. „weiche“ Faktoren, deren Einfluss analysiert werden soll, spielen hier keine Rolle.

3.1 Inhaltsanalyse

Die Menge der in den einzelnen Projektbeschreibungen veröffentlichten personenbezogenen Daten wurde mit Hilfe einer Inhaltsanalyse erhoben [Hol69]. Die Kreditprojekte variierten im Detailgrad des Beschreibungstextes, den freiwilligen Angaben im Benutzerprofil sowie möglicherweise hochgeladenen Bildern. Zwei zuvor geschulte Kodierer haben unabhängig voneinander die vorhandenen Projektbeschreibungen bewertet und vorgegebenen Kategorien zugeordnet, ohne jedoch die Hypothese zu kennen. Das Codebuch dieser Inhaltsanalyse definiert zehn verschiedene Kategorien personenbezogener Daten: Name des Kreditnehmers, Kontaktinformationen (Adresse, Telefonnummer, E-Mail, etc.), finanzielle Situation, Ausbildung, Beruf, weitere besondere Fähigkeiten, Wohnsituation, Gesundheit, Hobbys und Informationen über nahe stehende Verwandte (Partner, Kinder). Jede Kategorie wurde wiederum in verschiedene Subkategorien unterteilt, um genau erfassen zu können, wie detailliert Kreditsuchende Auskunft geben. Die Kodierer bewerteten ebenso den Gesamteindruck des Kreditprojekts (positiv, neutral, negativ) auf Grundlage aller zur Verfügung stehenden Informationen. Aus forschungsökonomischen Gründen konnten 125 der verfügbaren Projekte nicht vollständig kodiert werden. Trotzdem wurde ein Teil der Kreditprojekte von beiden Kodierern bewertet, um die Inter-Koder-Reliabilität zu errechnen. Nach der bekannten Formel aus [Hol69] war diese mit durchschnittlich 90 % Übereinstimmung recht gut.

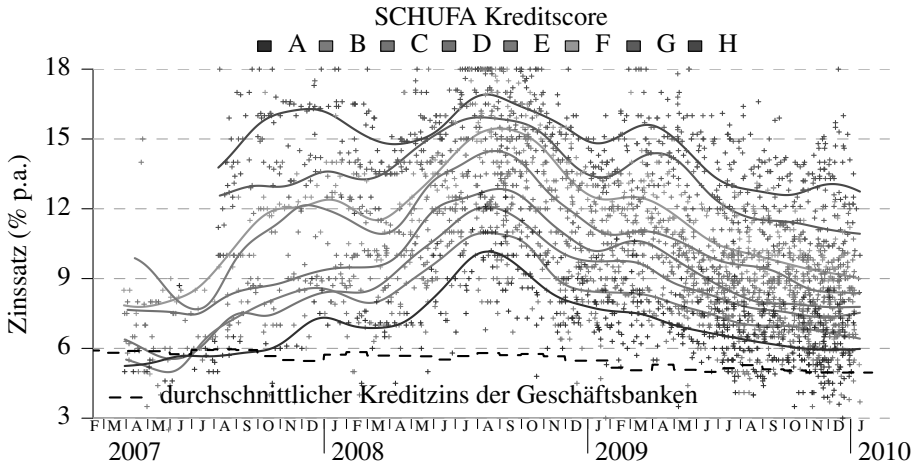


Abbildung 1: Kreditkonditionen auf *Smava.de* im Zeitverlauf, klassifiziert nach SCHUFA-Bonität

3.2 Weitere Datenquellen

Neben personenbezogenen Daten wurden für die Auswertung weitere Daten berücksichtigt, die die Höhe des Zinssatzes als abhängige Variable beeinflussen. Dazu zählen der von der Bundesbank ermittelte durchschnittliche Effektivzinssatz für Konsumentenkredite an private Haushalte [Bun] sowie generellen Schwankungen des Zinsniveaus auf *Smava.de*. Um die Auswirkungen dreier grundlegender Änderungen der Plattform zu kontrollieren, wurden entsprechende Dummy-Variablen generiert. Das erste Ereignis im Untersuchungszeitraum war die Erhebung von Gebühren für Kreditnehmer und Investoren im Februar 2009. Im Mai 2009 erfolgte eine Erweiterung der Funktionalitäten des Gebotsassistenten. Dieser steht Investoren zur Verfügung und ermöglicht das automatische Bieten auf Kreditprojekte mit zuvor spezifizierten Eigenschaften. Dabei ist zu beachten, dass neu eingestellte Kreditvorhaben maximal zu 50 % auf diese Weise finanziert werden können, die übrige Hälfte muss durch manuelle Gebote aufgefüllt werden. Die dritte und im relevanten Zeitraum letzte Neuerung war die Einführung von Sofortkrediten im Juli 2009. Sofortkredite werden zu 100 % automatisch mit Hilfe der Gebotsassistenten finanziert und sind deshalb für die vorliegende Untersuchung ausgeschlossen worden. Die Dummy-Variable ist dennoch notwendig, um einen möglichen Effekt dieses neuen Angebots auf das Zinsniveau der Plattform insgesamt zu erfassen.

4 Ergebnisse

Abbildung 1 zeigt die Zinssätze aller über *Smava.de* finanzierten Kreditprojekte im Zeitverlauf bis Januar 2010. Jedes Projekt entspricht einem Punkt in der Wolke, die farbliche Kodierung markiert die SCHUFA-Klasse von A (gute Bonität) bis H (schlecht). Die durchgezogenen Kurven sind geglättete Durchschnittszinssätze für die entsprechenden

SCHUFA-Klassen und die gestrichelte Linie zeigt den durchschnittlichen Effektivzinssatz der Geschäftsbanken für vergleichbare Kredite [Bun]. Die SCHUFA-Klassen G und H wurden erst ab Mitte 2007 zugelassen. Die Abbildung verdeutlicht die generell hohe Varianz der Kreditkonditionen sowie deren Fluktuation im Zeitverlauf. Außerdem ist ersichtlich, dass ein Großteil der Varianz durch „harte“ Information, wie die SCHUFA-Klasse, erklärt werden kann. Trotzdem verbleibt ein substanzieller Anteil an Streuung innerhalb jeder SCHUFA-Klasse, der Spielraum für Effekte der zu untersuchenden „weichen“ Informationen lässt. Weiterhin fällt auf, dass das Zinsniveau auf *Smava.de* selbst für Kredite mit guter Bonität sichtbar höher ist als das Zinsniveau der Geschäftsbanken. Im zeitlichen Verlauf lassen sich keine Korrelationen zwischen dem Zinsniveau des traditionellen und des neuen Kreditmarkts erkennen.

Die Wirkung personenbezogener Daten auf den Zinssatz aller vollständig finanzierten Kreditprojekte wurde mit Hilfe einer Regressionsanalyse geschätzt. Tabelle 1 zeigt zwei Spezifikationen mit geschätzten Koeffizienten und Tests auf statistische Signifikanz. In Modell 1 wurden nur „harte“ Informationen als Prädiktoren zugelassen und Modell 2 erweitert die Regression um „weiche“ Einflussfaktoren, wie die freiwillige Veröffentlichung personenbezogener Daten und die Verwendung bestimmter Argumentationstechniken. Mit korrigiertem R^2 von über 98 % erklären die Modelle einen extrem hohen Anteil der Zinsunterschiede, allerdings ist die in Modell 2 zusätzlich erklärte Varianz sehr gering.

Es ist wenig überraschend, dass der Kreditscore am meisten Einfluss auf den Zinssatz hat. Da die Modelle ohne Konstante geschätzt wurden, können die Schätzer der SCHUFA-Klassen direkt als durchschnittliche Zinssätze für die jeweilige Bonität interpretiert werden. Der Zinssatz wird signifikant nach oben korrigiert für Kreditnehmer, deren KDF größer als 40 % ist. Zum Schutz von Kreditnehmern und Investoren erlaubt *Smava.de* keine Kreditprojekte, bei denen dieses Verhältnis über 67 % liegen würde. Die Kontrolle der Auswirkungen von Veränderungen an der Plattform zeigt, dass nach der Modifikation des Gebotsassistenten der durchschnittliche Zinssatz für alle Kreditprojekte gesunken ist. Gleiches gilt für die Einführung der Sofortkredite. Die Gebührenerhöhung hatte keinen messbaren Einfluss, was aber auch an der Konfundierung mit einem Fixed-Effect zum gleichen Zeitpunkt liegen könnte. Bezüglich der monatlichen Schwankungen des Zinssatzes im Untersuchungszeitraum ist festzustellen, dass es im März 2009 zunächst eine punktuelle Steigerung gab und ab Juli 2009 der Zinssatz für das letzte Quartal signifikant gesunken ist. Grund für Letzteres könnte eine leichte Entspannung an den Kreditmärkten sein (die sich jedoch nicht in der offiziellen Zinsstatistik niederschlägt). Eine weitere Erklärung ist die zunehmende Berichterstattung über *Smava.de* als Anlagealternative, welche zu einem Überangebot durch das Hinzukommen von mehr neuen Investoren als Kreditnehmern führte.

Bei den Merkmalen der Projektbeschreibungen zeigt sich, dass höhere Kreditsummen und längere Laufzeiten mit einem höheren Zinssatz einher gehen. Anders als in [HADL08] berichtet, wirkt sich eine (Aufwärts-)Revision des angebotenen Zinssatzes nicht negativ auf die Kreditkonditionen aus. Übereinstimmend mit der Literatur finden wir eine signifikante Verbesserung der Kreditkonditionen, wenn das Projekt mit einem individuellen Bild illustriert wurde. Bilder im nur einen Klick weiter entfernten Benutzerprofil haben dagegen keinen messbaren Einfluss. Projekte mit hohen Geboten erhalten signifikant bessere

Konditionen, allerdings ist diese Variable teilweise endogen und ein Kausalschluss damit unzulässig. Sie wurde trotzdem in die Spezifikation einbezogen, um Heterogenität aus dem Datensatz zu entfernen. Da die Variable auch als Proxy für über „weiche“ Informationen vermittelte Qualitäten des Projekts interpretiert werden kann, wurde sie in einer separaten Analyse ausgeschlossen. Die Ergebnisse für die zusätzlichen Prädiktoren in Modell 2 blieben davon unberührt. Im Abschnitt zur Demographie zeigt sich erwartungskonform eine Benachteiligung von besonders alten oder besonders jungen Kreditnehmern. Der Alterskoeffizient im Modell wurde als absolute Abweichung in Jahren vom Median aller Kreditnehmer berechnet. Dieser lag bei 43 Jahren. Es zeigt sich keine geschlechterspezifische Benachteiligung, allerdings müssen Selbstständige tendenziell mehr Zinsen bezahlen.

Prädiktor	Modell 1		Modell 2	
Kreditscore				
SCHUFA Klasse A (gut)	5.90 ***	(0.813)	6.16 ***	(0.821)
SCHUFA Klasse B	6.68 ***	(0.813)	6.96 ***	(0.821)
SCHUFA Klasse C	7.71 ***	(0.811)	7.99 ***	(0.819)
SCHUFA Klasse D	8.22 ***	(0.810)	8.47 ***	(0.817)
SCHUFA Klasse E	9.36 ***	(0.819)	9.59 ***	(0.827)
SCHUFA Klasse F	10.09 ***	(0.815)	10.38 ***	(0.824)
SCHUFA Klasse G	11.93 ***	(0.811)	12.18 ***	(0.819)
SCHUFA Klasse H (schlecht)	13.33 ***	(0.815)	13.62 ***	(0.824)
Kapitaldienstfähigkeit (KDF)				
20–40 %	0.21	(0.138)	0.21	(0.138)
40–60 %	0.30 **	(0.132)	0.27 **	(0.132)
60–67 %	0.63 ***	(0.133)	0.61 ***	(0.133)
Zeitabhängige Faktoren				
Zinssatz der Geschäftsbanken	0.55	(1.130)	0.40	(1.142)
Dummy für Gebühren (1. Feb. 2009)	0.76	(0.704)	0.57	(0.707)
Dummy für Gebotsassistent (6. Mai 2009)	−0.65 **	(0.299)	−0.70 **	(0.300)
Dummy für Sofortkredite (16. Juli 2009)	−0.36 *	(0.187)	−0.32 *	(0.187)
Fixed-Effect: Nov. 2008	1.19	(1.101)	1.18	(1.113)
Dez. 2008	1.04	(0.893)	0.98	(0.903)
Jan. 2009	0.22	(0.868)	0.11	(0.876)
Feb. 2009	−0.24	(0.288)	−0.20	(0.289)
März 2009	0.72 **	(0.364)	0.81 **	(0.367)
Mai 2009	−0.01	(0.302)	0.04	(0.301)
Juni 2009	−0.46	(0.460)	−0.43	(0.466)
Juli 2009	−1.55 ***	(0.475)	−1.45 ***	(0.476)
Aug. 2009	−2.18 **	(0.917)	−2.06 **	(0.922)
Sep. 2009	−1.42 ***	(0.408)	−1.39 ***	(0.408)
Okt. 2009	−1.70 ***	(0.395)	−1.66 ***	(0.395)

... Tabelle wird auf der nächsten Seite fortgesetzt ...

Tabelle 1: Regressionsanalyse (OLS), abhängige Variable: Zinssatz p. a.

Prädiktor	Modell 1		Modell 2	
Merkmale der Projektbeschreibung				
Laufzeit (Dummy für 60 statt 36 Monate)	0.49^{***}	(0.082)	0.47^{***}	(0.082)
Betrag (Vielfaches von 250 Euro)	0.01^{***}	(0.002)	0.01^{***}	(0.002)
durchschnittliches Gebot (Schrittw. 250 Euro)	-0.14^{***}	(0.049)	-0.14^{***}	(0.049)
gewerbliches Kreditprojekt	-0.00	(0.103)	0.06	(0.108)
Revision des angebotenen Zinssatzes	-0.09	(0.090)	-0.06	(0.090)
Individuelles Projektbild vorhanden	-0.18^{**}	(0.090)	-0.15^{*}	(0.091)
Bild im Benutzerprofil vorhanden	0.11	(0.124)	0.10	(0.132)
Demographische Angaben				
Selbstständige Tätigkeit	0.13	(0.097)	0.18^{*}	(0.097)
Alter (absolute Abweichung vom Median)	0.02^{***}	(0.004)	0.02^{***}	(0.005)
Geschlecht (weiblich)	-0.00	(0.078)	-0.02	(0.080)
Angabe weiterer personenbezogener Daten				
Benutzerprofil ausgefüllt			0.12	(0.112)
Name und Kontaktinformationen [0..20]			0.00	(0.038)
Finanzielle Situation [0..13]			0.00	(0.029)
Beruf und Qualifikationen [0..23]			-0.05^{***}	(0.016)
Wohnsituation [0..4]			-0.02	(0.032)
Gesundheit [0..5]			-0.00	(0.067)
Hobbys [0..4]			0.02	(0.043)
Partner, Familienmitglieder [0..19]			-0.00	(0.018)
Stilmittel und Argumentation				
Schilderungen, die Mitleid hervorrufen			0.42^{**}	(0.188)
Direkter Aufruf zu helfen			-0.22^{*}	(0.122)
Eigene Hilfsbereitschaft (Reziprozität)			-0.09	(0.177)
Aussicht auf Bankkredit (Konkurrenz)			-0.22^{*}	(0.128)
Schulden bereits angefallen (Dringlichkeit)			0.16^{*}	(0.082)
Modellgüte: korrigiertes R^2 (Fälle)	98.72	(1098)	98.74	(1098)

Standardfehler in Klammern; Signifikanzniveaus: $^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$

Tabelle 1: (fortgesetzt) Regressionsanalyse, abhängige Variable: Zinssatz

4.1 Wirkung personenbezogener Daten

Die freiwillige Angabe personenbezogener Daten in den Projektbeschreibungen wurde nicht nur dichotom kodiert, sondern wie in Abschnitt 3.1 beschrieben, in verschiedenen Kategorien und Teilkategorien erfasst. Die Anzahl der Teilkategorien ist in Tabelle 1 in eckigen Klammern angegeben. Im Widerspruch zur postulierten Hypothese zeigt sich, dass die Veröffentlichung personbezogener Daten kaum Einfluss auf die Höhe des Zinssatzes hat. Der durchschnittliche Zinssatz ist etwas niedriger, wenn Kreditnehmer detaillierte An-

gaben zu ihrer Ausbildung, ihrem Beruf sowie sonstigen Qualifikationen und Fähigkeiten machen. Bezüglich der Verfügbarkeit von personenbezogenen Daten aus anderen Kategorien konnte kein Effekt gefunden werden. Damit ist die Hypothese mit den Daten dieser Untersuchung kaum vereinbar.

4.2 Argumentationstechnik und Plausibilitätsprüfung

Außer der generellen Ungültigkeit der Hypothese könnte dieser schwache Befund auch durch Schwierigkeiten bei der Messung von „weichen" Informationen oder durch unzureichende Kontrolle von Drittvariablen entstehen. Um Ersteres zu prüfen, wurden neben personenbezogenen Daten auch Stilmittel und Argumentationstechniken in den Projektbeschreibungen in das Regressionsmodell einbezogen.[1] Hier konnten deutlichere Effekte von „weichen" Informationen gemessen werden: Kreditnehmer, die direkt an das Hilfeverhalten der Investoren appellieren und solche, die auf die Möglichkeit hinweisen, alternativ einen Bankkredit zu bekommen, erhalten bessere Kreditkonditionen. Die Konditionen verschlechtern sich dagegen, wenn Kreditnehmer mit emotionalen Argumenten übertreiben und offensichtlich Mitleid bei den Kreditnehmern erregen wollen. Geht aus der Projektbeschreibung hingegen hervor, dass die Schulden bereits angefallen sind und der Kredit deshalb unbedingt notwendig ist, resultiert das in einem höheren Zinssatz. Interessant ist, dass Effekte in beide Richtungen sowohl bei eher rationalen (Konkurrenz, Dringlichkeit) als auch bei eher emotionalen Argumentationstechniken (Hilfsbereitschaft, Mitleid) beobachtbar sind. Lediglich das Betonen der eigenen Hilfsbereitschaft der Kreditnehmer und damit verbunden die Hoffnung auf reziprokes Verhalten [Kol00] hat keinen Effekt.

Eine Grundannahme der vorliegenden Untersuchung ist, dass Kreditnehmer grundsätzlich Daten veröffentlichen, die über die absolut notwendigen Angaben hinausgehen. Eine Überprüfung dieser Annahme bestätigt, dass praktisch alle Kreditprojekte eine spezifische Projektbeschreibung haben und dass 93 % dieser Projektbeschreibungen mindestens 50 Zeichen enthalten. 83 % der Projektbeschreibungen enthalten freiwillig angegebene personenbezogene Daten. Zur Kontrolle der Qualität der angegebenen personenbezogenen Daten haben die Kodierer den Gesamteindruck eines Projekts, basierend auf allen verfügbaren Informationen des Kreditnehmers und der Projektbeschreibung, bewertet. Wie die Ergebnisse in Tabelle 2 belegen, vermittelt die Mehrheit aller Projekte mit zusätzlichen personenbezogenen Daten einen positiven Eindruck. Wurden dagegen freiwillig keine personenbezogenen Daten in der Beschreibung genannt, wird der Eindruck im Durchschnitt nur als neutral bewertet. Dieser Unterschied ist statistisch signifikant. Das bedeutet, Kreditnehmer veröffentlichen vor allem solche personenbezogenen Daten, die sie selbst in einem positiven Licht erscheinen lassen und die zu einem positiven Gesamteindruck des Projekts bei Investoren beitragen. Damit kann ausgeschlossen werden, dass Kreditsuchende einfach die „falschen" personenbezogenen Daten auswählen und deshalb die Veröffentlichung von Details nicht zu besseren Kreditkonditionen führt.

Trotz der nicht zu unterschätzenden Schwierigkeiten bei der Quantifizierung von personenbezogenen Daten aus unstrukturiertem Textmaterial ist es also unwahrscheinlich, dass der Befund allein auf Messfehler zurückzuführen ist. Damit bleibt die Verbesserung der

[1] Ein schrittweiser Einschluss der Prädiktorblöcke ändert an den Ergebnissen nichts.

	Beschreibung enthält zusätzliche personenbezogene Daten?	
Gesamteindruck	nein ($n = 163$)	ja ($n = 627$)
positiv	19 %	66 %
neutral	52 %	28 %
negativ	28 %	5 %
		$\chi^2 = 141$, df $= 2$, $p < 0.001$

Tabelle 2: Gesamteindruck der Kreditprojekte (Kontrollvariable kodiert für 790 Fälle)

Drittvariablenkontrolle als Aufgabe für weitere Forschung. Explorative Analysen deuten darauf hin, dass sich in den Daten der dynamisch wachsenden Plattform *Smava.de* noch einige unerklärte Heterogenität befindet, die subtile Wirkungen personenbezogener Daten überdecken könnte.

5 Diskussion und Zusammenfassung

Dieser Beitrag dokumentiert einen ersten Versuch, Konflikte zwischen Datenschutzzielen und wirtschaftlichen Interessen am Beispiel von alternativen Kreditmärkten systematisch zu untersuchen. Der Nutzen von freiwilliger Veröffentlichung personenbezogener Daten zur Verbesserung der Kreditkonditionen wurde mit empirischen Daten der größten deutschen Plattform für Online Social Lending analysiert. Die Ergebnisse zeigen, dass Kreditnehmer, die Details bezüglich ihrer Ausbildung, Berufs sowie weiterer Qualifikationen angeben, tatsächlich etwas weniger Zinsen bezahlen und so im Mittel einen kleinen finanziellen Vorteil erzielen. Weitere personenbezogene Daten verbessern die Kreditkonditionen dagegen nicht messbar. Einer Offenlegung personenbezogener Daten gegenüber allen potenziellen Investoren, im Fall von Online Social Lending sind das derzeit *alle Internetnutzer*, stehen offenbar nur sehr geringe wirtschaftliche Vorteile gegenüber. Gleichzeitig birgt dieser freigiebige Umgang mit sensiblen Daten jedoch eine Reihe von Datenschutzrisken (Datenmissbrauch, Stalking, etc.). Kreditnehmer sollten also genau prüfen, welche personenbezogenen Daten sie im Internet einer sehr breiten Öffentlichkeit zugänglich machen. Mittelfristig sind Wissenschaftler sowie Anbieter von Online Social Lending aufgefordert, spezifische datenschutzfreundliche Technologien zu entwickeln und einzusetzen, die die Veröffentlichung von personenbezogenen Daten minimieren, ohne den Marktmechanismus irreparabel zu beschädigen.

Nicht zuletzt verspricht Online Social Lending, Finanzintermediation zu revolutionieren und Zugang zu Kredit zu „demokratisieren“. Es sei dahingestellt, ob diese Vorstellungen realistisch sind. Es wird jedoch deutlich, dass die Reduzierung von Ungleichheiten beim Zugang zu Kredit neue Ungleichheiten bei der Realisierung von Datenschutzzielen zur Folge hat: Wirtschaftlich Schwächere werden häufiger als Kreditnehmer auftreten und im Gegensatz zu Investoren mehr Auskunft über sich und ihr Leben erteilen müssen. Privatsphäre bleibt also vermutlich auch durch diesen Mechanismus ein „Luxusgut“ [BK07].

Danksagung Wir danken Alessandro Acquisti, Michael Ehrmann und Dagmar Schönfeld für ihre wertvollen Kommentare zu früheren Fassungen dieses Manuskripts sowie Ulrike Jani, Kristina Schäfer, Michael Sauer und Hagen Wahring für ihre Hilfe bei der Datenerhebung. Der Erstautor wurde durch ein Postdoc-Stipendium des Deutschen Akademischen Austauschdienstes (DAAD) gefördert. Die vorliegende Arbeit wurde zum Teil durch die Europäische Kommission innerhalb des siebten Rahmenprogrammes (FP7/2007-2013, Vertragsnr.: 216483) finanziell unterstützt.

Literatur

[Ake70] G. A. Akerlof. The Market for "Lemons": Quality Uncertainty and the Market Mechanism. *The Quarterly Journal of Economics*, 84(3):488–500, 1970.

[BC95] T. Besley und S. Coate. Group lending, repayment incentives and social collateral. *Journal of Development Economics*, 46:1–18, 1995.

[BG09] S. Berger und F. Gleisner. Emergence of Financial Intermediaries in Electronic Markets: The Case of Online P2P Lending. *BuR - Business Research*, 2(1):39–65, 2009.

[BK07] R. Böhme und S. Koble. On the Viability of Privacy-Enhancing Technologies in a Self-regulated Business-to-consumer Market: Will Privacy Remain a Luxury Good? *Workshop on the Economics of Information Security (WEIS)*, Carnegie Mellon University, Pittsburgh, PA, 2007.

[BP10] R. Böhme und S. Pötzsch. Privacy in Online Social Lending. In *AAAI Spring Symposium on Intelligent Privacy Management*, Seiten 23–28, Stanford University, 2010.

[Bun] Deutsche Bundesbank. Zeitreihe SUD114.

[CGL09] N. Chen, A. Ghosh und N. Lambert. Social Lending. In *ACM Conference on Electronic Commerce*, Seiten 335–344, Stanford University, 2009. ACM Press.

[FS08] A. Frerichs und M. Schuhmann. *Peer to Peer Banking – State of the Art*. Arbeitsbericht Nr. 02/2008, Institut für Wirtschaftsinformatik, Universität Göttingen, 2008.

[HADL08] M. Herzenstein, R. Andrews, U. Dholakia und E. Lyandres. The Democratization of Personal Consumer Loans? Determinants of Success in Online Peer-to-Peer Lending Communities, June 2008. `http://ssrn.com/abstract=1147856`.

[Hol69] O. R. Holsti. *Content analysis for the social sciences and humanities*. Addison-Wesley, 1969.

[Jen07] N. Jentzsch. *Financial Privacy*. Springer-Verlag, Berlin Heidelberg, 2007.

[Kol00] S.-C. Kolm. The Logic of Good Social Relations. *Annals of Public and Cooperative Economics*, 71(2):171–189, 2000.

[KW06] M. Kamp und T. Weichert. *Scoringsysteme zur Beurteilung der Kreditwürdigkeit. Chancen und Risiken für Verbraucher*. Unabhängigen Landeszentrum für Datenschutz Schleswig-Holstein (ULD), Kiel, 2006.

[KW08] D. Korczak und M. Wilken. *Scoring im Praxistest: Aussagekraft und Anwendung von Scoringverfahren in der Kreditvergabe und Schlussfolgerungen*. GP Forschungsgruppe, München, 2008.

[PB10] S. Pötzsch und R. Böhme. The Role of Soft Information in Trust Building: Evidence from Online Social Lending. In A. Acquisti, S.W. Smith und A.-R. Sadeghi, Hrsg., *TRUST 2010*, LNCS 6101, Seiten 381–395, Berlin Heidelberg, 2010. Springer-Verlag.

[Pet04] M. A. Petersen. Information: Hard and Soft. Mimeo, 2004.

[PS08] D. Pope und J. Syndor. What's in a Picture? Evidence of Discrimination from Prosper.com, 2008. `http://wsomfaculty.case.edu/sydnor/prosperpaper.pdf`.

[Rav07] E. Ravina. Beauty, Personal Characteristics and Trust in Credit Markets, Dezember 2007. `http://ssrn.com/abstract=972801`.

[Sti81] J. E. Stiglitz. Credit Rationing in Markets with Imperfect Information. *American Economic Review*, 71(3):393–410, 1981.

Security Analysis of OpenID

Pavol Sovis, Florian Kohlar, Joerg Schwenk
{*vorname.nachname*}@rub.de
Ruhr-University Bochum
Bochum, Germany

Abstract: OpenID is a user-centric and decentralized Single Sign-On system. It enables users to sign into Relying Partiesby providing an authentication assertion from an OpenID Provider. It is supported by many leading internet companies and there are over a billion accounts capable of using OpenID. We present a security analysis of OpenID and the corresponding extensions and reveal several vulnerabilities. This paper demonstrates how identity information sent within the OpenID protocol can be manipulated, due to an improper verification of OpenID assertions and no integrity protection of the authentication request.

1 Introduction

The web applications' de facto standard is the "username/password" authentication over an TLS/SSL [DA99, FKK96] secured connection. However, this mechanism is an unacceptable solution for the internet today, because it leads to several security problems, the number of usernames and passwords to remember being the worst. It leads to forgotten passwords, resulting in the need of a password renewal over e-mail and/or low-entropy passwords, which are easy to remember (and easy to guess as well). In order to solve these problems, Single Sign-on systems have been introduced and their goal is to authenticate a user only once and obviating the need of re-authentication. A prominent player on this field is OpenID [FRHH07]. It authenticates a user against a web application using an authentication assertion gained by a trusted third party. The biggest supporters of OpenID count Google, Microsoft, Yahoo, MySpace, Verisign, GMX/Web.DE or France Telecom and this provides for a solid user-base of more than a billion OpenID-capable users worldwide. Besides, many companies already support OpenID authentication, including Facebook, Sears, KMart or LiveJournal.

2 Related Work

Microsoft Passport has been analyzed by Kormann and Rubin [KR00] and several weaknesses have been found. They are based on redirecting the user to a bogus Passport server,

either by deploying a fake merchant site and luring unsuspicious users to this site (e.g. through phishing attacks) or actively by attacking the DNS or modifiying a response from a legitimate service provider. The bogus server may act as a proxy and thus obtain the user's credentials. Microsoft Cardspace - the successor of MS Passport - was analyzed by Gajek, Schwenk, Steiner and Chen [GSSX09], who were able to steal a user's security token and subsequently impersonate him. SAML, an XML-based standard also used widely to encorporate Single Sign-On, was analyzed by Groß [Gro03], who intercepted the authentication token from a referer tag by redirecting the user to a server under the adversary's control. His analysis led to a SAML revision, which was later proven by Groß and Pfitzman [GP06] to be also vulnerable. Pfitzman cooperated with Waidner [PW03] on an analysis of another SAML based Single Sign-On system - the Liberty Single Sign-On protocol - and found similar flaws.

OpenID has not yet been examined with respect to security thoroughly. Eugene and Vlad Tsyrklevich [TT07] presented several OpenID Authentication 1.0 related attacks at the Black Hat USA 2007 Conference and pointed out phishing and unauthenticated Die-Hellman Key Exchange as the biggest shortcomings of OpenID. Shakir James [Jam] analyzed Web Single Sign-On Systems in his report and again identified phishing as the major security issue regarding OpenID and called attention to the lack of security related material in the documentation of the OpenID Suite. Newman and Lingamneni [NL08] have conducted an attack which results in the victim being logged in at the Relying Party as an adversary (Session Swapping). This is possible due to the lack of any bond between a positive authentication assertion from the OpenID Provider and the victim's User agent. Barth, Jackson and Mitchell [BJM08] proposed a mechanism to mitigate this attacks by a secret token validation technique, where the Relying Party generates a fresh nonce at the start of each protocol flow, stores it in the user's browser-cookies and simultaneously appends it to the authentication request. The OpenID Provider returns the nonce in the authentication response. The user's cookie must match the nonce in this response in order for the User to become authenticated at the Relying Party.

3 OpenID Authentication 2.0

3.1 OpenID Roles

- The **User** is an entity wanting to authenticate against a Relying Party with his digital identity.
- The **Identifier** is generally a url, representing the User. It points to a resource, which holds information such as the User's OpenID Provider url, version of OpenID which the OpenID Provider is compatible with etc.
- The **Relying Party** is an entity accepting an assertion from an OpenID Provider, representing a digital identity of a specific User.
- The **OpenID Provider** or **Identity Provider** (interchangeable terms) is responsible

for authenticating the User against a Relying Party, therefore it is the trusted third party on which the User as well as the Relying Party rely. In order to do so, the User must authenticate against the OpenID Provider first and so prove his digital identity. This identity is then used to sign-in the User at the Relying Partyby accepting a security assertion from the OpenID Provider.

- The **Identifier host** is the host, where the Identifier-describing resource resides.

3.2 How does OpenID work?

OpenID is a suite of protocols, which enables users to authenticate against multiple web applications (Relying Parties) using only a single identity. In order to do this, the user must create such an identity at an OpenID Provider of his choice, link this identity to any Relying Party and use it afterwards as a key, proving his identity at the Relying Party. The concept of identity linking (shown in Figure 1) is a mechanism to create a trust relationship between the Relying Party and an OpenID Provider. Afterwards, the Relying Party recognizes the user by his OpenID Provider-identity. The OpenID Provider-identity is transported to the Relying Party in form of an assertion from the OpenID Provider.

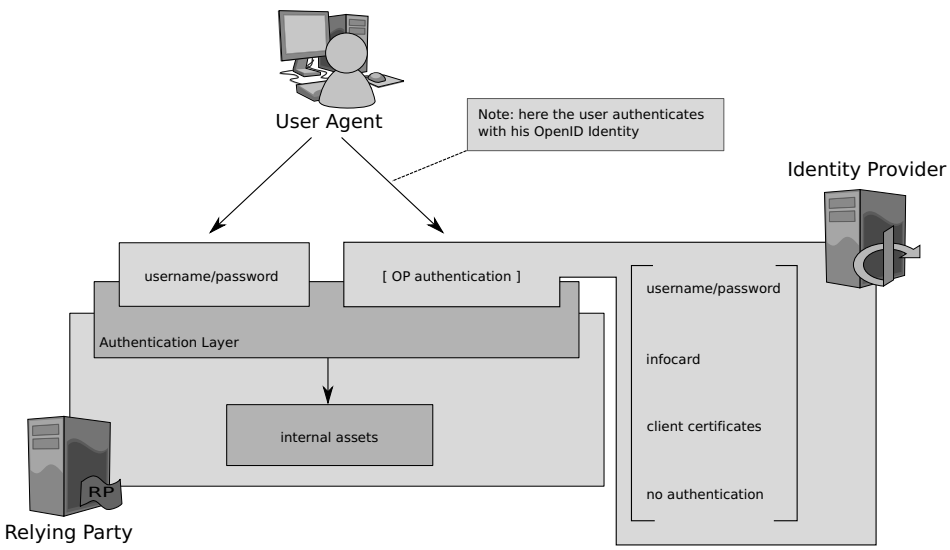


Figure 1: The OpenID Authentication Concept

A typical OpenID Authentication 2.0 Protocol Protocol flow corresponds to Figure 2 and runs as follows:

1. OpenID Authentication 2.0 Protocol Protocol is initiated by the User by requesting

the Relying Party's site.

2. The Relying Party responds with its login page presenting an input field for an Identifier.
3. The User enters his Identifier and submits the login page form, i.e. requests OpenID Authentication 2.0 Protocol.
4. The Relying Party performs discovery upon the received Identifier i.e. retrieves the data resource held at an Identifier's url and
5. subsequently receives metadata representing the User and his OpenID Provider.
6. Based upon metadata from the previous step, the Relying Party requests an association from the OpenID Provider, i.e. requests to exchange a shared secret.
7. The OpenID Provider responds with a shared key, which is encrypted (either using HTTPS as transport protocol or using Die-Hellman Key Exchange).
8. The Relying Party then redirects the User to the OpenID Provider by sending a HTTP(S) response with the redirect header pointing to the OpenID Provider's endpoint.
9. The User is presented with a login form at the OpenID Provider.
10. The User fills out the login form and submits it, hence authenticating against the OpenID Provider.
11. The OpenID Provider verifies the User's credentials and, if these are valid, redirects the User to the Relying Party along with the authentication result (MAC-protected by the previously established shared key). Again, this is done using a HTTP(S) redirect with the 'Location:' header pointing to the Relying Party's endpoint. The assertion in this request to the Relying Party indicates the login success from the OpenID Provider and the MAC ensures the integrity of the response.
12. According to the OpenID Provider's response, the User is either authenticated against the Relying Party or presented with an adequate error message.

The transport protocol used in OpenID Authentication 2.0 Protocol flow is either HTTP or HTTPS (HTTP used within a TLS/SSL secured channel). Independent of the method used (POST, GET), OpenID facilitates a key-value representation of its payload, e.g. "$openid.claimed\_id = http://sovo.myopenid.com$" or "$mode : error$". We refer to such pairs as OpenID parameters. OpenID uses `HTTP 302` status codes to redirect the user from the Relying Party to the OpenID Provider and vice versa. An example (based on Figure 2) of such redirection is given in the following listing:

1. The user initiates a HTTP request in step 3.
2. The user obtains a response to this request in step 8, comprising the HTTP 302 status code as well as the "Location:" header set to the desired destination, while the OpenID parameters are a part of the URL in this header.

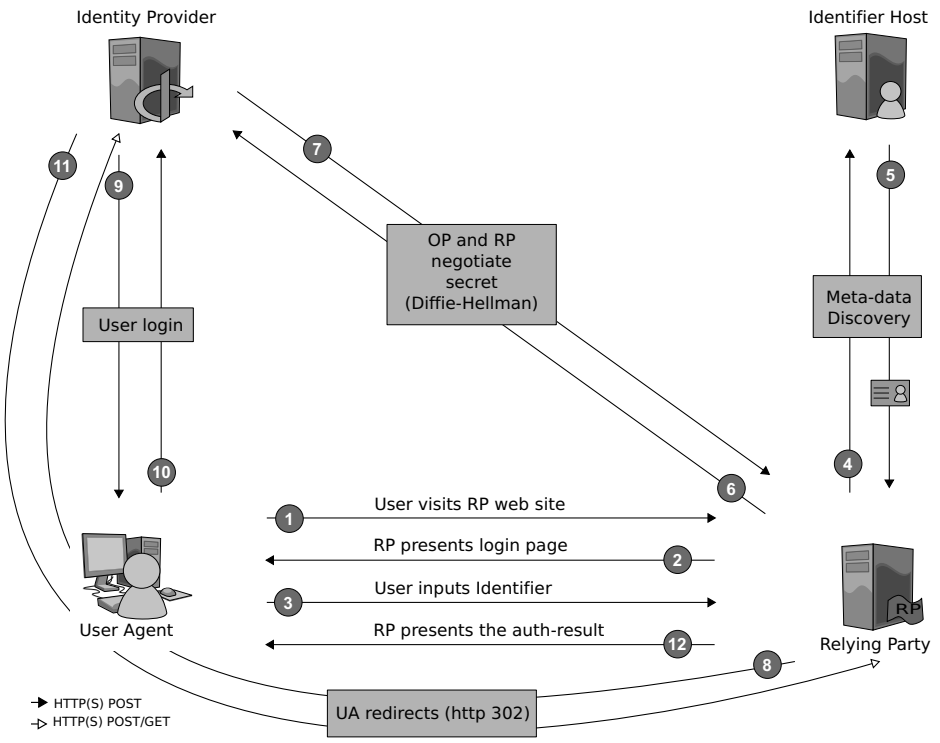


Figure 2: Typical OpenID Protocol Flow

3. The user requests the URL contained in the ”Location:” header from step 8.

An analog procedure is used to redirect the user back to the Relying Party. The security of OpenID messages can be divided into transport layer security (i.e. either using HTTP or HTTPS) and message level security (e.g. message authentication codes or MACs). A MAC is a hash-value generated over a specified list of parameter values xor-ed with the shared secret (pre-established in steps 6 and 7 in Figure 2), thus providing integrity of the OpenID message. Due to compatibility with specific OpenID flows which are not discussed in this paper, the only message in the whole OpenID flow, which is secured by the MAC, is the message from step 11 (see Figure 2).

3.3 Extensions

The basic OpenID parameters, which are compulsory and neccessary in any valid assertion, form the minimum (later also refered to as ”void”) assertion representing a positive or negative result of authentication at the OpenID Provider. However, OpenID al-

lows for extra identity information, such as email, name, date of birth and even self-defined parameters. These are facilitated through so called extensions. These can be appended to the compulsory parameters via a mechanism very similar to XML namespaces. Whereas standard parameters are key-value pairs in the form "[openid-prefix].[parameter-name]=[parameter-value]", e.g. "openid.identity=fooname.fooprovider.com", extensions must be defined first by a namespace of the form "[openid-prefix].[openid-extension-alias]=[extension-url]". For instance, a namespace may be set by "$openid.ns.sreg = http : //openid.net/extensions/sreg/1.1$" as is the case with OpenID Simple Registration Extension 1.0 [HDR06], and any parameters within this extension can be further addressed with help of the previously defined alias, e.g. "$openid.sreg.email = my@example.email$". Whereas OpenID Simple Registration Extension 1.0 can be used to send only a small set of predefined attributes, OpenID Attribute Exchange 1.0 [HBH07] allows to send custom attributes, which makes OpenID very flexible.

4 OpenID Security Analysis

In this section, we discuss several shortcomings that exist, if HTTP endpoints are used at the Relying Party and the OpenID Provider, though the User experiences HTTPS indicators at both of these parties.

4.1 Wrong Approaches on Transport Security

The endpoints of many OpenID Providers or Relying Parties are strictly HTTPS based. The problem is, if they are addressed via HTTP, they simply redirect the request to the HTTPS equivalent and proceed with the protocol flow (see Figure 3). This section comprises the dangers of such a workaround. The User's Identifier is responsible for the OpenID Provider's endpoint. In general, the User is given his identifier by the OpenID Provider, hence the OpenID Provider is overall responsible for the HTTP/HTTPS nature of its endpoint. Furthermore, the Relying Party sending an authentication request to the OpenID Provider is responsible for the $return\_to$ parameter representing the Relying Party's endpoint, where the User will later be redirected to. If both of these endpoints are HTTP URLs, then both of the User's redirects (steps 8 and 11 in Figure 2 or 9 and 15 in 3) are subject to forgery. The fact, that both of these parties may only allow communication over a TLS/SSL secured channel yields a false impression of security from the user's point of view.

The individual steps (of which 8,9 and 14,15 represent the redirects) represent the following workflow:

1 The User visits *http://www.plaxo.com* and klicks on the **Sign in** link.

2 The User receives a response redirecting him to *https://www.plaxo.com/signin?r=/events*.

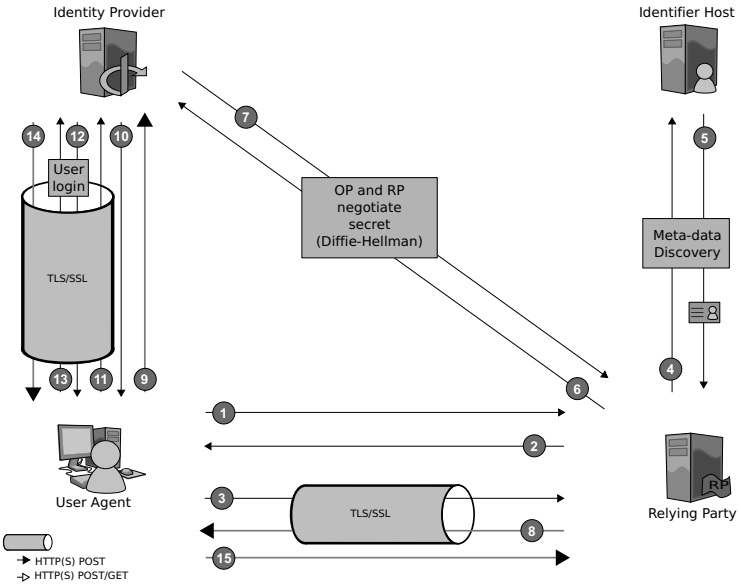


Figure 3: Reducing complexity to HTTP

3 The User chooses to sign in with OpenID, inserts his OpenID Identifier, and subsequently submits the form.

4,5 The Relying Party receives the User's meta-data and searches for the corresponding OpenID Provider.

6,7 An association must be made prior to exchanging a secret. This is done only once and left out in later iterations of the protocol flow.

8 The User receives a response within a secured channel, i.e. using the HTTPS protocol, with the location header pointing to $http://www.myopenid.com/server?openid.assoc$ ..., i.e. the unsecure HTTP protocol.

9,10 The initial request at the OpenID Provider is a HTTP request, hence resulting in another redirect advising the User to move to HTTPS.

11-14 The User inserts his credentials in a form and, in case he successfully authenticated against the OpenID Provider, he gets redirected back to the Relying Party with the authentication result (assertion) attached as GET parameter in the "Location" header.

15 The User evaluates the "Location" header from the previous response, containing the HTTP Endpoint of the Relying Party, and gets redirected there by the User agent.

In Figure 3, the authentication request (represented by the steps 8 and 9) can be modified after the User follows the redirect command (step 9). The authentication response (steps 14 and 15) can be modified as well (step 15). In Figure 4, we have stripped the communication up to these redirects. The two disctinct paths still represent a problem for a potential attacker, as he would need to attack these messages at two distinct network nodes. However, if we think of the User as a network node, which uses a gateway (e.g. an internet provider), both of these redirects may share several nodes on their way. If any of these shared nodes is attacked, then both of these redirects are susceptible to forgery as the information within is transported in plaintext. From the User's point of view, this attack is hard to detect, because it represents the standard OpenID flow and the User actually observes all neccessary HTTPS indicators.

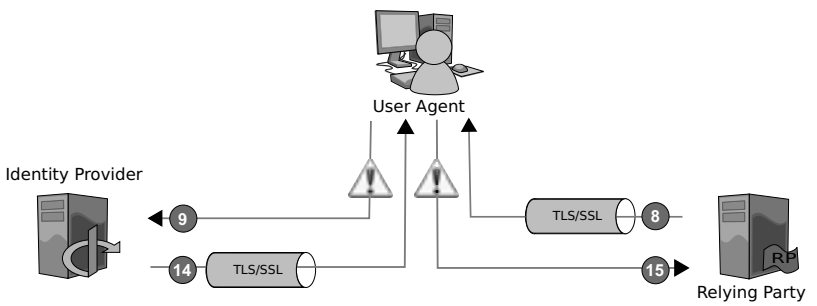


Figure 4: Reducing complexity to HTTP with emphasis on redirects

The topic discussed in this section does not present a threat on its own, it rather provides a perfect fundament for actual attacks discussed in the following sections of this document (i.e. Parameter Injection and Parameter Forgery).

4.2 Parameter Injection

In this section, we exploit the message level security mechanism of OpenID - MAC. With respect to MACs, the two most important OpenID parameters are `openid.sig`, representing the authentication code itself, and `openid.signed`, containing the hashvalue computed over all parameters and xor-ed with the pre-established shared key. The OpenID Authentication 2.0 Protocol Specification states, that if a positive assertion (meaning the User authenticated successfully) is received by the Relying Party, it must not be accepted until it is verified first. Any successful verification must satisfy, among others, the condition that 'the MAC of the assertion is valid and all required fields are MAC-protected'. Hence if a parameter is not defined as required (speaking of which, **none of the identity-related extension parameters are required**) and is not listed in `openid.signed`, it is automatically subject to forgery. In other words, appending arbitrary unused parameters to a MAC-protected message does not invalidate the assertion's MAC and the message stays

intact and valid in the eyes of the Relying Party.

The MAC-protected parameters are all part of the value of the ”openid.signed” parameter. Based on this parameter, we have the following options (examples make use of the OpenID Simple Registration Extension 1.0):

- parameter: ”openid.signed=...sreg.nickname,sreg.email,sreg.fullname...”
  - changing the ”openid.sreg.email” value in this setting would lead to a MAC verification missmatch, thus leading to an invalid assertion
  - however, appending the date of birth by appending the parameter ”openid.sreg.dob” would keep the MAC intact, leading to a valid assertion

Parameters returned to the Relying Party are affected by the Relying Party's request. The request contains a list of parameters, which should be returned by the OpenID Provider (e.g. ”openid.sreg.required=...”).
The OpenID Simple Registration Extension 1.0 states, that the ”openid.signed” list contained in the following response must include the returned ”sreg” parameter names and that the Relying Party bears responsibility of how to behave in case of missing required or additional unrequested parameters. As a consequence, the Relying Party may accept unsolicited parameters either as part of a normal behaviour, or as an implementation error. In fact, the attacker does not care which of both behaviours is the case, as long as such parameters are accepted. In the next section, we show how we can use such ”optimistic” behaviour to manipulate parameters, which have explicitely been requested and are MAC-protected.

4.3 Parameter Forgery

In the 'Parameter Injection' Section, we have shown how we can append our own parameters to the OpenID Authentication 2.0 Protocol response in the authentication phase. The problem, however, was that we were not able to append parameters which already were a component of the response, because they were part of the MAC and hence any modification would lead to a MAC-verification mismatch. Therefore we were only able to inject unused parameters. In the parameter forgery attack, we go a step further by removing parameters from the list of requested parameters ergo leaving it ”void”. As a result, in combination with parameter injection, we can modify any parameters we want, of course with the exception of obligatory OpenID Authentication 2.0 Protocol parameters, which must always be part of the MAC (marked as required in the specification).

Parameter Forgery is based on the fact, that although the OpenID Authentication 2.0 Protocol responses in the authentication phase are MAC-protected by the OpenID Provider, the request does not include any MAC and is therefore prone to forgery. The integrity of a request is in general secured either by the transport layer (using HTTPS) or not secured at all. In many scenarios, however, the integrity is naively achieved through the usage of

”semi-effective” HTTPS redirects, which do not take care of the integrity thoroughly, e.g. if the redirect is HTTPS, but the destination ’Location:’ header url inside is HTTP. Such a redirect is only secure on the way from the Relying Party to the User, but not further. Under such circumstances, there is no integrity on the transport layer and since there is no integrity at the application layer, any adversary acting as ’man-in-the-middle’ may modify the requests.

The reason why the whole request modification effort is performed is actually modifying the response by injecting parameters. According to a property of a modern identity metasystem - ’minimal disclosure’ - the OpenID Provider should protect its User’s privacy by returning only those parameters which it has explicitely been asked for, hence the OpenID Provider must check the request for requested parameters (this is not postulated by the OpenID Authentication 2.0 Protocol specification explicitely, but it is generally the case). These parameters are usually requested in two ways: as part of the `openid.sreg.required` field, meaning that these parameters are needed to successfully sign in the User, or they are listed in the `openid.sreg.optional` field, meaning they are desired, but the Relying Party does not rely on them to be returned from the OpenID Provider (on that matter, one can similarly attack any other OpenID extension, e.g. OpenID Attribute Exchange 1.0). It then depends on the OpenID Provider how it copes with such requested parameters, but if a parameter is not part of the ’required’ or ’optional’ field, it should not be sent (of course with the exception of the assertion-relevant required parameters, which are sent always, but generally do not contain any of the User’s private data). There is a direct relationship (the response-parameters dependend on the request-parameters) between the ’required’ and ’optional’ fields in the request and the returned fields in the response.

That being said, demanding no privacy-relevant parameters in the request inevitably leads to sending no privacy-relevant data in the response, ergo ”no parameters asked” means ”no parameters returned”. In such cases, only the basic assertion, specifying that the User has either successfully signed into the OpenID Provider or not, is sent back to the Relying Party.

The reason why such ’void’ assertions may be very interesting for an adversary lies in the ’Parameter Injection’ attack. If we strip the request of any demands (no parameters marked as required or optional), then there is no reason why an OpenID Provider would send any extra data back to the Relying Party (see Figure 5). Consequently, the OpenID Provider ends up sending a ’void’ assertion leaving all OpenID User relevant data vulnerable to the parameter injection. The adversary is then feasible to change almost anything.

We can use this along with modifying the extension parameters. Besides some extensions, which are solely informative and provide only data retrieval methods, OpenID also allows special extensions, which enable the Relying Parties to store data at the OpenID Provider. This way, the severity of this attack grows, because changing such parameters may affect

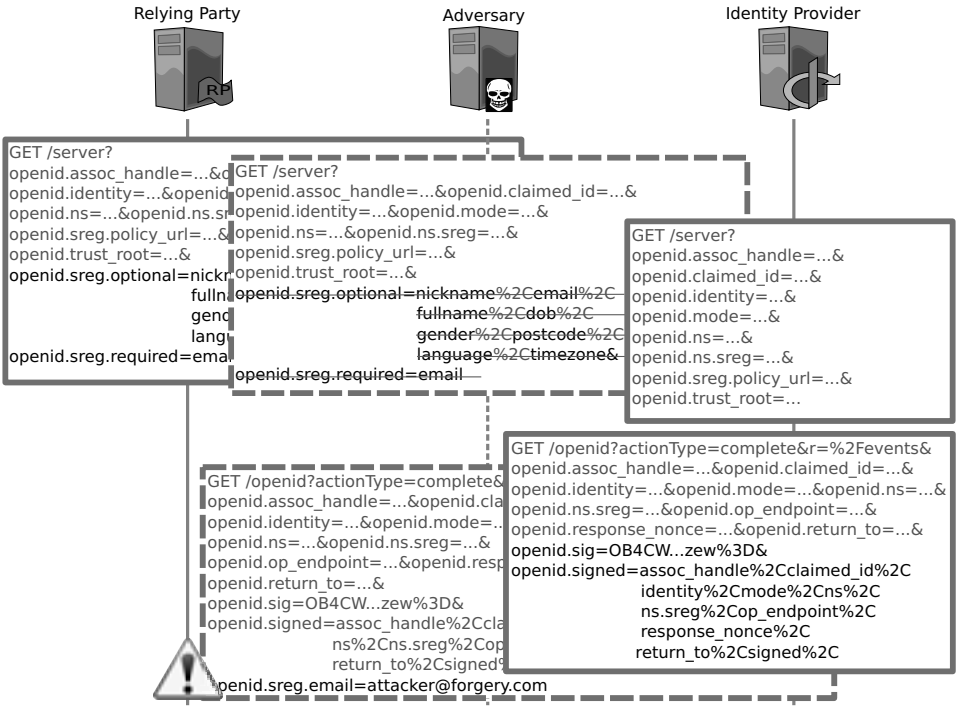


Figure 5: Parameter Forgery combined with Parameter Injection

the OpenID Provider and all Relying Parties therefrom.

5 Conclusion & Future Work

We have provided a description and analysis of the OpenID Single Sign-On protocol and its extensions. The model of OpenID seems to be a suitable Single Sign-On solution for the Internet of today. It has remarkable usability properties and the concept of extensions makes it very flexible. Besides that, giving the control in the user's hands with such a high grade of decentralization rises its popularity significantly. Unfortunately, there are a lot of drawbacks and OpenID has not yet learned from the mistakes of the past.

We have shown that an adversary is able to change arbitrary OpenID extensions' parameters. We recommend that Relying Parties accept only MAC-protected parameters and more importantly - protect the authentication requests with a MAC too. This becomes even more critical when OpenID Attribute Exchange 1.0 is used, due to its ability to change identity information at the OpenID Provider.

Although OpenID has a great potential, but yet again, a working protection against identity theft as one of the biggest challenges of browser-based Single Sign-On systems remains still unsolved.

References

[BJM08] Adam Barth, Collin Jackson, and John C. Mitchell. Robust Defenses for Cross-Site Request Forgery, 2008.

[DA99] T. Dierks and C. Allen. The TLS Protocol Version 1.0, January 1999.

[FKK96] A. Frier, P. Karlton, and P. Kocher. The SSL Protocol Version 3.0, November 1996.

[FRHH07] Brad Fitzpatrick, David Recordon, Dick Hardt, and Josh Hoyt. OpenID Authentication 2.0, December 2007.

[GP06] Thomas Groß and Birgit Pfitzmann. SAML Artifact Information Flow Revisited. In *In IEEE Workshop on Web Services Security (WSSS)*, pages 84–100, Berkeley, May 2006. IEEE.

[Gro03] Thomas Groß. Security Analysis of the SAML Single Sign-on Browser/Artifact Profile. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC'03)*. IEEE Computer Society Press, December 2003.

[GSSX09] Sebastian Gajek, Jörg Schwenk, Michael Steiner, and Chen Xuan. Risks of the CardSpace Protocol. In Pierangela Samarati, Moti Yung, Fabio Martinelli, and Claudio Agostino Ardagna, editors, *ISC*, volume 5735 of *Lecture Notes in Computer Science*, pages 278–293. Springer, 2009.

[HBH07] Dick Hardt, Johnny Bufu, and Josh Hoyt. OpenID Attribute Exchange 1.0, December 2007.

[HDR06] Josh Hoyt, Jonathan Daugherty, and David Recordon. OpenID Simple Registration Extension 1.0, June 2006.

[Jam] Shakir James. Web Single Sign-On Systems.

[KR00] David P. Kormann and Aviel D. Rubin. Risks of the Passport Single Signon Protocol, 2000.

[NL08] Ben Newman and Shivaram Lingamneni. CS259 Final Project: OpenID (Session Swapping Attack), 2008.

[PW03] B. Pfitzmann and M. Waidner. Analysis of liberty single-sign-on with enabled clients. *Internet Computing, IEEE*, 7(6):38–44, 2003.

[TT07] Eugene Tsyrklevich and Vlad Tsyrklevich. Single Sign-On for the Internet: A Security Story, July and August 2007.

Session Fixation – the Forgotten Vulnerability?

Michael Schrank[1] Bastian Braun[2] Martin Johns[3] Joachim Posegga[2]

[1] University of Passau, schrank@fim.uni-passau.de
[2] ISL Passau, {bb, jp}@sec.uni-passau.de
[3] SAP Research, martin.johns@sap.com

Abstract:
The term 'Session Fixation vulnerability' subsumes issues in Web applications that under certain circumstances enable the adversary to perform a session hijacking attack through controlling the victim's session identifier value. We explore this vulnerability pattern. First, we give an analysis of the root causes and document existing attack vectors. Then we take steps to assess the current attack surface of Session Fixation. Finally, we present a transparent server-side method for mitigating vulnerabilities.

1 Introduction and paper outline

Session Fixation has been known for several years. However, compared to vulnerability classes such as Cross-site Scripting (XSS), SQL Injection, or Cross-site Request Forgery (CSRF), this vulnerability class has received rather little attention. In this paper, we delve into Session Fixation in three ways: First we explore the vulnerability pattern and document existing attack methods (see Sec. 2). Secondly, we take steps to assess the current attack surface of Session Fixation (Sec. 3). Finally, we present a transparent server-side method for mitigating vulnerabilities (Sec. 4). We finish the paper with a look on related work (Sec. 5) and a conclusion (Sec. 6).

2 Exploring Session Fixation

2.1 Technical background: Session management

HTTP is a stateless protocol. Thus, HTTP has no protocol-level session concept. However, the introduction of dynamic Web applications resulted in workflows which consist in a series of consecutive HTTP requests. Hence, the need for chaining HTTP requests from the same user into usage sessions arose. For this purpose, session identifiers (SID) were introduced. A SID is an alphanumerical value which is unique for the corresponding Web session. It is the Web application's duty to implement measures that include the SID in every HTTP request that belongs to the same Web session. A SID mechanism can be implemented by transporting the value either in form of an HTTP cookie or via HTTP

parameters [FGM+99]. All modern Web application frameworks or application servers, such as PHP or J2EE, implement application transparent SID management out of the box.

Any request that contains the SID value is automatically regarded to belong to the same session and, thus, to the same user. After a user has authenticated himself against the application (e.g., by providing a valid password), his authorization state is also directly tied to his SID. Hence, the SID de facto becomes the user's authorization credential.

2.2 Session Fixation

Several classes of attacks that target SID-based authorization tracking have been documented in the past. Especially Session Hijacking via XSS [End02] has received considerable attention. This attack is based on the adversary's ability to obtain a user's valid and authenticated SID through the injection of JavaScript code into the application. Using this SID, the adversary is able to impersonate the victim in the context of the attacked application. Besides XSS other techniques to obtain valid SIDs exist, such as Sidejacking [Gra07] or brute-force session guessing [Kam09].

Session Fixation [Kol02] is a variant of Session Hijacking. The main difference to the variants discussed above is that Session Fixation does not rely on SID theft. Instead the adversary tricks the victim to send a SID that is controlled by the adversary to the server. See Figure 1 for a brief example:

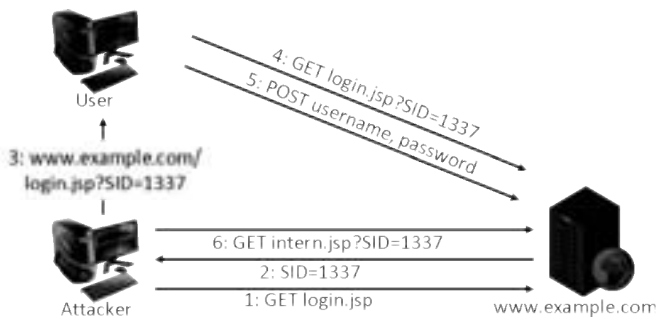


Figure 1: Exemplified Session Fixation attack [Kol02]

- The attacker obtains a SID value from the server (1,2).
- He tricks the victim to issue an HTTP request using this SID during the authentication process (3,4).
- The server receives a request that already contains a SID. Consequently, it uses this SID value for all further interaction with the user and along with the user's authorization state (5).

- Now, the attacker can use the SID to access otherwise restricted resources utilizing the victim's authorization context (6).

2.3 Why does Session Fixation exist?

At first glance, the Session Fixation attack pattern seems both contrived and unlikely. Why would an application accept a user's SID that was not assigned by the application to this user in the first place? The root problem lies within a mismatch of responsibilities: SID management is executed by the utilized programming framework or the application server. All related tasks, such as SID generation, verification, communication, or mapping between SID values to session storage, are all handled transparently to the application. The programmer does not need to code any of these tasks manually. However, the act of authenticating a user and subsequently adding authorization information to his session data is an integral component of the application's logic. Consequently, it is the programmers duty to implement the corresponding processes. The framework has no knowledge about the utilized authentication scheme or the employed authorization roles.

Finally, in general, modern Web applications assign SID values automatically with the very first HTTP response that is sent to a user. This is done to track application usage even before any authentication processes have been executed, e.g., while the user accesses the public part of the application. Often the programmer is not aware of this underlying SID functionality as he is not responsible for managing the SIDs. He simply relies on the framework provided automatism. The combination of the circumstances listed above lead to situations in which applications neglect to reissue SID values after a user's authorization state has changed. This in turn causes Session Fixation vulnerabilities.

2.4 Exploiting Session Fixation

Given the sources of Session Fixation vulnerabilities as described in Sec. 2.3, a broad field of attack vectors comes into play. All attacks strive to 'implant' a known SID to the victim's browser. As described in Sec. 2.1, SIDs are communicated either via HTTP parameters or cookies:

URL parameter: The easiest way to push a SID to the user's browser is a URL parameter (see Figure 1) in situations in which the attacked application accepts such SIDs. The attacker generates a valid SID, e.g. `1337`, of the vulnerable application on address `http://www.example.com`. Then, he sends the URL `http://www.example.com/?SID=1337` to the victim. For example, he could promise that the pictures from his last summer holiday are published there. The victim's browser sends a request to the application making use of the given SID if he clicks on this link. From the application's point of view, he already has a valid session. So, it does not need to deliver a new SID.

As a next step, the victim provides his credentials and logs in to the application. The session has not changed but he is now logged in. The application still identifies the victim

by his SID that is sent with every request. However, the attacker also knows this SID and can thus act as the victim in his account. Therefore, he does not even have to interfere in the communication between the victim and the application. The only thing to do is sending a request to the application like `http://www.example.com/account_balance.php?SID=1337`.

Cookies: A slightly more difficult way for the attacker are cookies set by the application. In this scenario, the adversary requests a session cookie from the application. Then, he needs to make his victim accept the same cookie. There are several options for him to reach his goal as we will show. When the victim owns the adversary's session cookie, his browser will provide this cookie to the application. As the adversary can use this cookie too, he can own his victim's session.

- **Setting cookies via XSS:** The adversary can use cross-site scripting (XSS) to set a cookie at his victim's site if the web application is vulnerable to this attack. Therefore, he inserts JavaScript code into a web page in the same domain, `.example.com`. When the victim visits this page, the cookie will be set. The adversary can set the expiry date to a date in distant future so that the cookie will not be deleted when the victim restarts his browser. Unlike a cookie stealing attack, the victim does not have to be logged in, thus, the attack also works at the public part of the application. In the past, we saw browser vulnerabilities that even allowed the attacker to set the cookie from a foreign domain [Zal06].

- **Setting cookies via `meta` tags:** Under certain circumstances, a Web application might allow user-provided HTML markup but filters user-provided JavaScript. In such cases, the attacker might be able to inject special `meta` tags. The tag `<meta http-equiv="Set-Cookie" Content="SID=1337;expires=Monday, 07-Mar-2011 12:00:00 GMT">` sets a cookie which is valid until March 2011.

- **Setting cookies via cross-protocol attacks:** The same-origin policy [Rud01] explicitly included the port of the URL that was utilized to retrieve a JavaScript as a mandatory component of its origin. In consequence, if a service in a given domain allows the adversary to execute JavaScript, e.g., via XSS, services in that domain which are hosted on different ports are unaffected by this.

  However HTTP cookies are shared across ports [KM00]. Thus, cross-protocol attacks come into play: Cross-protocol attacks [Top01, Alc07] (see below for further explanations) allow the adversary to create XSS like situations via exploiting non-HTTP servers, such as SMTP or FTP, that are hosted on the same domain.

  First, the adversary prepares a website with a specially crafted HTML form [Top01]. The form contains JavaScript code to set the attacker's cookie at his victim's browser. The target of this form is the targeted non-HTTP server[1]. To avoid URL encoding of the content, the adversary chooses the `enctype="multipart/form-data"` parameter. The target server interprets the HTTP request as a valid request of its own protocol, e.g. FTP [Dab09]. However, most of the incoming commands cannot

[1] NB: Some browser, e.g., Firefox, block HTTP requests to certain well known ports. In such cases the attack requires a susceptible server on a non-blocked port to function.

be understood and will be reflected with a respective error message. This message generally contains the erroneous input which is in our case the JavaScript code with the cookie. For compatibility reasons, browsers take all textual input as HTTP even without any valid HTTP header. So, the browser accepts the cookie since it comes from a server in the same domain.

- **Subdomain Cookie Bakery:** JavaScript's same origin policy prohibits the setting of cookies for other domains. Nevertheless this only applies to top level domains. Therefore a possible attack vector is setting a cookie using a vulnerable subdomain, e.g. JavaScript code or a `meta` tag on `vulnerable.example.com` can set a cookie which is subsequently sent by the victims browser for `example.com`. Hence a vulnerable application on a subdomain may compromise the session security of the whole top level domain.

- **HTTP Response Splitting and HTTP Header Injection:** In case the web application is also vulnerable to HTTP response splitting, the attacker could use this to send his cookie to the victim's browser [Kle04]. Therefore, he needs a redirected page that includes unfiltered user provided content and a proxy with a web cache.

  First, the attacker sends a crafted HTTP request through the proxy to `http://www.example.com/redirect.php`. The application takes information from the attacker's request to generate the target of the redirection, e.g. `/app_by_lang.php?lang=`*attacker's parameter*. This way, he can influence the `Location` header field where the redirection URL is denoted. He can then shape a request that finishes the HTTP response and append a new response. The latter is now fully controlled by himself, i.e. he can use the `Set-Cookie` header and deliver his cookie, e.g. `/redirect.php?lang=en%0d%0a Content-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aSet-Cookie:%20SID=1337%0d%0a Content-Length:%2019%0d%0a%0d%0a<html>Foobar</html>` [Kle04]. The proxy caches the second response if he manages to send a harmless request to `http://www.example.com/` at the right time. The actual answer to his second request will be discarded and considered superfluous. Finally, the proxy serves his specially crafted HTTP response to requests on `http://www.example.com/`.

Figure 2: A Header Injection Attack

The attack works similarly if the application takes user provided data to set a cookie. In this case, the `Set-Cookie` header serves as an entry point instead of the `Location` header. The attacker could run a HTTP header injection attack (see Figure 2) instead. Then, he prepares a special URL like `http://www.example.com/app_by_lang.php?lang=en%0d%0aSet-Cookie:%20SID=1337%0d%0a` and sends this to his victim. This attack is then analogical to the URL parameter attack. The victim has to click on the link and enter his login credentials. After that, the attacker can own his account. We tested the feasibility of header injection attacks and give results in Sec. 3.

2.5 Impact and discussion

Currently, Session Fixation is only a second stage attack that usually requires several preconditions.

The attacker needs another vulnerability to provide his cookie to the victim. Alternatively, he must mislead the victim into clicking on his link if the target web application allows URL parameters for session management. We presented different attack vectors in Sec. 2.4. In case the attacker is successful at the first step, he has to make the victim log into his account. The SID is useless as long as it does not belong to a logged in user. However, the attacker neither knows when the victim logs in nor when he logs out again. He has an unknown window of opportunity. Finally, the target web application has to be vulnerable as we pointed out in Sec. 2.2. Actually, it would be essential for a broad attack to provide a unique cookie for each victim. Otherwise, one victim would take the session of another. This is however not always easy to implement for the attacker depending on the attack vector.

When the conditions are met, though, Session Fixation is a severe attack that allows the attacker to fully impersonate the victim. It is generally not obvious to the victim to be under attack, especially if he is not familiar with Session Fixation. Most attack scenarios appear to be a software malfunction to the unexperienced user. The victim may even not notice the attack afterwards depending on the actions of the attacker on his behalf.

3 Practical experiments

In the context of this paper, we took first steps to assess to which degree Session Fixation currently poses a realistic threat. For this purpose, we conducted two series of practical experiments.

First we tested open source Content Management System (CMS) applications for Session Fixation issues. This way we aimed to get an estimate if developers are actually aware of the lingering threat of Session Fixation or if they unknowingly rely on the framework's protection mechanisms. The result of these tests suggested that a considerable fraction of existing applications indeed are susceptible to Session Fixation under circumstances that

allow the attacker to set cookies on the victim's browser (see Sec. 2.4). Consequently, we examined several popular web application frameworks with respect to their susceptibility to potential header injection vulnerabilities.

3.1 Examination of open source CMS applications

To assess an application's susceptibility to Session Fixation, we adhered to the following testing methodology (see Fig. 3): First we verified that the application indeed issues SIDs before any authentication processes have been undertaken. Then, we tested if the application leaves the SID unchanged in case a successful authentication process has happened. If these tests could be answered with 'yes', we concluded that under certain circumstances the application could expose susceptibility to Session Fixation. The final test probed which attack vectors (see Sec. 2.4) were applicable.

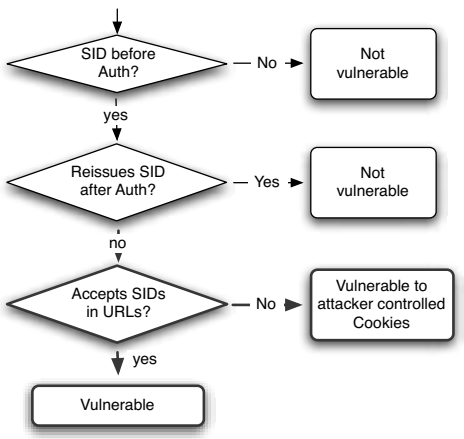


Figure 3: Testing methodolgy

We considered several open source Web applications in their configuration 'out-of-the-box'. The applications implement user management on their own. The results are given in Table 1. A plus sign (+) in a column denotes that the application is vulnerable to the respective attack vector. An application is vulnerable to *Cookie* if it accepts foisted cookies. The attacker can be successful if he manages to set a cookie at the victim's browser. In case the application allows session tracking via a URL parameter, it is vulnerable to *URL*. Finally, those applications which are vulnerable to *SID* allow the attacker to utilize arbitrary SID values that have not been generated by the application in the first place.

Application	Version	Cookie	URL	SID	Lang
Joomla	1.5	+	-	+	PHP
CMSmadesimple	1.6.6	+	-	+	PHP
PHPFusion	7.00.06	-	-	+	PHP
Redmine	0.9.2	+	-	-	PHP
XWiki	2.0.2.24648	+	-	-	Java
JAMWiki	0.9	+	+	+	Java
Wordpress	2.9.1	-	-	-	PHP
Novaboard	1.1.2	+	-	+	PHP
PHPBB	3.0.6	-	-	-	PHP
SimpleMachinesForum	1.1.11	-	-	-	PHP
Magento Shop	1.3.4.2	+	-	-	PHP
OSCommerce	2.2 RC 2a	+	-	-	PHP

Table 1: Test results

3.2 Header injection

We analyzed common web application frameworks to comprehend the feasibility of HTTP header injection attacks. Therefore, we implemented script files which either set a cookie or do forwarding respectively. The scripts take attacker controlled input to determine the forwarding target and the cookie value. The results are given in the next sections.

- **PHP:** Usually, HTTP forwarding in PHP is implemented by the use of the `header()` function. We implemented a file that takes one parameter and inserts this into the `Location` header field. We provided valid input but appended the payload to set our cookie. In the initial configuration (PHP 5.3.0, Apache 2.2.11), the cookie was not set but we got a warning message because `header()` may not contain more than a single header line since version 4.4.2 and 5.1.2 and our new line was detected. Then, we downgraded (PHP 5.1.1, Apache 2.0.63) and the attack was successful. As the forwarding worked as expected we were able to set the cookie 'drive-by' without any notice for the victim.

  Cookie setting is done with the `setcookie()` function. We appended a line break and a new `Set-Cookie` header. However, the function URL-encodes the cookie value and thus transforms our `:` and = characters to non-interpreted URL codes. We can avoid URL encoding of our input by using `setrawcookie()`. However, it prohibits control characters in the value field.

- **J2EE:** For the Java test scenario, we used Tomcat 6.0.20 as a servlet container. The redirection could not be successfully spoofed. The payload was treated as part of the URL. During our cookie setting approach, we got a `java.lang.IllegalArgumentException` due to the control characters in the payload.

- **CherryPy:** We made use of CherryPy (version 3.1.2) to test the Python functions. The injection to the `Location` header was successful whereas the cookie value turned out to be not exploitable.

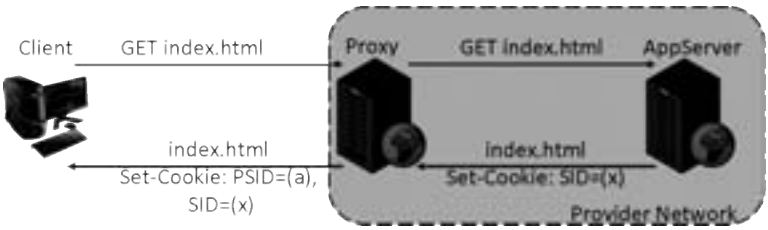


Figure 4: Introduction of the proxy session identifier.

- **Perl:** We also implemented a Perl (version 5.10.1 with `CGI.pm` version 3.48) script that does nothing but forwarding to a given site. Indeed, we managed to set a cookie at the victim's site, however, for some reason the equal sign between the cookie name and its value is finally coded as a colon and a blank. So, we got a cookie with a given value but without a name. Then, we inserted two equal signs and the second one was not recoded. Actually, we were able to set an arbitrary cookie which name ended with a colon. The cookie setting scenario was more difficult due to URL encoding of cookie names and values. We were not able to set our own cookie given that we could influence the value of an unimportant cookie.
- **Ruby on Rails:** For Rails (version 1.9.1), we omitted the tests for header injection in forwarding sites as this vulnerability was recently patched [Web08, Sec09]. During the cookie task, we faced the same situation as in the Perl scenario. Cookies must be declared and cannot be set as ordinary headers. The value that we inserted is thus first URL encoded and never interpreted. So, we did not find a way to escape the cookie value context.

4 Session Fixation Protection Proxy

In this section, we propose a method for transparent, frame-work level protection against Session Fixation attacks. Our approach allows to secure vulnerable applications without changing the application itself. This is achieved by introducing a proxy which monitors the communication between the user and the vulnerable application. The proxy implements a second level session identifier management (see Fig. 4). In addition to the SIDs that are set by the application, the proxy issues a second identifier (PSID). The set of SID and PSID is stored by the proxy. Only requests that contain a valid combination of these two values are forwarded to the application (see Fig. 5).

Requests that are received with an invalid combination are striped of all `Cookie` headers before sending them to the server. Furthermore, the proxy monitors the HTTP requests' data for incoming password parameters. If a request contains such a parameter, the proxy assumes that an authentication process has happened and renews the PSID value, adds an according `Set-Cookie` header to the corresponding HTTP response, and invalidates the

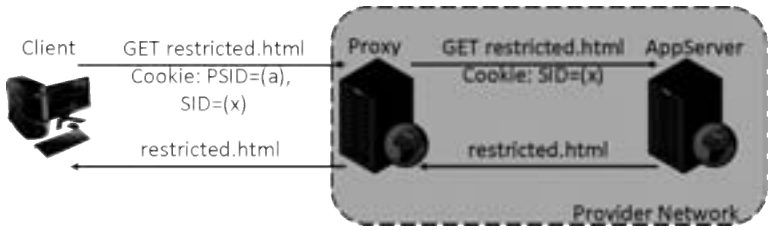


Figure 5: Verification of the proxy session identifier.

former PSID/SID combination.

Hence, the session's security does no longer lie in the SID of the vulnerable application but in an additional identifier issued by the proxy. Accordingly this proxy can be put in front of almost any vulnerable application, giving the application's developers time to fix it.

5 Related work

This paper serves two purposes: comprehensively documenting the Session Fixation vulnerability pattern and proposing a novel method for protecting against such attacks. We structure this section accordingly.

Vulnerability documentation: To the best of our knowledge, [Kol02] was the first public paper on Session Fixation. It describes the basic fundamentals of the attack and the most obvious attack vectors. In contrast to this paper, it provides no information about the spreading of the vulnerability or more advanced attack schemes. Furthermore, OWASP and WASC added articles about Session Fixation to their security knowledge bases. The OWASP article [(OW09] briefly names common Session Fixation issues and attack vectors. In contrast, the WASC article [(WA10] also provides small code examples of different attacks. Both sources name the HTTP response splitting attack as an attack vector, but they do not discuss its impact in today's world of web applications. Furthermore, they lack a general rating of the attack.

Related protection mechanisms: Web application firewalls are server-side proxies that aim to mitigate security problems. However, as the OWASP best practices guide on Web Application Firewalls (WAF) [OWA08] states, current WAFs can only prevent Session Fixation "if the WAF manages the sessions itself." In comparison, our approach does not touch the application level session management and only introduces additional security related information for each request. Furthermore, several protection techniques have been proposed that utilize proxies to mitigate related Web application vulnerabilities. [KKVJ06, Joh06] describe proxy solutions to counter Cross-site Scripting attacks, [JW06, JKK06] propose related techniques for Cross-site Request Forgery protection.

6 Conclusion and Future Work

In this paper, we thoroughly examined Session Fixation: We showed how Session Fixation attacks work (see Sec. 2.2), identified the root cause of this vulnerability class (Sec. 2.3), and documented existing attack vectors (Sec. 2.4).

As mentioned in the introduction, the public level of attention to Session Fixation is comparatively low. We tried to assess if this apparent ignorance is justified. For this purpose, we conducted two sets of tests (Sec. 3). First, we examined open source applications for evidence that their developers were aware of the vulnerability class, i.e., we tested if session identifier were changed after an authentication process. Out of 12 only 4 applications take this measure. Consequently, the remaining 8 applications would be vulnerable to Session Fixation if a supporting problem, such as a header injection flaw, exists (one application was vulnerable out of the box due to URL-support; the issue has been fixed in the meantime). Secondly, motivated by the outcome of the first set of tests, we examined various Web application programming frameworks with respect to protection against header injection flaws. These tests resulted in the observation that the majority of the regarded frameworks indeed take measures to protect against header injection vulnerabilities, thus, indirectly protecting otherwise vulnerable applications against Session Fixation attacks.

Finally, we outlined a novel method for framework-level, transparent protection against Session Fixation. Our solution introduced a dedicated server-side Web proxy that adds a second identifier value (PSID) to the application's outgoing requests. This value is treated as a companion to the application's SID. Only incoming SID values that appear together with a valid PSID are relayed to the application. Session Fixation attacks are successfully prevented as the PSID is renewed by the proxy after an authentication process has happened. As a next step we will implement and thoroughly test the proxy.

References

[Alc07] Wade Alcorn. Inter-Protocol Exploitation. Whitepaper, NGSSoftware Insight Security Research (NISR), http://www.ngssoftware.com/research/papers/InterProtocolExploitation.pdf, March 2007.

[Dab09] Arshan Dabirsiaghi. Cross-protocol XSS with non-standard service ports. TechNote, http://i8jesus.com/?p=75, August 2009.

[End02] David Endler. The Evolution of Cross-Site Scripting Attacks. Whitepaper, iDefense Inc., http://www.cgisecurity.com/lib/XSS.pdf, May 2002.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, http://www.w3.org/Protocols/rfc2616/rfc2616.html, June 1999.

[Gra07] Robert Graham. SideJacking with Hamster. [online], http://erratasec.blogspot.com/2007/08/sidejacking-with-hamster_05.html, (02/02/10), August 2007.

[JKK06] Nenad Jovanovic, Christopher Kruegel, and Engin Kirda. Preventing cross site request forgery attacks. In *Proceedings of the IEEE International Conference on Security and Privacy for Emerging Areas in Communication Networks (Securecomm 2006)*, 2006.

[Joh06] Martin Johns. SessionSafe: Implementing XSS Immune Session Handling. In Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors, *European Symposium on Research in Computer Security (ESORICS 2006)*, volume 4189 of *LNCS*, pages 444–460. Springer, September 2006.

[JW06] Martin Johns and Justus Winter. RequestRodeo: Client Side Protection against Session Riding. In Frank Piessens, editor, *OWASP Europe 2006*, May 2006.

[Kam09] Samy Kamkar. phpwn: Attack on PHP sessions and random numbers. Security Advisory, `http://samy.pl/phpwn/`, August 2009.

[KKVJ06] Engin Kirda, Christopher Kruegel, Giovanni Vigna, and Nenad Jovanovic. Noxes: A Client-Side Solution for Mitigating Cross Site Scripting Attacks. In *Security Track of the 21st ACM Symposium on Applied Computing (SAC 2006)*, April 2006.

[Kle04] Amid Klein. "Divide and Conquer" - HTTP Response Splitting, Web Cache Poisoning Attacks, and Related Topics. Whitepaper, Sanctum Inc., `http://packetstormsecurity.org/papers/general/whitepaper_httpresponse.pdf`, March 2004.

[KM00] D. Kristol and L. Montulli. HTTP State Management Mechanism. RFC 2965, `http://www.ietf.org/rfc/rfc2965.txt`, October 2000.

[Kol02] Mitja Kolsek. Session Fixation Vulnerability in Web-based Applications. Whitepaper, Acros Security, `http://www.acrossecurity.com/papers/session_fixation.pdf`, December 2002.

[(OW09] The Open Web Application Security Project (OWASP). Session Fixation. TechNote, `http://www.owasp.org/index.php/Session_Fixation`, February 2009.

[OWA08] OWASP German Chapter. OWASP Best Practices: Use of Web Application Firewalls. [whitepaper], `http://www.owasp.org/index.php/Category:OWASP_Best_Practices:_Use_of_Web_Application_Firewalls`, July 2008.

[Rud01] Jesse Ruderman. The Same Origin Policy. [online], `http://www.mozilla.org/projects/security/components/same-origin.html` (01/10/06), August 2001.

[Sec09] SecurityFocus. Ruby on Rails 'redirect_to' HTTP Header Injection Vulnerability. TechNote, `http://www.securityfocus.com/bid/32359`, December 2009.

[Top01] Jochen Topf. The HTML Form Protocol Attack. TechNote, `http://www.remote.org/jochen/sec/hfpa/hfpa.pdf`, August 2001.

[(WA10] The Web Application Security Consortium (WASC). Session Fixation. TechNote, `http://projects.webappsec.org/Session-Fixation`, January 2010.

[Web08] Heiko Webers. Header Injection And Response Splitting. TechNote, `http://www.rorsecurity.info/journal/2008/10/20/header-injection-and-response-splitting.html`, October 2008.

[Zal06] Michal Zalewski. Cross Site Cooking. Whitepaper, `http://www.securiteam.com/securityreviews/5EP0L2KHFG.html`, January 2006.

CAPTCHAs: The Good, the Bad, and the Ugly

Paul Baecher Marc Fischlin Lior Gordon Robert Langenberg
Michael Lützow Dominique Schröder

Darmstadt University of Technology, Germany
www.minicrypt.de

Abstract. A CAPTCHA is a program that generates challenges that are easy to solve for humans but difficult to solve for computers. The most common CAPTCHAs today are text-based ones where a short word is embedded in a cluttered image. In this paper, we survey the state-of-the-art of currently deployed CAPTCHAs, especially of some popular German sites. Surprisingly, despite their importance and the large-scale deployment, most of the CAPTCHAs like the ones of the "Umweltprämie", the Bundesfinanzagentur, and the Sparda-Bank are rather weak. Our results show that these CAPTCHAs are subject to automated attacks solving up to 80% of the puzzles. Furthermore, we suggest design criteria for "good" CAPTCHAs and for the system using them. In light of this we revisit the popular reCAPTCHA system and latest developments about its security. Finally, we discuss some alternative approaches for CAPTCHAs.

1 Introduction

A CAPTCHA is an automated challenge and response program to distinguish humans from computers. The term CAPTCHA was coined by van Ahn et al. [vAMM+08] as an acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart". Roughly, CAPTCHAs are small puzzles which a human being should be able to solve easily, whereas an automated attack is infeasible.

State-of-the-Art. CHAPTCHAs are paramount in today's Internet: almost every user encounters a CAPTCHA in his or her daily usage of the Internet. CAPTCHAs are used to protect websites like googlemail or wordpress, to prevent spam in blogs, and to protect e-mail addresses. Despite their importance there are still a lot of seemingly bad CAPTCHAs out there. We highlight this by revisiting three popular German examples.

In March 2009 the stimulus program "Umweltprämie" offered car owners up to 2,500 EUR for a new car, scrapping their old one. Users could register online and, since the bonus was given on a first-come first-serve basis, the immense amount of accesses repeatedly caused unavailability problems of the hosting site (`www.ump.bafa.de`). About 2 Million car owners eventually received a bonus; we are not aware of the additional number of unsuccessful applications. As part of completing the online application form users had

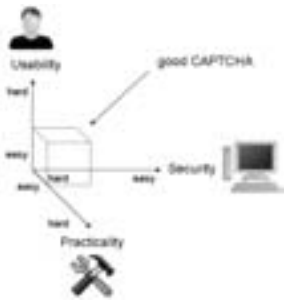


Figure 1: Requested Properties of CAPTCHAs.

to solve a CAPTCHA, seemingly to prohibit automated attempts to secure the bonus. The website is no longer operational since the program has ended.

In a similar vein, the Bundesfinanzagentur, specifically the Bundeswertpapierverwaltung (German Federal Securities Administration), supports the sale and custody of Governmental bonds, treasury financing papers and federal notes. For private investors it is free of charge, and currently used by about 450,000 private investors, with the agency's goal to increase this number to 1 Million in 2013 (according to their homepage). Around June 2009 the agency has augmented the log-in process for online banking by a CAPTCHA to "protect unauthorized persons to log into the Internet banking automatically and anonymously" (translated from German).

The Sparda-Bank, although divided into regional groups like Sparda-Bank Hessen or Sparda-Bank Berlin, uses a central log-in system for the online banking. Every user has to solve a CAPTCHA to log in, preventing unauthorized access since "reading the code by computer programs is very hard" (translated from German).

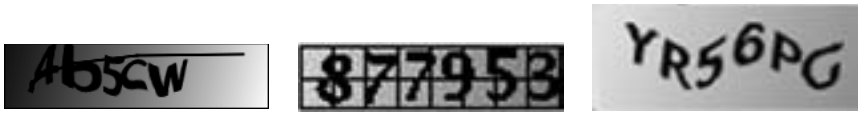
Here we show that none of the above CAPTCHAs achieves its intended goal; they are easily solvable by automated attacks, with success rates of $70\% - 87\%$. It is also worth noting that the governmental sites do not offer barrier-free alternatives for users with visual impairment (note that, while citizens with impaired vision may not drive a car, they may still own one and would have been eligible to sign up for the Umweltprämie).

Usability vs. Security vs. Practicality. Developing a CAPTCHA is always a trade-off between different goals. We identify three orthogonal requirements (see Figure 1):

Usability refers to the difficulty of solving the CAPTCHA for a human, as well as the time that a human actually needs to find the solution.

Security on the other hand, gives a rough guide how difficult it is to find a solution for the computer.

Practicality refers to the effort to realize the CAPTCHAs in practice, e.g., if it requires only a standard web browser or can be used easily on a smart phone. Practicality could also refer to the acceptance on the users' side.



(a) (b) (c)

Figure 2: Three insecure text-CAPTACHAs. Figure 2a has been used by the German government for the "Umweltprämie". The second CAPTCHA is part of the Sparda Bank, and the third one is implemented on the BFA website.

Ideally, a CAPTCHA should be highly usable by humans, provide strong security against automated attacks, and is easy to realize. As an example how practicality affects other dimensions consider an image-based CAPTCHA where users should interpret the semantics of the image. If, for ease of implementation, the CAPTCHA uses a public database with many freely available images then automated attacks may take advantage by trying to find the image in the database.

From a practicality point of view text-based CAPTCHAs, where a word or a sequence of digits or characters is embedded into a slightly distorted image, are easy to implement and are thus very popular. The main question regarding such text CAPTCHAs is the trade-off between usability and security. An in-depth discussion focused on usability of CAPTCHA designs can be found in [YA08].

When is a CAPTCHA broken? The question if a CAPTCHA is secure or not depends heavily on the application. For example, protecting spammers from posting advertisement in a blog may not be as dangerous as a bot that tries to hack an email account. As a rule of thumb, the developers of reCAPTCHA [vAMM+08] ask that computers can solve at most 5% of the generated puzzles, or else the CAPTCHA system should be considered broken. Note that this explicitly rules out attacks in which the CAPTCHAs are solved by exploiting cheap human labor, or via so-called "porn-relay" attacks in which CAPTCHAs are forwarded to other sites and then solved by humans trying to access these sites.

2 Bad Text-CAPTCHAs Do not Die Out

In this section, we discuss several in-use text-CAPTCHAs that are completely insecure. Figure 2 depicts the text-CAPTCHAs from the German government, the Sparda Bank, and the BFA. These CAPTCHAs have in common that they are built on a fixed-length sequence of black characters (or, the characters at least have the darkest color). The background consists of a color gradient from white to gray. The CAPTCHA of the "Umweltprämie" has in addition a black line that always ends in the right upper part of the picture and the Sparda Bank CAPTCHA uses a grid in the background.

2.1 Preparing the CAPTCHAs, or How to Segment the Characters

Since all discussed CAPTCHAs follow roughly the same idea, we describe a general method to break them all using only one program. We first remove the background. This is possible because the characters always have the darkest color. Thus, we parse the image and set the darkest gray value as a threshold, such that all lighter shades of gray values below this threshold are set to white.

Since the CAPTCHA of the "Umweltprämie" has a black line through the picture, we have to remove it. The main observation is that line starts in the right upper part of the picture. To remove this line we fist apply a line detection algorithm starting at the first upper black pixel and we then set all detected pixels to white. Likewise, the grid in the background of the Sparda-CAPTCHA can be removed easily because it is static and always appears at a fixed position (some care must be taken in order to remove genuine parts of the grid only).

In the last step, we transform each picture into one where each character has exactly 32x32 pixels. To do so, we start with a vertical line at the left part of the picture and move it towards the right part until we "hit" the first black pixel. This pixel must correspond to the first character. Afterwards, we use an algorithm that finds the shortest path from the bottom to the top (which is ideally a straight line). See Figure 3a for an example of the algorithm. Note that this partition algorithm is applicable to these CAPTCHAs because the characters are not connected. The algorithm then returns all separate characters.

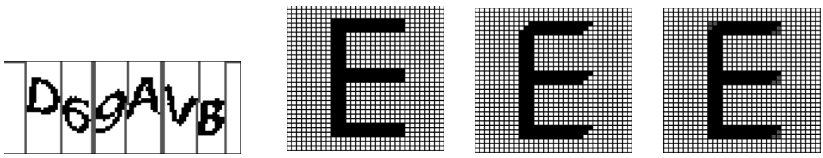
2.2 The K-Means Algorithm, or How to Break the CAPTCHAs

To break the CAPTCHAs of the Finanzagentur and the Umweltprämie with a single algorithm, we apply the *k-Means* algorithm [Mac67]. While it is in principle also possible to employ this algorithm also against the Sparda-Bank CAPTCHA, a simple OCR job with Tesseract after the preprocessing stage is sufficient in that case. For the other two CAPTCHAs the k-means algorithm yields stronger results with only a little more effort.

The k-Means algorithm has been developed for digital image processing and it is a so called clustering algorithm. Clustering means that this algorithm compares the given image to a database that contains several partitions and it returns the partition to which the image fits best. We illustrate the main idea of this algorithm for the following example.

The first part of the algorithm is a learning phase where it gets as input a set of images. The algorithm then computes on this basis its partitions. Figure 3 depicts the learning phase of the k-Means algorithm. The images 3b and 3c are two training images that the algorithm get as input. The k-Means algorithm then computes the difference between both pictures and it treats this difference as a tolerance. This is shown in Figure 3d. The black parts of the letter "E" are fixed wheres the gray parts may be contained (nor not) in the challenge picture. This way the algorithm also handles italic letters of different font types.

After having finished the learning phase, the k-Means algorithm is applicable to the challenge CAPTCHA. Whenever it gets as input a CAPTCHA, then it simply compares char-



(a) (b) (c) (d)

Figure 3: Partition of the picture and learning phase of the k-Means algorithm.

acter per character with its partitions (via the common Euclidean norm) and outputs the partition that fits best.

Results. The following results for Finanzagentur and Umweltprämie are based on a semi-automatic learning phase. Semi-automatic means that whenever an a certain letter matches to a wrong cluster, then we create a new cluster with this image. The advantage is that we get more partitions such that different fonts match to better to certain partitions. The results are based on a test set of 100 CAPTCHAs each (which are distinct from the CAPTCHAs in the learning phase).

For the Sparda Bank CAPTCHA 330 challenges are preprocessed to remove the background grid and then simply fed to the Tesseract OCR software.

Finanzagentur	Sparda Bank	Umweltprämie
70 %	87 %	68 %

Note that the sample size (especially with 100 CAPTCHAs) appears to be rather small, and that a "lucky" set of test samples may not allow to extrapolate to the average success probability. However, since the success rate is quite high we can use the Chernoff bound to conclude that these CAPTCHAs should be considered broken, in the sense that one can solve significantly more than 5% of the puzzles by an automated attack (after an initial training phase which is inherent to many OCR or k-Means approaches). To be more precise, the Chernoff bound says that, given the statistics above that one can solve at least 70% of the random test sample of 100 CAPTCHAs, the probability that an individual CAPTCHA can be solved with more than 50% success rate, is more than $99,5\%$.

3 Best Practice for Designing Text CAPTCHAs

Given the history of weak CAPTCHA systems and their continuing use on popular sites, we aim to provide a set of recommendations on how to build secure CAPTCHAs and to avoid common pitfalls in this section. In the first subsection, we discuss high-level design issues that apply to the actual challenges. The second subsection discusses implementation questions which are easily overlooked but equally important to the security of the system.

clipfish.de n24.de freeCap 1.4

Figure 4: Careless of use of color that makes segmentation and extraction of the characters trivial.

3.1 Design Considerations

When designing a text-based CAPTCHA, it may be tempting to use dictionary words as challenges. While this provides superior usability since the user might be familiar with the word and thus be able to read it under extreme circumstances, it comes with several disadvantages. The most obvious aspect here is that the attacker can easily verify if a given attempt to pass the test successfully offline. Consequently, she is able to submit only those solutions which are likely to be considered valid by the challenger. This increases the difficulty to detect automated clients even further, since hopeless responses can be detected and discarded offline. Another drawback of dictionary words is the comparatively limited challenge space. This sets the stage for a class of attacks where a database containing every possible challenge can be precomputed and the best match for a given challenge selected. Finally, the usability advantage applies only if the testee is familiar with the language or even with the character set in question.

The usability penalty when using random character strings can be easily offset by allowing a small error in the given response. An edit distance metric that allows one bad character for example is suitable in this case.

Careless usage of different colors is arguably the most effective way to diminish the security of any CAPTCHA. The fundamental problem is that any change in color, slight or drastic, constitutes immediately a hint on how to segment the image. For example, rendering a string of overlapping characters where each character uses a distinct color is a trivial segmentation task. One may assume that this is well-understood, but instances of this design have been found in the wild as Figure 4 demonstrates.

Adding background clutter is another popular method that is supposed to complicate character recognition. There are, however, numerous instances where this kind of noise is entirely irrelevant for an adversary, including the examples in Section 2. This is somewhat related to the color discussion above: In the extreme case, characters are typeset in one color and some seemingly complex background uses any other color. A simple binarization operation is sufficient to separate background from characters in this case. Well-designed background clutter on the other hand is highly random and uses character-like shapes that share the same color and the same line width with the actual characters. Unfortunately, this is exactly what degrades usability and is one possible explanation why many strong CAPTCHAs use a plain backgrounds.

In general, one can differentiate between character segmentation and recognition. The former is the task of correctly decomposing a string of overlapping characters into individual characters while the latter deals with the recognition of a given (segmented) character. It is

(a) (b) (c)

Figure 5: One challenge, three versions; allowing for independent analysis of each letter. Used on `digg.com`.

(a) (b) (c) (d)

Figure 6: CAPTCHA used on `quoka.de`; three versions of the same challenge (a–c) with different backgrounds. A pixel-wise multiplication eliminates background noise almost entirely (d).

important to understand that segmentation is relatively hard for machines but recognition is easy given a correct segmentation. This has been studied extensively in [CLSC05]. A CAPTCHA designer should therefore rely on both the difficulty of both segmentation and recognition rather than recognition alone.

A very elegant way to generate new challenges automatically is to take advantage of the human user. This concept has been used by reCAPTCHA [vAMM+08] and [GKB09]. The basic idea is to present one challenge whose legitimate response is unknown together with $k-1$ known challenges at once. If the user is able to answer the $k-1$ challenges correctly, it is likely that she also provided the correct response to the unknown challenge. If a sufficient number of users agree on the response, this pair can be stored as known. It is, however, important that known and unknown challenges are indistinguishable to the user. Otherwise, the system can be exploited easily by a large number of users that always provide the same bogus response to all unknown challenges. Eventually, the database is polluted and accepts the bogus solutions when newly "learned" challenges are generated. Note that this kind of attack is still possible for small values of k even if the type of the challenges is indistinguishable.

3.2 Realization Advice

As an attacker only needs to target the weakest component of any CAPTCHA system, it is important to pay close attention to the surrounding implementation that is not strictly part of the generated challenge.

Prevent replay attacks. If allowed, this is perhaps the most effective type of attack. The system needs to keep track either of challenges that are yet to be answered or challenges that have been answered already (both correct or incorrect). Otherwise, a human attacker may solve one single instance and reuse the answer multiple times in a subsequent automated attack. This also implies that some kind of stateful server is necessary – an unattractive requirement in high-load environments, but vital nevertheless.

(a) (b) (c)

Figure 7: Evolution of reCAPTCHA.

One version per challenge. One specific challenge should be presented in one *version* only, i.e., the according image should be created in a deterministic fashion. Generating multiple versions of the same challenge by applying different transformations, as in Figure 5, gives a great advantage to the attacker. By requesting the same challenge multiple times, unrecognizable characters in one version can still be recognized in another version of the same challenge. In fact, since the same characters are used, the problem is reduced to correctly recognizing one single character independently from a set of samples instead of an entire string at once. This increases the success probability for the attacker exponentially in the number of requested versions.

A particular weak variant of this flaw arises when only background clutter varies over different versions. Figure 6 is one such example, where a pixel-wise multiplication operation essentially removes the background noise completely (Figure 6d). This CAPTCHA is used on "Quoka.de" as of March 2010, a popular online marketplace that claims[1] to have had over 78 million page impressions in July 2009. Note that this CAPTCHA can be broken by even less sophisticated means; it only serves as an example here.

Impose timeouts. A challenge should be valid only for a limited amount of time in order to restrict the adversary's attack frame. Note that the timeout should not be chosen too short either, since the user may be required to fill out some form first[2]. It is, however, unreasonable to accept challenges that have been given out several hours or days before submission.

Adapt to attackers. If distinct "users" can be identified, it makes sense to increase the difficulty of the test after several unsuccessful attempts have been made by the same user to solve it. This can be done by stronger distortions and/or more background clutter. For many public services it is impossible to identify distinct users, though. Using the testee's IP address is a popular choice in practice but generates many false positives within institutions that share one public address. Hence this option should be used with great care.

4 reCAPTCHA

reCAPTCHA (`recaptcha.net`) is perhaps the most successful and robust CAPTCHA systems of the last few years [vAMM+08]. The crucial property that separates it from regular text-based CAPTCHAs is the method of generating challenges—they are a by-product of digitizing huge amounts of text. Whenever OCR software fails to recognize one word properly, it is considered hard to digitize and is added to the reCAPTCHA system, after undergoing even further distortions. This immediately ensures that the word in question is not trivially readable by machines.

Since the correct responses to these challenges are not known, the system learns from its users as described in Section 3.2 by presenting one known and one unknown challenge. The system in its entirety also constitutes a remarkable win-win situation: We either get a secure CAPTCHA implementation or, if an attack exists, a significant advance in the accuracy of OCR systems. This has lead to enormous success: reCAPTCHA is used on many of the most popular sites on the Internet, including Facebook, Twitter and Craigslist. Over 200 million challenges are solved each day by its users according to reCAPTCHA.

Conceptually reCAPTCHA is also unique in the sense that it is a centralized service rather than downloadable software. One important implication is that there is no definite version of reCAPTCHA; it is a highly dynamic system that changes both slightly and drastically (cf. Figure 7) over time. Occasional claims that reCAPTCHA has been broken are therefore to be taken with a grain of salt, since this can possibly only refer to one point in time. It is consequently often difficult to verify such claims without a large test set of challenges from the relevant period of time. Even then, prompt changes by reCAPTCHA may have rendered the attack irrelevant in the meantime. Nevertheless, specific versions of reCAPTCHA are known to be prone to attacks from which we can learn. We discuss these in the next section.

Attacks on reCAPTCHA. In its original version, as depicted in Figure 7a, reCAPTCHA added a horizontal line to each challenge word as well as per-word distortion. In [Wil09] the author describes how challenges of this version can be broken by first applying a set of morphological image processing operations, specifically erosion (remove pixels on the border of objects) and dilation (add pixels on the border). Standard off-the-shelf OCR software, namely Ocropus, then processes the resulting images. The exact parameters for the morphological operations are learned from a training set of 200 labeled challenges. This approach is claimed to yield a 5% success rate for a test set of 200 challenges.

Likewise, a more recent version (Figure 7b) of reCAPTCHA can be solved automatically in 5% of the time by simply passing it to the Tesseract OCR software, without any preprocessing, according to the aforementioned author. Considering that the previous version was completely resistant against this trivial attack, there is some indication that reCAPTCHA in fact weakened their system between these two evolution steps.

The latest version of reCAPTCHA, as of April 2010, inverts the colors within a randomly

[1] `http://www.quoka-gmbh.de/resource/pdfs/Quoka_Mediaunterlagen_Online.pdf`
[2] Ideally, the CAPTCHA is presented separately before or after the actual form completion.

Figure 8: One of reCAPTCHA's current challenges, edge detection and binarization operations applied.

chosen, elliptic area that overlaps with the challenge words; Figure 7c depicts an instance of this method. This change seems to be reCAPTCHAs response to the publication of Wilkins [Wil09] in December 2009, since it has been introduced shortly after. This modification is remarkable in two aspects.

First, the statistics in [Wil09] appear to be quite fragile. The (basic) attack reported in [Wil09] guarantees about 5% on a sample space of only 200 challenges. This sample space is rather small compared to the low success probability. Assuming independent distributions with a success probability of 5% each, and approximating the (averaged) sum of successes for the 200 challenges by a normal distribution, we consider a confidence interval of 80%, i.e., the actual mean value really falls into the interval with at least 80% (leaving 20% for possibly lower or higher values). Then this confidence interval still ranges from 3% to 7%. One may quite reliably observe 5% success even though the actual success rate is much lower. As an example, for an actual success rate of 4% the 80% confidence interval would be approximately $[2.3\%, 5.7\%]$. The problem does not occur for our examples in the previous section as the success rate there is sufficiently high for the number of samples.

We also note that the optimistic success rate of up to 17.5% mentioned in [Wil09] is only valid under the assumption that, of the 25% of the additional cases where only one of the two words is recognized, half of these solutions are for the unknown word. It is unclear if this assumption is justified. In the worst case, the overall success rate could still be as low as 5% (or even lower, as discusses above).

In summary, it must be observed, though, that the actual recognition rate could in fact be higher than the estimates. It is only safe to say that the statistics, as given, do not provide a reliable lower bound.

The second issue about reCAPTCHA's evolution is that it is rather unclear to us if the aforementioned modification significantly improves the security. Although we are not aware of any attacks at this stage, there are a few properties that make this type of distortion possibly vulnerable:

1. There is little variance in the general shape of the object. It is always well-rounded and can be approximated roughly with an ellipse.

2. The edges of the distortion object are very "clean", it is therefore easy to detect these with standard algorithms. As shown in Figure 8, these edges have low curvature compared to the curves of characters and they often extend to areas where they stand out (top left and bottom right here). This makes for an easy start point for a line trace algorithm that also takes the shape of the object, an ellipse, into account.

3. The shape of the object forms a relatively large blob of black pixels in the verbatim challenge, making it easy to locate. For example, a vertical pixel projection reveals two maxima that correspond directly to the objects.

Once this distortion or large parts of it can be removed, the same attacks that apply to the previous version should work. With these considerations in mind, we expect that it is only a matter of time until reCAPTCHA is forced to adjust their system again.

5 Other CAPTCHA Types

So far we dealt exclusively with text-based CAPTCHAs. While this type of CAPTCHA is certainly the most common one, other approaches have been proposed and implemented, most predominantly image-based systems. There are many different reasons why these are interesting alternatives. We first discuss a few of these reasons and then give some concrete examples of implementations.

First, OCR systems constantly improve and have reached a level that demands very strong distortions. These, in turn, affect the usability of course and make it harder for humans to solve the challenge. Another aspect is the increasing number of small mobile devices that are capable of using standard web services. Often these devices do not have a hardware-based keyboard and the user is forced to use an alternative—possibly cumbersome—input method to solve a text-based CAPTCHA. Additionally, many web-based applications rely strongly on a pointing device for navigation where the user's hand spends most of the time on this device. It can be annoying if such a "click-based" navigation flow is suddenly interrupted by a text-based CAPTCHA challenge. Finally, it seems that CAPTCHAs which resemble some kind of small game challenge are more likely accepted than writing down a string of characters.

A typical issue of CAPTCHA systems that are not based on text input is a small response space. For example, the large class of binary decisions is of limited use only. Since the attacker can always be successful with probability $\frac{1}{2}$ by guessing, many of such challenges have to be combined in order to achieve a reasonably sized response space: A guessing probability of $0.1\% \approx 2^{-10}$ requires 10 challenges to be solved at once. Likewise—in the case of image-based systems—whenever the response is a position (more precisely a tolerance area) within the challenge image, comparatively large images are needed to reach 0.1%. A circular response area of 16 pixels diameter for instance would require approximately a 450×450 image, which is prohibitively large for mobile displays.

The systems we present here are a selected set of research projects mostly. We find these particularly interesting, despite their availability or readiness for productive use. Some of them are, in fact, already broken but they demonstrate novel approaches to this topic.

Image Rotation. This novel CAPTCHA system, presented in [GKB09], asks the user to rotate randomly oriented images to their natural upright position. While this task is easy to automate for a certain class of images, such as photographs of faces, type or clear horizon

lines, it is considered hard for a wide range of images in general. Experimental results indicate that this system works well if the images do have a natural upright orientation. The actual challenge here is to filter the other, infeasible, images for which the authors suggest an automated method. However, unsuitable images may still slip through this preprocessing stage and need to be filtered out in the live system by evaluating the variance of the chosen user rotations.

Asirra. Asirra [EDHS07] is an image-based CAPTCHA developed at Microsoft Research. It relies on the presumed difficulty to automatically tell pictures of dogs and cats apart. The system presents a set of 12 pictures at once and asks the user to select those which depict a cat. Surprisingly, an automated attack [Gol08] exists which is able to distinguish these two types of photos very reliably with a probability of 82.7% and is thus able to solve the Asirra challenge in 12.2% of the time. This result highlights how difficult it is to design a strong CAPTCHA, even if the challenge it may appear hard at first.

6 Conclusions

Designing good CAPTCHAs is a tedious business. Text CAPTCHAs achieve a high level of practicality, but very often fall short of providing a good balance between usability and security. Currently, the best choice seems to be reCAPTCHA: it is easy to incorperate, achieves appropriate security and usability levels, and because of the centralized structure behind reCAPTCHA it allows fast migration in response to emerging threats.

Acknowledgments. We thank the anonymous reviewers for valuable comments. Paul Baecher, Marc Fischlin and Dominique Schröder are supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG). This work was also supported by CASED (`www.cased.de`).

References

[CLSC05] Kumar Chellapilla, Kevin Larson, Patrice Y. Simard, and Mary Czerwinski. Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs). In *HIP*, volume 3517 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2005.

[EDHS07] Jeremy Elson, John R. Douceur, Jon Howell, and Jared Saul. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In *ACM Conference on Computer and Communications Security*, pages 366–374. ACM, 2007.

[GKB09] Rich Gossweiler, Maryam Kamvar, and Shumeet Baluja. What's up CAPTCHA?: a CAPTCHA based on image orientation. In *WWW*, pages 841–850. ACM, 2009.

[Gol08] Philippe Golle. Machine learning attacks against the Asirra CAPTCHA. In *ACM Conference on Computer and Communications Security*, pages 535–542. ACM, 2008.

[Mac67] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[vAMM+08] Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, 321(5895):1465–1468, 2008.

[Wil09] Jonathan Wilkins. Strong CAPTCHA Guidelines v1.2. 2009.

[YA08] Jeff Yan and Ahmad Salah El Ahmad. Usability of CAPTCHAs or usability issues in CAPTCHA design. In *SOUPS*, ACM International Conference Proceeding Series, pages 44–52. ACM, 2008.

GI-Edition Lecture Notes in Informatics

P-1 Gregor Engels, Andreas Oberweis, Albert Zündorf (Hrsg.): Modellierung 2001.

P-2 Mikhail Godlevsky, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications, ISTA'2001.

P-3 Ana M. Moreno, Reind P. van de Riet (Hrsg.): Applications of Natural Language to Information Systems, NLDB'2001.

P-4 H. Wörn, J. Mühling, C. Vahl, H.-P. Meinzer (Hrsg.): Rechner- und sensorgestützte Chirurgie; Workshop des SFB 414.

P-5 Andy Schürr (Hg.): OMER – Object-Oriented Modeling of Embedded Real-Time Systems.

P-6 Hans-Jürgen Appelrath, Rolf Beyer, Uwe Marquardt, Heinrich C. Mayr, Claudia Steinberger (Hrsg.): Unternehmen Hochschule, UH'2001.

P-7 Andy Evans, Robert France, Ana Moreira, Bernhard Rumpe (Hrsg.): Practical UML-Based Rigorous Development Methods – Countering or Integrating the extremists, pUML'2001.

P-8 Reinhard Keil-Slawik, Johannes Magenheim (Hrsg.): Informatikunterricht und Medienbildung, INFOS'2001.

P-9 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Innovative Anwendungen in Kommunikationsnetzen, 15. DFN Arbeitstagung.

P-10 Mirjam Minor, Steffen Staab (Hrsg.): 1st German Workshop on Experience Management: Sharing Experiences about the Sharing Experience.

P-11 Michael Weber, Frank Kargl (Hrsg.): Mobile Ad-Hoc Netzwerke, WMAN 2002.

P-12 Martin Glinz, Günther Müller-Luschnat (Hrsg.): Modellierung 2002.

P-13 Jan von Knop, Peter Schirmbacher and Viljan Mahni_ (Hrsg.): The Changing Universities – The Role of Technology.

P-14 Robert Tolksdorf, Rainer Eckstein (Hrsg.): XML-Technologien für das Semantic Web – XSW 2002.

P-15 Hans-Bernd Bludau, Andreas Koop (Hrsg.): Mobile Computing in Medicine.

P-16 J. Felix Hampe, Gerhard Schwabe (Hrsg.): Mobile and Collaborative Busi-ness 2002.

P-17 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Zukunft der Netze –Die Verletzbarkeit meistern, 16. DFN Arbeitstagung.

P-18 Elmar J. Sinz, Markus Plaha (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2002.

P-19 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3.Okt. 2002 in Dortmund.

P-20 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3.Okt. 2002 in Dortmund (Ergänzungsband).

P-21 Jörg Desel, Mathias Weske (Hrsg.): Promise 2002: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen.

P-22 Sigrid Schubert, Johannes Magenheim, Peter Hubwieser, Torsten Brinda (Hrsg.): Forschungsbeiträge zur "Didaktik der Informatik" – Theorie, Praxis, Evaluation.

P-23 Thorsten Spitta, Jens Borchers, Harry M. Sneed (Hrsg.): Software Management 2002 – Fortschritt durch Beständigkeit

P-24 Rainer Eckstein, Robert Tolksdorf (Hrsg.): XMIDX 2003 – XML-Technologien für Middleware – Middleware für XML-Anwendungen

P-25 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Commerce – Anwendungen und Perspektiven – 3. Workshop Mobile Commerce, Universität Augsburg, 04.02.2003

P-26 Gerhard Weikum, Harald Schöning, Erhard Rahm (Hrsg.): BTW 2003: Datenbanksysteme für Business, Technologie und Web

P-27 Michael Kroll, Hans-Gerd Lipinski, Kay Melzer (Hrsg.): Mobiles Computing in der Medizin

P-28 Ulrich Reimer, Andreas Abecker, Steffen Staab, Gerd Stumme (Hrsg.): WM 2003: Professionelles Wissensmanagement – Erfahrungen und Visionen

P-29 Antje Düsterhöft, Bernhard Thalheim (Eds.): NLDB'2003: Natural Language Processing and Information Systems

P-30 Mikhail Godlevsky, Stephen Liddle, Heinrich C. Mayr (Eds.): Information Systems Technology and its Applications

P-31 Arslan Brömme, Christoph Busch (Eds.): BIOSIG 2003: Biometrics and Electronic Signatures

P-32 Peter Hubwieser (Hrsg.): Informatische Fachkonzepte im Unterricht – INFOS 2003

P-33 Andreas Geyer-Schulz, Alfred Taudes (Hrsg.): Informationswirtschaft: Ein Sektor mit Zukunft

P-34 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 1)

P-35 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 2)

P-36 Rüdiger Grimm, Hubert B. Keller, Kai Rannenberg (Hrsg.): Informatik 2003 – Mit Sicherheit Informatik

P-37 Arndt Bode, Jörg Desel, Sabine Rathmayer, Martin Wessner (Hrsg.): DeLFI 2003: e-Learning Fachtagung Informatik

P-38 E.J. Sinz, M. Plaha, P. Neckel (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2003

P-39 Jens Nedon, Sandra Frings, Oliver Göbel (Hrsg.): IT-Incident Management & IT-Forensics – IMF 2003

P-40 Michael Rebstock (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2004

P-41 Uwe Brinkschulte, Jürgen Becker, Dietmar Fey, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle, Thomas Runkler (Edts.): ARCS 2004 – Organic and Pervasive Computing

P-42 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Economy – Transaktionen und Prozesse, Anwendungen und Dienste

P-43 Birgitta König-Ries, Michael Klein, Philipp Obreiter (Hrsg.): Persistance, Scalability, Transactions – Database Mechanisms for Mobile Applications

P-44 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): Security, E-Learning. E-Services

P-45 Bernhard Rumpe, Wofgang Hesse (Hrsg.): Modellierung 2004

P-46 Ulrich Flegel, Michael Meier (Hrsg.): Detection of Intrusions of Malware & Vulnerability Assessment

P-47 Alexander Prosser, Robert Krimmer (Hrsg.): Electronic Voting in Europe – Technology, Law, Politics and Society

P-48 Anatoly Doroshenko, Terry Halpin, Stephen W. Liddle, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications

P-49 G. Schiefer, P. Wagner, M. Morgenstern, U. Rickert (Hrsg.): Integration und Datensicherheit – Anforderungen, Konflikte und Perspektiven

P-50 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 1) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-51 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 2) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-52 Gregor Engels, Silke Seehusen (Hrsg.): DELFI 2004 – Tagungsband der 2. e-Learning Fachtagung Informatik

P-53 Robert Giegerich, Jens Stoye (Hrsg.): German Conference on Bioinformatics – GCB 2004

P-54 Jens Borchers, Ralf Kneuper (Hrsg.): Softwaremanagement 2004 – Outsourcing und Integration

P-55 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): E-Science und Grid Adhoc-Netze Medienintegration

P-56 Fernand Feltz, Andreas Oberweis, Benoit Otjacques (Hrsg.): EMISA 2004 – Informationssysteme im E-Business und E-Government

P-57 Klaus Turowski (Hrsg.): Architekturen, Komponenten, Anwendungen

P-58 Sami Beydeda, Volker Gruhn, Johannes Mayer, Ralf Reussner, Franz Schweiggert (Hrsg.): Testing of Component-Based Systems and Software Quality

P-59 J. Felix Hampe, Franz Lehner, Key Pousttchi, Kai Ranneberg, Klaus Turowski (Hrsg.): Mobile Business – Processes, Platforms, Payments

P-60 Steffen Friedrich (Hrsg.): Unterrichtskonzepte für inforrmatische Bildung

P-61 Paul Müller, Reinhard Gotzhein, Jens B. Schmitt (Hrsg.): Kommunikation in verteilten Systemen

P-62 Federrath, Hannes (Hrsg.): „Sicherheit 2005“ – Sicherheit – Schutz und Zuverlässigkeit

P-63 Roland Kaschek, Heinrich C. Mayr, Stephen Liddle (Hrsg.): Information Systems – Technology and ist Applications

P-64 Peter Liggesmeyer, Klaus Pohl, Michael Goedicke (Hrsg.): Software Engineering 2005

P-65 Gottfried Vossen, Frank Leymann, Peter Lockemann, Wolffried Stucky (Hrsg.): Datenbanksysteme in Business, Technologie und Web

P-66 Jörg M. Haake, Ulrike Lucke, Djamshid Tavangarian (Hrsg.): DeLFI 2005: 3. deutsche e-Learning Fachtagung Informatik

P-67 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 1)

P-68 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 2)

P-69 Robert Hirschfeld, Ryszard Kowalcyk, Andreas Polze, Matthias Weske (Hrsg.): NODe 2005, GSEM 2005

P-70 Klaus Turowski, Johannes-Maria Zaha (Hrsg.): Component-oriented Enterprise Application (COAE 2005)

P-71 Andrew Torda, Stefan Kurz, Matthias Rarey (Hrsg.): German Conference on Bioinformatics 2005

P-72 Klaus P. Jantke, Klaus-Peter Fähnrich, Wolfgang S. Wittig (Hrsg.): Marktplatz Internet: Von e-Learning bis e-Payment

P-73 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): “Heute schon das Morgen sehen“

P-74 Christopher Wolf, Stefan Lucks, Po-Wah Yau (Hrsg.): WEWoRC 2005 – Western European Workshop on Research in Cryptology

P-75 Jörg Desel, Ulrich Frank (Hrsg.): Enterprise Modelling and Information Systems Architecture

P-76 Thomas Kirste, Birgitta König-Riess, Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Informationssysteme – Potentiale, Hindernisse, Einsatz

P-77 Jana Dittmann (Hrsg.): SICHERHEIT 2006

P-78 K.-O. Wenkel, P. Wagner, M. Morgenstern, K. Luzi, P. Eisermann (Hrsg.): Land- und Ernährungswirtschaft im Wandel

P-79 Bettina Biel, Matthias Book, Volker Gruhn (Hrsg.): Softwareengineering 2006

P-80 Mareike Schoop, Christian Huemer, Michael Rebstock, Martin Bichler (Hrsg.): Service-Oriented Electronic Commerce

P-81 Wolfgang Karl, Jürgen Becker, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle (Hrsg.): ARCS´06

P-82 Heinrich C. Mayr, Ruth Breu (Hrsg.): Modellierung 2006

P-83 Daniel Huson, Oliver Kohlbacher, Andrei Lupas, Kay Nieselt and Andreas Zell (eds.): German Conference on Bioinformatics

P-84 Dimitris Karagiannis, Heinrich C. Mayr, (Hrsg.): Information Systems Technology and its Applications

P-85 Witold Abramowicz, Heinrich C. Mayr, (Hrsg.): Business Information Systems

P-86 Robert Krimmer (Ed.): Electronic Voting 2006

P-87 Max Mühlhäuser, Guido Rößling, Ralf Steinmetz (Hrsg.): DELFI 2006: 4. e-Learning Fachtagung Informatik

P-88 Robert Hirschfeld, Andreas Polze, Ryszard Kowalczyk (Hrsg.): NODe 2006, GSEM 2006

P-90 Joachim Schelp, Robert Winter, Ulrich Frank, Bodo Rieger, Klaus Turowski (Hrsg.): Integration, Informationslogistik und Architektur

P-91 Henrik Stormer, Andreas Meier, Michael Schumacher (Eds.): European Conference on eHealth 2006

P-92 Fernand Feltz, Benoît Otjacques, Andreas Oberweis, Nicolas Poussing (Eds.): AIM 2006

P-93 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 1

P-94 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 2

P-95 Matthias Weske, Markus Nüttgens (Eds.): EMISA 2005: Methoden, Konzepte und Technologien für die Entwicklung von dienstbasierten Informationssystemen

P-96 Saartje Brockmans, Jürgen Jung, York Sure (Eds.): Meta-Modelling and Ontologies

P-97 Oliver Göbel, Dirk Schadt, Sandra Frings, Hardo Hase, Detlef Günther, Jens Nedon (Eds.): IT-Incident Mangament & IT-Forensics – IMF 2006

P-98 Hans Brandt-Pook, Werner Simonsmeier und Thorsten Spitta (Hrsg.): Beratung in der Softwareentwicklung – Modelle, Methoden, Best Practices

P-99 Andreas Schwill, Carsten Schulte, Marco Thomas (Hrsg.): Didaktik der Informatik

P-100 Peter Forbrig, Günter Siegel, Markus Schneider (Hrsg.): HDI 2006: Hochschuldidaktik der Informatik

P-101 Stefan Böttinger, Ludwig Theuvsen, Susanne Rank, Marlies Morgenstern (Hrsg.): Agrarinformatik im Spannungsfeld zwischen Regionalisierung und globalen Wertschöpfungsketten

P-102 Otto Spaniol (Eds.): Mobile Services and Personalized Environments

P-103 Alfons Kemper, Harald Schöning, Thomas Rose, Matthias Jarke, Thomas Seidl, Christoph Quix, Christoph Brochhaus (Hrsg.): Datenbanksysteme in Business, Technologie und Web (BTW 2007)

P-104 Birgitta König-Ries, Franz Lehner, Rainer Malaka, Can Türker (Hrsg.) MMS 2007: Mobilität und mobile Informationssysteme

P-105 Wolf-Gideon Bleek, Jörg Raasch, Heinz Züllighoven (Hrsg.) Software Engineering 2007

P-106 Wolf-Gideon Bleek, Henning Schwentner, Heinz Züllighoven (Hrsg.) Software Engineering 2007 – Beiträge zu den Workshops

P-107 Heinrich C. Mayr, Dimitris Karagiannis (eds.) Information Systems Technology and its Applications

P-108 Arslan Brömme, Christoph Busch, Detlef Hühnlein (eds.) BIOSIG 2007: Biometrics and Electronic Signatures

P-109 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 1

P-110 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 2

P-111 Christian Eibl, Johannes Magenheim, Sigrid Schubert, Martin Wessner (Hrsg.) DeLFI 2007: 5. e-Learning Fachtagung Informatik

P-112 Sigrid Schubert (Hrsg.) Didaktik der Informatik in Theorie und Praxis

P-113 Sören Auer, Christian Bizer, Claudia Müller, Anna V. Zhdanova (Eds.) The Social Semantic Web 2007 Proceedings of the 1st Conference on Social Semantic Web (CSSW)

P-114 Sandra Frings, Oliver Göbel, Detlef Günther, Hardo G. Hase, Jens Nedon, Dirk Schadt, Arslan Brömme (Eds.) IMF2007 IT-incident management & IT-forensics Proceedings of the 3rd International Conference on IT-Incident Management & IT-Forensics

P-115 Claudia Falter, Alexander Schliep, Joachim Selbig, Martin Vingron and Dirk Walther (Eds.) German conference on bioinformatics GCB 2007

P-116 Witold Abramowicz, Leszek Maciszek (Eds.) Business Process and Services Computing 1st International Working Conference on Business Process and Services Computing BPSC 2007

P-117 Ryszard Kowalczyk (Ed.) Grid service engineering and manegement The 4th International Conference on Grid Service Engineering and Management GSEM 2007

P-118 Andreas Hein, Wilfried Thoben, Hans-Jürgen Appelrath, Peter Jensch (Eds.) European Conference on ehealth 2007

P-119 Manfred Reichert, Stefan Strecker, Klaus Turowski (Eds.) Enterprise Modelling and Information Systems Architectures Concepts and Applications

P-120 Adam Pawlak, Kurt Sandkuhl, Wojciech Cholewa, Leandro Soares Indrusiak (Eds.) Coordination of Collaborative Engineering - State of the Art and Future Challenges

P-121 Korbinian Herrmann, Bernd Bruegge (Hrsg.) Software Engineering 2008 Fachtagung des GI-Fachbereichs Softwaretechnik

P-122 Walid Maalej, Bernd Bruegge (Hrsg.) Software Engineering 2008 - Workshopband Fachtagung des GI-Fachbereichs Softwaretechnik

P-123 Michael H. Breitner, Martin Breunig, Elgar Fleisch, Ley Pousttchi, Klaus Turowski (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Technologien, Prozesse, Marktfähigkeit
Proceedings zur 3. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2008)

P-124 Wolfgang E. Nagel, Rolf Hoffmann, Andreas Koch (Eds.)
9th Workshop on Parallel Systems and Algorithms (PASA)
Workshop of the GI/ITG Speciel Interest Groups PARS and PARVA

P-125 Rolf A.E. Müller, Hans-H. Sundermeier, Ludwig Theuvsen, Stephanie Schütze, Marlies Morgenstern (Hrsg.)
Unternehmens-IT: Führungsinstrument oder Verwaltungsbürde
Referate der 28. GIL Jahrestagung

P-126 Rainer Gimnich, Uwe Kaiser, Jochen Quante, Andreas Winter (Hrsg.)
10th Workshop Software Reengineering (WSR 2008)

P-127 Thomas Kühne, Wolfgang Reisig, Friedrich Steimann (Hrsg.)
Modellierung 2008

P-128 Ammar Alkassar, Jörg Siekmann (Hrsg.)
Sicherheit 2008
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 4. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
2.-4. April 2008
Saarbrücken, Germany

P-129 Wolfgang Hesse, Andreas Oberweis (Eds.)
Sigsand-Europe 2008
Proceedings of the Third AIS SIGSAND European Symposium on Analysis, Design, Use and Societal Impact of Information Systems

P-130 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
1. DFN-Forum Kommunikations-technologien Beiträge der Fachtagung

P-131 Robert Krimmer, Rüdiger Grimm (Eds.)
3rd International Conference on Electronic Voting 2008
Co-organized by Council of Europe, Gesellschaft für Informatik and E-Voting.CC

P-132 Silke Seehusen, Ulrike Lucke, Stefan Fischer (Hrsg.)
DeLFI 2008:
Die 6. e-Learning Fachtagung Informatik

P-133 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik
Band 1

P-134 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik
Band 2

P-135 Torsten Brinda, Michael Fothe, Peter Hubwieser, Kirsten Schlüter (Hrsg.)
Didaktik der Informatik – Aktuelle Forschungsergebnisse

P-136 Andreas Beyer, Michael Schroeder (Eds.)
German Conference on Bioinformatics GCB 2008

P-137 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2008: Biometrics and Electronic Signatures

P-138 Barbara Dinter, Robert Winter, Peter Chamoni, Norbert Gronau, Klaus Turowski (Hrsg.)
Synergien durch Integration und Informationslogistik
Proceedings zur DW2008

P-139 Georg Herzwurm, Martin Mikusz (Hrsg.)
Industrialisierung des Software-Managements
Fachtagung des GI-Fachausschusses Management der Anwendungsentwick-lung und -wartung im Fachbereich Wirtschaftsinformatik

P-140 Oliver Göbel, Sandra Frings, Detlef Günther, Jens Nedon, Dirk Schadt (Eds.)
IMF 2008 - IT Incident Management & IT Forensics

P-141 Peter Loos, Markus Nüttgens, Klaus Turowski, Dirk Werth (Hrsg.)
Modellierung betrieblicher Informations-systeme (MobIS 2008)
Modellierung zwischen SOA und Compliance Management

P-142 R. Bill, P. Korduan, L. Theuvsen, M. Morgenstern (Hrsg.)
Anforderungen an die Agrarinformatik durch Globalisierung und Klimaveränderung

P-143 Peter Liggesmeyer, Gregor Engels, Jürgen Münch, Jörg Dörr, Norman Riegel (Hrsg.)
Software Engineering 2009
Fachtagung des GI-Fachbereichs Softwaretechnik

P-144 Johann-Christoph Freytag, Thomas Ruf, Wolfgang Lehner, Gottfried Vossen (Hrsg.)
Datenbanksysteme in Business, Technologie und Web (BTW)

P-145 Knut Hinkelmann, Holger Wache (Eds.)
WM2009: 5th Conference on Professional Knowledge Management

P-146 Markus Bick, Martin Breunig, Hagen Höpfner (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Entwicklung, Implementierung und Anwendung
4. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2009)

P-147 Witold Abramowicz, Leszek Maciaszek, Ryszard Kowalczyk, Andreas Speck (Eds.)
Business Process, Services Computing and Intelligent Service Management
BPSC 2009 · ISM 2009 · YRW-MBP 2009

P-148 Christian Erfurth, Gerald Eichler, Volkmar Schau (Eds.)
9th International Conference on Innovative Internet Community Systems
I²CS 2009

P-149 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
2. DFN-Forum Kommunikationstechnologien
Beiträge der Fachtagung

P-150 Jürgen Münch, Peter Liggesmeyer (Hrsg.)
Software Engineering
2009 - Workshopband

P-151 Armin Heinzl, Peter Dadam, Stefan Kirn, Peter Lockemann (Eds.)
PRIMIUM
Process Innovation for Enterprise Software

P-152 Jan Mendling, Stefanie Rinderle-Ma, Werner Esswein (Eds.)
Enterprise Modelling and Information Systems Architectures
Proceedings of the 3rd Int'l Workshop EMISA 2009

P-153 Andreas Schwill, Nicolas Apostolopoulos (Hrsg.)
Lernen im Digitalen Zeitalter
DeLFI 2009 – Die 7. E-Learning Fachtagung Informatik

P-154 Stefan Fischer, Erik Maehle Rüdiger Reischuk (Hrsg.)
INFORMATIK 2009
Im Focus das Leben

P-155 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2009:
Biometrics and Electronic Signatures
Proceedings of the Special Interest Group on Biometrics and Electronic Signatures

P-156 Bernhard Koerber (Hrsg.)
Zukunft braucht Herkunft
25 Jahre »INFOS – Informatik und Schule«

P-157 Ivo Grosse, Steffen Neumann, Stefan Posch, Falk Schreiber, Peter Stadler (Eds.)
German Conference on Bioinformatics 2009

P-158 W. Claupein, L. Theuvsen, A. Kämpf, M. Morgenstern (Hrsg.)
Precision Agriculture Reloaded – Informationsgestützte Landwirtschaft

P-159 Gregor Engels, Markus Luckey, Wilhelm Schäfer (Hrsg.)
Software Engineering 2010

P-160 Gregor Engels, Markus Luckey, Alexander Pretschner, Ralf Reussner (Hrsg.)
Software Engineering 2010 – Workshopband
(inkl. Doktorandensymposium)

P-161 Gregor Engels, Dimitris Karagiannis Heinrich C. Mayr (Hrsg.)
Modellierung 2010

P-162 Maria A. Wimmer, Uwe Brinkhoff, Siegfried Kaiser, Dagmar Lück-Schneider, Erich Schweighofer, Andreas Wiebe (Hrsg.)
Vernetzte IT für einen effektiven Staat
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI) 2010

P-163 Markus Bick, Stefan Eulgem, Elgar Fleisch, J. Felix Hampe, Birgitta König-Ries, Franz Lehner, Key Pousttchi, Kai Rannenberg (Hrsg.)
Mobile und Ubiquitäre Informationssysteme
Technologien, Anwendungen und Dienste zur Unterstützung von mobiler Kollaboration

P-165 Gerald Eichler, Peter Kropf, Ulrike Lechner, Phayung Meesad, Herwig Unger (Eds.)
10th International Conference on Innovative Internet Community Systems (I²CS) – Jubilee Edition 2010 –

P-166 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
3. DFN-Forum Kommunikationstechnologien
Beiträge der Fachtagung

P-167 Robert Krimmer, Rüdiger Grimm (Eds.)
4th International Conference on Electronic Voting 2010
co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC

P-168 Ira Diethelm, Christina Dörge, Claudia Hildebrandt, Carsten Schulte (Hrsg.)
Didaktik der Informatik
Möglichkeiten empirischer Forschungsmethoden und Perspektiven der Fachdidaktik

P-169 Michael Kerres, Nadine Ojstersek Ulrik Schroeder, Ulrich Hoppe (Hrsg.)
DeLFI 2010 - 8. Tagung der Fachgruppe E-Learning der Gesellschaft für Informatik e.V.

P-170 Felix C. Freiling (Hrsg.)
Sicherheit 2010
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 5. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
5.–7. Oktober 2010, Berlin

P-171 Werner Esswein, Klaus Turowski, Martin Juhrisch (Hrsg.)
Modellierung betrieblicher Informationssysteme (MobIS 2010)
Modellgestütztes Management

The titles can be purchased at:

Köllen Druck + Verlag GmbH
Ernst-Robert-Curtius-Str. 14 · D-53117 Bonn
Fax: +49 (0)228/9898222
E-Mail: druckverlag@koellen.de