GESELLSCHAFT
FÜR INFORMATIK

Christian Wressnegger, Delphine Reinhardt, Thomas Barber,
Bernhard C. Witt, Daniel Arp, Zoltan Mann (Hrsg.)

SICHERHEIT 2022

Sicherheit, Schutz und Zuverlässigkeit

Konferenzband der 11. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)

5.-8. April 2022
in Karlsruhe

Gesellschaft für Informatik e.V. (GI)

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

Volume P-323

ISBN 978-3-88579-717-3
ISSN 1617-5468

Volume Editors
Prof. Dr. Christian Wressnegger
Karlsruher Institut für Technologie, Kaiserstraße 12, 76131 Karlsruhe
Prof. Dr. Delphine Reinhardt (Universität Göttingen, Goldschmidtstr. 7, 37077 Göttingen)
Dr. Thomas Barber (SAP Research, Vincenz-Prießnitz-Straße 1, 76131 Karlsruhe)
Bernhard C. Witt (it.sec GmbH, Einsteinstr. 55, 89077 Ulm)
Dr. Daniel Arp (TU Berlin, Marchstraße 23, 10587 Berlin)
Dr. Zoltán Mann (Universität Amsterdam, Science Park 904, 1098 XH Amsterdam)

printed by Köllen Druck+Verlag GmbH, Bonn

Vorwort

Die SICHERHEIT 2022 ist die elfte Ausgabe der regelmäßig stattfindenden Fachtagung des Fachbereichs "Sicherheit, Schutz und Zuverlässigkeit" der Gesellschaft für Informatik e.V. (GI). Sie bietet einem Publikum aus Forschung, Entwicklung und Anwendung ein Forum zur Diskussion von Herausforderungen, Trends, Techniken und neuesten wissenschaftlichen sowie praxisnahen Ergebnissen in der Cybersicherheit. Es werden alle Aspekte der Sicherheit informationstechnischer Systeme abgedeckt und so eine Brücke zwischen den Themen IT Security, Safety und Dependability geschlagen.

Zusätzlich zu dem gewohnten Programm aus wissenschaftlichen Beiträgen bietet die SICHERHEIT 2022 auch dieses Jahr wieder einen „Practitioners Track" sowie ein Doktorand·innenforum. Zum einen soll dadurch die Nähe und Vernetzung mit der Industrie gefördert werden. Zum anderen bieten wir dem wissenschaftlichen Nachwuchs die Möglichkeit sich sowohl untereinander als auch mit erfahrenen Forscher·innen im Bereich Cybersicherheit auszutauschen. Die Relevanz der Nachwuchsförderung wird außerdem durch das im Rahmen der Konferenz stattfindende Finale des CAST/GI-Promotionspreises IT-Sicherheit unterstrichen. Wir bieten Forscher·innen, die sich mit einer überdurchschnittlichen Leistung in ihrer Promotion verdient gemacht haben, eine Bühne und die Möglichkeit ihre Arbeiten vorzustellen.

Für das Programm der SICHERHEIT 2022 wurden aus insgesamt 41 Einreichungen 10 Beiträge als reguläre Papiere im wissenschaftlichen Hauptteil akzeptiert. Im Sinne der Nachwuchsförderung werden darüber hinaus auch 3 Kurzpapiere vorgestellt, die thematisch besonders gut geeignet sind, um interessante Diskussionen anzustoßen. Weiterhin wurden jeweils 4 Papiere im Rahmen des Doktorand·innenforums und des „Practitioners Tracks" ausgewählt. Die Beiträge sind traditionsgemäß in deutscher als auch in englischer Sprache verfasst.

Ergänzend zu den Präsentationen der eingereichten Beiträge ist es uns außerdem gelungen namhafte und fest in der deutschsprachigen Sicherheitsgemeinschaft verankerte Persönlichkeiten für eingeladene Vorträge zu gewinnen. Im Speziellen werden Prof. Dr. Konrad Rieck (TU Braunschweig), Prof. Dr. Rainer Böhme (Universität Innsbruck) und Prof. Dr. Stefan Katzenbeisser (Universität Passau) das Programm bereichern.

Wir bedanken uns vielmals bei allen Mitwirkenden der Fachtagung. Herzlichen Dank den Autor·innen, die uns Ihre Arbeiten zugesandt haben, den vielen eifrigen Gutachter·innen ohne die die Auswahl des Programms nicht zu stemmen gewesen wäre, sowie unseren großzügigen Sponsoren. Einen besonderen Dank gebührt auch der Gesellschaft für Informatik für die tatkräftige Unterstützung, die Planung und die fabelhafte Umsetzung der Veranstaltung.

Ohne dem Zusammenspiel aller wäre dies nicht möglich gewesen. Vielen Dank!

Christian Wressnegger
Delphine Reinhardt
Bernhard C. Witt

Karlsruhe, April 2022

Sponsoren

Wir danken den folgenden Unternehmen und Institutionen für Ihre Unterstützung.

ERNW Research
(Gold Partner)

genua GmbH
(Gold Partner)

Connecting Media
(Silber Partner)

SAP SE
(Silber Partner)

aramido GmbH
(Bronze Partner)

Darkdefense GmbH
(Bronze Partner)

KASTEL Security Research Labs
(Keynote Partner)

Fachbereich Sicherheit – Schutz und Zuverlässigkeit

Der GI-Fachbereich "Sicherheit - Schutz und Zuverlässigkeit" wurde 2002 gegründet und vernetzt zwei „Communities“ miteinander: Während die „Safety-Community“ vor allem den Schutz der Umwelt vor IT-Systemen (beispielsweise Sicherheit des Menschen vor schwerwiegenden Systemfehlern in Flugzeugen, Kernreaktoren und Kraftwerken) sowie Fehlertoleranzmaßnahmen (z.B. Systemausfälle als Folge von Ermüdungserscheinungen, Softwarefehlern und Naturereignissen) im Blick hat, beschäftigt sich die „Security-Community“ hauptsächlich mit dem Schutz der IT-Systeme und ihrer Umgebung vor Bedrohungen von außen, insbesondere vor Gefahren, die von bösartigen Angriffen (durch Menschen) ausgehen. Sicherheit ist ein Querschnittsthema. Für den Fachbereich gilt daher eine hohe Flexibilität hinsichtlich der Möglichkeiten zur Quervernetzung verschiedener Gruppen und Themen innerhalb und außerhalb der GI. Diese Quervernetzung wird sowohl in gemeinsamen Veranstaltungen als auch in der starken Berücksichtigung anderer Themen aus der Informatik deutlich. Sicherheit ist kein Selbstzweck, sondern wichtig zur Erfüllung gesellschaftlicher und wirtschaftlicher Bedürfnisse.

Tagungsleitung

Christian Wressnegger, Karlsruher Institut für Technologie
Delphine Reinhardt, Universität Göttingen
Bernhard C. Witt, it.sec GmbH

Programmkomitee

Chair: Christian Wressnegger, Karlsruher Institut für Technologie
Co-Chair: Delphine Reinhardt, Universität Göttingen

Andreas Berl	Technische Hochschule Deggendorf
Andreas Mayer	Hochschule Heilbronn
Andreas Noack	Hochschule Stralsund
Andriy Panchenko	BTU Cottbus
Anne Koziolek	Karlsruher Institut für Technologie
Arno Wacker	Universität der Bundeswehr München
Basel Katt	Norwegian University of Science and Technology
Bernd Freisleben	Philipps-Universität Marburg
Bernhard Beckert	Karlsruher Institut für Technologie
Bernhard C. Witt	it.sec GmbH
Bernhard Fechner	FernUniversität in Hagen
Bernhard Hämmerli	Hochschule Luzern
Bettina Buth	HAW Hamburg
Björn Scheuermann	TU Darmstadt
Chris Dietrich	Westfälische Hochschule
Christian Kraetzer	Otto-von-Guericke-Universität Magdeburg
Christian Rossow	CISPA Helmholtz-Zentrum für Informationssicherheit
Christoph Karg	Hochschule Aalen

Christoph Krauß	Hochschule Darmstadt
Christoph Striecks	AIT Austrian Institute of Technology
Dieter Gollmann	TU Hamburg-Harburg
Dieter Hogrefe	Universität Göttingen
Dieter Hutter	DFKI
Dirk Westhoff	Hochschule Offenburg
Dominik Engel	Fachhochschule Salzburg
Dominik Herrmann	Otto-Friedrich-Universität Bamberg
Dominik Merli	Hochschule Augsburg
Eberhard Zehendner	FSU Jena
Eckehard Hermann	Fachhochschule Oberösterreich
Edgar Weippl	Universität Wien
Erhard Plödereder	Universität Stuttgart
Erik Buchmann	Hochschule für Telekommunikation Leipzig
Evren Eren	Hochschule Bremen
Felix Freiling	Friedrich-Alexander-Universität Erlangen-Nürnberg
Florian Tschorsch	Humboldt-Universität zu Berlin
Frank Kargl	Universität Ulm
Friedbert Kaspar	Hochschule Furtwangen
Hannes Federrath	Universität Hamburg
Hannes Hartenstein	Karlsruher Institut für Technologie
Hanno Langweg	HTWG Konstanz
Harald Baier	Universität der Bundeswehr München
Heiko Roßnagel	Fraunhofer IAO
Henning Schnoor	CAU Kiel
Herbert Leitold	A-SIT
Holger Morgenstern	Hochschule Albstadt-Sigmaringen
Hubert B. Keller	WTB Karlsruhe
Ingo Stengel	Hochschule Karlsruhe
Isabel Münch	Bundesamt für Sicherheit in der Informationstechnik
Jan Jürjens	Universität Koblenz-Landau
Jana Dittmann	Otto-von-Guericke Universität Magdeburg
Jens-Peter Akelbein	Hochschule Darmstadt
Jernej Tonejc	Bundesamt für Sicherheit in der Informationstechnik
Joachim Posegga	Universität Passau
Johann Uhrmann	Hochschule Landshut
Johanna Ullrich	Universität Wien, SBA Research
Johannes Kinder	Universität der Bundeswehr München
Jörg Helbach	HSPV NRW
Jörg Keller	FernUniversität in Hagen
Jörn Eichler	Freie Universität Berlin
Jörn Müller-Quade	Karlsruher Institut für Technologie
Juraj Somorovsky	Universität Paderborn
Jürgen Schönwälder	Jacobs University Bremen
Kai Rannenberg	Goethe-Universität Frankfurt
Kerstin Lemke-Rust	Hochschule Bonn-Rhein-Sieg
Klaus-Peter Kossakowski	HAW Hamburg

Konrad Rieck	TU Braunschweig
Konstantin Knorr	Hochschule Trier
Kristin Weber	Hochschule Würzburg-Schweinfurt
Luigi Lo Iacono	Hochschule Bonn-Rhein-Sieg
Maciej Liskiewicz	Universität zu Lübeck
Manuel Duque-Anton	Hochschule Kaiserslautern
Marian Margraf	Freie Universität Berlin
Maritta Heisel	Universität Duisburg-Essen
Marko Schuba	FH Aachen
Markus Dürmuth	Ruhr-Universität Bochum
Martin Hübner	HAW Hamburg
Martin Johns	TU Braunschweig
Martin Kappes	Frankfurt University of Applied Sciences
Martina Zitterbart	Karlsruher Institut für Technologie
Mathias Fischer	Universität Hamburg
Matthias Hollick	TU Darmstadt
Matthias Meyer	MAMEDO IT-Consulting GmbH
Matthias Schunter	Intel Labs
Michael Meier	Universität Bonn, Fraunhofer FKIE
Michael Sonntag	Johannes Kepler Universität Linz
Nils Aschenbruck	Universität Osnabrück
Nils gentschen Felde	Universität der Bundeswehr München
Nils Gruschka	University of Oslo
Nils Ole Tippenhauer	CISPA Helmholtz-Zentrum für Informationssicherheit
Norbert Pohlmann	Institut für Internet-Sicherheit
Pascal Reisert	Universität Stuttgart
Peter Schartner	Alpen-Adria-Universität Klagenfurt
Rainer Boehme	Universität Innsbruck
Ralf Reussner	Karlsruher Institut für Technologie, FZI
Rolf Drechsler	Universität Bremen
Ruediger Grimm	Fraunhofer SIT Darmstadt
Ruediger Reischuk	Universität zu Lübeck
Sachar Paulus	Hochschule Mannheim
Sebastian Pape	Goethe-Universität Frankfurt
Sebastian Schinzel	FH Münster
Sebastian Schmerl	Arctic Wolf Networks Germany
Sebastian Schrittwieser	Universität Wien
Stefan Katzenbeisser	Universität Passau
Steffen Helke	Fachhochschule Südwestfalen
Steffen Wendzel	Hochschule Worms
Stephan Trahasch	Hochschule Offenburg
Tobias Heer	Hochschule Esslingen
Ulrich Greveler	Hochschule Rhein-Waal
Ulrike Meyer	RWTH Aachen
Van Bang Le	Universität Rostock
Viorica Sofronie-Stokkermans	Universität Koblenz-Landau
Wilhelm Zugaj	FH Joanneum

Practitioners Track

Chair: Thomas Barber, SAP Research
Co-Chair: Bernhard C. Witt, it.sec GmbH

Andreas Dewald	ERNW Research GmbH
Andreas Schaad	Hochschule Offenburg
Bastian Braun	mgm security partners GmbH
Bernhard C. Witt	it.sec GmbH
Dirk Koschuetzki	Hochschule Furtwangen
Hans Pongratz	Stiftung für Hochschulzulassung, TU Dortmund
Isabel Münch	Bundesamt für Sicherheit in der Informationstechnik
Jörn Vossbein	UIMC DR. VOSSBEIN GmbH & Co KG
Jürgen Neuschwander	Hochschule Konstanz
Karin Schuler	Karin Schuler - Datenschutz und IT-Sicherheit
Klaus Kirst	PTLV
Lukas Rist	The Honeynet Project
Mathias Kohler	SAP Security Research
Michael Heinl	Fraunhofer AISEC
Patrizia Russ	Munich Re F&C
Peer Reymann	ITQS GmbH
Steffen Schulz	Intel

Doktorand·innenforum

Chair: Daniel Arp, TU Berlin
Co-Chair: Zoltán Mann, Universität Amsterdam

Andreas Berl	Technische Hochschule Deggendorf
Andreas Noack	Hochschule Stralsund
Bernhard Hämmerli	Hochschule Luzern
Bettina Buth	HAW Hamburg
Christoph Striecks	Austrian Institute of Technology
Dieter Gollmann	TU Hamburg-Harburg
Dieter Hutter	DFKI
Dominik Engel	Fachhochschule Salzburg
Dominik Merli	Hochschule Augsburg
Eberhard Zehendner	Friedrich-Schiller-Universität Jena
Erik Buchmann	Hochschule für Telekommunikation Leipzig
Felix Freiling	Friedrich-Alexander-Universität Erlangen-Nürnberg
Frank Kargl	Universität Ulm
Hannes Hartenstein	Karlsruher Institut für Technologie
Henning Schnoor	Christian-Albrechts-Universität zu Kiel
Herbert Leitold	A-SIT, Zentrum für sichere Informationstechnologie
Holger Morgenstern	Hochschule Albstadt-Sigmaringen
Hubert Keller	Karlsruher Institut für Technologie
Isabel Münch	Bundesamt für Sicherheit in der Informationstechnik
Johann Uhrmann	Hochschule Landshut

Jörg Helbach	HSPV NRW
Jörg Keller	Fernuniversität in Hagen
Jörn Eichler	Freie Universität Berlin
Konrad Rieck	TU Braunschweig
Marian Margraf	Freie Universität Berlin
Martin Johns	TU Braunschweig
Mathias Fischer	Universität Hamburg
Michael Meier	Universität Bonn
Nils Aschenbruck	Universität Osnabrück
Nils gentschen Fede	Universität der Bundeswehr München
Pascal Reisert	Universität Stuttgart
Peter Mayer	Karlsruher Institut für Technologie
Peter Schartner	Universität Klagenfurt
Rüdiger Grimm	Universität Koblenz
Simone Fischer-Hübner	Karlstad University
Stefan Katzenbeisser	Universität Passau

Organisationsteam

Publicity Chair:

Anne Henning	Karlsruher Institut für Technologie

Event Organisation:

Alexander Scheibe	Gesellschaft für Informatik (GI)

Inhaltsverzeichnis

Session 1

N. Demir, D. Theis, T. Urban, N. Pohlmann
First-Party Cookie Tracking in the Field 19

M. Ohm, L. Kempf, F. Boes, M. Meier
Towards Detection of Malicious Software Packages 35

J. Gruber, F.C. Freiling
Fighting Evasive Malware With a VMI-based Human Interaction Simulator 49

Session 2

M. Güdemann
SMT-Based Verification of Concurrent Critical Systems 67

D. Doerner, J. Mechler, J. Müller-Quade
Hardening the Security of Server-Aided MPC 83

C. Burkert, J. Balack, H. Federrath
PrivacyDates: A Framework for More Privacy-Preserving Timestamp Data Types 101

F. Jacob, S. Bayreuther, H. Hartenstein
On CRDTs in Byzantine Environments 113

Session 3

V. Garske, A. Noack
Recovering information from pixelized credentials 129

C. Saatjohann, F. Ising, M. Gierlings, D. Noss, S. Schimmler, A. Klemm, L. Grundmann, T. Frosch, S. Schinzel
Sicherheit medizinischer Systeme und Protokolle 143

R. Müller, J. Ruppert, K. Will, L. Wüsteney, T. Heer
Analyzing the Software Patch Discipline Across Different Industries and Countries 159

Short Papers

N. Mayer, S. Wendzel, J. Keller
Gender Citation-gap bei Cybersecurity-Publikationen 173

J. Breig, D. Westhoff
Debating Ethics with Cybersecurity Students 183

M. Gruber, C. Höfig, M. Große-Kampmann, T. Urban, N. Pohlmann
Business Chat ist verwirrt. Es hat sich vor Verwirrung selbst verletzt! . . . 193

Practitioners Track

F. Lochmann, S. Schmerl
Cyber-Defense "Gemessen, Bewertet und Ausgerichtet " 205

C. Strotmann, R. van Rijswijk-Deij, P. Koetter, M. de Brün, A. Kölligan
Gefahren für das DNS durch IP-Fragmentierung 209

B. Beckert, J. Budurushi, A. Grunwald, R. Krimmer, O. Kulyk, R. Küsters, A. Mayer, J. Müller-Quade, S. Neumann, M. Volkamer
Recent Developments in the Context of Online Elections and Digital Polls in Germany 213

T. Hinrichs
Automated Data Set Generation 219

Doktorand·innenforum

A. Hennig
Your website has been hijacked 225

B. Fraune
Automated Monitoring of Operational Technology Security and Compliance for Power Grids 231

H. Heseding
Reinforcement Learning-Controlled Mitigation of Volumetric DDoS Attacks 237

J. Möller
Differential Testing 243

Session 1

Towards Understanding First-Party Cookie Tracking in the Field

Nurullah Demir[1], Daniel Theis[2], Tobias Urban[3], Norbert Pohlmann[4]

Abstract: Third-party tracking is a common and broadly used technique on the Web. Different defense mechanisms have emerged to counter these practices (e. g. browser vendors that ban all third-party cookies). However, these countermeasures only target third-party trackers and ignore the first party because the narrative is that such monitoring is mostly used to improve the utilized service (e. g. analytical services). In this paper, we present a large-scale measurement study that analyzes tracking performed by the first party but utilized by a third party to circumvent standard tracking preventing techniques. We visit the top 15,000 websites to analyze first-party cookies used to track users and a technique called "DNS CNAME cloaking", which can be used by a third party to place first-party cookies. Using this data, we show that 76% of sites effectively utilize such tracking techniques. In a long-running analysis, we show that the usage of such cookies increased by more than 50% over 2021.

Keywords: first-party tracking, cookies, privacy, CNAME cloaking, tracking methods

1 Introduction

The business model of many modern (Web) applications relies on the revenue generated by "renting" space on their services to advertisement companies. These ad-tech companies try to place advertisements on the sites that meet the users' interests motivating that they will interact with the ad, and ultimately buy the advertised product or service. To place such targeted ads, ad-tech companies track users across the Web, by assigning a unique identifier to each of them, and try to understand their interests by building so-called *behavioral profiles* [MC10]. The unique user identifiers are often stored in the third-party context (e. g. in an HTTP cookie). Some consider this large-scale tracking as privacy-invasive because it often happens without users' explicit consent or knowledge [TH11], nor is the tracking made transparent to the user. The desire for more privacy and the need for more user data led to an arms race between anti-tracking tools, and novel techniques to track users. One recent (technical) step in this race was the announcement of major browser vendors to ban

[1] Institute for Internet Security – if(is), Westphalian University of Applied Sciences and KASTEL Security Research Labs, Karlsruhe Institute of Technology demir@internet-sicherheit.de
[2] Institute for Internet Security – if(is), Westphalian University of Applied Sciences, Neidenburger Straße 43, 45897, Gelsenkirchen, Germany, theis@internet-sicherheit.de
[3] Institute for Internet Security – if(is), Westphalian University of Applied Sciences and secunet Security Networks AG, urban@internet-sicherheit.de
[4] Institute for Internet Security – if(is), Westphalian University of Applied Sciences, Neidenburger Straße 43, 45897, Gelsenkirchen, Germany, pohlmann@internet-sicherheit.de

doi:10.18420/sicherheit2022_01

third-party cookies within the next years [Go20a; Mo20a]. While there is no immediate problem with third-party cookies, previous work showed that they are overwhelmingly used for advertisement purposes [Ur20a]. Hence, trackers need to find different ways to persist their identifiers on the users' devices. One known way to do so is the computation of *browser fingerprints*, which are distinct identifiers that are computed based on properties of the user's device or browser [EN16; La20]. However, these fingerprints change over time and, therefore, one cannot simply rely on them for tracking purposes [GLB18].

One way to cache such identifiers is to store them in a first-party context, and send them to a third party if needed. More specifically, a tracking script embedded in the first-party context of a site could store the identifier (cookie) in the first-party context and send it to the tracker in a dedicated request. Hence, deleting, or banning third-party cookies does not affect them. In this paper, we perform a large-scale measurement study on the top 15,000 sites to analyze the presence of such first-party tracking techniques in the field and use the HTTP Archive [HT21] to analyze the development of such tracking techniques. In our measurement, we find that roughly 90% of all sites include a first-party object that tracks users, and leaks the identifier to a third party. Furthermore, our results show that 10,354 (69%) of the sites in our analysis corpus hide the presence of a tracker by redirection of request on DNS level ("DNS CNAME cloaking"). Using the HTTPArchive [HT21], we show that first-party tracking, in cooperation with a third party, has been a common phenomenon in the past with a growing popularity. Our results show that the leakage of such cookies to a third party increased by nearly 50%. Previous work that analyzed the tracking ecosystem almost exclusively focused on third-party trackers from various perspectives (e. g. [Ac14; En15; EN16; Fo20; GLB18]). Other works focused on analyzed tracking attempts performed in the first-party context using CNAME cloaking [Di21; DMF20; Re21] (see Section A). CNAME cloaking is one way to pace for a third party to place an identifier in the first-party context of a site (see Section 3.2). In this work, we focus on first-party tracking via cookies and do not aim to analyze tracking enabled by CNAME cloaking, but we see it as one way to place such cookies. To be more precises, we analyze the combined usage of CNAME cloaking and first-party cookie tracking in the field – and not each technology in isolation. One important insight from our work is that tracking is no longer a phenomenon only present in the third-party context but has arrived in the first-party context at large scale. Consequently, many of the current anti-tracking tools might have to be revised to match this new tracking scope, that future studies might have to reconsider their measurement setups, and that banning third-party cookies might have less impact than it is currently expected [Lt20]. We make the following key contributions:

1. We perform a large-scale web measurement of the top 15,000 sites on the Web, and analyze if they utilize first-party identifiers. Our results suggest that first-party tracking – at the behest of a third party – arrived at scale on the Web.
2. In our experiment, over 85% of the analyzed sites store potential tracking identifiers in a first-party cookie, and send them to a third party. Utilizing the *HTTPArchive*, we show that this technique has already been used in 2019 and has ever grown since.

3. Finally, we analyze the companies participating in the first-party tracking ecosystem. We find that trackers who dominate the third-party tracking market also dominate the first-party tracking market.

2 Method

As basis for our analysis, we chose to use the *Tranco* top 15,000 list generated on 07/16/2020 (ID: PX8J)[5], which is an aggregation of other popular top lists [Le19], in our analysis. We visited each site on the list and collected 30 distinct first-party hyperlinks to pages on the same site, which we used in our measurement crawls. If possible, we repeated the process to recursively collect up to 30 subsites. We chose to visit 30 pages instead of only analyzing the landing page of a site because recent work has shown that "subsites" show increased usage of privacy-invasive technologies and that visiting 30 sites is a good approximation to attest the privacy impact of a website [Aq20; Ur20a]. Due to this need for a vertical measurement setup, the number of distinct websites we analyze is limited compared to other measurement studies (e. g. [En15]).

To measure the usage of first-party tracking and the leakage of such cookies, we utilize the popular *OpenWPM* framework [EN16], which uses the Firefox browser to visit websites. We configured the framework to log all (1) HTTP requests/responses and (2) data stored in the local storage or cookie jar. Furthermore, we instrumented Firefox's DNS API [Mo20b] to log the DNS resolution of all hostnames, which is necessary to identify CNAME cloaking (Section 2.1). A detailed description of our framework can be found in Appendix B and a detailed description of CNAME cloaking is given in Appendix A. We use four different browser configurations ("profiles") for our measurement: (1) No particular configuration, (2) disabled 3rd party cookies, (3) active anti-tracking tool (i. e. *uBlock Origin* [Hi20]; henceforth, "*uBlock*"), and (4) disabled 3rd party cookies, and an active anti-tracking tool (i. e. (2) and (3) combined). We chose to use the option to block third-party cookies to test if trackers use other techniques (e. g. first-party tracking techniques) if they cannot store their identifiers as they usually do. We used *uBlock* because – to the best of our knowledge – it is the only tool at the time of writing this paper that claims to identify and block CNAME cloaking. Hence, the tool provides protection against trackers that utilize this new technique, and we can assess, concerning the other profiles, how effective it is. While the named defence mechanisms do not actively aim to protect against tracking performed by a third party they might block requests to a third party that contains the first-party identifier, which would effectively limit the tracking impact. For each profile, we performed a separate measurement crawl. Hence, we visited each page on our website corpus four times, once with each profile.

[5] Available at https://tranco-list.eu/list/PX8J.

2.1 Measuring CNAME Cloaking

CNAME cloaking is increasingly used to track users [Di21; DMF20]. We analyze this practice's presence by inspecting the DNS resolutions on the client by instrumenting the DNS resolution of the Firefox browser. More specifically, we log the URL handled by the browser and the responding DNS resolution observed by the browser, which we instrumented. For example, if we observe a request to *tracking.foo.com/search=foo* whose domain is then cloaked to *cloaked-party.com*, we use the URL *cloaked-party.com/search=foo* in our analysis but treat it a first-party request. In contrast to previous work in the field of CNAME cloaking, this allows us to analyze the real DNS resolution of each request. Previous work often performed the DNS resolutions isolated from the web measurement on different machines and at different times [Di21; DMF20]. This comes with the non-negligible assumption that DNS resolutions will not change over time and/or based on the location of users, which could change the outcome of a study. However, we want to highlight that our contribution is not the measurement of tracking that is enabled by third parties but that we are interested in tracking identifiers that are stored in a third-party context. One way to store such cookies is via CNAME cloaking. In order to compare our results to previous work, we match the resolved URL against two standard tracking filter lists, as of 04/16/2021 (i. e. the *EasyList* and *EasyPrivacy* lists [Ea20]). Such lists always come with the downside that they might not contain all tracking URLs [Fo20; Me17] and, therefore, our results can be seen as a lower bound. However, since we used lists generated after the crawl, we think they should contain most trackers. As we describe in Appendix A, CNAME cloaking is not a problem per se, but it could be used to hide the real recipient of a request. For example, a service provider who wants to host a domain on a content delivery network but wants to keep his domain (brand) name often uses CNAME cloaking to do so [DMF20].

2.2 First-Party Cookie Tracking

Since browser vendors limit the use of third-party cookies [Go20a; Mo20a], trackers need to find different means to store them. One way to store them is to use first-party cookies because these are currently not restricted. One challenge in this approach is that we have to distinguish between session cookies, which are usually not stored in third-party cookies, and user identifiers. If the same identifier reoccurs on different visits to pages of the same site, we assume that it is a user identifier and not a session ID. This assumption is valid because we perform a stateless crawl, and session IDs will reset between two visits since the local storage and cookie jar are reset. We assess a cookie to be a first-party cookie if it was set by a request or script that was loaded in a first-party context. Hence, if we visit *foo.com* and the site sets a cookie, we count it as a first-party cookie. However, let us assume *bar.com* embeds an object from *foo.com* that sets a cookie; we do not count this cookie in our first-party analyses – because it was set in a third-party environment. In theory, cookies are simple `name=value` pairs, yet Gonzalez et al. show that a single cookie often contains multiple values in proprietary formats (e. g. `key=[v1=foo;v2=bar]`) [Go17].

To identify a tracking cookie, we use the following commonly accepted definition [Ac14; En15; KTK20; Ur20a; Ur20b] : (1) it is not a session cookie and has a lifetime of more than 90 days; (2) has a length of at least eight bytes (to hold enough entropy); (3) is unique in each measurement run, and the length of each value only differs by up to 25%; and (4) The values of the cookie are similar according to the Ratcliff/Obershelp [RM88] string comparison algorithm (≤60%). We resort to these definitions to allow a basic comparison of our and previous work. It is essential to mention that first-party cookies might hold an identifier to implement analytical or similar services, which are essentially a form of local user tracking. If services send the (local) identifier to a third party, which offers the service, this third party could track the user. For example, if we find the first-party cookie `OptanonConsent=ABC-123`, which is probably used by *OneTrust*, and observe a third-party request that holds this ID (e. g. *tracking.foo.com?**consentId=ABC-123***), we assume that *tracking.foo.com* could potentially track the user with ID *ABC-123*. Hence, to find potential tracking that utilized first-party cookies, one has to find instances where they leak it to any third party. In our analysis, we identify such cookie leakage by inspecting the HTTP GET and POST parameters of all third-party requests. We compare them with all cookie values, that could be used for tracking (see the definition above). If a first-party cookie is leaked in such a way, we assume that it can be potentially used to track users. One challenge in this approach is that we have to distinguish between session cookies, which are usually not stored in third-party cookies, and user identifiers. If the same identifier reoccurs on different visits to pages of the same site, we assume that it is a user identifier and not a session ID. This assumption is valid because we perform a stateless crawl, and session IDs will reset between two visits since the local storage and cookie jar are reset.

2.2.1 Cookie Tracking Over Time

The introduced methods potentially used to perform first-party tracking do not rely on any new or recently introduced techniques. Hence, it is reasonable to assume that if such practices are used that they were used in the past. To understand whether such practices were used in the past we utilize the *HTTPArchive* [HT21]. HTTPArchive is an initiative to measure and analyze how the modern Web is built. To achieve this, HTTPArchive crawls the landing pages of the most popular origins, based on the *Chrome User Experience Report* (CrUX) [Go20b], on a monthly basis since 01/2019. Thus, we can analyze the first-party tracking capabilities of all websites in the archive over time. Due to the size of the archive, which includes millions of websites in each measurement, we decided to perform the analysis on a quarterly basis. Hence, the first measurement point in our dataset is 01/19, the second one 04/19, and the last one 07/21. In the analysis, we extract all cookie values from the raw data provided in the archive. Using the raw data, we can access all necessary data that we need in our analysis to identify tracking cookies (e. g. lifetime). Similar to our approach in our active measurement, we first identify first-party cookies that could be used for tracking purposes and then test if such cookies are leaked to a third party in an HTTP GET or POST request. An important limitation of this approach is that it excludes all cookies set via JavaScript (i. e. we only see cookies set via the HTTP protocol).

#	Configuration	Date	# Sites	# Pages	1st-party c.	3rd-party c.	CNAME Cl.
1	Plain browser	07/20	11,471	272,659	408,509	155,193	14,493,533
2	No 3rd-party cookies	07/26	10,368	246,493	355,208	—*	11,017,643
3	*uBlock Origin* active	08/01	10,203	230,327	139,115	89,876	5,829,786
4	#2 and #3 combined	08/06	10,153	225,315	142,608	—*	5,831,884

Tab. 1: Overview of all measurements. ∗ Firefox deletes these cookies after creation.

3 Results

First, we want to provide an overview of the measured dataset. We conducted all four measurement runs in three consecutive weeks (each measurement took approx. five days). We conducted the first measurement (#1—no particular configuration) on 07/20/2020, and the last measurement (#4) ended on 08/12/2020. Across all measurement, we visited 272,659 distinct pages on 11,471 sites. In total, we visited 974,794 URLs across our four measurements. In total, 1,290,509 cookies were set during these visits (1,045,440 first-party and 245,069 third-party cookies). Across all measurements, we observed 37,172,846 instances of CNAME cloaking of which 783,917 (2%) ultimately resulted in tracking attempts, based on the classification of the resolved URL. A summary of the measurement runs is given in Table 1. The drop in analyzed sites and, consequently, pages can be attested to sites no longer being available (e. g. HTTP 404 errors). According to the numbers, it seems that blocking third-party cookies has no effect on the usage of first-party cookies, but using an anti-tracking tool, *uBlock* in our case, leads to a decrease in the usage of both types of cookies. Furthermore, the tool seemingly helps to block CNAME cloaking attempts.

3.1 First-Party Cookie Leakage

In the following, we discuss the leakage of first-party tracking cookies. Furthermore, we provide a general analysis of CNAME-based first-party tracking cookies in Appendix C. As we have shown 20% of the first party cookies could be used to track different users across a site (see Appendix C). In this section, we evaluate to what extent these cookies are leaked to third parties. A straightforward way to send such cookies is to include them in HTTP GET or POST parameters. Overall, we found that 68% (143,216) of the cookies that we identified as first-party trackers are sent to a single third party. A common practice in the tracking ecosystem is the so-called *cookie-syncing*, which means that two or more third parties exchange a user identifier. In our experiment, we find that first-party tracking cookies are shared, on average, with 1.3 third parties, across all profiles. 1,182 (0.8%) are shared with more than one third party. Hence, real cross-domain tracking and “cookie-syncing” has not arrived in the first-party context at scale. We identified 2,253 distinct domains that received such cookies, in 5,931,727 requests. Our results show that first-party tracking cookies are often leaked to a third party and they could be used to track users. A more detailed analysis of the companies receiving cookies this way can be found in Section 3.3.

Defense Mechanisms In profile #3 and #4, we enabled the *uBlock* extension to test its effectiveness in protecting users. The tool does not primarily aim to protect users' from cookie value leakage. However, since it blocks some URLs that are knowingly used for tracking purposes it might also prevent first-party cookie leakage. In our analysis, we observed 43% fewer requests that contained a cookie value. This decrease consequently leads to a decrease of 98,819 (31%) tracking cookies leaked and 456 (20%) fewer third parties getting access to the cookie. Similar to our findings on first-party tracking cookies, the leakage of them is also positively impacted by the used extension. This is probably also explained by the fact that the tool blocks requests to different third parties

3.1.1 First-Party Cookie Tracking Over Time

Our active measurement shows that first-party cookie tracking and leakage are severe and joint problems on the Web. In the following, we want to analyze the development of first-party cookie-based tracking usage. To achieve this, we use data from 2019, 2020, and 2021 that we extracted from the *HTTPArchive* (see Section 2.2.1). Across all 11 measurement points (four in 2019 and 2020 each and three in 2021), we analyzed 15,805,385 websites, which issued 5,592,914,767 requests. Our dataset contains 119,676,680 first-party cookies. Of those cookies 9,489,765 (8%) could potentially be used to track users, according to our definition. In total, 42,068 (0.3%) of the websites actively sent a cookie to a third party, and 321,644 (0.3% of the first-party cookies, and 3.4% of the tracking cookies) were leaked in such away. In the following, we only analyse sites that appeared in all measurement points (1,777,206 sites). In total, 13,116 (0.7%) of the websites actively sent a cookie to a third party, and 63,235 (0.2%) of the first-party cookies, and 2.4% of tracking cookies were leaked in such away. We reason that this high discrepancy to our active measurement is because of the large amount of less popular sites in the *HTTPArchive* and our finding that popular sites tend to use first-party tracking more often than less popular ones.

Figure 1 provides an overview of the temporal development of first-party cookie based tracking on the Web. As one can see, the number of leaking parties and leaked cookies are directly related to each other. This relation shows that a number of websites seem to use a more or less fixed set of services (e. g. a consent management system) that use the first-party context to track users. Overall, the data point at 07/21 contains 118% more first parties that use the analyzed tracking cookies than the first measurement point (07/19). The number of third-party trackers that receive such cookies remains relatively stable, with an increase from 456 (01/19) to 621 (07/21), which accounts for 36%. This development indicates that there is a steadily growing set of players in the ecosystem that utilize this technique (see Section 3.3), but more and more websites adopt these services of these players.

3.2 First-Party Cookie Tracking via CNAME Cloaking

Previously, we discussed straightforward attempts of first-party cookie tracking, which presumably happens in coordination with a third party. Another option for a third party to

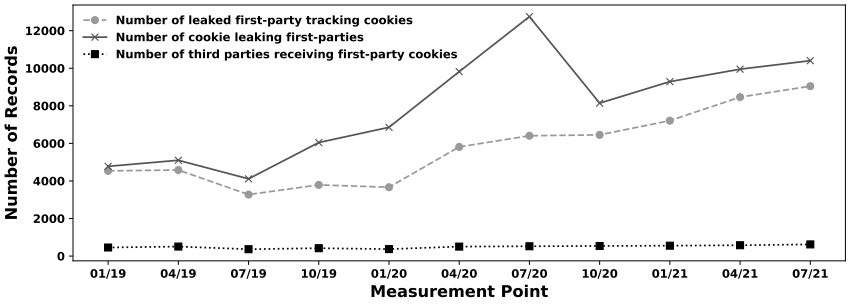


Fig. 1: Number of distinct sites and third parties that engage in first-party tracking and number of leaked first-party tracking cookies, over time.

set and access cookies in a first-party context is that the first party redirects requests on DNS level to another domain ("CNAME cloaking" – see Section 2.1). Utilizing CNAME cloaking, HTML objects (e. g. JavaScript) operate in the first-party context (e. g. *t.foo.com*) on browser level but are actually loaded from another domain (e. g. *tracking.org*) on DNS level. Across all four measurements in our dataset, we found over 37M requests for which the DNS name was cloaked (45% of all requests). Of those requests 2.1% (783,917) are sent to an URL that is marked to be used for tracking purposes, by the lists mentioned in the Section 2.1 *after* the cloaking of the CNAME.

Cloaked First-Party Cookies: In this work, we focus on the usage of first-party cookies to track users. Aside from cloaking the tracking URL, one way to abuse CNAME cloaking for tracking is to place tracking cookies in the first-party context. Therefore, we are interested in analysing if cloaked requests use the first-party context to store (tracking) cookies. Overall, we found 69,961 (6.7%) of all first-party cookies that are set by such requests. According to our definition of tracking cookies (see Section 2.2), 28% (19,676) of those cookies can be used to track users. Overall, the identified tracking cookies are utilized by 8,508 (74%) sites in our dataset. Hence, it seems that the third parties actively try to circumvent anti-tracking methods. In our dataset, more popular sites (lower rank) and sites of a specific category (e. g. "News") utilized CNAME based cookie tracking.

Defense Mechanisms *uBlock* provides tracking protection for CNAME cloaking based tracking as it actively inspects the DNS resolution of requests and makes decisions based on the result of it. Other tools operate on the objects (e. g. URLs) accessible on browser level and, therefore, cannot detect DNS level tracking approaches. Therefore, we are interested in the effect of the 'lower level' DNS blocking and the impact on users' privacy. In our measurement, we used the *uBlock* browser extension in profile #3 and #4. In both profiles, we see a decrease of CNAME cloaking of around 46% what results in a decrease of 15% in identified trackers, which use this technology. However, we still found trackers that were not detected by *uBlock*, which is in line with previous work that showed that list based blockers

miss some trackers [Me17]. The blocking of the identified requests resulted in a decrease of 101,387 (21%) first-party tracking cookies. Our results show that modern anti-tracking tools need to adapt to modern first-party tracking techniques – especially CNAME tracking.

3.3 First-Party Cookie Tracking Ecosystem

In the previous section, we have analyzed the extent of first-party tracking – emphasizing first-party tracking cookies – which are set through various means. In the following, we shed some light on the ecosystem behind this first-party tracking and the companies that are active in it. Concerning the websites that utilize first-party cookies for tracking, we found that 10,719 (91%) sites use them. To get a better understanding of who uses such techniques, we use, on the one hand, the ranking (according to the *Tranco* list [Le19]) and, on the other hand, the category of the website's content (see Section 2) to test for usage differences. We used the X^2 test ($\alpha = 0.5$) to evaluate statistical significance in these categories. Both categories (rank and category) show a strong correlation (p-value < 0.0001) in the usage of first-party tracking cookies. More specifically, we found that popular websites (rank $\leq 3,000$) and websites of categories like "Business", "News", or "Education" tend to use more of such tracking techniques. This finding hints that websites are more aware of upcoming changes respectively current protection mechanisms and try to circumvent them.

Companies Active in the Ecosystem We use the *Cookiepedia* (see Section B) to map identified cookies to companies using them. Using this approach, we could classify 95,236 (36%) of the observed cookies. These can be attributed to 283 distinct companies. As previously briefly noted, the aggregated view of active companies, that use first-party tracking cookies, shows that *Google* (54%) and *Facebook* (14%) are the most common ones, followed by *Adobe* (8%) and *OneTrust* (3%). However, we see a different picture when we look at the companies to which the cookies are leaked. *Dynamic Yield* (17%) and *Adobe* (15%) are the top companies that receive them, these results are in line with previous work [DMF20]. *Google* only receives 7% of all leaked cookies.

4 Related Work

Online tracking has received a lot of attention from the academic community and the industry alike and, therefore, a huge body of work exists that analyzes tracking methods along different dimensions. Quantifying Web tracking though measurement is a popular way to analyze different tracking techniques like device fingerprinting (e. g. [EN16; Va18]), tracking cookies (e. g. [Ac14; En15]), connections between tracking companies (e. g. [PKM18]), other ways to track users (e. g. [Fo20]), and the efficiency of privacy-preserving tools to prevent racking (e. g. [Me17; RKW12]). The impact of newly introduced legislation on online tracking (like the CCPA or GDPR) was also analyzed in detail by different works (e. g. [Sa19; SK19; Ur20b]). However, all of these works focus on tracking performed by third parties that are

embedded into a website. Our work analyzes a new trend in the tracking market that moves the tracking code to the first party.

Recently, different works focused on CNAME cloaking. In mid 2020, Dao et al. were the first ones to analyze first-party tracking by performing a large scale Web measurement to identify the usage of CNAME cloaking [DMF20]. In their paper, they show that this technique has already been utilized for several years and they highlight the limitation of current privacy preserving technologies to counter such tracking. Dao et al. also provide a first mitigation strategy for CNAME based cloaking, using machine learning [DF20]. Complementary to our work, Ren et al. take a look at the effect of CNAME cloaking on browser cookie policies and how they propagate in the ecosystem [Re21]. They find that CNAME cloaking is often used to circumvent browser policies and that sensitive data is leaked by this bypass. Dimova et al. perform a rigorous analysis of CNAME cloaking-based tracking and discuss privacy and security issues that arise by the usage of this technique [Di21]. Finally, the dangers of CNAME cloaking is also discussed in several non-academic articles(e. g. [Co19; La21]) and on browser vendor pages (e. g. [Wi20]), which hints the piratical relevance of the problem. Most recently, Quan et al. also took a look at cookie-based first-party tracking [Ch21]. They use an elaborated JavaScript taint analysis framework to understand the process how first-party cookies are set and by whom. They show that nearly all of the popular websites (97%) utilize first-party cookies for user tracking. Our approach differs from the named work because, on the one hand, we do not limit our analysis to one technique only but analyse first-party tracking along different axes and take a look at the implications of first-party cookie tracking, sometimes enabled by CNAME cloaking (e. g. by analysing defence mechanisms, the ecosystem, or temporal development). Hence, we do not analyze novel tracking techniques in isolation but aim to understand how they are used together.

5 Discussion and Conclusion

This paper analyzed the scale of (potential) first-party tracking techniques on the top 15,000 websites. In contrast to previous work, we have shown that tracking is not exclusively implemented through third-party means but that first-party content is used for such purposes as well. Hence, limiting or blocking third-party content on websites will not prevent that users will be tracked by third parties. Our work shows that first-party cookies and requests that are redirected on DNS level are used for these purposes. One takeaway from these findings is that privacy-preserving technologies need to extend their attempts to the first-party context while still covering third-party contexts. First-party cookies that hold a personal identifier constitute a privacy risk since users might assume that those are exclusively used by the first party and are needed to run the service. However, we have shown that they are often leaked to a third party, which could abuse them for tracking purposes. While the extend of tracking and used methods (e. g. first-party cookie syncing) are not yet present at scale we argue that they will gain in importance once first-party tracking is more commonly used.

Acknowledgements This work was partially supported by the German Federal Ministry for Economic Affairs and Energy (grant 01MK20008E "Service-Meister" and grant 01MN21002H "IDunion"), the Ministry of Culture and Science of North Rhine-Westphalia (MKW grant 005-1703-0021 "MEwM"). Furthermore, we would like to thank Norman Schmidt and Katharina Meyer for their efforts in developing the analysis pipeline.

References

[Ac14] Acar, G.; Eubank, C.; Englehardt, S.; Juarez, M.; Narayanan, A.; Diaz, C.: The Web Never Forgets: Persistent Tracking Mechanisms in the Wild. In: ACM Conference on Computer and Communications Security. CCS '14, 2014.

[Aq20] Aqeel, W.; Chandrasekaran, B.; Feldmann, A.; Maggs, B. M.: On Landing and Internal Web Pages: The Strange Case of Jekyll and Hyde in Web Performance Measurement. In: ACM SIGCOMM Internet Measurement Conference. IMC'20, 2020.

[Ch21] Chen, Q.; Ilia, P.; Polychronakis, M.; Kapravelos, A.: Cookie Swap Party: Abusing First-Party Cookies for Web Tracking. In: International Conference on World Wide Web. WWW, 2021.

[Cl18] Cliqz GmbH: WhoTracks.me Data—Tracker database, `https://github.com/cliqz-oss/whotracks.me/tree/master/whotracksme/data`, 2018.

[Co19] Cointepas, R.: CNAME Cloaking, the Dangerous Disguise of Third-party Trackers, https://medium.com/nextdns/cname-cloaking-the-dangerous-disguise-of-third-party-trackers-195205dc522a, 2019.

[DF20] Dao, H.; Fukuda, K.: A Machine Learning Approach for Detecting CNAME Cloaking-based Tracking on the Web, 2020.

[Di21] Dimova, Y.; Acar, G.; Olejnik, L.; Joosen, W.; Van Goethem, T.: The CNAME of the Game: Large-scale Analysis of DNS-based Tracking Evasion. Privacy Enhancing Technologies Symposium, PoPETS '21 3/, 2021.

[DMF20] Dao, H.; Mazel, J.; Fukuda, K.: Characterizing CNAME Cloaking-Based Tracking on the Web. In: Network Traffic Measurement and Analysis Conference. TMA, 2020.

[Ea20] EasyList: EasyPrivacy, `https://easylist.to/`, 2020.

[En15] Englehardt, S.; Reisman, D.; Eubank, C.; Zimmerman, P.; Mayer, J.; Narayanan, A.; Felten, E. W.: Cookies That Give You Away: The Surveillance Implications of Web Tracking. In: International Conference on World Wide Web. WWW, 2015.

[EN16] Englehardt, S.; Narayanan, A.: Online Tracking: A 1-Million-Site Measurement and Analysis. In: ACM Conference on Computer and Communications Security. CCS, 2016.

[Fo20] Fouad, I.; Bielova, N.; Legout, A.; Sarafijanovic-Djukic, N.: Missed by Filter Lists: Detecting Unknown Third-Party Trackers with Invisible Pixels. Privacy Enhancing Technologies Symposium, PoPETS '20 2/, 2020.

[GLB18] Gómez-Boix, A.; Laperdrix, P.; Baudry, B.: Hiding in the Crowd: An Analysis of the Effectiveness of Browser Fingerprinting at Large Scale. In: International Conference on World Wide Web. WWW '18, 2018.

[Go17] Gonzalez, R.; Jiang, L.; Ahmed, M.; Marciel, M.; Cuevas, R.; Metwalley, H.; Niccolini, S.: The cookie recipe: Untangling the Use of Cookies in the Wild. In: Network Traffic Measurement and Analysis Conference. TMA, 2017.

[Go20a] Google Inc.: Building a more private web: A path towards making third party cookies obsolete, https://blog.chromium.org/2020/01/building-more-private-web-path-towards.html, 2020.

[Go20b] Google Inc.: Chrome User Experience Report | Tools for Web Developers, https://developers.google.com/web/tools/chrome-user-experience-report?hl=de, 2020.

[Hi20] Hill, Raymond: uBlock Origin, https://github.com/gorhill/uBlock/, 2020.

[HT21] HTTP Archive: The HTTP Archive Tracks How the Web is Built, https://httparchive.org, 2021.

[KTK20] Koop, M.; Tews, E.; Katzenbeisser, S.: In-Depth Evaluation of Redirect Tracking and Link Usage. Privacy Enhancing Technologies Symposium, PoPETS '20 4/, 2020.

[La20] Laperdrix, P.; Bielova, N.; Baudry, B.; Avoine, G.: Browser Fingerprinting: A Survey. ACM Transactions on the Web 14/2, 2020.

[La21] Lakshmanan, R.: Online Trackers Increasingly Switching to Invasive CNAME Cloaking Technique, https://thehackernews.com/2021/02/online-trackers-increasingly-switching.html, 2021.

[Le19] Le Pochat, V.; Van Goethem, T.; Tajalizadehkhoob, S.; Korczyński, M.; Joosen, W.: Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In: Symposium on Network and Distributed System Security. NDSS, 2019.

[Lt20] Ltd., R. M.: When the third-party cookie crumbles, https://www.raconteur.net/marketing/third-party-cookies/, 2020.

[MC10] McDonald, A. M.; Cranor, L. F.: Americans' Attitudes About Internet Behavioral Advertising Practices. In: ACM Workshop on Privacy in the Electronic Society. WPES, 2010.

[Mc20] McAfee LLC: Customer URL Ticketing System, https://trustedsource.org/, 2020.

[Me17] Merzdovnik, G.; Huber, M.; Buhov, M.; Nikiforakis, N.; Neuner, S.; Schmiedecker, M.; Weippl, E.: Block Me If You Can: A Large-Scale Study of Tracker-Blocking Tools. In: IEEE European Symposium on Security and Privacy. EuroS&P, 2017.

[Mo20a] Mozilla: Today's Firefox Blocks Third-Party Tracking Cookies and Cryptomining by Default, https://blog.mozilla.org/blog/2019/09/03/todays-firefox-blocks-third-party-tracking-cookies-and-cryptomining-by-default/, 2020.

[Mo20b] Mozilla Inc.: DNS JavaScript API, https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/dns, 2020.

[On19] OneTrust LLC: Cookiepedia, https://cookiepedia.co.uk/, 2019.

[PKM18] Papadopoulos, P.; Kourtellis, N.; Markatos, E. P.: The Cost of Digital Advertisement: Comparing User and Advertiser Views. In: International Conference on World Wide Web. WWW, 2018.

[Re21] Ren, T.; Wittman, A.; De Carli, L.; Davidson, D.: An Analysis of First-Party Cookie Exfiltration due to CNAME Redirections. In: Workshop on Measurements, Attacks, and Defenses for the Web. MADWeb, 2021.

[RKW12] Roesner, F.; Kohno, T.; Wetherall, D.: Detecting and Defending Against Third-party Tracking on the Web. In: Conference on Networked Systems Design and Implementation. NSDI, 2012.

[RM88] Ratcliff, J. W.; Metzener, D. E.: Pattern Matching: The Gestalt Approach. Dr Dobbs Journal 13/7, 1988.

[Sa19] Sanchez-Rola, I.; Dell'Amico, M.; Kotzias, P.; Balzarotti, D.; Bilge, L.; Vervier, P.-A.; Santos, I.: Can I Opt Out Yet?: GDPR and the Global Illusion of Cookie Control. In: ACM Symposium on Information, Computer and Communications Security. AsiaCCS, 2019.

[SK19] Sørensen, J. K.; Kosta, S.: Before and After GDPR: The Changes in Third Party Presence at Public and Private European Websites. In: International Conference on World Wide Web. WWW, pp. 1590–1600, 2019.

[TH11] TRUSTe; Harris Interactive: Consumer Research Results—Privacy and Online Behavioral Advertising, https://www.eff.org/files/truste-2011-consumer-behavioral-advertising-survey-results.pdf, 2011.

[Ur20a] Urban, T.; Degeling, M.; Holz, T.; Pohlmann, N.: Beyond the Front Page: Measuring Third Party Dynamics in the Field. In: International Conference on World Wide Web. WWW, 2020.

[Ur20b] Urban, T.; Tatang, D.; Degeling, M.; Holz, T.; Pohlmann, N.: Measuring the Impact of the GDPR on Data Sharing. In: ACM Symposium on Information, Computer and Communications Security. AsiaCCS, 2020.

[Ut19] Utz, C.; Degeling, M.; Fahl, S.; Schaub, F.; Holz, T.: (Un)informed Consent: Studying GDPR Consent Notices in the Field. In: ACM Conference on Computer and Communications Security. CCS, 2019.

[Va18] Vastel, A.; Laperdrix, P.; Rudametkin, W.; Rouvoy, R.: FP-STALKER: Tracking Browser Fingerprint Evolutions. In: IEEE Symposium on Security and Privacy. S&P, 2018.

[Wi20] Wilander, J.: CNAME Cloaking and Bounce Tracking Defense, https://webkit.org/blog/11338/cname-cloaking-and-bounce-tracking-defense/, 2020.

A Background on First-Party Tracking

In contrast to traditional third-party tracking, in which the first party (website) does not necessarily know which trackers might be embedded into the website [Ur20a], in our tracking model, the first party needs to support the tracker actively. The high-level idea is that the first party stores the tracking information in its first-party context and then sends it to a third party. Figure 2 displays the tracking (attack) model that we assume in this work. Our model consist of three parties: (1) the tracker (`tracking.org`), (2) the visited website (`foo.com`), and (3) the users' browser. More specifically, the website places a (static) interactive object (e. g. a JavaScript code) in its first-party context that establishes a connection to a known third-party interface (e. g. by issuing an HTTP GET request). Once the user visits a website, the first-party object checks if an identifier for the user exists in the local first-party context. If it does not exists, a new identifier is set (❶ in Figure 2). This identifier can either be computed based on the device's properties (i. e. a device fingerprint) or be generated by the third-party (❷). The browser will store the identifier in the first-party context of the site since it was set by a first-party object (❸). On consecutive visits the first-party tracking object can read the previously set tracking identifier (❹) and send it to the third party (`tracking.org` – ❺). Utilizing this mechanism, the tracker can bypass defense mechanisms that aim to limit or eliminate third-party cookies.

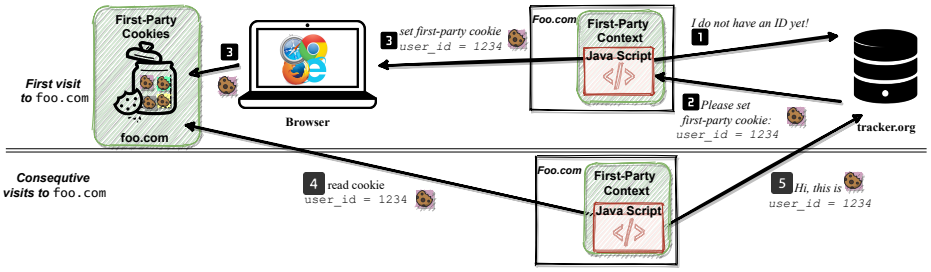


Fig. 2: Overview of first-party tracking methods.

In this work, we call this practice potential "*first-party cookie tracking*" because it works like traditional tracking only that the identifier is stored in the first-party context. This method comes with the drawback that the identifier is not accessible cross-site per-se since it has to be set for each site and might be accessed by other third parties that get access to the storage. Naturally, an identifier that is stored by the first-party cannot be used by a third party to track the user. However, once the identifier is leaked to a third party, it can potentially be used by that third party to track the user across a site. This could be the case for consent management services that assign a (first-party) identifier to a user, and, consequently, track them across the site. In this case, tracking is done with an eligible purpose (consent management), but the third party *could* also abuse it.

Canonical Name Cloaking: One popular way to link a third-party resource with a first-party element is to use so called *CNAME cloaking*. Within the DNS protocol, the *canonical name record* (CNAME) is used to map a domain name to another. This mapping is commonly used to host multiple services (e. g. *news.foo.com*, *ftp.foo.com*, and *mail.foo.com*) on the same IP. To do so, one creates an alias (CNAME) for each service that all refer to the same `DNS A` record of *foo.com*. However, the CNAME can also point to another server. For example, the CNAME of *news.foo.com* could point to *bar.com*. On the Web, this means that the browser will load the content from *bar.com*, and not *foo.com*. CNAME cloaking is commonly used on the Web. However, one can also exploit this to avoid that users block specific content (e. g. ads or tracking) as this is commonly done based on the loaded URL [Me17]. For example, if a user wants to avoid content loaded by *tracker.com* and blocks all requests to the domain on the application level (e. g. by using an ad-blocker), a web service could circumvent that by resolving all requests to *t.foo.com* to the tracking domain – on DNS level.

B Measurement Framework

In this section, we provide further details on our measurement Framework.

Experimental Setup For each visited page, we configured the framework to use the same user agent (`Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:77.0) Gecko/20100101`

`Firefox/77.0`) and desktop resolution (`1366x768`) to ensure that these attributes will not impact the computation of browsers fingerprints. According to Urban et al. [Ur20a], websites tend to use more cookies if the cookie jar and local storage of a browser instance is populated. Therefore, we created a base profile that is loaded by *OpenWPM* upon each page visit. To populate this profile, we successively visited each landing page in our dataset once and used the resulting Firefox profile in our measurement. The base profile is not updated after each visit (i. e. we perform a *stateless* crawl), which, on the one hand, implies that the order of visited websites has no impact on the results and, on the other hand, that trackers have to re-set their identifiers. Hence, this setup allows us to distinguish between user identifiers and session IDs.

We conducted the measurement from a university network within the EU. To perform the measurement, we profit of two machines which in total have 95 GB of RAM, 56 CPUs (*Intel Xeon CPU @ 2.10GHz*), and more than 5 TB of hard disk space. When visiting a website we used a 30 second timeout interval to compensate for e. g. site crashes or slowly or not loading websites. We did not retry a page visit in case it failed in the first attempt. In our crawl, we used simple faked user interactions (i. e. random scrolling and mouse jiggling) because previous work found that this increases the number of observed tracking attempts [Ur20a].

Identifying Third Parties In our analysis, we use the *WhoTracks.Me* database to map domains (eTLD+1) to the organizations running them [Cl18]. This clustering allows us to discuss the impact of a company rather than the impact of a domain because one company can run multiple tracking domains. Previous works have shown that some types of websites tend to use more (third-party) user tracking techniques than others (e. g. "News" websites) [SK19]. Therefore, we analyze if similar observations can be made for first-party tracking techniques as well. We use the *McAfee SmartFilter Internet Database* service to identify the primary purpose ("category") of a domain [Mc20]. This mapping allows us to understand which categories of sites utilize the technique of interest and gives us a brief impression of who mainly uses it. The *Cookiepedia* [On19] is a crowd sourced index for cookies, their purposes, and "owning" companies. In this work, we use the service to map identified cookies to the respective companies who "own" and use them.

Limitations: Like most large-scale web measurement studies, our work comes with the limitation that our crawler is not logged in to any website, nor do we perform any meaningful interaction with the websites. Hence, we will miss some pages and features of sites that are only triggered after a user interaction (e. g. payment processes) or after successful login (e. g. loading of personal data). Regarding tracking technologies, this means that some sites that wait until the user opts-in, might not set any cookies. Since the literature suggests that these opt-in choices ("cookie banners") are often not honored [Sa19; Ut19], our results can only be seen as a lower bound to the extend of the problem. Furthermore, our automated crawler might be detected by a page, which might then show different content than for real user visits.

C First-Party Tracking Cookies

Using our approach to classify tracking cookies, we found that, on average, each site uses 6 potential first-party tracking cookies (max: 598; min: 0) and over all profiles 19% of all first-party cookies are used to track users. Overall, we found that 210,721 (20%) of all first-party cookies could be used to track users. In total, we found such tracking cookies on 10,700 (93%) of the analyzed sites. Hence, a vast majority in our dataset of websites offer third parties their first-party storage to place identifiers. We used the name of a cookie as an indicator to distinguish which third-party utilizes the cookie. We observed a long tail distribution that shows that some prominent non-trivial cookie names are used by several websites, which indicates that they all originate from commonly used services or libraries. According to the classification of *Cookiepedia* [On19], the top cookies, of the aforementioned classified cookies, are used by *Google* (`_ga` 28%), *Facebook* (`_fbp`, 14%), and again *Google* (`__gads`, 7% and `_gcl_au` 6%). The results show that the large, known tracking companies are also utilizing first-party tracking tools, in addition to third-party tools. In absolute numbers, popular websites (i. e. sites with a low rank on the *Tranco* list) tend to use more first-party cookies and also utilize more of them to track users. In our experiment, popular websites (rank $\leq 3,000$) use on average 45 cookies (SD: 65) of which 21% (SD: 21) are used to track users while less popular websites (rank $> 3,000$) use, on average, 34 (SD: 44) cookies and 20% (SD: 61) of them were classified as trackers. We chose the top 3,000 sites because less popular websites show a different behavior, number wise (see the standard deviations). The X^2 test ($\alpha = 0.5$) also found a strong correlation between the absolute number of present trackers and the rank of the website (p-value < 0.001). Hence, such cookies are more likely to appear on popular sites. However, if we look at the relative number of first-party tracking cookies in relation to other cookies, we find that popular websites use fewer trackers than other sites.

Defense Mechanisms One simple defense mechanism to avoid third-party cookies all together is to tell the browser never to store them. While third-party cookies are seemingly unrelated to first-party cookie sit is still interesting to analyses whether websites switch their tracking techniques if the preferred one is actively blocked (i. e. do they store identifier in the first-party context if the third-party context is disabled). In our measurement, this applies to profiles two and four (i. e. we used Firefox's option to "never" store third-party cookies). It is worth noting that the browser accepts and sets a third-party cookie but deletes it right away. Hence, the script or request that sets the cookie can do this 'normally' but cannot access the cookie afterward. Overall, we observed a change of roughly 13% in the use of first-party cookies if third-party cookies are disabled. This decrease can probably be attributed to the slight reduction of visited websites and the usual noise of large-scale Web measurements. Hence, blocking third-party cookies has a negligible effect on the use of first-party tracking cookies, which means that trackers do not use this way to evade the deletion of third-party cookies.

Towards Detection of Malicious Software Packages Through Code Reuse by Malevolent Actors

Marc Ohm[1] Lukas Kempf[2] Felix Boes[3] Michael Meier[4]

Abstract: Trojanized software packages used in software supply chain attacks constitute an emerging threat. Unfortunately, there is still a lack of scalable approaches that allow automated and timely detection of malicious software packages and thus most detections are based on manual labor and expertise. However, it has been observed that most attack campaigns comprise multiple packages that share the same or similar malicious code. We leverage that fact to automatically reproduce manually identified clusters of known malicious packages that have been used in real world attacks, thus, reducing the need for expert knowledge and manual inspection. Our approach, AST Clustering using MCL to mimic Expertise (ACME), yields promising results with a F_1 score of 0.99. Signatures are automatically generated based on characteristic code fragments from clusters and are subsequently used to scan the whole npm registry for unreported malicious packages. We are able to identify and report six malicious packages that have been removed from npm consequentially. Therefore, our approach can support the detection by reducing manual labor and hence may be employed by maintainers of package repositories to detect possible software supply chain attacks through trojanized software packages.

Keywords: Software Supply Chain; Malware; Abstract Syntax Tree; Markov Cluster Algorithm

1 Introduction

Modern software is developed on top of an ever-growing pool of third party tools, libraries and packages. Hence, the integrity of software projects depend directly on the integrity of the underlying software supply chain. Over the past few years, software supply chain attacks that leverage trojanized software packages kept emerging [Oh20]. A central role in that ecosystem is held by maintainers of package repositories like npm or Python Package Index (PyPI). These platforms are repeatedly abused for the distribution of trojanized software packages that are part of a software supply chain attack.

From the point of view of a malware author, distributing trojanized software packages to open source package repositories is a both cost-efficient and effective for a number of reasons. This is because open source packages are frequently used in various software projects. However, the packages code base is unlikely to be audited by the user of the

[1] University of Bonn, Institute for Computer Science 4, Friedrich-Hirzebruch-Allee 8, 53115 Bonn, Germany ohm@cs.uni-bonn.de
[2] University of Bonn & Hochschule Bonn-Rhein-Sieg kempf@uni-bonn.de
[3] University of Bonn boes@cs.uni-bonn.de
[4] University of Bonn & Fraunhofer FKIE mm@cs.uni-bonn.de

doi:10.18420/sicherheit2022_02

package. Therefore, malicious packages tend to be available for roughly 200 days before being identified and removed [Oh20]. Clearly, an improvement of automated capabilities for timely detection of such attacks is mandatory. It must be noted that at least npm and PyPI are taking effort to implement the detection of malicious software packages [Ha21; Py22].

Based on a manual annotated dataset, we evaluate various automated approaches to mimic the manual clustering of malicious packages which is typically performed by an expert. Following the idiom "if you've seen one, you've seen them all" we automatize the identification of related malicious packages to keep the upper hand in the arms race of software supply chain attacks. Consequentially, we propose a timely detection of malicious packages based on signatures derived from identified clusters. To this end, we leverage Abstract Syntax Trees (AST) that are generated from known malicious packages that have previously been used in real world attacks. Eventually, clusters of packages that share source code are identified through Markov Cluster Algorithm (MCL).

Our results indicate excellent performance ($F_1 = 0.99$) on the annotated dataset and good scalability for practical application. This way, suspicious packages are detected as soon as they are published to a package repository. It supports the detection by notifying an analyst about suspicious packages and giving hints about possible connections to already known malicious packages. Thus, the contribution of this paper is the automated and signature-based detection of possibly malicious packages which may then be analyzed by an expert. Eventually, we were able to detect and report six incidents of malicious packages on npm that share code with previously distributed malicious packages.

The remainder of this paper is organized as follows. Sect. 2 provides related work to frame the academic context of our approach. The underlying methodology for our approach is depicted in Sect. 3. Our results are presented and discussed in Sect. 4. Sect. 5 concludes the paper and provides an outlook for future work.

2 Related Work

The detection of similar code fragments is being used for the detection of software plagiarism. To this end, a vast majority of approaches leverage ASTs [CDR09; CJ11; DG13; NJK19; RKC18].

The detection of source code similarity also found application in cyber security. Especially, the detection of software vulnerabilities leverages code similarity detection in order to identify known but currently undetected vulnerabilities in software. Known vulnerable source code is used to generate signatures which are used to scan other source codes for these known patterns. [Bi20; Ch20; Li16; YLR12]

In contrast to vulnerable software components, the research field of malicious software packages is comparably new. Most often, they try to identify possible typosquatting attacks [Ta20; Ts16; Vu20]. Nonetheless, there are approaches that try to detect malicious

packages based on their source code. For instance, by looking for anomalies in a package's source codes compared to all other packages [Ča19; Ga19] or by leveraging heuristics [Du20; PO17] of presumably malicious characteristics. Moreover, dynamic analysis of suspicious packages might be considered in order to detect malicious behavior [Du20; OSM20]. Furthermore, Ohm et al. [Oh20] systematized software supply chain attacks by collecting and analyzing a large dataset of malicious open source packages that have been used in real world attacks.

Now, with an annotated dataset of known malicious packages at hand, it is possible to leverage insights and ground truth to develop suitable detection techniques. Our contribution differs from the approaches mentioned beforehand by leveraging evidence-based characteristics of known malicious packages. In this work, we focus on static code analysis in order to detect malicious packages based on automatically generated signatures.

3 Methodology

As observed by Ohm et al. [Oh20], malicious packages tend to have characteristic code fragments in common. This might be because they are employed in the same attack campaign or were simply copied by other malware authors. As discussed in the introduction, an automated approach that solves the time-consuming and tedious tasks to identify and search for said characteristic code fragments is highly anticipated. Our approach consists of three automated steps: (1) calculation of the source code similarity between all known malicious packages, (2) clustering based on these similarities, and (3) generation of signatures for each cluster.

After providing our metric that measures how well the automated clustering mimics the manual approach, we evaluate the clustering on an annotated dataset. In order to show that our tool is efficient and feasible for practical application on the large scale, it is evaluated on the full npm repository in Sect. 4.3. Hereby, we are able to identify and report six malicious packages that have been removed from npm consequentially.

3.1 Embedding of Packages

Each package consists of a number of source files and each source file consists of a number of functions. In our approach, each function is represented as Abstract Syntax Tree. AcornJs [Ma20], a lightweight parser for JavaScript, is used to transform source code into AST representation. Through abstraction of source code into a structural representation, naming of identifiers is of no matter. Hereby, we treat member functions as independent functions and group all statements in the global scope into a function. Similarity of two ASTs, i.e., two packages, is calculated according to the Tree Edit Distance introduced by Zhang-Shasha [ZS89] using the Python package `zss` [HJ13]. The similarity of two packages,

A and B, is the smallest Tree Edit Distance when comparing all ASTs, i.e., all functions, of package A to all ASTs of package B. This way, we can quantify the similarity between all malicious packages at hand which is then used to identify clusters among these.

3.2 Clustering of Packages

The defined goal is to group known malicious packages. Under the assumption that the malicious packages mostly share the malicious code, we can now identify the relevant code fragment for each cluster and use it as signature. To this end, we evaluate several unsupervised clustering algorithms on the packages (using the distances between their representations) and compare the resulting clusters with the experts clustering of malicious packages. Hereby, unsupervised methods have been chosen to reduce the need for prior knowledge about the dataset. We evaluate connected component (*ccomp*) and maximal cliques (*clique*) by leveraging the Python package `NetworkX` [HSS08]. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is implemented by using the Python package `scikit-Learn` [Pe11] and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) by `hdbscan` [MHA17]. Last, we examine Markov Cluster Algorithm (MCL) [va00] for which we leverage the Python package `Markov-Clustering` [Al20]. We expect the automated approach to identify the same amount of clusters containing the same set of malicious packages as if performed manually.

All approaches are evaluated on the "Backstabber's Knife Collection" dataset [Oh20][5] as ground truth. This dataset contains malicious packages harvested from npm, PyPI, and RubyGems. Furthermore, a manual clustering with expert knowledge of these was performed that formed seven clusters comprising 109 malicious packages from npm. For sake of brevity and with respect to the amount of npm packages in the dataset, we focus on packages written for Node.js in JavaScript. However, our approach is transferable to all kind of programming languages.

In order to compare the manual clustering by Ohm et al. [Oh20] with a fixed automated clustering approach, the conceiving metrics Precision, Recall and F_1-score are employed as follows. First of all, we assume that the manual clustering is complete and accurate, i.e., every malicious code similarity is found and packages are clustered correctly. Then, a pair of packages is said to be a **true positive** if the two packages are in the same cluster in both approaches, a **true negative** if the two packages are in different clusters in both approaches, a **false positive** if the two packages are in different manually generated clusters but in the same automatically generated cluster, and a **false negative** if the two packages are in the same manually generated clusters but in the different automatically generated clusters.

With this definition, the metrics Precision, Recall and F_1-score are interpreted as follows. By definition, Precision is the ratio of true positives in all positives. Therefore, the Precision

[5] For reproducibility we use the version of the dataset as described in the publication.

is high if the number of false positives is relatively low. Observe that this is the case if the automated approach generates clusters that are overall finer or as fine as the manual approach. Observe analogously that the Recall is high if the automated approach generates clusters that are overall coarser or as coarse as the manual approach. Consequentially, the F_1-score (which is the harmonic mean of Precision and Recall) measures how well the automated clustering mimics the manual approach.

3.3 Derivation of Signatures

It turns out that AST Clustering using MCL to mimic Expertise (ACME) mimics the manual clustering of an expert nearly perfectly with an F_1-score of 0.99. Based on top of ACME, the signature of a cluster of malicious packages is derived by constructing a so called fingerprint from each characteristic code fragment.

Our approach leverages fingerprinting as proposed by Chilowicz et al. [CDR09]. From each function f represented in an AST G, we derive a so called fingerprint $\mathcal{H}_f$. To this end, we focus our attention to the subgraph $G_f \subset G$ associated to f after all (nested) functions that are defined inside f are discarded. For each node $v \in G_f$, we concentrate on its type $t(v)$. After fixing SHA-256 as hash function C, the fingerprint of f is defined recursively as follows. Given an arbitrary node $v \in G_f$ with children $w_1, w_2, \ldots$, we define $\mathcal{H}(v)$:

$$\mathcal{H}(v) = C\left(t(v) \parallel \mathcal{H}(w_1) \parallel \mathcal{H}(w_2) \parallel \ldots\right) \tag{1}$$

Denoting the root of G_f by r_f, the fingerprint of f is

$$\mathcal{H}_f = \mathcal{H}(r_f) \tag{2}$$

We remark that we leverage a different function t than Chilowicz et al. Our $t(v)$ solely takes the type of node into account. Thus, our subgraph G_f is very focused on the structure of the code fragment by discarding nonstructural information like operators. For instance the code fragments $a + b$ and $a * b$ result in the same fingerprint. However, $a + (a + a)$ and $(a + a) + a$ yield different fingerprints. Let us remark further that we group code into a dummy global function if it resides outside of functions, i.e., in global scope. Furthermore, we treat functions inside classes as independent functions.

For each cluster c, one or more characteristic code fragment are chosen to serve as signature S_c as follows. By construction, a cluster consists of a set of ASTs and to each AST, we associate a fingerprint $\mathcal{H}$. Roughly speaking, a fingerprint is simply a hash of a given AST. Now a fingerprint $\mathcal{H}$ is "characteristic" if the following conditions are met: (1) The code fragment $\mathcal{H}$ is unique to its cluster, i.e., $\mathcal{H}$ is not derived from any package in any other cluster, (2) the code fragment $\mathcal{H}$ is derived from at least two packages in its cluster, and (3)

the code fragment $\mathcal{H}$ cannot be derived from one of the most depended upon packages[6] from npm.

Observe that (1) ensures that the signatures of the clusters are pairwise disjoint. This means that a newly classified package is assigned to one cluster only. Observe further that the condition (2) focuses signatures on common and hence descriptive code for the analyzed cluster. Condition (3) forbids code fragments that can be found in popular and hence supposedly benign packages.

The signature of a cluster c comprises all characteristic code fragments, i.e., fingerprints, of that cluster. The use of ASTs and the leveraged abstraction level are able to detect exact clones of known malicious code by definition. By discarding identifier names, the approach becomes resilient against obfuscation through unreadable names and renaming in general. Now, a package p matches the signature $\mathcal{S}_c$ of cluster c if at least one of p's fingerprints $h_1^p, \ldots, h_N^p$ matches a fingerprint $h \in \mathcal{S}_c$.

4 Results

This section summarizes our results from experiments as introduced in the previous sections. First, we evaluate which clustering approach is suited best in combination with AST to reproduce the results of the manual clustering in Sect. 4.1. The best approach is leveraged in Sect. 4.2 to automatically generate and optimize signatures based on identified clusters. These signatures are subsequently used in Sect. 4.3 to scan the whole npm registry for unreported malicious packages that have code fragments common to known malicious packages.

4.1 Reproduction of Clustering

Recall from Sect. 3 that we aim to automate the tedious and time-consuming task of manually finding (variations of) recognized malicious code fragments in a given package repository. Hereby, packages with similar malicious code fragments are clustered using various unsupervised cluster algorithms. In this subsection, we evaluate the quality of these approaches that attempt to reproduce the result of the manual clustering of Ohm et al. [Oh20].

Tab. 1 displays the performance of ASTs in conjunction with all clustering algorithms. It is noticeable that most clustering algorithms yield either high Precision or high Recall. Solely, MCL is capable of reaching both high Precision and high Recall thus recreating the manual cluster as similar as possible. With a Precision of 0.97 and a Recall of 1.00 the F_1 score is

[6] `https://www.npmjs.com/browse/depended`, we are limited to the 108 most depended upon packages due to technical issues of the website.

at 0.99. Through the use of ACME we are able to recreate the manual clustering performed by expert almost perfectly.

Having a suitable, automated, and unsupervised clustering at hand, signatures can now be derived to describe the syntactic characteristics of malicious packages of that cluster. Using these signatures, packages related to the same attack campaign, i.e., sharing some code fragments, are detected and identified automatically. The upcoming subsection evaluates the quality of the automatically generated signatures to quantify their practical suitability.

4.2 Quality of Signatures

The previous subsection shows that the experts task of manually clustering malicious code is automated almost perfectly by combining ASTs with MCL. Using the ACME approach described in Sect. 3.3, a signature S_c is derived for each cluster c. In this subsection, we discuss the sizes of the clusters and we demonstrate that the first two conditions yield signatures with a promising Recall.

In Tab. 2, the resulting clusters, their sizes, and corresponding number of signatures are shown. Our approach automatically identified seven clusters that cover 97 packages. As stated initially, 104 packages belong to a manual created cluster in the dataset. This is because the manual clustering by Ohm et al. also took dependency into account for clustering. Our approach solely relies on code syntax similarity and hence may not cluster all packages as in the dataset. However, only few clusters based on dependency exists and hence ACME was able to reproduce the manual results almost perfectly. Nonetheless, if the manual clustering is flawed, our approach also contains these flaws.

It is noticeable that the sizes of clusters varies heavily ($\sigma^2 = 230.40$). The smallest ones comprise two packages while the biggest clusters are of size 38 and 36 respectively. The size of a signature weakly correlates with the size of the corresponding cluster (Pearson $r = 0.65$, $p = 0.12$). However, there are outliers. For instance cluster 1 and 5 yield very large signature compared to their size and in contrast to that cluster 2 yields a very small signature.

Tab. 1: Results of AST and all clustering approaches with employed parameters, sorted by F_1 score.

Clustering	Parameter	Precision	Recall	F_1
MCL	exp = 2, inf = 2	0.9747	0.9958	0.9851
ccomp		0.6761	0.9958	0.8054
DBSCAN	$\varepsilon = 1$, minPts = 2	0.6761	0.9958	0.8054
HDBSCAN	minClst = 2	0.6580	0.9967	0.7927
clique		0.9878	0.6074	0.7522

In order to demonstrate that condition (3) is mandatory, we test the quality of the signatures associated to all fingerprints satisfying only conditions (1) and (2) as follows. In a 10-fold cross validation, we cluster the 114 packages containing malicious code with ACME and derive the signatures associated to all fingerprints satisfying conditions (1) and (2). These signatures are evaluated against the 10% of the split in the cross validation and against 108 benign packages. In this context, a package is positive if the automatically generated signature matches the package. On average, the Recall is 0.88 but the number of false positives is 46%. This is because the signatures contain too many fingerprints of benign functions, i.e., condition (3) is mandatory to reduce the number of false positives.

In total 3,875 fingerprints satisfying only conditions (1) and (2) are derived from the malicious packages of the seven clusters. Considering relevant fingerprints, i.e., after applying condition (3), the seven clusters yield 3,396 (-12.36%) fingerprints in total.

4.3 Large Scale Evaluation

For large scale evaluation of our signatures, we harvest the npm repository on 25$^{\text{th}}$ of September 2020. At this time, 1,396,447 packages were listed and respectively 1,396,413 are obtained in their "latest" version. In total, 20,017,543 files and 749,558,178 function are inspected.

Tab. 2 also lists the number of matches ($|\mathcal{M}_c|$) our signatures produced per cluster. After the automated removal of false positive fingerprints, the total amount of matches is reduced from 283,887 to 136,157 (-52.04%). For manual optimization, we inspect the 50 most matching fingerprints for each cluster. This takes roughly 10 minutes per cluster and resulted in 133 signatures (3.92%) being removed. This further reduces the amount of matches from 136,157 to 70,432 (-48.27%).

Tab. 2: Identified clusters based on ACME sorted by size. The size of corresponding signatures $\mathcal{S}_c$ and matches M_c are based on the level of optimization: Consider all fingerprints satisfying only conditions (1) and (2), all conditions, and all conditions plus manual optimization.

		cond. (1) + (2)		cond. (1), (2) + (3)		cond. (1), (2), (3) + manual	
No.	Size	$\lvert\mathcal{S}_c\rvert$	$\lvert M_c\rvert$	$\lvert\mathcal{S}_c\rvert$	$\lvert M_c\rvert$	$\lvert\mathcal{S}_c\rvert$	$\lvert M_c\rvert$
1	38	3,752	278,473	3,282	131,842	3,232	70,228
2	36	1	1	1	1	1	1
3	14	40	1,137	34	694	2	0
4	3	3	3	3	3	3	3
5	2	75	4,191	72	3,536	22	200
6	2	2	81	2	81	1	0
7	2	2	0	2	0	2	0
$\sum$	97	3,875	283,887	3,396	136,157	3,263	70,432

It must be noted that the signatures of cluster 1 are responsible for most of the matches (99.71%). This indicates that our approach fails to choose descriptive fingerprints from this cluster and hence produces many false positive matches. This indicates that a more sophisticated selection of relevant fingerprints might be needed. Solely 204 suspicious packages need manual inspection when leaving out matches from that cluster. However, for the remaining clusters the ACME is able to pre-select a reasonable number of suspicious packages that need further manual inspection. Furthermore, ACME gives a hint to the analyst by stating suspicious functions of matching packages.

In addition to automatically generated signatures, we manually create signatures for packages that did not belong to a cluster. To this end, we extract the fingerprint of malicious functions by hand. This results in eight new pseudo-cluster with corresponding signatures. However, this yields only one additional match.

4.3.1 Detected Packages

By the construction of the signatures, see Sect. 3.3, every match is treated as suspicious and hence needs manual inspection to verify actual maliciousness. Eventually, we were able to identify seven unreported but malicious packages that have code in common with known malicious packages. We identify the packages `nodetest199`[7], `nodetest1010`[8], and `plutov-slack-client`[9] based on the signature of cluster 4. All of them try to establish a reverse shell upon installation in order to give the attacker full control of the victims' systems.

Furthermore, ACME detects the packages `revshell` and `node-shells` that claim to be proof of concept packages. Thus, npm security did not publish a security advisory for `revshell` but nonetheless removed it from the registry. However, the package `node-shells` was published by Adam Baldwin, head of security at npm, and was thus not reported by us.

The package `hellhun_homelibrary` which was found by a manually generated signature was indeed not a malicious package itself. It is affected by `flatmap-stream`, the malicious package that was used in the `event-stream` incident. Hence, npm security decided to inform the developer about it instead of removing it.

The publication dates of the packages related to cluster 4, namely `nodetest1010` (2018-08-03), `nodetest199` (2018-08-02), and `plutov-slack-client` (2018-03-30), fit the overall time frame of known malicious packages from that cluster (2018-03-06, 2018-09-08, 2018-03-25). Thus, we conclude that we identified remnants of a previous attack.

However, the package `npmpubman`[10] matches the signature of cluster 2 and it aimed at the

[7] `https://www.npmjs.com/package/nodetest199`
[8] `https://www.npmjs.com/package/nodetest1010`
[9] `https://www.npmjs.com/package/plutov-slack-client`
[10] `https://www.npmjs.com/package/npmpubman`

exfiltration of data about the victims' systems. It was published on 2020-09-13 which is way later than the average package from cluster 2. We reported it on 2020-09-28, only 15 days after it was published.

Overall, we conclude that our automated generation of signatures reduces the workload for manual analysis drastically. However, manual optimization is still required to further boil down the amount of matches, thus further reducing the amount of suspicious packages for manual inspection. Nonetheless, our straight forward approach already yields feasible results. Hence, ACME can be leveraged to search packages for variations of known malicious code.

5 Conclusion

In this paper, we examine how source code similarities of known malicious packages can be leveraged to support the detection of software supply chain attacks. Our main goal is to automatically detect malicious software packages that are uploaded to large package repositories like npm. To this end, we automatize the clustering and signature generation of known malicious packages which is typically based on expert knowledge and manual inspection. On a dataset of 114 malicious npm packages that are used in real world attacks, we evaluate several approaches to find and group syntactical similarities in source codes. Based on that, clusters of packages with similar structure are identified automatically and unsupervised.

Compared to the manual clustering of these packages at hand, our best approach yields promising results ($F_1 = 0.99$). It leverages Abstract Syntax Trees (AST) to compare source code of multiple packages and Markov Cluster Algorithm (MCL) to identify clusters among these and is hence called AST Clustering using MCL to mimic Expertise (ACME). Our approach can be used to support the detection by pre-selecting suspicious packages based on signatures of known malicious code fragments for manual inspection. This reduces the need for expert knowledge and manual inspection drastically.

To demonstrate the effectiveness of ACME, a scan of the whole npm registry based on all generated signatures is performed. This revealed seven previously unreported packages in total. A manual inspection shows that four of them are indeed malicious packages and are therefore reported by us and subsequently removed from npm. Two of the remaining three are proof of concept packages. The last package itself is not malicious but it contains a full copy of the known malicious package `flatmap-stream` as dependency.

In conclusion, this means that our approach is feasible to automatically generate signatures for known malicious packages which then may be used to scan packages for remnants, variations, and imitators of known malicious packages. Through the use of ASTs the approach is resilient to modifications of the source code like renaming of variables and minor structural modifications. Furthermore, ACME is transferable to any other programming language. However, automatically generated signatures reduce the manual work needed but still cause

many false positives which may be removed manually. Nonetheless, our naive approach already yields promising results and good scalability.

For future work we plan to optimize our signature generation and enhance support for structural modifications of the source code. Eventually, we would like to expand our approach to other software ecosystems like Python Package Index (PyPI) and RubyGems.

Acknowledgment

This work is funded under the SPARTA project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 830892.

References

[Al20] Allard, G.: Markov Clustering, 2020, URL: https://github.com/GuyAllard/markov_clustering, visited on: 10/29/2020.

[Bi20] Bilgin, Z.; Ersoy, M. A.; Soykan, E. U.; Tomur, E.; Çomak, P.; Karaçay, L.: Vulnerability Prediction From Source Code Using Machine Learning. IEEE Access 8/, pp. 150672–150684, 2020.

[Ča19] Čarnogursky, M.: Attacks on Package Managers, MA thesis, Masaryk University, Faculty of Informatics, 2019.

[CDR09] Chilowicz, M.; Duris, E.; Roussel, G.: Syntax tree fingerprinting for source code similarity detection. In: 2009 IEEE 17th International Conference on Program Comprehension. IEEE, pp. 243–247, 2009.

[Ch20] Chinthanet, B.; Ponta, S. E.; Plate, H.; Sabetta, A.; Kula, R. G.; Ishio, T.; Matsumoto, K.: Code-based Vulnerability Detection in Node. js Applications: How far are we? arXiv preprint arXiv:2008.04568/, 2020.

[CJ11] Cosma, G.; Joy, M.: An approach to source-code plagiarism detection and investigation using latent semantic analysis. IEEE transactions on computers 61/3, pp. 379–394, 2011.

[DG13] Djurić, Z.; Gašević, D.: A source code similarity system for plagiarism detection. The Computer Journal 56/1, pp. 70–86, 2013.

[Du20] Duan, R.; Alrawi, O.; Kasturi, R. P.; Elder, R.; Saltaformaggio, B.; Lee, W.: Measuring and preventing supply chain attacks on package managers. arXiv preprint arXiv:2002.01139/, 2020.

[Ga19] Garrett, K.; Ferreira, G.; Jia, L.; Sunshine, J.; Kästner, C.: Detecting suspicious package updates. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER). IEEE, pp. 13–16, 2019.

[Ha21] Hanley, M.: GitHub's commitment to npm ecosystem security, 2021, URL: https://github.blog/2021-11-15-githubs-commitment-to-npm-ecosystem-security/, visited on: 01/06/2022.

[HJ13] Henderson, T.; Johnson, S.: Zhang-Shasha: Tree edit distance in Python, 2013, URL: https://pythonhosted.org/zss, visited on: 01/15/2021.

[HSS08] Hagberg, A.; Swart, P.; S Chult, D.: Exploring network structure, dynamics, and function using NetworkX, tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[Li16] Li, Z.; Zou, D.; Xu, S.; Jin, H.; Qi, H.; Hu, J.: VulPecker: an automated vulnerability detection system based on code similarity analysis. In: Proceedings of the 32nd Annual Conference on Computer Security Applications. Pp. 201–213, 2016.

[Ma20] Marijn Haverbeke, I. S. et al.: Acorn, 2020, URL: https://github.com/acornjs/acorn, visited on: 10/21/2020.

[MHA17] McInnes, L.; Healy, J.; Astels, S.: hdbscan: Hierarchical density based clustering. Journal of Open Source Software 2/11, p. 205, 2017.

[NJK19] Novak, M.; Joy, M.; Kermek, D.: Source-code similarity detection and detection tools Used in academia: a systematic review. ACM Transactions on Computing Education (TOCE) 19/3, pp. 1–37, 2019.

[Oh20] Ohm, M.; Plate, H.; Sykosch, A.; Meier, M.: Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2020.

[OSM20] Ohm, M.; Sykosch, A.; Meier, M.: Towards detection of software supply chain attacks by forensic artifacts. In: Proceedings of the 15th International Conference on Availability, Reliability and Security. ACM, pp. 1–6, 2020.

[Pe11] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V., et al.: Scikit-learn: Machine learning in Python. the Journal of machine Learning research 12/, pp. 2825–2830, 2011.

[PO17] Pfretzschner, B.; ben Othmane, L.: Identification of Dependency-based Attacks on Node. js. In: Proceedings of the 12th International Conference on Availability, Reliability and Security. Pp. 1–6, 2017.

[Py22] PyPI Warehouse: Malware Checks, 2022, URL: https://warehouse.pypa.io/development/malware-checks.html, visited on: 01/06/2022.

[RKC18] Ragkhitwetsagul, C.; Krinke, J.; Clark, D.: A comparison of code similarity analysers. Empirical Software Engineering 23/4, pp. 2464–2519, 2018.

[Ta20] Taylor, M.; Vaidya, R. K.; Davidson, D.; De Carli, L.; Rastogi, V.: SpellBound: Defending Against Package Typosquatting. arXiv preprint arXiv:2003.03471/, 2020.

[Ts16] Tschacher, N. P.: Typosquatting in programming language package managers, BA thesis, Universität Hamburg, Fachbereich Informatik, 2016.

[va00] vanDongen, S.: A cluster algorithm for graphs. Information Systems [INS]/R 0010, 2000.

[Vu20] Vu, D.-L.; Pashchenko, I.; Massacci, F.; Plate, H.; Sabetta, A.: Typosquatting and Combosquatting Attacks on the Python Ecosystem. Proc. of EuroS&PW'20/ , 2020.

[YLR12] Yamaguchi, F.; Lottmann, M.; Rieck, K.: Generalized vulnerability extrapolation using abstract syntax trees. In: Proceedings of the 28th Annual Computer Security Applications Conference. Pp. 359–368, 2012.

[ZS89] Zhang, K.; Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM journal on computing 18/6, pp. 1245–1262, 1989.

Fighting Evasive Malware: How to Pass the Reverse Turing Test By Utilizing a VMI-Based Human Interaction Simulator

Jan Gruber, Felix C. Freiling[1]

Abstract: Sandboxes are an indispensable tool in dynamic malware analysis today. However, modern malware often employs sandbox-detection methods to exhibit non-malicious behavior within sandboxes and therefore evade automatic analysis. One category of sandbox-detection techniques are reverse Turing tests (RTTs) to determine the presence of a human operator. In order to pass these RTTs, we propose a novel approach which builds upon virtual machine introspection (VMI) to automatically reconstruct the graphical user interface, determine clickable buttons and inject human interface device events via direct control of virtualized human interface devices in a stealthy way. We extend the VMI-based open-source sandbox DRAKVUF with our approach and show that it successfully passes RTTs commonly employed by malware in the wild to detect sandboxes.

Keywords: Malware Analysis; Sandboxing; Virtual Machine Introspection; Reverse Turing Test

1 Introduction

To cope with the vast amount of new malware samples found in the wild, automated analysis approaches are required to classify and triage the bulk of theses samples. One of the standard techniques to perform this is *automated dynamic malware analysis* using sandboxes. A sandbox is an isolated but closely monitored execution environment that tracks a program's system interactions and makes the resulting analysis available to further scrutiny.

Malware sandboxes were pioneered in Windows XP using a technique called *API-hooking* to catch system calls [WHF07]. Following the successes of these automated techniques, threat actors came up with a myriad of techniques to hinder or evade analyses [GLL14]. A common strategy of malware today is to exhibit a "split personality" meaning to refrain from the execution of its malicous payload if they were able to successfully detect that it is running in a sandbox [Ba10]. The resulting arms race between attackers and defenders initiated a trend of moving monitoring capabilities gradually deeper down the system stack towards the hardware, culminating in the use of hypervisor hooks and other hardware-assisted virtualization features like *Two-Dimensional Paging* [WHH13]. This has resulted in sandboxes that are stealthy regarding ongoing monitoring [Le14].

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Department Informatik, Martensstr. 3, 91058, Erlangen, Germany; jan.gruber@fau.de, felix.freiling@fau.de

doi:10.18420/sicherheit2022_03

While being stealthy, hypervisor-based analysis environments are also rather involved. Therefore, user-level sandbox approaches like the well-known and widely used open-source sandbox *Cuckoo*[2] have remained popular. But they are restricted to user-level malware [MK21] and can also be actively evaded with methods like unhooking, API hammering and the direct use of system calls as shown by the project *anticuckoo*.[3] Therefore, it is advisable to rely on introspection-based monitoring, in order to get reliable analysis results. The open source project DRAKVUF[4] provides a powerful virtualization-based agentless black-box binary analysis system, which is presently the only open-source VMI framework that provides a frontend to function as full-fledged sandbox.[5]

1.1 Reverse Turing Tests

Apart from attempting to detect any (user-level) specific changes of an execution environment, another highly effective class of sandbox detection approaches as described by Bulazel and Yener [BY17] are *reverse Turing tests* (RTTs). The name is derived from the classical Turing test, where a human attempts to distinguish between an actual person and a computer. In this case, however, the malicious computer program tries to identify the presence of a human, in order to determine which one of its split personalities to show. Thus, instead of trying to detect technical changes of the execution environment, RTTs gather evidence of a human user interacting with the system which regularly revolve around human interface device (HID) events.

Generally, one can distinguish between three different ways in which RTTs can be performed:

(1) passively observe events related to human use (usually HID events, e.g., mouse movement and keyboard input),
(2) demand active interaction (e.g., by presenting a dialog box or generating the need to scroll down to the bottom of a window), and
(3) check signs of the usual "wear and tear" of human use in the system (e.g., clipboard content or recently opened files).

In order to remain stealthy, real-world malware commonly employs such observation techniques with respect to HID events. Since providing meaningful interactions with a running program automatically via the graphical user interface (GUI) is a difficult problem, demands for active interactions appear effective and were recently observed in the wild as well [HO20]. Therefore, we explore the utilization of VMI-based approaches to fool evasive malwares demanding user interaction.

[2] See `https://www.honeynet.org/projects/active/cuckoo-sandbox/`,
[3] See `https://github.com/therealdreg/anticuckoo`, Commit 4a1e7f2
[4] See `https://github.com/tklengyel/drakvuf`
[5] Maintained by CERT.pl, see `https://github.com/CERT-Polska/drakvuf-sandbox` and `https://drakvuf-sandbox.readthedocs.io/en/latest/`, accessed on 04.11.2021

1.2 Contributions

The paper presents a method that allows sandbox environments to pass the reverse Turing test. We designed and implemented a proof-of-concept system based on the open source binary analysis system DRAKVUF which reconstructs the GUI of an analysis guest running a Windows 7 operating system by means of virtual machine introspection to find clickable buttons.[6] The system then injects mouse clicks at those locations via the *QEMU Monitor Protocol* in a targeted manner. This is a novel approach to imitate user behavior and a first step in the direction of "meaningful interaction" with a running program, which can be conceptually transferred to other operating systems as well. We evaluate the system and show that it successfully passes all RTTs that are commonly deployed by malware in the wild.[7]

1.3 Paper Outline

We summarize the design goals of our system in Section 2. Subsequently, Section 3 describes its integration into the architecture of DRAKVUF and its building blocks as well as the inner working of our proof-of-concept to simulate human-like behaviour. In Section 4, we evaluate the efficacy of our implementation, before we summarize the paper, discuss remaining challenges, and give directions for future work in Section 5.

Due to the limited extent of this paper, we give a brief overview of anti-sandbox techniques in general and present several examples of RTTs in the appendix, which can be read independently.

2 Design Goals

Our goal was to build a system that is able to simulate human interaction and thereby counter reverse Turing tests, in order to support reliable sandbox-based dynamic analyses with low false negative detection rate.

One approach of commercial sandbox suppliers is to enable the user to remotely control the sandbox much like she would be sitting in front of a real machine after submitting her program for analysis.[8] This approach, however, just shifts the RTT to another level without solving the underlying problem. Although it leads to passing the RTTs reliably, it does not scale at all. Since there is a need to address this issue, the well-known open-source sandbox

[6] Our implementation has been published under the terms of version 2 of the GNU General Public License at `https://github.com/tklengyel/drakvuf/tree/master/src/plugins/hidsim`, Commit bc7708b

[7] The RTTs which implemented for the evaluation of the system have been added to the open-source test framework *pafish* under the terms of version 3 of the GPL and can be found at `https://github.com/a0rtega/pafish/pull/72`, Commit f68d74f

[8] Refer to the online sandbox `https://any.run/` for example

Cuckoo[9] employs countermeasures to fool the malware in regard to passive as well as active RTTs. This is accomplished by running a Python agent inside the guest VM that moves the cursor to a random location every second and clicks the left mouse button. Furthermore, it scans for windows and dialogs to automatically simulate the button clicking [Ra13] by using API-functions, like `SendMessageW` and `EnumWindows` exposed by `user32.dll`, which is accessed via Python's ctypes-interface.[10] However, as mentioned above, agent-based approaches like *Cuckoo* are inherently limited.

Our aim was to bring the capabilities to counter RTTs into a sandbox based on hypervisor hooks. We chose the black-box malware analysis system DRAKVUF as the basis for our system. Until now, its engine, which follows the paradigm of being absolutely stealthy [Le14], did neither accomodate HID event emulation nor a mechanism to provide some kind of meaningful interaction with the programs under test.

The upmost design goal for our solution was to stay absolutely transparent, following DRAKVUF's overall maxime that no analysis-related code, regardless of kernel-level or user-level, should run inside the analysis guest. At the same time, we aim for enhancing the sandbox's capabilities by a simulator of human behaviour to trick the malware into revealing its malicious intent.

Based on the above design choices, a HID-emulation module had to be built and integrated, which specifically sought to achieve the following tasks:

(a) Inject randomized HID events to pass passive RTTs,
(b) detect GUI updates and reconstruct GUI elements via VMI, and
(c) interact with dialogs.

3 Implementation

We now describe the most important aspects of the implementation of our prototype.

3.1 Integrating Into the DRAKVUF Architecture

DRAKVUF is built on top of the Xen virtual machine monitor. It runs in the control domain (*dom0*) and thereby makes use of direct memory access (DMA) through the *LibVMI* library, which was pioneered by B.D. Payne [Pa12], to gain access to the state of the fully virtualized Xen HVM (hardware virtual machine) guests, which are observed from the aforementioned privileged domain [Le14, p. 387], [LS20, p. 2].

[9] See `https://www.honeynet.org/projects/active/cuckoo-sandbox/`,

[10] See `https://github.com/cuckoosandbox/cuckoo/blob/master/cuckoo/data/analyzer/windows/modules/auxiliary/human.py`, Commit b26f88b

Within the control domain, DRAKVUF utilizes hypervisor features to control virtualization extensions provided by the CPU, such as the *Extended Page Tables* (EPT). While it relies on several techniques to transfer control to the hypervisor (VMEXIT), the utilization of breakpoint injection is the commonly used way to trap the execution. In order to do this, the #BP instruction – INT3 with instruction opcode `0xCC` – is written into the guest's memory and the CPU is configured to issue a VMEXIT, when breakpoints are executed. Xen, in turn, is setup to forward those events to the control domain, where DRAKVUF is running, which handles those events with previously configured callbacks [Le14, p. 388]. For the case of Windows guest systems, the code locations to inject INT3-instructions into, are determined by the utilization of debug symbols provided by Microsoft, which are converted with the memory forensics tool *Volatility* into a textual representation, in order to reliably infer the physical addresses of various kernel datastructures and establish a map of those with the help of the Kernel Processor Control Region at runtime [Le14, p. 389].

Another important capability utilized by DRAKVUF is Xen's alt2pm feature, which allows to create multiple EPTs and to switch between those at runtime [LS20, p. 3]. By restricting EPT access permissions of pages, it is possible to stealthily trace memory accesses, because EPT permission violations trap into the hypervisor, where the pages, which have been manipulated by breakpoint injection, could be switched transparently and thus remain hidden from Window's patchguard or a malware scanning for INT3-instructions as well as other modifications. This means, that the ongoing observation is not detectable from the perspective of the monitored virtual machine [Le16].

In addition to this core functionality, DRAKVUF houses a plugin architecture, which allows for flexible extension by the capability of installing traps and event handlers per plugin. If one of the expected events occurs in the VM, control is transferred to DRAKVUF running in the control domain, where the list of handlers for this type of event is traversed, in order to execute each subscribed handler [Ko18, pp. 112 f.].

3.2 The Human Interaction Simulator's Architecture

To implement human-like simulation capabilities, we built upon DRAKVUF's plugin architecture and introduced a component called *hidsim*, whose lifecycle is controlled by DRAKVUF's "main loop". The component is multi-threaded to a) inject HID events and b) install an event handler to parse the memory on updates of the user interface, in order to identify and extract GUI-related datastructures, as Figure 1 illustrates.

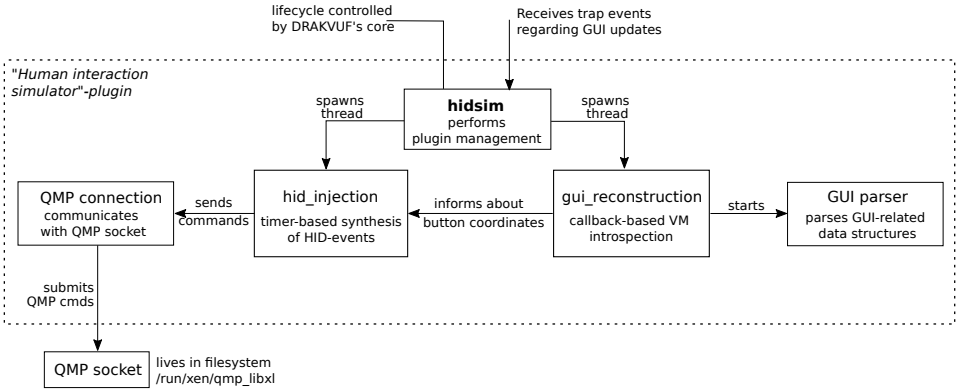


Fig. 1: High-level overview of the implemented plugin.

3.2.1 Injection of Human Interface Device Events

Since Xen uses the QEMU device model, we could rely on the QEMU monitor protocol (QMP)[11], which allows applications to control a QEMU instance in a fine grained way by sending JSON-commands, defined by the so-called QAPI, directly to its Unix domain socket. Several of those commands allow the sending of HID events over a QMP-socket to a QEMU instance, so that its handlers update the virtualized human interface devices accordingly. From the analysis guest's perspective, this is an approach which is fully transparent. By doing so, the HID injection subcomponent is able to send either pre-recorded or randomized HID events, like cursor updates or keystrokes, following a timer-based approach. However, in order to achieve the indistinguishable illusion of a person sitting in front of the sandbox's alleged screen, the actual coordinates to click on have to be provided by a sub-component which is responsible for the reconstruction of the GUI.

3.2.2 Reconstruction of the Graphical User Interface

At the start of the analysis of a Windows 7-guest, we set up a trap on the system call `NtUserShowWindow`, which is exposed by `win32k.sys` – the kernel-mode portion of the Windows GUI subsystem [RSI12, p. 50]. We accomplish this by locating the system call-handler with the help of the GUI system service descriptor table,[12] which contains the array of pointers for each system call-handler related to USER and GDI services implemented in `win32k.sys` [RSI12, pp. 137 and 273]. Then, we insert a breakpoint, so that the plugin gets notified about updates of the GUI without employing some kind of busy-waiting, so that a

[11] See `https://wiki.qemu.org/Documentation/QMP`, accessed on 28.10.2021
[12] Also called *W32pServiceTable*

second thread can start to reconstruct the GUI based on the in-memory data structures with the help of *LibVMI* and scan for clickable buttons.

This is done by identifying and iterating all interactive window stations and their desktops, in order to retrieve the active instance of the session. Afterwards, all windows, which are represented as `tagWND`-structs in memory to be specific, are traversed to build up a depth-ordered list, which is done by following an algorithm pioneered by B. Dolan-Gavitt [Li12a].[13] Since buttons are represented as `tagWND`-structs as well, potentially clickable buttons of interest can then be determined by looking at the top n list elements and employing several heuristics regarding size, class ID, visibility and textual content of the `tagWnd`-struct in question. After having identified the top-most structure which fulfills the requirements, the visible part of the area and its centroid are determined and passed to the injection thread, where a corresponding HID command will be constructed and sent to the guest via QEMU's monitor protocol, so that the cursors will be moved smoothly to the given coordinates and click on those. For implementation details, the interested reader might refer to the publicly available source code.[14]

4 Evaluation

In order to evaluate the efficacy of our approach, we implemented test cases using the Win32-API in C which perform several RTTs, which were briefly addressed in Section 1.1 and will be discussed in depth in Appendix A. With the aim to contribute to the open-source community, we decided to extend the popular open-source framework *pafish*[15] by those tailored but nevertheless realistic tests. *Pafish* bundles a collection of several sandbox detection techniques used by malware in the wild [Hu15] and has already been taken up in other academic work [Yo16] but lacked most RTTs until now. The newly added implementations were published as a patch and have been incorporated into *pafish*.[16] We decided to use these handcrafted implementations, which were derived from checks found in real world malware samples, as presented in Appendix A at detail, instead of using evasive samples themselves for practical reasons. By doing so, we gain quantifiability as well as reproducibility. This is especially notable, because RTTs are often found in loaders, whose sole task is to download the actual payload from malware distribution sites, which are usually only available for a few days and would make it hard to determine, whether the RTT was passed or not.

We compared our implementation[17] with the popular open-source sandbox *Cuckoo* in version 2.0.7 with its before mentioned auxiliary module called `human.py` using VirtualBox

[13] See the originally implementation of Brendan Dolan-Gavitt in Volatility's `screenshot`-plugin

[14] See especially `https://github.com/tklengyel/drakvuf/tree/master/src/plugins/hidsim/gui`, commit 63fdbf6

[15] See `https://github.com/a0rtega/pafish`, building upon commit 62dad68.

[16] Refer to `https://github.com/a0rtega/pafish/pull/72` for our patched version of *pafish* which has been integrated into version 0.6 in the meantime

[17] Using `https://github.com/tklengyel/drakvuf/tree/master/src/plugins/hidsim`, commit bc7708b

machinery by submitting the enhanced *pafish* executable for analysis. The generated log files, where all failed tests are recorded, were extracted after each submission, in order to determine the RTTs which failed. Both sandboxes run their analyses on 32-bit Windows 7 SP1-guests, which are still common for the use case of malware analysis. We performed the analysis for exactly 60 seconds and used both systems with and without their respective plugin that is responsible for the simulation of human interaction, in order to establish causality between the tooling and the resulting observations. By performing three runs each, in which we did not observe any ambiguities, we are certain that we have eliminated fluctuations and inconsistencies. The results of this practical evaluation are depicted in Table 1.

Tab. 1: Results of running the RTTs defined in the extended *pafish* in Cuckoo and DRAKVUF, both with and without their respective human simulator plugin; The test cases correlate to the functions implemented in `rtt.c` of our patched version of *pafish*.

Sandbox	Mouse Presence	Mouse Movement	Natural Mouse Speed	Single Clicking	Double Clicking	Dialog Confirmation	Plausible Dialog Confirmation
Cuckoo without simulation	✓	✗	✗	✗	✗	✗	✗
Cuckoo with human.py	✓	✓	✗	✓	✓	✓	✗
DRAKVUF without simulation[18]	✓	✗	✗	✗	✗	✗	✗
DRAKVUF with hidsim[19]	✓	✓	✓	✓	✓	✓	✓

As a preliminary remark, we would like to mention that we did not compare analysis results, but considered only the efficacy of human behaviour simulation. For a comparison of the analysis capabilities, refer to [MK21].

Looking at the results, it is obvious that without any enhancements both *Cuckoo* and DRAKVUF fail all tests except the one for mouse presence. When the simulators are involved, the detection results change drastically. Our plugin passes all tests, whereas *Cuckoo*'s auxiliary package failed the check for the cursor's speed limit, since it randomly places the cursor on the screen. In addition, it failed the plausibility check added to the dialog confirmation, since it just uses `SendWindowMessageW(...)` to send a message of type `BM_CLICK` to the dialog, without taking the current cursor position into account. However, as a matter of fact, this is a made-up test, which has not been found in real world malware but is an obvious approach to determine simulation.

[18] Command: `drakvuf -d $DOM_ID -r "$KR_JSON" -W "$WIN32K_JSON" -i $PID -e "$REMOTE_PAFISH" -c "$REMOTE_CWD" -t $TIMEOUT -a regmon -a filetracer`

[19] Command as before with the following additional arguments: `-a hidsim --hid-monitor-gui --hid-random-clicks`

To summarize the evaluation, it is clearly shown, that our implementation passes all implemented RTTs and delivers a performance which is even more effective than *Cuckoo*'s auxiliary module, while it is not running any code inside the analysis guest – neither for cursor movement nor for button identification. Therefore, we claim that the proof-of-concept fulfills all of the design goals stated in Section 2.

5 Conclusion and Outlook

In the present paper, we reviewed reverse Turing tests (RTTs) as one class of evasion techniques commonly employed by malware. We surveyed implementations found in real world samples to motivate the need for appropriate countermeasures. Given the trend to move the sandboxes' monitoring measures deeper down the technology stack, we identified the necessity to realize the simulation of a human's presence without running any agent in the monitored system itself. In order to put this need into practice, we extended the VMI-based open-source sandbox DRAKVUF, which employs an absolutely stealthy monitoring approach, by a plugin to simulate a human's presence implementing a novel approach. We utilized virtual machine introspection to monitor the analysis guest for updates of its graphical user interface and to reconstruct the GUI-elements presented to the user, when such an update was detected. By doing so and utilizing QEMU's monitor protocol to control the virtualized human interface devices, we were able to perform meaningful interactions with the program under test to a certain extent, which was done by clicking on buttons of dialogs popping up. We evaluated our approach by extending the open-source testing tool called *pafish* by the RTTs found in malware samples and comparing the efficacy of our implementation with the popular open-source sandbox *Cuckoo*. This showed, that in regard to the simulation of human behaviour, our implementation is at least as effective as *Cuckoo*'s agent-based approach, but does not rely on runnning any code inside the analysis guest. Therefore, it is unsusceptible from this perspective.

Future work should try to enhance the meaningfulness of the simulated human behaviour, in order to pass more sophisticated active RTTs which malware might employ in the future. Another goal is to port our prototype to work with more recent operating systems, reconstruct the GUI in a richer way and tackle the challenge of being able to determine elements drawn by higher level GUI frameworks. Furthermore, a comparison between VMI-based and machine vision-based approaches in regard to efficacy and performance seems to be of interest. Obviously, passing the RTT in its generality remains an unsolved problem and will be subject of further research. However, the presented proof-of-concept can at least deal with several RTTs commonly employed by malware and trick it to reveal its harmful intent.

Acknowledgements

The practical implementation, on which this paper is based, was created in the course of a *Google Summer of Code* project, which was hosted by the *The Honeynet Project*.[20] Therefore, we would like to thank Michał Leszczyński and Adam Kliś (both from CERT.pl at the time) and DRAKVUF's creator Tamas K. Lengyel for their support and code reviews. Furthermore, we would like to thank Jonas Röckl, Tobias Latzo and the anonymous reviewers for helpful comments on the draft as well as the members of the Research and Training Group 2475 "Cybercrime and Forensic Computing" funded by *Deutsche Forschungsgemeinschaft* (DFG, German Research Foundation (grant number 393541319/GRK2475/1-2019) for fruitful discussions.

References

[Ba10] Balzarotti, D.; Cova, M.; Karlberger, C.; Kirda, E.; Kruegel, C.; Vigna, G.: Efficient Detection of Split Personalities in Malware. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2010, San Diego, California, USA, 28th February - 3rd March 2010. The Internet Society, 2010, URL: https://www.ndss-symposium.org/ndss2010/efficient-detection-split-personalities-malware.

[BY17] Bulazel, A.; Yener, B.: A survey on automated dynamic malware analysis evasion and counter-evasion: Pc, mobile, and web. In: Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium. Pp. 1–21, 2017.

[CS20] Chaffey, E. J.; Sgandurra, D.: Malware vs Anti-Malware Battle - Gotta Evade 'em All! In (Kohlhammer, J.; Angelini, M.; Bryan, C.; Gómez, R. R.; Prigent, N., eds.): 17th IEEE Symposium on Visualization for Cyber Security, VizSec 2020, Virtual Event, USA, October 28, 2020. IEEE, pp. 40–44, 2020, URL: https://doi.org/10.1109/VizSec51108.2020.00012.

[Fo19] Fois, Q.: Threat Actor "Cold River": Network Traffic Analysis and a Deep Dive on Agent Drable, tech. rep., Lastline Inc., Jan. 2019, URL: https://www.lastline.com/labsblog/threat-actor-cold-river-network-traffic-analysis-and-a-deep-dive-on-agent-drable/, visited on: 10/28/2021.

[GLL14] Gao, Y.; Lu, Z.; Luo, Y.: Survey on malware anti-analysis. In: Fifth International Conference on Intelligent Control and Information Processing. Pp. 270–275, 2014.

[HO20] Haughom, J.; Ortolani, S.: Evolution of Excel 4.0 Macro Weaponization, tech. rep., Lastline Inc., June 2020, URL: https://www.lastline.com/labsblog/evolution-of-excel-4-0-macro-weaponization/, visited on: 11/08/2021.

[20] See https://summerofcode.withgoogle.com/archive/2021/projects/5750586108018688/, accessed on 27.10.2021.

[Hu15] Hund, R.: Pafish: How to Test your Sandbox Against Virtualization Detection, 2015, URL: https://www.vmray.com/cyber-security-blog/a-pafish-primer/, visited on: 12/17/2020.

[Ko18] Kovalev, S. G.: Reading the contents of deleted and modified files in the virtualization based black-box binary analysis system Drakvuf. In: Proceedings of ISP RAS. Vol. 30. 5, 2018.

[Le14] Lengyel, T. K.; Maresca, S.; Payne, B. D.; Webster, G. D.; Vogl, S.; Kiayias, A.: Scalability, fidelity and stealth in the DRAKVUF dynamic malware analysis system. In (Jr., C. N. P.; Hahn, A.; Butler, K. R. B.; Sherr, M., eds.): Proceedings of the 30th Annual Computer Security Applications Conference, ACSAC 2014, New Orleans, LA, USA, December 8-12, 2014. ACM, pp. 386–395, 2014, URL: https://doi.org/10.1145/2664243.2664252.

[Le16] Lengyel, T.: Stealthy monitoring with Xen alt2pm, tech. rep., Xen Project, Apr. 2016, URL: https://xenproject.org/2016/04/13/stealthy-monitoring-with-xen-altp2m/, visited on: 10/29/2021.

[Li12a] Ligh, M. H.: MoVP 4.2 Taking Screenshots from Memory Dumps, tech. rep., The Volatility Foundation, Oct. 2012, URL: https://volatility-labs.blogspot.com/2012/10/movp-43-taking-screenshots-from-memory.html, visited on: 10/30/2021.

[Li12b] Ligh, M. H.: What do Upclicker, Poison Ivy, Cuckoo, and Volatility Have in Common?, tech. rep., The Volatility Foundation, Dec. 2012, URL: https://volatility-labs.blogspot.com/2012/12/what-do-upclicker-poison-ivy-cuckoo-and.html, visited on: 11/07/2021.

[LS20] Leszczyński, M.; Stopczański, K.: A new open-source hypervisor-level malware monitoring and extraction system – current state and further challenges. Virus Bulletin 12/, 2020.

[MK21] Melvin, A. A. R.; Kathrine, G. J. W.: A Quest for Best: A Detailed Comparison Between Drakvuf-VMI-Based and Cuckoo Sandbox-Based Technique for Dynamic Malware Analysis. In: Intelligence in Big Data Technologies—Beyond the Hype. Springer, pp. 275–290, 2021.

[Pa12] Payne, B. D.: Simplifying virtual machine introspection using LibVMI./, Sept. 2012, URL: https://www.osti.gov/biblio/1055635.

[Ra13] Rapid7: Fooling malware like a boss with Cuckoo Sandbox, tech. rep., Rapid7, Apr. 2013, URL: https://www.rapid7.com/blog/post/2013/04/16/fooling-malware-like-a-boss-with-cuckoo-sandbox/, visited on: 10/29/2021.

[RSI12] Russinovich, M. E.; Solomon, D. A.; Ionescu, A.: Windows Internals, Part 1: Covering Windows Server 2008 R2 and Windows 7. Microsoft Press, USA, 2012, ISBN: 0735648735.

[SK12] Singh, A.; Khalid, Y.: Don't Click the Left Mouse Button: Introducing Trojan UpClicker, tech. rep., Fireeye Inc., Dec. 2012, URL: https://webcache.googleusercontent.com/search?q=cache:NeVZ4J1Y-cQJ:https://www.fireeye.com/blog/threat-research/2012/12/dont-click-the-left-mouse-button-trojan-upclicker.html+&cd=1&hl=en&ct=clnk&gl=de, visited on: 11/07/2021.

[VS14] Vashisht, S. O.; Singh, A.: Turing Test in Reverse: New Sandbox-Evasion Techniques Seek Human Interaction, tech. rep., Fireeye Inc., Nov. 2014, URL: https://www.fireeye.com/blog/threat-research/2014/06/turing-test-in-reverse-new-sandbox-evasion-techniques-seek-human-interaction.html, visited on: 10/28/2021.

[WHF07] Willems, C.; Holz, T.; Freiling, F. C.: Toward Automated Dynamic Malware Analysis Using CWSandbox. IEEE Secur. Priv. 5/2, pp. 32–39, 2007, URL: https://doi.org/10.1109/MSP.2007.45.

[WHH13] Willems, C.; Hund, R.; Holz, T.: CXPinspector: Hypervisor-based, hardware-assisted system monitoring. Ruhr-Universitat Bochum, Tech. Rep/, p. 12, 2013.

[Yo16] Yokoyama, A.; Ishii, K.; Tanabe, R.; Papa, Y.; Yoshioka, K.; Matsumoto, T.; Kasama, T.; Inoue, D.; Brengel, M.; Backes, M.; Rossow, C.: SandPrint: Fingerprinting Malware Sandboxes to Provide Intelligence for Sandbox Evasion. In (Monrose, F.; Dacier, M.; Blanc, G.; García-Alfaro, J., eds.): Research in Attacks, Intrusions, and Defenses - 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings. Vol. 9854. Lecture Notes in Computer Science, Springer, pp. 165–187, 2016, URL: https://doi.org/10.1007/978-3-319-45719-2%5C_8.

A A Brief Survey of Reverse Turing Tests Found in Real World Malware

To address general interest, we now survey several reverse Turing tests found in the wild. The appendix merely illustrates the technical mechanisms of RTTs and is not necessary to understand the main part of the paper.

A.1 Passive Observation

A.1.1 A Test for Mouse Presence Found in a Malicious Document of *ColdRiver*

In its simplest form, an RTT might be to determine the availability of a mouse on the system. This is illustrated by Listing 1, which shows a part of a VBA-macro code extracted from a malicous document utilized by an APT actor called *ColdRiver* [Fo19]. There is a predicate,

which ensures, that the base64-encoded payload is only fetched and decoded, if the check for the presence of a mouse device succeeded beforehand.

```
1  Sub Document_Open()
2    <snip>
3    If Application.MouseAvailable Then
4      Set DM = CreateObject("Microsoft.XML" & "DOM")
5      Set EL = DM.createElement("t" & "mp")
6      EL.DataType = "bin.bas" & "e64"
7    <snip>
8  End Sub
```

List. 1: Passive reverse Turing test found in a VBA-macro of *ColdRiver*[21]

A.1.2 A Test for Mouse Movement Found in *IFSB/Gozi*

A slighty more elaborated but nevertheless primitive example for a passive observation test can be found in the leaked source code of the malware family *IFSB/Gozi*.

```
1574 #ifdef _WAIT_USER_INPUT
1575         do
1576         {
1577                 ULONG Seed = AvGetCursorMovement();
1578
1579                 Sleep(100);
1580
1581                 if (!Seed)
1582                 {
1583                         Status = ERROR_BADKEY;
1584                         continue;
1585                 }
1586
1587                 Status = CsDecryptSection(g_CurrentModule, Seed % 9);
1588         } while(Status == ERROR_BADKEY);
1589 #else
1590         Status = CsDecryptSection(g_CurrentModule, 0);
1591 #endif
```

List. 2: Passive reverse Turing test found in leaked source code of the malware family *IFSB/Gozi*; taken from `install.c`, l. 1574 – l. 1591.

As it is depicted in Listing 2, the malware does only decrypt and execute its malicious payload, if cursor movements could be detected beforehand. This could be determined by repeatedly calling the subroutine `AvGetCursorMovement()`, where the API-function `GetCursorPos(&Point)` is used to calculate the delta to the previous result (Listing 3,

[21] The macro was shortened and formatted for readability after its extraction via `olevba` from the malware sample with MD5-hash 48320f502811645fa1f2f614bd8a385a.

ll. 204 ff.). If there is no such delta, the program will infinitely check for movement every 100 ms until termination (Listing 2, ll. 1575 ff.).

```
196 #ifdef _WAIT_USER_INPUT
197
198 // // Returns mouse cursor position relative to previously saved coordinates. //
199 ULONG AvGetCursorMovement(VOID)
200 {
201         POINT Point;
202         ULONG Movement = 0;
203
204         GetCursorPos(&Point);
205
206         if (g_Point.x && g_Point.y)
207           Movement = Point.x - g_Point.y + ((Point.y - g_Point.y) << 16);
208
209         g_Point.x = Point.x;
210         g_Point.y = Point.y;
211
212         return(Movement);
213 }
214
215 #endif // _WAIT_USER_INPUT
```

List. 3: Calculation of cursor movement found in leaked source code of the malware family *IFSB/Gozi*; taken from `av.c`, l. 196 – l. 217.[22]

The above mentioned code snippets show original source code as it was written by the author of *IFSB/Gozi* also known as *Ursnif*. They were taken from a leak of the malware in version 2.13.24.1 provided by vx-underground.org.[23]

A.1.3 A Test for Mouse Clicks Found in *UpClicker*

In 2012, the trojan *UpClicker*, which delivered a remote access toolkit called *Poison Ivy* at that time [Li12b], was one of the first malware samples which checked for mouse clicks. It utilized the API-function `SetWindowsHookExA` with the parameter `0xE` standing for `WH_MOUSE_LL` to catch mouse input events [SK12], as it is illustrated by the decompilation in Listing 4, l. 38. By doing so, an application-defined hook procedure is installed into a hook chain, which is called every time a new mouse input event is about to be posted into a thread input queue, so that it can effectively catch all of those.

```
36 void FUN_00401000(void)
37   <snip>
```

[22] As a side note: This seems to be implemented in a flawed but nevertheless working way, because coordinate axes were mixed up, as the following expression taken from the code illustrates: `Movement = Point.x - g_Point.y + ((Point.y - g_Point.y) << 16);`.

[23] See `https://github.com/vxunderground/MalwareSourceCode/blob/main/Leaks/Win32/Win32.Gozi.rar`, accessed 31.10.2021

```
38   DAT_00406000 = SetWindowsHookExA(0xe,FUN_004010b0,(HINSTANCE)hmod,dwThreadId);
39   iVar1 = GetMessageA((LPMSG)&local_e0,(HWND)0x0,0,0);
40   while (iVar1 != 0) {
41     TranslateMessage((MSG *)&local_e0);
42     DispatchMessageA((MSG *)&local_e0);
43     iVar1 = GetMessageA((LPMSG)&local_e0,(HWND)0x0,0,0);
44   }
45   UnhookWindowsHookEx(DAT_00406000);
46                     /* WARNING: Subroutine does not return */
47   FUN_0040129d(0);
48 }
```

List. 4: Decompiled view on the installation of the mouse hook procedure found *UpClicker*.[24]

The malware was dormant until the left mouse button was released after a click, which was determined by checking the event type of each received message and comparing it with the value `0x202` resembling (`WM_LBUTTONUP`), as it is shown in l. 10 of the installed callback-function presented in Listing 5. When this happened, the malware showed its real personality and injected its malicious payload into another process (l. 12).

```
1  void FUN_004010b0(int param_1,int param_2,undefined4 param_3)
2  {
3    <snip>
4    if (param_1 == 0) {
5      if (param_2 == 0x200) { // WM_MOUSEMOVE
6           <snip>
7      }
8      else {
9        if (param_2 != 0x201) {
10         if (param_2 == 0x202) { // WM_LBUTTONUP
11           UnhookWindowsHookEx(DAT\_00406000);
12           FUN_00401170();
13                     /* WARNING: Subroutine does not return */
14           FUN_0040129d(0);
15         }
16         goto LAB_004010f9;
17       }
18 <snip>
```

List. 5: Decompiled view of the relevant part of the LowLevelMouseProc installed by *UpClicker*, which inspects the mouse events.

Several variations of passive RTTs have been observed in the wild concerning movement speed, movement patterns, clicking intervals or scrolling [CS20]. Some malware, for example, used the API-Call `GetAsyncKeyState` to wait on multiple clicks, others would check the cursor's speed, and would not execute, if the mouse cursor traveled to quickly [VS14].

[24] The sample has the MD5-Hash `ce69dee5307d58db4e2a6fdbcbf87e9d`; The decompilation of was done by *Ghidra* in version 10

[24] The decompiled code was shortened and formatted for readability

A.2 Demand for Active Interaction

Since passive tests are relatively primitive and error-prone, malware author's came up with active reverse Turing tests, which utilize the graphical user interface to demand some kind of (inter)action from an eventually present human operator. Those tests can often be found in malicious documents sent by e-mail, whose sole purpose is to function as a first stage loader to retrieve the actual payload from a malware distribution site (MDS).

One instance of this technique is presenting a dialog, which asks the user for the approval of an alleged repair action after opening an Excel document. The according XLM-macro code is presented in Listing 6. The return value of the dialog is used as a predicate, which acts as an active reverse Turing test as line 3 illustrates. If, and only if, the user presses the button labelled "OK", the content of the following cells is executed, which ultimately leads to an obfuscated call of `urlmon`'s `UrlDownloadToFileA` in line 10, in order to retrieve the second stage payload.

```
1  auto_open: auto_open->Macro251!$A$1
2  SHEET: Macro251, Macrosheet
3  CELL:A1 , =IF(ALERT("We found a problem with some content. Do you want to try to recover as
       ↪  much as we can?",1.0),,CLOSE(TRUE))
4  CELL:A2 , =GET.WORKSPACE(1.0)
5  CELL:A3 , =IF(ISNUMBER(SEARCH("Windows",A2)),,CLOSE(TRUE))
6  CELL:A4 , =GET.WORKSPACE(32.0)
7  CELL:A5 , =IF(ISNUMBER(SEARCH("Office",A4)),,CLOSE(TRUE))
8  CELL:A6 , =GET.WINDOW(1.0)
9  CELL:A7 , =IF(ISNUMBER(SEARCH("Fax",A6)),,CLOSE(TRUE))
10 CELL:A8 , =IF(GET.WORKSPACE(19.0),CALL("ur"&C6,"UR"&C7&"nloa"&C8&"ileA","JJCCJJ",0.0,GET.
       ↪ NOTE(D8),GET.NOTE(E8),0.0,0.0),CLOSE(TRUE))
11 CELL:A9 , =WAIT(NOW()+"00:00:05")
12 <snip>
13 CELL:C6 , None , lmon
14 CELL:C7 , None , LDow
15 CELL:C8 , None , dToF
```

List. 6: Active reverse Turing test found in XLM-macro.[25]

However, showing dialogs – like in this rather recent campaign from 2020 – is not the only instance of active Turing tests. Back in 2014, security analysts discovered an RTF-exploit, that would only trigger after scrolling to the second page of the document, which is what usually a curious human would do, but won't happen in an automated sandbox [VS14].

[25] Extracted with a tool called `xlmdeobfuscator` from the malicious document with MD5-hash 50d518246c2b61f5b427948f87a0aa24

Session 2

SMT-Based Verification of Concurrent Critical Systems

Matthias Güdemann[1]

Abstract: Petri nets are a widely used formalism to describe and analyze critical systems. It is in particular well suited for systems with concurrency like cache coherence protocols, fault-tolerant distributed systems or security critical protocols. The verification approaches for Petri nets are most often based on enumerative approaches which allow for analyzing complex, often temporal, properties.

Dataflow languages are widely used in safety critical systems. There are several state-of-the-art model-checkers for these languages. While the properties that can be verified are generally limited to invariants, it is possible to encode some interesting properties of Petri nets as invariants which makes them accessible for powerful analysis methods based on modern SMT and SAT solvers.

The SPiNAT approach transforms Petri net into synchronous dataflow language models. This allows for using predicate abstraction and the theory of unbounded integers allows to analyze the potentially unbounded markings of Petri nets using model-checking tools for languages like Lustre. The presented approach is orthogonal to enumeration based approaches for Petri net analysis and allows benefiting from any increase in efficiency of industrial strength SMT-based model-checkers like kind2 and JKind.

1 Introduction

In many critical systems there is some inherent concurrency, in particular if a system contains distributed components or uses a protocol. The behavior of such systems is often hard to understand as the different execution sequences make it easy to build systems that might deadlock or have parts which are never activated. Due to the complex possible behavior, it is most often not possible to apply testing in an exhaustive way.

To overcome these problems there are different formalisms to describe and analyze critical systems with concurrent behavior. Process algebras like CSP [Ho78] and languages like LNT [GLS17] are higher level description mechanisms which allow specification of concurrent systems. Their formal analysis is often conducted via transformation into Petri nets. Such formalisms have been used, e.g., to analyze safety of fault-tolerant distributed systems [Am11], TLS handshake [Bo18], fault-tolerant routing on a network-on-chip system [Zh16], system-on-chip cache coherence protocols [KS15] or autonomous cloud coordination managers [ASDP17]. The programming language Go [La] takes its use of channels from process algebras. The language Erlang [Er] uses a mechanism similar to synchronization in process algebras to exchange messages between concurrent light-weight processes. Many of the approaches to analyze such models of concurrent systems use Petri

[1] Munich University of Applied Science HM, Department of Computer Science and Mathematics, Lothstrasse 64, 80335 München, Germany matthias.guedemann@hm.edu

doi:10.18420/sicherheit2022_04

nets and are based on enumeration. While this allows for the analysis of complex properties it also makes the analyses susceptible to the state-space explosion problem.

Symbolic verification based on SAT and SMT solvers has become common in HW verification. Bounded model checking is very successful in bit-precise Software Model-checking [KT14]. On the one hand there is the drawback that these approaches are mainly limited to invariant properties. On the other hand, several interesting properties of concurrent systems can be expressed as invariant or reachability problems using observers. In these cases, it is possible to make use of the advanced algorithms, automated theorem provers and industrial strength model-checking tools for dataflow languages.

The main contribution of this work is the SMT-based Petri Net Analysis Technique (SPiNAT) to analyze concurrent systems using synchronous languages. It is meant to be an approach orthogonal to other analysis methods of these systems. In its current form, it could be used in a portfolio approach to verification where different algorithms are used in parallel. The overall result is simply the first finished analysis. The first experimental results show its applicability to Petri net models of the Model Checking Contest (MCC).

2 Background

2.1 Petri Nets

Petri nets are a low-level formalism to describe concurrent systems. For example the CADP [Ga13] toolbox translates the LNT [GLS17] language into Petri nets for further analysis. Petri nets, also called place transition or P/T nets.

Definition 1 (Petri Net) *A Petri net is a tuple $N = (P, T, F)$ with a set of places P, a set of transitions T and a relation of arcs F where $F \subseteq (P \times T) \cup (T \times P)$*

Definition 2 (Marking) *For a Petri net $N = (P, T, F)$ a marking M is a mapping $M : P \mapsto \mathbb{N}$ which represents the number of tokens at each place.*

A marking of a Petri net therefore expresses the number of token on each place of the net.

Definition 3 (Pre- and post-set) *For a Petri net $N = (P, T, F)$ the pre-set is ${}^\bullet x := \{y | yFx\}$ and the post-set is $x^\bullet := \{y | xFy\}$.*

Definition 4 (Labelling and Enabled Transitions) *A labelling is a function $L : F \mapsto \mathbb{N}$ which assigns a natural number to each arc.*

A transition $t \in T$ is enabled *iff*

$$\forall p \in {}^\bullet t : L(p,t) \leq M(p)$$

This means that a transition t is *enabled* if each incoming arc (p,t) is labelled with a number less than or equal to the marking of the place p. Informally this means that each place in the pre-set of t must hold enough tokens to enable the transition.

Definition 5 (Transition Firing) *If a transition t is enabled and the current marking is M, then the transition t can* fire *which changes the marking to M'. This is often written as $M \xrightarrow{t} M'$, where for new marking M' the following holds*

$$\begin{aligned} \forall p : & (p \in {}^\bullet t \rightarrow M'(p) = M(p) - L(p,t)) \\ & \wedge (p \in t^\bullet \rightarrow M'(p) = M(p) + L(t,p)) \\ & \wedge (p \in P \setminus ({}^\bullet t \cup t^\bullet) \rightarrow M'(p) = M(p)) \end{aligned}$$

Informally this means that when a transition t fires, the current marking changes by removing tokens from the places in the pre-set of t and adding tokens to the post-set of t. The number of tokens is equal to the label of the incoming (respective outgoing) arc of t.

Every Petri net has an initial marking M_I, i.e., an initial distribution of tokens on places. A marking M is *reachable* if there exists a finite sequence of transition firings such that the initial marking is transformed into M, i.e., $M_I \xrightarrow{t_1} \cdots \xrightarrow{t_n} M$. For more details see [Re13].

2.2 Model Checking using Satisfiability

Many of the most efficient approaches to verification use satisfiability or constraint problem formulation and rely on efficient SAT or SMT solvers to find solutions. SMT allows for a richer modeling language and can also use higher level reasoning similar to automated theorem provers. These tools support bit-precise formal analysis for fixed width bitvectors as well as additional decidable theories, e.g., the theory of unbounded integers for infinite state systems [KT14, CG12, La19].

The basic approach of bounded model-checking (BMC) unrolls the transition relation until a state that violates the property is reached. This method is incomplete but can be made complete using k-induction [BC00]. Newer approaches use for example the ic3/pdr [Br11] (property directed reachability) method to strengthen the property and to make it 1-inductive. In this paper we will use the verification of invariants or reachability checking to verify properties of Petri nets.

2.3 Lustre Dataflow Language

Dataflow languages are well studied and widely used for safety critical systems modeling and implementation. Lustre is a synchronous dataflow language which has a long history of use in safety critical systems in both commercial applications and academia. A program is built from hierarchical nodes. Special temporal operators allow the use of state variables with a defined initial value and a function to calculate the value for the next time step. At each time step the outermost node reads its inputs and uses these values to calculate simultaneously all values of each inner node and the output value of the outermost node. For details see [JRH16].

Several verification tools for Lustre have been developed, including commercial ones. There are modern open source industrial grade model checkers, kind2 [Ch16] and JKind [Ga18]. These tools allow for using different state of the art SMT engines, e.g., Z3 [DMB08] or CVC4 [Ba11] and use different algorithms in parallel in a portfolio approach to verification.

3 Expressing the Semantics of Petri Nets in Lustre

Here, the expression of Petri net semantics in Lustre follows an example driven explanation. A fully formal specification is under current development. In the SPiNAT approach, Lustre state variables are used to encode the places of Petri nets. Every state variable is of type integer (unbounded) and holds the number of tokens at the corresponding place. Transition firing changes the marking of the Petri net, adjusting the value of the state variables corresponding to the impacted places.

Two transitions are *independent* if their post-sets (or pre-sets) have no common element. If progress is possible, then at least one transition of a set of independent transitions fires. This is ensured by defining one Boolean *activation* input parameter for each transition. This uses the *assert* mechanism of Lustre to add the constraint that if progress is possible, then exactly one transition set can have activated transitions. Each activated transition in a set of independent transitions fires if it is enabled, i.e., its incoming places hold enough tokens. Via an additional *assert* it is ensured that at each step if there is a transition that is enabled there also exists an activated and enabled transition, guaranteeing progress if possible.

3.1 Example Petri Net

Figure 1 shows an example Petri net [wi]. In the initial marking the place p_1 contains one token. All other places are empty, each arc is labelled with a weight of value 1. The encoding of the Petri net in Lustre works as follows. For each of the four transitions the *main* node of the Lustre model has one Boolean input parameter in the form of act_t for a transition t. Each of these parameters models the potential *activation* of a transition, an

activated transition can fire if it is *enabled*, too. In the later analysis, a model-checker can used non-deterministic values for each activation input parameter of the transitions. The output of the *main* node is the content of the three places modeled as integers and the information which transition fires.

```
node main(act_t0, act_t1, act_t2, act_t3 : bool)
returns (p1,p2,p3 : int; firing_t0,firing_t1,firing_t2,firing_t3 : bool);
```

For each transition there exists a Boolean flag which is true iff the transition is enabled. For t_0 for example this means that p_1 and p_3 hold at least one token.

```
enable_t0 = p1 >= 1 and p3 >= 1;
```

As no transitions are independent, each transition set only contains a single element and each transition set is activated iff the contained transition is activated. Correct transition firing is modelled as follows via Lustre *assert*. This ensures that if a transition is enabled, then exactly one set of independent transitions has activated transitions. It also ensures that at least one activated transition is also enabled.

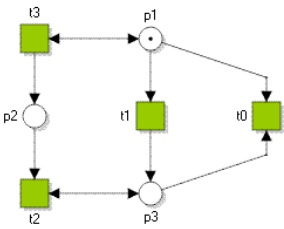


Fig. 1: Example Petri Net

```
trans_set0 = act_t0;
trans_set1 = act_t1;
trans_set2 = act_t2;
trans_set3 = act_t3;

assert (not (enable_t0 or enable_t1 or enable_t2 or enable_t3)
         or (bool2int(trans_set0) + bool2int(trans_set1) +
             bool2int(trans_set2) + bool2int(trans_set3) = 1));
assert ((enable_t0 or not act_t0) and (enable_t1 or not act_t1)
         and (enable_t2 or not act_t2) and (enable_t3 or not act_t3));
```

The first constraint models that if at least one transition is enabled, then exactly one independent set can contain activated transitions. In this example each transition set contains exactly one transition, the *bool2int* function equals 1 for true and 0 for false. Note, that if no transition is enabled a deadlock is reached and no progress is possible any more. The second constraint models that if there is an activated transition there is also enabled transition which fires.

For each transition t there exists a Boolean flag which represents that t fires, e.g., for transition t_0 in the example:

```
firing_t0 = act_t0 and enable_t0;
```

Places are modelled as state variables. The initial value of p_1 is 1, the value of p_1 in the next time step is defined by the expression after the arrow operator `->`. The `pre` operator represents the value in the previous time step. A firing of transition t_3 removes a token from p_1 and adds a token to it. Therefore, in that case the change is the difference between the weight of the incoming arc and the weight of the outgoing arc which is 0. For the place p_1 in the example this translates to (cf. Def. 5):

```
p1 = 1 -> if pre firing_t0 then pre p1 - 1
  else if pre firing_t1 then pre p1 - 1
  else if pre firing_t3 then pre p1 - 0
  else  pre p1;
```

3.2 Petri Net Properties for SPiNAT

There are some standard properties of Petri nets which can be directly expressed as invariants. For other properties full temporal logic is necessary and those are not yet supported in SPiNAT.

- *no dead transition / quasi-liveness* in a marking is a transition property which states that every transition can be enabled eventually.
- *no dead places* is a property on places which states that no place exists which never holds a token.
- *deadlock* in a marking states that no transition is enabled and no more progress is possible.
- *n-safety* is a property on the number of tokens at each place. A net is called *n-safe* if there are at most n tokens on each place for each reachable marking. An often used special case of this property is *1-safety*.

Quasi-liveness is checked via recording whether at least one transition is never enabled. Any counterexample to this invariant proves quasi-liveness. Dead places exist if the invariant holds that at least one place has value zero in all reachable markings. This requires observing whether each place holds a non-zero value of tokens eventually. A deadlock corresponds to observing that no transition is enabled.

```
enabled_t0 = false -> if enable_t0 then true else pre enabled_t0;
marked_p1 = false -> if 0 < p1 then true else pre marked_p1;
deadlock = not (enable_t0 or enable_t1 or enable_t2 or enable_t3);
```

The *n-safety* property is the conjunction of the boundedness of the places. In the example 1-safety of the net is formulated as follows:

```
prop_oneSafe = 1 >= p1 and 1 >= p2 and 1 >= p3;
```

In the example no dead places exist, i.e., for each place p there is a reachable marking where p holds at least one token. The net is not 1-safe because after two firings of the transition t_3 there are 2 tokens at place p_2. There exists a deadlock, i.e., if t_1 fires in the initial marking, no other transition can fire afterwards. The transition t_0 can never be enabled, either p_1 or p_3 holds a token, but never both. For the example net in Fig. 1 we get the following analysis results for the above properties.

```
<Failure> prop_noDeadlock is invalid by PDR for k=1 after 0.035s.
<Failure> prop_oneSafe is invalid by BMC for k=2 after 0.035s.
<Failure> prop_deadPlaceExists is invalid by BMC for k=2 after 0.035s.
<Success> prop_deadTransExists is valid by PDR after 0.100s.
```

The above properties are generated automatically. This is done by generating the necessary observers states and the necessary Boolean variables during the transformation from the Petri net description to Lustre. Obviously there can be other interesting properties, but most often application and net specific properties which can be modelled manually.

4 Model Checking Examples

The Model Checking Contest (MCC) [mc] provides example Petri nets in the XML-based Petri Net Modeling Language (PNML) [Hi10] format. Many of the models are parametric and can therefore be analyzed in different sizes and/or difficulties. For this experimental study, a small subset of the available models and their parametric variants was selected and analyzed. It it interesting to compare different analysis tools for Lustre among each other and then also compare SpiNat with an analysis tool for Petri nets.

4.1 SpiNat Variants and Results

There exist different model checkers which can check Lustre programs. The most modern versions of these are JKind and kind2. JKind is written in Java and allows for using the built-in SMTInterpol [CHN12] SMT solver or an external one based on the standardized SMTLIB2 format [BFT17]. kind2 is a reimplementation of the pkind model checker. It is written in OCaml and supports several SMT solvers with Z3 being the standard choice. Both model checkers implement different analysis engines with the main ones being bounded model-checking, k-induction, invariant generation and pdr/ic3.

The analysis runs were executed with a 5-minute timeout using JKind version 4.4.4 using Z3 version 4.8.1[2] as external solver and kind2 1.4.0 with Z3 version 4.8.12 on a quad-core i7-8665U with 1.90 GHz. Timeouts are marked with —, ✓ marks a valid property and ∅ marks an invalid property. The elapsed time is shown in parentheses after the result. The numbers beside the name indicate the variant of the model. For the *no dead transitions* / *no dead places* properties, a validity means reaching a marking and an property corresponds to an invariant. For *no deadlock* and *1-safety* a valid property is proven as an invariant and an invalid one via reaching a marking.

Table 1 shows the results for some selected model variants, comparing the results of JKind and kind2. Both analysis tools implement a portfolio approach for their analyses and are capable to use multiple processors. Neither tool is strictly superior to the other. For each one there are models where it is better than the other and vice-versa. For example, JKind is significantly faster for the referendum-10 model whereas kind2 is significantly faster for the 2 PhaseLocking model. For the viral 03 1 1 2 model JKind manages to strictly analyze more properties than kind2, while for the satellite 100-3 model the opposite is true.

[2] More recent versions of Z3 are currently not supported because of a format change.

Name	no dead transitions	no dead places	no deadlock	1-safety
JKind				
GPU FP 40 a	∅2m8.686s	∅4m1.755s	∅23.715ss	—
CloudOps	∅1.355s	∅1.355s	—	∅0.329s
Satellite 100-3	—	✓0.298s	—	∅0.235s
Referendum 10	∅6.587s	∅12.533s	∅0.336s	✓12.533s
Election 2020	✓0.754s	✓1.358s	—	∅0.429s
2 PhaseLocking	✓0.321s	✓0.321s	∅1m5.75s	∅0.228s
Raft 02	∅2m50.069s	∅2m50.069s	—	—
viral 03 1 1 2	∅1m25.486s	4m40.131s	∅1.271s	—
kind2				
GPU FP 40 a	—	—	∅51.575s	—
CloudOps	∅1.915s	∅1.915s	—	∅0.083s
Satellite 100-3	—	✓0.183s	✓22.284s	∅0.053s
Referendum 10	∅39.335s	∅39.335s	∅0.224s	✓3m34.42s
Election 2020	✓1.429s	✓3.345s	—	∅0.517s
2 PhaseLocking	✓0.242s	✓0.242s	∅23.010s	∅0.054s
Raft 02	∅3m9.877s	∅3m9.877s	—	—
viral 03 1 1 2	—	—	∅2.388s	—

Tab. 1: Results for different SPiNAT solver variants

4.2 Comparing SPiNAT to TINA

Here we present more results with different models, comparing the results of an analysis using SPiNAT with TINA [BRV04], a specialized Petri Net analysis tool which has been successful in the MCC.

The set of generated models for different examples of Petri nets from MCC is available[3]. The computer setup is the same as for the analyses in the previous section and used the same timeout of 5 minutes. The times reported for SPiNAT are the minimum of runs with kind2 and JKind. The reasoning here is that using both solvers and keeping the first result is easy to do in a portfolio approach.

Whether it takes longer to prove an invariant or compute a reachable marking depends on how the net is structured. Generally SAT / SMT based analysis is well suited to prove invariants via inductive arguments and finding reachable markings which are *not far* from the initial marking. *Deep* markings which require a specific, long sequence of transition firings can be costly to compute, either in terms of memory for BMC or in terms of time for pdr.

[3] `https://guedemann.org/downloads/pnml_in_lustre.zip`

TINA has direct support for 1-safety and deadlock freeness, other properties need to be expressed explicitly in its modeling language. Table 2 shows the comparison of analysis results using TINA and SpiNat for deadlocks and 1-safety. For TINA this was done using the *sift* tool using the `-dead` and `-f safe` option.

	no deadlock			1-safety		
Name	valid	SpiNat	TINA	valid	SpiNat	TINA
GPU FP 24 a	∅	8.073*s*	0.013*s*	—	—	—
GPU FP 28 a	∅	15.27*s*	0.017*s*	—	—	—
GPU FP 32 a	∅	13.643*s*	0.029*s*	—	—	—
GPU FP 36 a	∅	18.204*s*	0.067*s*	—	—	—
GPU FP 40 a	∅	23.715*s*	0.071*s*	—	—	—
CloudOps	—	—	—	∅	0.083*s*	0.00*s*
Satellite 100-3	✓	22.284*s*	0.160*s*	∅	0.053*s*	0.00*s*
Satellite 1000-32	✓	28.325*s*	25.791*s*	∅	0.053*s*	0.00*s*
Satellite 1500-46	✓	27.352*s*	1*m*2.946*s*	∅	0.084*s*	0.00*s*
Satellite 3000-94	✓	27.159*s*	3*m*36.982*s*	∅	0.074*s*	0.00*s*
Satellite 65535-2048	✓	28.814*s*	—	∅	0.074*s*	0.00*s*
Referendum 10	∅	0.224*s*	0.349*s*	✓	12.533*s*	0.361*s*
Referendum 15	∅	0.307*s*	2*m*13.568*s*	✓	34.352*s*	2*m*15.971*s*
Referendum 20	∅	0.417*s*	—	✓	1*m*12.267*s*	—
Referendum 50	∅	1.036*s*	—	—	—	—
Noc3x3 1 1 a	✓	3*m*24.345*s*	—	—	—	—
Election 2020	—	—	—	∅	0.429*s*	0.00*s*
2 PhaseLocking	∅	23.01*s*	0.002*s*	∅	0.054*s*	0.00*s*
Raft 02	✓	—	0.045*s*	✓	—	0.026*s*
viral 03 1 1 02	∅	2.388*s*	3.851*s*	—	—	—
viral 04 1 1 02	∅	2.858*s*	—	—	—	—
viral 04 1 1 03	∅	27.464*s*	—	—	—	—
viral 08 1 1 02	∅	1*m*52.008*s*	—	∅	1.375*s*	0.00*s*

Tab. 2: Comparing SpiNat and TINA on Selected Petri Net Models

It is clear that in some cases TINA is more efficient in its analysis and in other cases SpiNat is faster. If a counterexample can be found close to the initial marking TINA is efficient, almost independent of the size of the model variant, e.g., as seen for the GPU model variants. In contrast, for the satellite model the whole state space must be constructed for the valid deadlock property. The state space grows quickly, increasing the analysis time for TINA. For this model the induction based verification of SpiNat shows almost no increase in run-time for the deadlock property.

5 Related Work

Most Petri net analysis and verification methods are based on enumerative approaches. A lot of interesting properties are expressed in temporal logic which is more expressive than the simple invariants here. In [PCM14] the authors apply SMT based verification on a constrained subset of timed Petri nets. They use BMC which allows for checking reachable markings but cannot prove invariants. BMC also has the challenge that with every time step the problem for the underlying solver gets bigger and more complex.

In [ABDZ21] the authors combine SMT-based verification with an abstraction technique called polyhedral abstraction for Petri nets. They show that this works well together and allows for solving a lot of instances of the MCC competition. A similar approach is followed in [TM20] and the corresponding ITS-Tools. It would be interesting to see how to combine these approaches with the one described here.

In [BSS09] the authors propose a related approach which expresses the semantics of Lustre in an asynchronous framework called BIP (Behavior, Interaction, Priority). BIP models are then executed via a transformation into a special form of Petri nets. It could be interesting to see if it is possible to combine the framework in with SPiNAT to allow for analysis of systems with both synchronous and asynchronous behavior.

6 Conclusion and Outlook

This work shows that it is possible to express Petri nets in the synchronous dataflow language Lustre. This allows for using tools like JKind and kind2 to analyze certain properties of critical systems with concurrency. The approach is orthogonal to other verification methods. First experiments show that there are properties and Petri nets where the required analysis time is comparable or even better than specialized Petri net tools.[4] Using SMT-based verification for Petri nets is not new but to my knowledge dataflow languages as intermediate representation have not yet been proposed.

There is room to increase the efficiency of the current implementation. For larger Lustre models the parsers of the model checkers sometimes have problems. Therefore it might make sense to also provide the possibility to use them as libraries and use their API instead of text files to exchange models.

The current implementation supports basic Petri nets. A possible extension would be support for colored Petri nets and hierarchic Petri nets. To increase efficiency of the analyses it would make sense to support annotations, e.g., convert the information that a system is *n-safe* into a system constraint in order to speed up verification of other properties. By using *incremental inductive CTL* [HBS12] it would be possible to support temporal logic properties. This technique would have to be integrated into the model checkers directly.

[4] **Threat to validity** The author is no expert on using the TINA toolbox. Therefore, It is possible that with different options and / or some pre-processing, TINA can be faster than reported in the table.

Bibliography

[ABDZ21] Amat, Nicolas; Berthomieu, Bernard; Dal Zilio, Silvano: , On the Combination of Polyhedral Abstraction and SMT-based Model Checking for Petri nets, 2021.

[Am11] Ameur-Boulifa, Rabéa; Halalai, Raluca; Henrio, Ludovic; Madelaine, Eric: Verifying Safety of Fault-Tolerant Distributed Components – Extended Version. Research Report RR-7717, INRIA, September 2011.

[ASDP17] Abid, Rim; Salaün, Gwen; De Palma, Noel: Asynchronous synthesis techniques for coordinating autonomic managers in the cloud. Science of Computer Programming, 146:87–103, 2017.

[Ba11] Barrett, Clark; Conway, Christopher L; Deters, Morgan; Hadarean, Liana; Jovanović, Dejan; King, Tim; Reynolds, Andrew; Tinelli, Cesare: CVC4. In: International Conference on Computer Aided Verification. Springer, pp. 171–177, 2011.

[BC00] Bjesse, Per; Claessen, Koen: SAT-based verification without state space traversal. In: International Conference on Formal Methods in Computer-Aided Design. Springer, pp. 409–426, 2000.

[BFT17] Barrett, Clark; Fontaine, Pascal; Tinelli, Cesare: The SMT-LIB Standard: Version 2.6. Technical report, Department of Computer Science, The University of Iowa, 2017. Available at `www.SMT-LIB.org`.

[Bo18] Bozic, Josip; Marsso, Lina; Mateescu, Radu; Wotawa, Franz: A formal TLS handshake model in LNT. arXiv preprint arXiv:1803.10319, 2018.

[Br11] Bradley, Aaron R: SAT-based model checking without unrolling. In: International Workshop on Verification, Model Checking, and Abstract Interpretation. Springer, pp. 70–87, 2011.

[BRV04] Berthomieu, Bernard; Ribet, P-O; Vernadat, François: The tool TINA–construction of abstract state spaces for Petri nets and time Petri nets. International journal of production research, 42(14):2741–2756, 2004.

[BSS09] Bozga, Marius; Sfyrla, Vassiliki; Sifakis, Joseph: Modeling Synchronous Systems in BIP. In (Chakraborty, Samarjit; Halbwachs, Nicolas, eds): 9th ACM & IEEE International conference on Embedded software, EMSOFT 2009. ACM, Grenoble, France, pp. 77–86, October 2009.

[CG12] Cimatti, Alessandro; Griggio, Alberto: Software model checking via IC3. In: International Conference on Computer Aided Verification. Springer, pp. 277–293, 2012.

[Ch16] Champion, Adrien; Mebsout, Alain; Sticksel, Christoph; Tinelli, Cesare: The Kind 2 model checker. In: International Conference on Computer Aided Verification. Springer, pp. 510–517, 2016.

[CHN12] Christ, Jürgen; Hoenicke, Jochen; Nutz, Alexander: SMTInterpol: An interpolating SMT solver. In: International SPIN Workshop on Model Checking of Software. Springer, pp. 248–254, 2012.

[DMB08] De Moura, Leonardo; Bjørner, Nikolaj: Z3: An efficient SMT solver. In: International conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, pp. 337–340, 2008.

[Er] Erlang: , The Erlang Programming Language. https://www.erlang.org/.

[Ga13] Garavel, Hubert; Lang, Frédéric; Mateescu, Radu; Serwe, Wendelin: CADP 2011: a toolbox for the construction and analysis of distributed processes. International Journal on Software Tools for Technology Transfer, 15(2):89–107, 2013.

[Ga18] Gacek, Andrew; Backes, John; Whalen, Mike; Wagner, Lucas; Ghassabani, Elaheh: The JKind model checker. In: International Conference on Computer Aided Verification. Springer, pp. 20–27, 2018.

[GLS17] Garavel, Hubert; Lang, Frédéric; Serwe, Wendelin: From LOTOS to LNT. In: ModelEd, TestEd, TrustEd, pp. 3–26. Springer, 2017.

[HBS12] Hassan, Zyad; Bradley, Aaron R; Somenzi, Fabio: Incremental, inductive CTL model checking. In: International Conference on Computer Aided Verification. Springer, pp. 532–547, 2012.

[Hi10] Hillah, Lom-Messan; Kordon, Fabrice; Petrucci, Laure; Treves, Nicolas: PNML Framework: an extendable reference implementation of the Petri Net Markup Language. In: Proceedings of Petri Nets. Springer, pp. 318–327, 2010.

[Ho78] Hoare, Charles Antony Richard: Communicating sequential processes. Communications of the ACM, 21(8):666–677, 1978.

[JRH16] Jahier, Erwan; Raymond, Pascal; Halbwachs, Nicolas: The Lustre V6 reference manual. Verimag, Grenoble, Dec, 2016.

[KS15] Kriouile, Abderahman; Serwe, Wendelin: Using a formal model to improve verification of a cache-coherent system-on-chip. In: Proceedings of TACAS. Springer, 2015.

[KT14] Kroening, Daniel; Tautschnig, Michael: CBMC–C bounded model checker. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, pp. 389–391, 2014.

[La] Language, Go Programming: , The Go Programming Language. https://golang.org/.

[La19] Lange, Tim; Neuhäußer, Martin R; Noll, Thomas; Katoen, Joost-Pieter: IC3 software model checking. International Journal on Software Tools for Technology Transfer, pp. 1–27, 2019.

[mc] mcc: , Model Checking Contest. https://mcc.lip6.fr/models.php.

[PCM14] Półrola, Agata; Cybula, Piotr; Męski, Artur: SMT-based reachability checking for bounded time Petri nets. Fundamenta Informaticae, 135(4):467–482, 2014.

[Re13] Reisig, Wolfgang: Understanding Petri Nets: Modeling techniques, analysis methods, case studies. Springer, 2013.

[Sa] Satellite: , Memory Model. https://mcc.lip6.fr/pdf/SatelliteMemory-form.pdf.

[TM20] Thierry-Mieg, Yann: Structural Reductions Revisited. In: International Conference on Applications and Theory of Petri Nets and Concurrency. Springer, pp. 303–323, 2020.

[wi] wikipedia: , Petri Net Example. https://en.wikipedia.org/wiki/Petri_net.

[Zh16] Zhang, Zhen; Serwe, Wendelin; Wu, Jian; Yoneda, Tomohiro; Zheng, Hao; Myers, Chris: An improved fault-tolerant routing algorithm for a network-on-chip derived with formal analysis. Science of Computer Programming, 118:24–39, 2016.

A Satellite Model Example

The Satellite Memory model is an industrial case study from the aerospace domain. It has been submitted to MCC as an example model and is described in [Sa]. It is a Petri net with multiple variants. Figure 2 shows the initial marking for the variant $X = 100$ and $Y = 6$.

This models a satellite which contains memory in the form of a circular buffer. The size of the buffer in the number of available sectors is modelled as the parameter X. The circular buffer has a write pointer and a read pointer. The parameter Y models the minimal difference between the sector currently used for writing and the sector used for reading. More details on the case study can be found in [Sa].

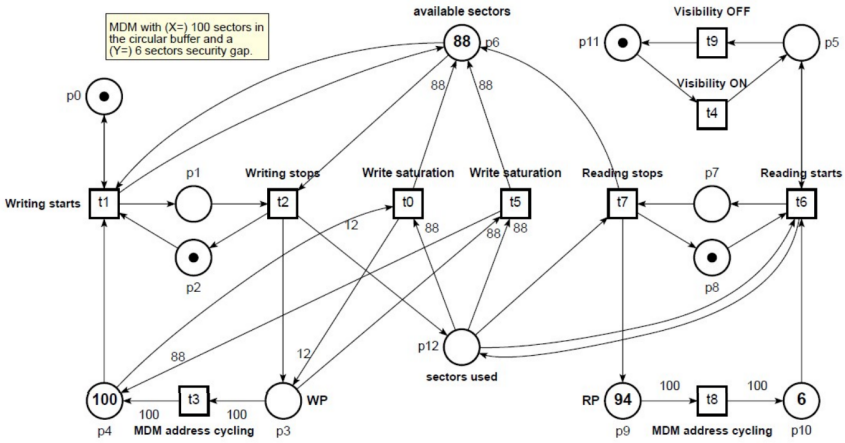


Fig. 2: Satellite Memory Model Variant $X = 100, Y = 6$ [Sa]

A.1 Lustre version of the Satellite Model

```
node main(act_t0, act_t1, act_t2, act_t3, act_t4, act_t5, act_t6,
   act_t7, act_t8, act_t9 : bool)

returns (p0, p1, p10, p11, p12, p2, p3, p4, p5, p6, p7, p8, p9 : int;
  firing_t0, firing_t1, firing_t2, firing_t3, firing_t4, firing_t5,
  firing_t6, firing_t7, firing_t8, firing_t9 : bool);

var
deadlock : bool;

trans_set0, trans_set1, trans_set2, trans_set3, trans_set4 : bool;

enabled_t0, enabled_t1, enabled_t2, enabled_t3, enabled_t4,
   enabled_t5, enabled_t6, enabled_t7, enabled_t8, enabled_t9 : bool;
```

```
enable_t0, enable_t1, enable_t2, enable_t3, enable_t4, enable_t5,
   enable_t6, enable_t7, enable_t8, enable_t9 : bool;

marked_p0, marked_p1, marked_p10, marked_p11, marked_p12, marked_p2, marked_p3,
  marked_p4, marked_p5, marked_p6, marked_p7, marked_p8, marked_p9 : bool;

prop_oneSafe : bool;
prop_deadTransExists : bool;
prop_deadPlaceExists  : bool;
prop_noDeadlock : bool;

let
trans_set0 = act_t0 or act_t4 or act_t8;
trans_set1 = act_t1 or act_t6;
trans_set2 = act_t2;
trans_set3 = act_t3 or act_t7 or act_t9;
trans_set4 = act_t5;

assert (not (enable_t0 or enable_t1 or enable_t2 or enable_t3 or enable_t4
          or enable_t5 or enable_t6 or enable_t7 or enable_t8 or enable_t9)
        or (bool2int(trans_set0) + bool2int(trans_set1) +  bool2int(trans_set2)
            + bool2int(trans_set3) + bool2int(trans_set4) = 1));

assert ((enable_t0 or not act_t0) and (enable_t1 or not act_t1) and
        (enable_t2 or not act_t2) and (enable_t3 or not act_t3) and
        (enable_t4 or not act_t4) and (enable_t5 or not act_t5) and
        (enable_t6 or not act_t6) and (enable_t7 or not act_t7) and
        (enable_t8 or not act_t8) and (enable_t9 or not act_t9));

enable_t0 = true and p12 >= 94 and p4 >= 6;
enable_t1 = true and p0 >= 1 and p2 >= 1 and p4 >= 1 and p6 >= 1;
enable_t2 = true and p1 >= 1 and p6 >= 1;
enable_t3 = true and p3 >= 100;
enable_t4 = true and p11 >= 1;
enable_t5 = true and p12 >= 94 and p3 >= 94;
enable_t6 = true and p10 >= 1 and p12 >= 1 and p5 >= 1 and p8 >= 1;
enable_t7 = true and p12 >= 1 and p7 >= 1;
enable_t8 = true and p9 >= 100;
enable_t9 = true and p5 >= 1;

deadlock = not (enable_t0 or enable_t1 or enable_t2 or enable_t3 or enable_t4
             or enable_t5 or enable_t6 or enable_t7 or enable_t8 or enable_t9);

p1 = 0 -> if pre firing_t1 then pre p1 + 1
  else if pre firing_t2 then pre p1 - 1
  else  pre p1;

p12 = 0 -> if pre firing_t0 then pre p12 - 94
  else if pre firing_t2 then pre p12 + 1
  else if pre firing_t5 then pre p12 - 94
  else if pre firing_t6 then pre p12 - 0
  else if pre firing_t7 then pre p12 - 1
  else  pre p12;
```

```
p3 = 0 -> if pre firing_t0 then pre p3 + 6
  else if pre firing_t2 then pre p3 + 1
  else if pre firing_t3 then pre p3 - 100
  else if pre firing_t5 then pre p3 - 94
  else  pre p3;

p5 = 0 -> if pre firing_t4 then pre p5 + 1
  else if pre firing_t6 then pre p5 - 0
  else if pre firing_t9 then pre p5 - 1
  else  pre p5;

p7 = 0 -> if pre firing_t6 then pre p7 + 1
  else if pre firing_t7 then pre p7 - 1
  else  pre p7;

p0 = 1 -> if pre firing_t1 then pre p0 - 0
  else  pre p0;

p11 = 1 -> if pre firing_t4 then pre p11 - 1
  else if pre firing_t9 then pre p11 + 1
  else  pre p11;

p2 = 1 -> if pre firing_t1 then pre p2 - 1
  else if pre firing_t2 then pre p2 + 1
  else  pre p2;

p8 = 1 -> if pre firing_t6 then pre p8 - 1
  else if pre firing_t7 then pre p8 + 1
  else  pre p8;

p10 = 3 -> if pre firing_t6 then pre p10 - 1
  else if pre firing_t8 then pre p10 + 100
  else  pre p10;

p6 = 94 -> if pre firing_t0 then pre p6 + 94
  else if pre firing_t1 then pre p6 - 0
  else if pre firing_t2 then pre p6 - 1
  else if pre firing_t5 then pre p6 + 94
  else if pre firing_t7 then pre p6 + 1
  else  pre p6;

p9 = 97 -> if pre firing_t7 then pre p9 + 1
  else if pre firing_t8 then pre p9 - 100
  else  pre p9;

p4 = 100 -> if pre firing_t0 then pre p4 - 6
  else if pre firing_t1 then pre p4 - 1
  else if pre firing_t3 then pre p4 + 100
  else if pre firing_t5 then pre p4 + 94
  else  pre p4;

firing_t0 = act_t0 and p12 >= 94 and p4 >= 6;
firing_t1 = act_t1 and p0 >= 1 and p2 >= 1 and p4 >= 1 and p6 >= 1;
```

```
firing_t2 = act_t2 and p1 >= 1 and p6 >= 1;
firing_t3 = act_t3 and p3 >= 100;
firing_t4 = act_t4 and p11 >= 1;
firing_t5 = act_t5 and p12 >= 94 and p3 >= 94;
firing_t6 = act_t6 and p10 >= 1 and p12 >= 1 and p5 >= 1 and p8 >= 1;
firing_t7 = act_t7 and p12 >= 1 and p7 >= 1;
firing_t8 = act_t8 and p9 >= 100;
firing_t9 = act_t9 and p5 >= 1;

enabled_t0 = false -> if enable_t0 then true else pre enabled_t0;
enabled_t1 = false -> if enable_t1 then true else pre enabled_t1;
enabled_t2 = false -> if enable_t2 then true else pre enabled_t2;
enabled_t3 = false -> if enable_t3 then true else pre enabled_t3;
enabled_t4 = false -> if enable_t4 then true else pre enabled_t4;
enabled_t5 = false -> if enable_t5 then true else pre enabled_t5;
enabled_t6 = false -> if enable_t6 then true else pre enabled_t6;
enabled_t7 = false -> if enable_t7 then true else pre enabled_t7;
enabled_t8 = false -> if enable_t8 then true else pre enabled_t8;
enabled_t9 = false -> if enable_t9 then true else pre enabled_t9;

marked_p0 = false -> if 0 < p0 then true else pre marked_p0;
marked_p1 = false -> if 0 < p1 then true else pre marked_p1;
marked_p10 = false -> if 0 < p10 then true else pre marked_p10;
marked_p11 = false -> if 0 < p11 then true else pre marked_p11;
marked_p12 = false -> if 0 < p12 then true else pre marked_p12;
marked_p2 = false -> if 0 < p2 then true else pre marked_p2;
marked_p3 = false -> if 0 < p3 then true else pre marked_p3;
marked_p4 = false -> if 0 < p4 then true else pre marked_p4;
marked_p5 = false -> if 0 < p5 then true else pre marked_p5;
marked_p6 = false -> if 0 < p6 then true else pre marked_p6;
marked_p7 = false -> if 0 < p7 then true else pre marked_p7;
marked_p8 = false -> if 0 < p8 then true else pre marked_p8;
marked_p9 = false -> if 0 < p9 then true else pre marked_p9;

prop_noDeadlock = not deadlock;

prop_deadTransExists = (not enabled_t0) or (not enabled_t1) or (not enabled_t2)
 or (not enabled_t3) or (not enabled_t4) or (not enabled_t5) or (not enabled_t6)
 or (not enabled_t7) or (not enabled_t8) or (not enabled_t9);

prop_deadPlaceExists = (not marked_p0) or (not marked_p1) or (not marked_p10)
    or (not marked_p11) or (not marked_p12) or (not marked_p2) or (not marked_p3)
    or (not marked_p4) or (not marked_p5) or (not marked_p6) or (not marked_p7)
    or (not marked_p8) or (not marked_p9);

prop_oneSafe = 1 >= p0 and 1 >= p1 and 1 >= p10 and 1 >= p11 and
  1 >= p12 and 1 >= p2 and 1 >= p3 and 1 >= p4 and 1 >= p5 and
  1 >= p6 and 1 >= p7 and 1 >= p8 and 1 >= p9;

--%PROPERTY prop_noDeadlock;
--%PROPERTY prop_oneSafe;
--%PROPERTY prop_deadTransExists;
--%PROPERTY prop_deadPlaceExists;
tel;
```

Hardening the Security of Server-Aided MPC Using Remotely Unhackable Hardware Modules

Dominik Doerner[1], Jeremias Mechler[2], Jörn Müller-Quade[3]

Abstract:

Garbling schemes are useful building blocks for enabling secure multi-party computation (MPC), but require considerable computational resources both for the garbler and the evaluator. Thus, they cannot be easily used in a resource-restricted setting, e.g. on mobile devices. To circumvent this problem, server-aided MPC can be used, where circuit garbling and evaluation are performed by one or more servers.

However, such a setting introduces additional points of failure: The servers, being accessible over the network, are susceptible to remote hacks. By hacking the servers, an adversary may learn all secrets, even if the parties participating in the MPC are honest.

In this work, we investigate how the susceptibility for such remote hacks in the server-aided setting can be reduced. To this end, we modularize the servers performing the computationally intensive tasks. By using data diodes, air-gap switches and other simple remotely unhackable hardware modules, we can isolate individual components during large parts of the protocol execution, making remote hacks impossible at these times. Interestingly, this reduction of the attack surface comes without a loss of efficiency.

Keywords: multi-party computation; garbling schemes; universal composability; fortified universal composability

1 Introduction

Secure Multi-Party Computation (MPC) allows mutually distrusting parties to jointly perform tasks on private data via a protocol π. Examples for such tasks include

- the secure evaluation of a function f, where protocol parties learn nothing but the result [GMW87],
- first-price sealed-bid auctions, where nothing but the winner and the highest bid is revealed or, for example,
- COVID contact tracing [Be21].

[1] Constreo Systems, dominik.doerner@student.kit.edu
[2] KASTEL, Karlsruhe Institute of Technology, jeremias.mechler@kit.edu
[3] KASTEL, Karlsruhe Institute of Technology, joern.mueller-quade@kit.edu

doi:10.18420/sicherheit2022_05

More generally, MPC can be used for every efficiently computable task while guaranteeing properties such as *correctness*, *privacy* or the *independence of inputs*. These guarantees hold even if some parties are corrupted and deviate from the protocol in an arbitrary manner.

For MPC, there is a long list of feasibility results (e.g. [GMW87; Ya86]) that give a protocol for *any* efficiently computable function (*general MPC*). To this end, the function-to-be-evaluated f is transformed into a circuit C that is jointly evaluated. For *secret-sharing* MPC (e.g. [Da12; GMW87]), this evaluation happens online and gate by gate, incurring a round complexity that is linear in the circuit's depth. For high performance, low latency between the protocol parties is desirable.

For MPC with *garbled circuits*, the circuit C is "garbled" into a garbled circuit C' by a garbler. After an interaction phase for distributing the inputs (which is independent of the circuit depth), C' can be evaluated by the evaluator without further interaction. While garbled circuit protocols are generally constant-round, the garbled circuit C' is usually much larger than C, leading to a high communication overhead. In this setting, a high bandwidth is more important than a low latency.

Common to virtually all solutions for general MPC is the high computational or communication overhead. In a setting where the parties participating in such a protocol have very limited resources, e.g. on mobile devices, this overhead is prohibitive. As such, MPC has not seen wide-spread adoption, despite its promising security guarantees.

A partial solution to the limited efficiency is to outsource the most expensive parts of an MPC to one or more *servers*, e.g. in the cloud. If they can be partially trusted, for example in the sense that some kind of corruption threshold is tolerated by the protocol, security in such a setting can be shown. In practice, the assumption of e.g. a honest majority of the serves is often justified by placing the servers at different (competing) cloud providers. In such an outsourcing setting, the resources of the clients' devices are mostly irrelevant. Assuming the use of different cloud providers, it may be plausible to assume high bandwidths together with latency that is higher than e.g. within the same data center. In the following, we thus consider MPC based on garbled circuits, which is suitable for this particular setting.

Depending on the protocol, cloud-based MPC may not require the clients to communicate with each other. This may be highly desirable, e.g. for security reasons.

Possible applications include private set intersection, e.g. for contact discovery with messengers on smartphones or privacy-preserving analytics on e.g. smart-meter data. Generally, a setting with a complex function to be evaluated and small in- and outputs is favored.

1.1 Related Work

Garbling-based multi-party computation. Garbling schemes have proven to be a valuable tool in implementing MPC protocols. Goyal et al. [GMS08] designed an efficient MPC protocol using the cut-and-choose principle while Ben-David et al. [BNP08] designed a system called FairplayMP, in which the function to be executed is compiled to a boolean circuit and executed jointly and securely against semi-honest adversaries.

Server-aided MPC. More practice-oriented variations include the works of Alam et al. [Al18], for server-aided privacy-aware access control, and Baldimtsi et al. [Ba17], for online queries using data of offline users in the use case of social networks. Another subclass of the research has focused on improving security by utilizing multiple servers, as used by Jakobsen et al. [JNO14], Kerschbaum [Ke15] and Bugiel et al. [Bu11]. Thirdly, many MPC protocols utilize the cut-and-choose principle to ensure that the circuit is garbled correctly. Notable example include the series of single-server-aided MPC protocols by Kamara et al. [KMR11] and the Salus system by Kamara et al. [KMR12].

MPC using secure hardware. Additional hardware assumptions provide a way to enable security under majority collusion. Some hardware assumptions, such as Trusted Execution Environments (TEE), provide great possibilities, but require the assumption of complete tamper-proof processors that can execute arbitrary functions. These have been used by Gupta [Gu16], Bahmani et al. [Ba16] and Benenson et al. [Be06]. Less restrictive assumptions include tamper-proof hardware tokens for cryptographic operations, as used by Katz [Ka07] or Dowsley et al. [DMN15].

Fortified Multi-Party Computation. In the established adaptive corruption model, an adversary may corrupt parties during any point of a protocol execution, being able to learn and modify a corrupted party's inputs and outputs. As such, even protocol parties that are initially honest are not afforded any kind of protection from the consequences of adaptive corruptions. This problem is addressed by Broadnax et al. [Br21], who introduce a new corruption model that distinguishes between *physical attacks* and *remote hacks*. By using simple remotely unhackable hardware modules like data diodes or air-gap switches, a protocol party can (temporarily) isolate itself, preventing an adversary to remotely hack it. By using an appropriate architecture, they construct a protocol where initially honest protocol parties are protected against the *consequences* of remote hacks: Unless all parties are corrupted or a party is corrupted at the very beginning of a protocol execution, the remote hacking of a party will *not* result in the adversary learning or being able to modify the hacked party's inputs or outputs. Being a theoretical feasibility result, their construction is not practically efficient.

1.2 Our Contribution

In previous solutions for server-aided MPC, the problem of server corruptions is usually addressed by defining a corruption threshold up to which security is guaranteed. However, no

technical measures are taken to justify such corruption thresholds. In practice, the problem is exacerbated by the fact that a server, presumably used for many MPC executions, may be an attractive target for hackers. It is thus very important to reduce the attack surface of such servers as much as possible.

Inspired by the work of Broadnax et al. [Br21], we investigate the use of simple remotely unhackable hardware modules like data diodes (which allow unidirectional communication only) or air-gap switches (which allow a party to temporarily disconnect itself) in the context of server-aided MPC in order to reduce the servers' attack surface. We present a server-aided general MPC protocol where the most expensive computations are outsourced. We split the servers into several parts with different responsibilities. By isolating parts of the servers from the network, they are not susceptible to remote hacks anymore. We prove the security of our construction in the Fortified UC framework [Br21].

To the best of our knowledge, we are the first to explore the use of isolation assumptions such as data diodes or air-gap switches in the field of server-aided MPC.

1.3 Guide for the Reader

In Sect. 2, we give a short introduction into the concepts used throughout the paper. In Sect. 3, we present our construction, along with a proof sketch together with a short theoretical evaluation of its performance. For more details, see the full version.

2 Preliminaries

In the following, we give a short and informal overview of important concepts and building blocks used in this paper. A detailed version can be found in in the full version.

Oblivious transfer (OT) [EGL85; Ra81] allows a receiver with an input $b \in \{0, 1\}$ to interact with a sender with inputs m_0 and m_1. At the end of the interaction, the receiver learns the value m_b, but nothing about m_{1-b}. Conversely, the sender does not learn the bit b. This notion of 1-out-of-2-OT can be extended, e.g. to the case of n-out-of-k-OT.

When combined with an OT protocol, a *garbling scheme* [BHR12; Ya86] allows two parties P_1, P_2 to jointly evaluate a circuit C on their respective private inputs x_1 and x_2. To this end, the *garbler* creates an obfuscated variant of C with its input x_1 hard-coded into the circuit. The *evaluator* P_2 can obtain "keys" for its input x_2 via OT from the garbler P_1. Having these keys, it can evaluate C on inputs x_1 and x_2 to learn the result $y = C(x_1, x_2)$. With respect to security, it is guaranteed that P_1 learns nothing about x_2 and P_1 learns nothing about x_1 that it cannot compute from y and x_2. Also, P_2 cannot evaluate the circuit repeatedly on different inputs.

Universal Composability [Ca01] is a security notion for MPC protocols in a setting where multiple (possibly adversarially chosen) protocols are executed concurrently. As an extension of the *real-ideal paradigm*, the security of a protocol is defined through an ideal functionality it (presumptively) realizes. *Fortified Universal Composability* (Fortified UC) [Br21] extends UC security to capture isolation properties of remotely unhackable hardware modules such as data diodes or air-gap switches and introduces a new corruption model that distinguishes between *physical attacks* and *remote hacks*.

3 Our Protocol

In the following, we will describe our MPC protocol, which is cast in the Fortified UC Framework [Br21]. We begin with a description of the protocol and its architecture, followed by the ideal functionality we realize and the formal protocol definition.

3.1 Protocol Architecture

Our protocol π_{MPC} allows a set of n players ($\mathcal{P}$), each with their own private input $x_i \in X$, to securely compute a globally known circuit $C : X^n \to Y^n$ and yet keep their inputs x_i and respective output $y_i \in Y$ private. Apart from the players, the parties include an initializer **I**, a server **S**, a hybrid functionality $\mathcal{F}_{\text{nOT}}$, a remotely unhackable module **M** and the environment $\mathcal{Z}$.

The complete architecture is depicted in Fig. 1. Using the conventions for graphical depictions as defined by Broadnax et al. [Br21], rounded boxes represent main parties while sub-parties are represented by sharp corners. A cloud shape is meant to represent hybrid functionalities, a gray background that the party is online during the protocol execution and bold borders that a (sub-)party is unhackable. Air-gap switches, represented with —⁄—, can be controlled by the party next to the hinge, allowing it to disconnect itself via this connection. Similarly, data diodes, represented with —▷|—, allow unidirectional communication only.

The initializer **I** is connected to the environment via an initially connected air-gap switch. To the other entities, it is connected via data diodes. After it has received initialization messages from the environment, the initializer can disconnect the air-gap switch. Then, it is no longer able to receive messages and thus cannot be hacked remotely. At the same time, it can still send protocol messages to the outside via its data diodes.

Connected to the initializer is a remotely unhackable module **M** that will act as a surrogate for **I** for any interaction with the other parties. As we will see later, this is necessary as this interaction cannot be performed by uni-directional messages from behind a data diode. Due to its simplicity, we assume that the module **M** can be implemented in a way that makes it immune to hacks, e.g. as a fixed-function circuit that can be verified for correctness.

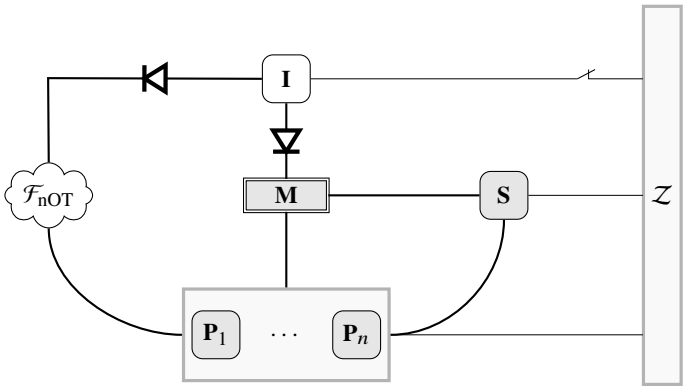


Fig. 1: Architecture of π_{MPC}

Finally, the server **S** interacts with the module **M** and the players $\mathcal{P}$. Due to the communication required in the protocol, we cannot protect it via data diodes or air-gap switches. Also, due to its complexity, we do not assume that it is remotely unhackable.

The protocol makes use of a single hybrid functionality $\mathcal{F}_{\text{nOT}}$ that interacts with **I** and the players $\mathcal{P}$:

Definition 3.1 (Ideal Functionality for $n \times OT_2^1$)**.** *The ideal functionality $\mathcal{F}_{\text{nOT}}$ for non-adaptive n-fold 1-out-of-2 l-bit oblivious transfer ($n \times OT_2^1$) interacts with a sender* **S***, a receiver* **R** *and an adversary $\mathcal{A}$.*

It is parameterized with the number of repetitions n as well as the word length l. These are fixed and known to all parties involved.

On receiving $(\texttt{sid}, \texttt{initiate}, m = ((m_1^0, m_1^1), \ldots, (m_n^0, m_n^1)))$ *from* **S***, with* $m_i^v \in \{0,1\}^l$*:*

- *If no message* $(\texttt{sid}, \texttt{initiate}, \cdot)$ *has yet been received, store* $(m_1, \ldots, m_n)$.
- *Send* $(\texttt{sid}, \texttt{initiate})$ *to* $\mathcal{A}$.

On receiving $(\texttt{sid}, \texttt{request}, b = (b_1, \ldots, b_n))$ *from* **R***, with* $b_i \in [0,1]$*:*

- *Send* $(\texttt{sid}, \texttt{output-request})$ *to* $\mathcal{A}$ *and wait for an answer* $(\texttt{sid}, \texttt{output-response}, \phi)$ *with* $\phi \in \{\texttt{pass}, \texttt{abort}\}$.
  - *if* $\phi = \texttt{pass}$*, continue.*
  - *if* $\phi = \texttt{abort}$*, abort.*
- *Check if a message* $(\texttt{sid}, \texttt{request}, \cdot)$ *was previously received and answered.*
  - *if not, continue.*
  - *if yes, abort.*
- *Check if a message* $(\texttt{sid}, \texttt{initiate}, \cdot)$ *was previously received.*
  - *if not, abort.*

  - *if yes, continue.*
- *Send* $(\texttt{sid}, \texttt{response}, m^* = (m_1^{b_1}, \ldots, m_n^{b_n}))$ *to* $\mathbf{R}$.

All main parties (namely $\mathbf{I}$, $\mathbf{S}$ and $\mathcal{P}$) are connected to the environment $\mathcal{Z}$, although $\mathbf{I}$ is connected only via an air-gap switch and will disconnect it upon first activation.

The only parties to receive any input from the environment are the players $\mathcal{P}$, each of which receives an input value $x_i \in X$.

As we will see later, the above architecture is suitable to provide security against adversaries that adhere to the following corruption model.

3.2 Corruption Model

We assume that the initializer $\mathbf{I}$ can only be corrupted semi-honestly, meaning that an adversary learns all information, but does not deviate from the protocol. We make this assumption for the following reasons: i) We envision that $\mathbf{I}$, $\mathbf{M}$ and $\mathbf{S}$ are placed in the cloud. Cloud providers are often assumed to be honest-but-curious. This is captured by semi-honest corruption. ii) As $\mathbf{I}$ is placed behind data diodes and an air-gap switch, the attack surface from the "outside" is greatly reduced, making remote hacks infeasible. Moreover, semi-honest corruption also captures e.g. side-channel attacks, which are still possible in this setting. They would only leak the internal state of the party and not allow for an active manipulation of the protocol flow.

Additionally, we assume that an adversary can only corrupt either $\mathbf{I}$ or $\mathbf{S}$, but not both at the same time. This assumption is practically motivated: Standard security practices would dictate to use different providers to host both parties, such as two different major cloud providers.

The server $\mathbf{S}$ as well as the players are subject to adaptive byzantine corruptions throughout the whole protocol execution.

3.3 Ideal Functionality

The ideal functionality $\mathcal{F}_{\text{MPC}}$ for secure multi-party computation interacts with an initializer $\mathbf{I}$, a server $\mathbf{S}$, a given set of players $\mathcal{P} := \{\mathbf{P}_1, \ldots, \mathbf{P}_n\}$ and an adversary $\mathcal{A}$.

It is parameterized with a circuit $C : (\{0,1\}^l)^n \to (\{0,1\}^l)^n$ to execute as well as the word length l. These are fixed and known to all parties involved.

Every player $\mathbf{P}_i$ holds an input $x_i \in \{0,1\}^l$.

Phase 1: Input
On receiving $(\texttt{sid}, \texttt{input}, x_i)$ from $\mathbf{P}_i \in \mathcal{P}$, with $x_i \in \{0,1\}^l$:

- If no message $(\texttt{sid}, \texttt{input}, \cdot)$ has yet been received from $\mathbf{P}_i$, store x_i and ignore all further `input` messages from P_i.
- Send $(\texttt{sid}, \texttt{input-received}, \mathbf{P}_i)$ to $\mathcal{A}$.
- Once a message $(\texttt{sid}, \texttt{input}, \cdot)$ has been received from every player $\mathbf{P}_j \in \mathcal{P}$, mark this phase as complete.

Phase 2: Calculation

- Store $(y_1, \ldots, y_n) := C(x_1, \ldots, x_n)$ and mark this phase as complete.

Phase 3: Output

- For every player $\mathbf{P}_j \in \mathcal{P}$, send $(\texttt{sid}, \texttt{output-request}, \mathbf{P}_i)$ to $\mathcal{A}$ and wait for an answer $(\texttt{sid}, \texttt{output-response}, \phi)$ with $\phi \in \{\texttt{pass}, \texttt{abort}\}$. If $\phi = \texttt{pass}$, send $(\texttt{sid}, \texttt{output}, y_i)$ to $\mathbf{P}_i$. If $\phi = \texttt{abort}$, send $(\texttt{sid}, \texttt{output}, \texttt{abort})$ to $\mathbf{P}_i$.

Corruption

- Upon corruption of a player $\mathbf{P}_i$, send its input x_i to the adversary (if existent) and let it replace it with an input x_i'.
- Upon corruption of the server $\mathbf{S}$, do nothing.
- Upon corruption of the initializer $\mathbf{I}$, do nothing.

3.4 Protocol

In the following, we present our construction.

Roughly, the protocol works as follows: First, the initializer $\mathbf{I}$ creates a garbling of the circuit C and distributes the input labels to the players via the ideal functionality for oblivious transfer $\mathcal{F}_{\text{nOT}}$. The garbled circuit and the output labels are given to the module $\mathbf{M}$, which sends the garbled circuit to the server $\mathbf{S}$. According to their inputs, the players will obtain their input labels via oblivious transfer and forward them to the server. With these labels, $\mathbf{S}$ can evaluate the garbled circuit, leading to an encrypted and authenticated result that it distributes to the parties. Using the output labels provided by $\mathbf{M}$, a player $\mathbf{P}_i$ can reconstruct its output y_i.

Unless an adversary controls both the initializer and the server, this implies the following (informal) security guarantees:

- Privacy: For an honest player $\mathbf{P}_i$, the adversary is unable to learn its input or output. If the server is corrupted, the adversary only sees a player's input labels, but does not know the corresponding input. Conversely, the output is encrypted. If the initializer is corrupted, the adversary learns all input and output labels. However, as we assume that the server is honest in this case, it does not learn which labels are actually used.

- Integrity: The adversary cannot modify the computation's result. To this end, it would need private information held by the initializer together with the ability to modify the server's messages to the parties. As the adversary can only corrupt the server or the initializer, this is not possible.
- Independence of Inputs: In order to choose the input of a corrupted player $\mathbf{P}_j$ in dependence of an honest player $\mathbf{P}_i$, the adversary would have to perform the OT of $\mathbf{P}_j$ with choice bits depending on the choice bits of $\mathbf{P}_i$. By the definition of $\mathcal{F}_{\text{nOT}}$, this is not possible. Alternatively, even if the initializer is corrupted, the adversary will not learn the choice bits of $\mathbf{P}_i$ in the OT, making it impossible to use the corresponding input labels for $\mathbf{P}_j$ (which are then known to the adversary for all possible inputs of $\mathbf{P}_j$).
- No Repeated Circuit Evaluations: In order to evaluate the circuit multiple times in the case of a corrupted server (and one or more corrupted players), the adversary needs to know additional input labels. Due to the security of the OT and the fact that the initializer must not be corrupted together with the server, this is not possible.

These only very informal security guarantees are implied by the ideal functionality $\mathcal{F}_{\text{MPC}}$, which is provably realized by our construction.

In the following, we give the formal protocol description.

Protocol 1 π_{MPC}, a secure server-aided multi-party computation protocol

---

Parties. Initializer $\mathbf{I}$, secure module $\mathbf{M}$, server $\mathbf{S}$ and set of players $\mathcal{P} := \{\mathbf{P}_1, \ldots, \mathbf{P}_n\}$.

Parameterization. The protocol is parameterized with a session identifier `sid`, a circuit $C : (\{0,1\}^l)^n \to (\{0,1\}^l)^n$, a security parameter κ, an input length l and a projective, output-projective and output-key-secure garbling scheme[4] $\mathcal{G} = (\mathtt{Gb}_{\mathcal{G}}, \mathtt{En}_{\mathcal{G}}, \mathtt{Ev}_{\mathcal{G}}, \mathtt{De}_{\mathcal{G}})$. In the following, we use $\vec{x}$ to abbreviate $x_1, \ldots, x_n$.

Inputs. Every $\mathbf{P}_i \in \mathcal{P}$ holds an input $x_i \in \{0,1\}^l$. The other parties hold no initial input.

Hybrid functionality. The protocol makes use of the ideal functionality $\mathcal{F}_{\text{nOT}}$ for n-fold oblivious transfer (see Definition 3.1).

Channels. $\mathbf{I}$, $\mathbf{S}$ and all $\mathbf{P}_i \in \mathcal{P}$ are connected to $\mathcal{Z}$ via air-gap switches that are initially connected. $\mathbf{I}$ is connected to $\mathbf{M}$ via a data diode. Every player $\mathbf{P}_i \in \mathcal{P}$ is connected via a (secure) standard channel to both $\mathbf{M}$ and $\mathbf{S}$.

---

[4] A garbling scheme $\mathcal{G}$ is called *projective* if the encoding function consists of $2nm$ input keys (also called "wire labels"), where n is the number of input values and m is the number of bits in each input value.

We call a garbling scheme *output-projective* if the decoding information consists of 2 wire labels for each output bit in each output value, one corresponding to each possible value of that bit.

Informally, output-key secure (OKS) garbling schemes, introduced by Jafargholi et al. [JSW17], maintain their security properties even if the adversary is provided with an unordered set of output keys, i.e. the possible labels that can be produced by $\mathtt{Ev}_{\mathcal{G}}(\cdot)$ for each output bit, but without knowledge which bit value they each encode.

Protocol Phase 1: Garbling

- **MGarble**(C), executed by **I** on first activation
  1. Close the air-gap switch to $\mathcal{Z}$.
  2. Set $(\tilde{C}, e, d) \leftarrow \mathtt{Gb}_{\mathcal{G}}(1^{\kappa}, C)$.
  3. Parse $e =: \{L^{v}_{in,i,j}\}_{i\in[n],j\in[l],v\in\{0,1\}}$ and $d =: \{L^{v}_{out,i,j}\}_{i\in[n],j\in[l],v\in\{0,1\}}$.
  4. For each player $\mathbf{P}_i \in \mathcal{P}$:
     a) Initialize a new instance of $\mathcal{F}_{\text{nOT}}$ with session identifier $\mathtt{sid}_{ot,i}$.
     b) Send $(\mathtt{sid}_{ot,i}, \mathtt{initiate}, L_{in,i} = \big((L^{0}_{in,i,1}, L^{1}_{in,i,1}), \ldots, (L^{0}_{in,i,l}, L^{1}_{in,i,l})\big))$ to $\mathcal{F}_{\text{nOT}}$.
  5. Send $(\mathtt{init}, \tilde{C}, \overrightarrow{\mathtt{sid}}_{ot}, \vec{L}_{out})$ to **M**.

Protocol Phase 2: Distribution

- **MDistReq**$(\tilde{C}, \overrightarrow{\mathtt{sid}}_{ot}, \vec{L}_{out})$, executed by **M** after receiving a message $(\mathtt{init}, \tilde{C}, \overrightarrow{\mathtt{sid}}_{ot}, \vec{L}_{out})$ from **I**
  1. Send $(\mathtt{circuit}, \tilde{C})$ to **S**.
  2. Save $\vec{L}_{out}$.
  3. For each player $\mathbf{P}_i \in \mathcal{P}$, send $(\mathtt{labels}, \mathtt{sid}_{ot,i})$ to $\mathbf{P}_i$.
- **MDistServer**$(\tilde{C})$, executed by **S** after receiving a message $(\mathtt{circuit}, \tilde{C})$ from **M**
  1. Save $\tilde{C}$.
- **MDistPlayer**$(x_i, \mathtt{sid}_{ot,i})$, executed by $\mathbf{P}_i$ after receiving a message $(\mathtt{labels}, \mathtt{sid}_{ot,i})$ from **M** and an input (x_i) from $\mathcal{Z}$
  1. Send $(\mathtt{sid}_{ot,i}, \mathtt{request}, x_i)$ to $\mathcal{F}_{\text{nOT}}$ and receive $\tilde{x}_i$.
  2. Send $(\mathtt{input}, \tilde{x}_i)$ to **S**.

Protocol Phase 3: Computation

- **MCalcReq**$(\mathbf{P}_i, \tilde{x}_i)$, executed by **S** after receiving a message $(\mathtt{input}, \tilde{x}_i)$ from $\mathbf{P}_i$.
  1. Save $(\mathbf{P}_i, \tilde{x}_i)$.
  2. If an entry has been saved for every $\mathbf{P}_j \in \mathcal{P}$:
     a) Set $\vec{\tilde{y}} = \mathtt{Ev}_{\mathcal{G}}(\tilde{C}, \vec{\tilde{x}} = (\tilde{x_1}, \ldots, \tilde{x_{|\mathcal{P}|}}))$.
     b) If the execution of $\mathtt{Ev}_{\mathcal{G}}$ aborts, send $(\mathtt{output}, \mathtt{abort})$ to every player $\mathbf{P}_i \in \mathcal{P}$
     c) Else, send $(\mathtt{output}, \tilde{y}_i)$ to every player $\mathbf{P}_i \in \mathcal{P}$.
- **MCalcRes**$(\tilde{y}_i)$, executed by $\mathbf{P}_i$ after receiving a message $(\mathtt{output}, \tilde{y}_i)$ from **S**
  1. Save $\tilde{y}_i$.
  2. Send $(\mathtt{decode\text{-}request})$ to **M**.

Protocol Phase 4: Output Decoding

- **MDecSend**$(\vec{L}_{out}, \mathbf{P}_i)$, executed by **M** after receiving a message $(\mathtt{decode\text{-}request})$ from $\mathbf{P}_i \in \mathcal{P}$.
  1. Save $(\mathbf{P}_i, \mathtt{decode})$.
  2. If an entry has been saved for every player $\mathbf{P}_j \in \mathcal{P}$:

a) Send (`decode-response`, $L_{out,i}$) to every player $\mathbf{P}_i \in \mathcal{P}$.

- **MDecEx**($\tilde{y}_i, L_{out,i}$), executed by $\mathbf{P}_i$ after receiving a message (`decode-response`, $L_{out,i}$) from $\mathbf{M}$.
  1. Set $y_i := \mathtt{De}_{\mathcal{G}}(L_{out,i}, i, \tilde{y}_i)$.
  2. Return y_i.

We can now state our main theorem:

Theorem 3.1. *The protocol π_{MPC} Fortified-UC-realizes the ideal functionality $\mathcal{F}_{\text{MPC}}$ in the $\mathcal{F}_{\text{nOT}}$-hybrid model for adversaries adhering to the corruption model of Sect. 3.2.*

3.5 Proof Sketch

In the following, we will give a short proof sketch for Theorem 3.1. The complete proof can be found in the full version.

In order to show Theorem 3.1, we have to show the indistinguishability of a real execution of π_{MPC} and the execution of $\mathcal{F}_{\text{MPC}}$ with a simulator that "simulates" the execution of π_{MPC}. To this end, the simulator must provide an *interactive distinguisher* (the environment) with protocol messages that are indistinguishable from messages of a real execution. The environment may influence the execution, e.g. by (adaptively) giving inputs to parties, learning the outputs of honest parties and by communicating with the adversary.

The architecture has been specifically designed to allow simulation:

If the initializer is corrupted, it must follow the protocol flow and expose its secrets to the simulator by passing them to $\mathbf{M}$ (which is simulated by the simulator). Since $\mathbf{I}$ and $\mathbf{S}$ cannot be corrupted simultaneously according to our corruption model, the adversary cannot decrypt (fake) values sent by the simulator or check their validity. With at least one of both being controlled by the simulator, the environment has not enough information to verify the circuit execution.

If any player is corrupted, they need to garble their inputs before passing them to the server $\mathbf{S}$. As such, the simulator can learn their input from $\mathcal{F}_{\text{nOT}}$, as players would only be able to garble their inputs themselves if $\mathbf{I}$ is corrupted as well. Following the reasoning above, the simulator would learn the input labels from $\mathbf{M}$ and can then decrypt the input given to the simulated $\mathbf{S}$.

If the simulator has committed itself to encrypted random inputs for honest players, which the environment then learns by adaptively corrupting $\mathbf{S}$, then the simulator is still able to decrypt the encrypted circuit output to the correct output values by manipulating the output labels.

We now state the simulator for one considered corruption pattern. The other corruption cases are handled similarly. For the sake of simplicity, we consider static corruptions only.

Definition 3.2 (Simulator for corrupted players and a corrupted server)**.** *The simulator $\mathcal{S}$ works as follows:*

1. *Set* $(\tilde{C}, \vec{L}_{in}, \vec{L}_{out}) \leftarrow \mathtt{Gb}_{\mathcal{G}}(1^{\kappa}, C)$.
2. *Send the message* $(\mathtt{circuit}, \tilde{C})$ *to* $\mathbf{S}^*$.
3. *For each corrupted* $\mathbf{P}_i^* \in \mathcal{P}^*$*:*
   a) *Create a new dummy instance of* $\mathcal{F}_{\text{nOT}}$ *with session identifier* $\mathtt{sid}_{ot,i}$.
   b) *Send the message* $(\mathtt{labels}, \mathtt{sid}_{ot,i})$ *to* $\mathbf{P}_i^*$.
4. *When receiving a message* $(\mathtt{sid}, \mathtt{input\text{-}received}, \mathbf{P}_i)$ *from* $\mathcal{F}_{\text{MPC}}$ *for an honest* $\mathbf{P}_i$*:*
   a) *Choose* $\dot{x}_i$ *at random.*
   b) *Set* $\tilde{\dot{x}}_i \leftarrow \mathtt{En}_{\mathcal{G}}(L_{in,i}, i, \dot{x}_i)$.
   c) *Send the message* $(\mathtt{input}, \tilde{\dot{x}}_i)$ *to* $\mathbf{S}^*$.
5. *When a corrupted* $\mathbf{P}_i^*$ *sends a message* $(\mathtt{sid}_{ot,i}, \mathtt{request}, x_i)$ *meant for* $\mathcal{F}_{\text{nOT}}$*:*
   a) *Set* $\tilde{x}_i \leftarrow \mathtt{En}_{\mathcal{G}}(L_{in,i}, i, x_i)$.
   b) *Have* $\mathcal{F}_{\text{nOT}}$ *return* $(\mathtt{sid}_{ot,i}, \mathtt{response}, \tilde{x}_i)$ *to* $\mathbf{P}_i^*$.
6. *When receiving the first message* $(\mathtt{output}, \cdot)$ *from* $\mathbf{S}^*$ *meant for an honest player*
   a) *For every corrupted player* $\mathbf{P}_i^* \in \mathcal{P}^*$
      i. *Let* x_i *be the input that* $\mathbf{P}_i^*$ *has sent to* $\mathcal{F}_{\text{nOT}}$*. Set it randomly* ($x_i \xleftarrow{R} X$) *if* $\mathbf{P}_i^*$ *has not interacted with* $\mathcal{F}_{\text{nOT}}$.
      ii. *Send the message* $(\mathtt{sid}, \mathtt{input}, x_i)$ *as* $\mathbf{P}_i^*$ *to* $\mathcal{F}_{\text{MPC}}$.
7. *When* $\mathcal{F}_{\text{MPC}}$ *sends a message* $(\mathtt{sid}, \mathtt{output\text{-}request}, \mathbf{P}_i)$ *to an honest player* $\mathbf{P}_i \notin \mathcal{P}^*$*, delay sending a response until later on. Let all* `output-requests` *to corrupted players pass.*
8. *When* $\mathbf{S}^*$ *sends a message* $(\mathtt{output}, \phi)$ *to an honest player* $\mathbf{P}_i$
   a) *If* $\phi = \mathtt{abort}$*, send* $(\mathtt{sid}, \mathtt{output\text{-}response}, \mathtt{abort})$ *to* $\mathcal{F}_{\text{MPC}}$.
   b) *If* ϕ *is an encoded value* $\tilde{y}_i^{*\mathtt{server}}$*, mark that a* `decode-request`*-message was received from* $\mathbf{P}_i$.
9. *When a* `decode-request`*-message was received from every honest and corrupted player in* $\mathcal{P}$
   a) *Evaluate* $y^{\mathtt{sim}} = C(x)$*, with* x *being the input of the corrupted players as it was given to* $\mathcal{F}_{\text{MPC}}$ *and the input of the honest players as it was given (in an encoded form) to* $\mathbf{S}^*$.
   b) *For every corrupted player* $\mathbf{P}_i^*$
      i. *Wait for a message* $(\mathtt{sid}, \mathtt{output}, y_i^{\mathtt{ideal}})$ *from* $\mathcal{F}_{\text{MPC}}$ *meant for* $\mathbf{P}_i^*$.
      ii. *Create a copy* $\bar{L}_{out,i}$ *of* $L_{out,i}$ *as follows:*
         - *For each* $j \in [l]$
           - *If* $y_{i,j}^{\mathtt{sim}} = y_{i,j}^{\mathtt{ideal}}$*, set* $\bar{L}_{out,i,j}^{v} := L_{out,i,j}^{v}, v \in [0,1]$.
           - *Else, set* $\bar{L}_{out,i,j}^{v} := L_{out,i,j}^{1-v}, v \in [0,1]$.
      iii. *Send* $(\mathtt{decode\text{-}response}, \bar{L}_{out,i})$ *to* $\mathbf{P}_i^*$.
   c) *For every honest player* $\mathbf{P}_i^*$
      - *If* $\tilde{y}_i^{*\mathtt{server}}$ *can be decoded successfully, send the response* $(\mathtt{sid}, \mathtt{output\text{-}response}, \mathtt{pass})$ *to* $\mathcal{F}_{\text{MPC}}$ *to let the output pass to (this single)* $\mathbf{P}_i$.
      - *Else, send the response* $(\mathtt{sid}, \mathtt{output\text{-}response}, \mathtt{abort})$ *to* $\mathcal{F}_{\text{MPC}}$.

We will now briefly argue why the environment's view is computationally indistinguishable between an execution of π_{MPC} and a real-world adversary and the execution of $\mathcal{F}_{\text{MPC}}$ and the simulator of Definition 3.2 for the case of corrupted players and a corrupted server.

To this end, we define a series of games, starting with the real execution. Finally, we reach the ideal execution through indistinguishable changes.

Game 1 We now consider an execution with an ideal functionality $\langle \mathcal{F}_{\text{MPC}} \rangle$ that is identical to $\mathcal{F}_{\text{MPC}}$, but also tells the simulator all inputs and lets it determine all outputs. In this execution, the simulator performs the protocol for the honest parties, using the inputs provided by the functionality and making outputs through it. Also, $\mathcal{F}_{\text{nOT}}$ is simulated honestly. As the changes are only syntactical and oblivious for the environment, it is easy to see that the environment's view is identically distributed.

Game 2 In Game 2, the simulator initializes an internal vector \$INPUT that saves the inputs x_i of all players. Initially set all inputs to random values. If a corrupted player $\mathbf{P}_i$ sends an input to $\mathcal{F}_{\text{nOT}}$, replace the saved input with the given value.

When $\mathbf{S}^*$ first sends out a `output`-message to any (honest or corrupted) player, send $(\texttt{sid}, \texttt{input}, x_i)$ to $\langle \mathcal{F}_{\text{MPC}} \rangle$ for all players with $x_i = \$\text{INPUT}[i]$ for every $\mathbf{P}_i$.

Replace the output labels $L_{out,i}$ in the message $(\texttt{decode-response}, L_{out,i})$ with a copy $\tilde{L}_{out,i}$, where the output labels for the same bit $(L^0_{out,i,j}, L^1_{out,i,j})$ are conditionally switched so that the garbled output produced by $\texttt{Ev}_{\mathcal{G}}(\tilde{C}, \texttt{En}_{\mathcal{G}}(L_{in}, \$\text{INPUT}))$ decodes to the output returned by $\langle \mathcal{F}_{\text{MPC}} \rangle$.

It is obvious that when \$INPUT holds the correct server input of every player it follows that the result produced by the ideal functionality is identical to the one to be expected from the server and thus no changes to the output labels will be made. Note that the actual behavior of the corrupted server has no effect on this change. Thus, the games are indistinguishable.

Game 3 Let us assume that the simulator does not know all inputs. We define a changed game where an honest player $\mathbf{P}_i$ is chosen at random by the simulator. $\mathbf{P}_i$ is mostly simulated honestly, but will choose a (random) input $\dot{x}_i \neq x_i$ such that the resulting $\dot{y}_i := \texttt{Ev}_{\mathcal{G}}(\dot{x}_i, \cdot) \neq \texttt{Ev}_{\mathcal{G}}(x_i, \cdot)$. $\dot{x}_i$ is given to $\mathcal{F}_{\text{nOT}}$ and (in an encoded form) to $\mathbf{S}^*$ and thus saved in `$INPUT`. The original input x_i is still saved and sent in a message $(\texttt{sid}, \texttt{input}, x_i)$ to $\langle \mathcal{F}_{\text{MPC}} \rangle$. This results in at least one row of $L_{out,i}$ to be rearranged due to the simulator instructions as to preserve the correctness of the player outputs.

If a player in Game 3 were to send different inputs to the hybrid functionality $\mathcal{F}_{\text{nOT}}$ and the ideal functionality $\langle \mathcal{F}_{\text{MPC}} \rangle$, then the resulting game is still indistinguishable from Game 2 under the output-key secure obliviousness of $\mathcal{G}$.

Let us assume that there exists an environment $\mathcal{Z}$ that can distinguish between Games 2 and 3, thereby determining for at least one tuple $(L^{v}_{out,i,j}, i, j, w)$ that $v \neq w$. Such an environment $\mathcal{Z}$ can be used to construct a new adversary $\mathcal{B}$ that can break the output-key secure obliviousness of $\mathcal{G}$. Due to the security of $\mathcal{G}$, we thus conclude that Game 2 and Game 3 are indistinguishable.

Game 4 We now replace replace all (simulated) honest players in $\mathcal{P}\backslash\mathcal{P}^*$ with their ideal world counterparts as in Definition 3.2.

When $\langle\mathcal{F}_{\text{MPC}}\rangle$ sends an `input-received`-message to the simulator informing about the input of an honest player $\mathbf{P}_i \notin \mathcal{P}^*$, send the message $(\texttt{input}, \tilde{x}_i)$ to $\mathbf{S}^*$, where $\tilde{x}_i$ is the encoding of a randomly chosen input x_i.

When $\mathbf{S}^*$ sends an `abort` to an honest $\mathbf{P}_i$, respond with `abort` to any `output-request` by $\langle\mathcal{F}_{\text{MPC}}\rangle$ for $\mathbf{P}_i$. If $\mathbf{S}^*$ instead sends an output $\tilde{y}^*_i$ to $\mathbf{P}_i$, mark that a `decode-request`-message from $\mathbf{P}_i$ was received.

After sending out `decode-response`-messages, try to decode the result $\tilde{y}^*_i$ received from the server for any honest $\mathbf{P}_i$ and respond with `pass` to any `output-request` by $\langle\mathcal{F}_{\text{MPC}}\rangle$ for $\mathbf{P}_i$ if that is the case and `abort` otherwise.

Game 4 is indistinguishable from Game 3 for any environment $\mathcal{Z}$ under the output-key secure obliviousness, authenticity and the output-projectiveness of $\mathcal{G}$.

Game 5 Replace the ideal functionality $\langle\mathcal{F}_{\text{MPC}}\rangle$ with the ideal functionality $\mathcal{F}_{\text{MPC}}$. Game 5 is perfectly indistinguishable from Game 4 for any environment $\mathcal{Z}$.

We remark that $\mathcal{S}$ in the latest adaptation is no longer dependent on the input or output of any party. Since these values are not used in the simulator's instructions and both ideal functionalities are identical apart from that, this change is naturally perfectly indistinguishable.

Note that the protocol described in Game 5 is identical to our simulator instructions. The indistinguishability of the presented games proves our claim.

For the complete proof, see the full version.

3.6 Efficiency

Regarding the efficiency of the protocol we have multiple aspects of complexity that we can differentiate. In the following, we will present a summary of our findings regarding the communication cost, total information flow and computational complexity of the protocol. The full efficiency analysis can be found in the full version.

The communication cost of the protocol, with which we denote the total number of messages sent between protocol participants, with n being the number of players, is $2 + 5n$ for the execution of π_{MPC}. As such, the communication cost is linear in the number of players.

The information flow is our designation for the total number of bits transferred between parties and is more difficult to determine than the communication cost as it depends on details of the garbling scheme and the circuit to be evaluated. Using variables for the number of input bits per player l, the maximum size of input and output keys m and the information size of the garbled circuit $\tilde{c}$, we can determine an upper bound of $\Theta(\tilde{c} + n \cdot (\log(n) + ml))$ bit for the execution of π_{MPC} alone, which is quasi-linear in the number of players n.

The computation complexity heavily depends not only on the implementation of the protocol and its schemes but also the size of the used circuit and the underlying hardware. We note that only three sub-tasks incur noteworthy computation efforts, namely the garbling of the circuit, the execution of $\mathcal{F}_{\text{nOT}}$ and the evaluation of the garbled circuit. It can be seen that our protocol has the advantage that all players are burdened identically and have outsourced both the garbling and execution to other participants, namely **S** and **I**. Furthermore, even **I** can realistically be presented by low-performance hardware, since the initial garbling can be pre-processed and stored securely before initiation of the protocol with concrete players if the circuit is known beforehand.

4 Conclusion

While many solutions for server-aided MPC exist, they do not aim to protect the servers from remote hacks performed by an adversary.

In this paper, we take a first step towards addressing this problem: We have presented a composable general MPC protocol in the setting of server-aided MPC. By modularizing the servers and using remotely unhackable hardware modules such as data diodes and air-gap switches, we can greatly reduce the attack surface.

The resulting protocol is plausibly efficient and provides security going beyond the state of the art.

Acknowledgements

We woud like to thank Lukas Beeck, Markus Raiber and Rebecca Schwerdt for helpful discussions on a related but unpublished project.

Jeremias Mechler, Jörn Müller-Quade: This work was supported by funding from the topic Engineering Secure Systems of the Helmholtz Association (HGF) and by KASTEL Security Research Labs.

References

[Al18] Alam, M.; Emmanuel, N.; Khan, T.; Khan, A.; Javaid, N.; Choo, K. K. R.; Buyya, R.: Secure policy execution using reusable garbled circuit in the cloud. Futur. Gener. Comput. Syst. 87/, pp. 488–501, 2018.

[Ba16] Bahmani, R.; Barbosa, M.; Brasser, F.; Portela, B.; Sadeghi, A.-r.: Secure Multiparty Computation from SGX, tech. rep., Cryptology ePrint Archive, 2016.

[Ba17] Baldimtsi, F.; Papadopoulos, D.; Papadopoulos, S.; Scafuro, A.; Triandopoulos, N.: Server-Aided Secure Computation with Off-line Parties. ESORICS Part 1/LNCS 10492, pp. 103–123, 2017, arXiv: 9780201398298.

[Be06] Benenson, Z.; Fort, M.; Freiling, F.; Kesdogan, D.; Penso, L. D.: TrustedPals: Secure Multiparty Computation Implemented with Smart Cards. Eur. Symp. Res. Comput. Secur./, pp. 34–48, 2006.

[Be21] Beskorovajnov, W.; Dörre, F.; Hartung, G.; Koch, A.; Müller-Quade, J.; Strufe, T.: ConTra Corona: Contact Tracing against the Coronavirus by Bridging the Centralized-Decentralized Divide for Stronger Privacy. In (Tibouchi, M.; Wang, H., eds.): Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II. Vol. 13091. Lecture Notes in Computer Science, Springer, pp. 665–695, 2021.

[BHR12] Bellare, M.; Hoang, V. T.; Rogaway, P.: Foundations of garbled circuits. Proc. ACM Conf. Comput. Commun. Secur./, pp. 784–796, 2012.

[BNP08] Ben-David, A.; Nisan, N.; Pinkast, B.: FairplayMP - A system for secure multiparty computation. Proc. ACM Conf. Comput. Commun. Secur./, pp. 257–266, 2008.

[Br21] Broadnax, B.; Koch, A.; Mechler, J.; Müller, T.; Müller-Quade, J.; Nagel, M.: Fortified Multi-Party Computation: Taking Advantage of Simple Secure Hardware Modules. Proc. Priv. Enhancing Technol. 2021/4, pp. 312–338, 2021.

[Bu11] Bugiel, S.; Stefan, N.; Sadeghi, A.-r.; Schneider, T.: Twin Clouds : Secure Cloud Computing with Low Latency./, pp. 32–44, 2011.

[Ca01] Canetti, R.: Universally Composable Security: a New Paradigm for Cryptographic Protocols. In: 42nd Found. Comput. Sci. Conf. Vol. 16, 2001.

[Da12] Damgård, I.; Pastro, V.; Smart, N. P.; Zakarias, S.: Multiparty Computation from Somewhat Homomorphic Encryption. In (Safavi-Naini, R.; Canetti, R., eds.): Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings. Vol. 7417. Lecture Notes in Computer Science, Springer, pp. 643–662, 2012.

[DMN15] Dowsley, R.; Müller-Quade, J.; Nilges, T.: Weakening the Isolation Assumption of Tamper-proof Hardware Tokens./1, 2015, arXiv: 1502.03487.

[EGL85] Even, S.; Goldreich, O.; Lempel, A.: A Randomized Protocol for Signing Contracts. Commun. ACM 28/6, pp. 637–647, 1985, arXiv: arXiv:1011.1669v3.

[GMS08] Goyal, V.; Mohassel, P.; Smith, A.: Efficient two party and multi party computation against covert adversaries. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 4965 LNCS/, pp. 289–306, 2008.

[GMW87] Goldreich, O.; Micali, S.; Wigderson, A.: How To Play Any Mental Game OR A Completeness Theorem for Protocols with Honest Majority. In: Proc. ACM STOC. Pp. 218–229, 1987, ISBN: 0897912217.

[Gu16] Gupta, D.: Practical and Deployable Secure Multi-Party Computation, Yale University, 2016.

[JNO14] Jakobsen, T. P.; Nielsen, J. B.; Orlandi, C.: A Framework for Outsourcing of Secure Computation. Proc. 6th Ed. ACMWorkshop Cloud Comput. Secur. CCSW '14/, pp. 81–92, 2014.

[JSW17] Jafargholi, Z.; Scafuro, A.; Wichs, D.: Adaptively indistinguishable garbled circuits. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 10678 LNCS/, pp. 40–71, 2017.

[Ka07] Katz, J.: Universally composable multi-party computation using tamper-proof hardware. EUROCRYPT 2007 4515/, pp. 115–128, 2007.

[Ke15] Kerschbaum, F.: Oblivious outsourcing of garbled circuit generation. Proc. ACM Symp. Appl. Comput. 13-17-Apri/, pp. 2134–2140, 2015.

[KMR11] Kamara, S.; Mohassel, P.; Raykova, M.: Outsourcing Multi-Party Computation./, 2011.

[KMR12] Kamara, S.; Mohassel, P.; Riva, B.: Salus : A System for Server-Aided Secure Function Evaluation./, 2012.

[Ra81] Rabin, M. O.: How To Exchange Secrets with Oblivious Transfer, 1981.

[Ya86] Yao, A. C. C.: How To Generate and Exchange Secrets. Annu. Symp. Found. Comput. Sci./1, pp. 162–167, 1986.

PrivacyDates: A Framework for More Privacy-Preserving Timestamp Data Types

Christian Burkert[1], Jonathan Balack[2], Hannes Federrath[3]

Abstract: Case studies of application software data models indicate that timestamps are excessively used in connection with user activity. This contradicts the principle of data minimisation which demands a limitation to data necessary for a given purpose. Prior work has also identified common purposes of timestamps that can be realised by more privacy-preserving alternatives like counters and dates with purpose-oriented precision. In this paper, we follow up by demonstrating the real-world applicability of those alternatives. We design and implement three timestamp alternatives for the popular web development framework Django and evaluate their practicality by replacing conventional timestamps in the project management application Taiga. We find that our alternatives could be adopted without impairing the functionality of Taiga.

Keywords: Privacy by design; data minimisation; timestamps

1 Introduction

The design of software is today probably one of the biggest factors for everyday privacy. Since using software becomes virtually inescapable, it is increasingly application data modelling that decides how much of our personality and about our activities is recorded. Previous work [BF19] indicates that data models make excessive use of timestamps, the data type that adds the particularly sensitive temporal dimension to profiling. Timestamps have been previously observed to fulfil various functions in programming that not even require temporal properties. Instead, timestamps are frequently used for ordering or determining state (e. g., maintain order in which attachments were added). Function that can easily be achieved with less privacy-invasive alternatives. But also in cases where their temporal functions like universal comparability are actually used (e. g., time a bug report was filed), there appears to be room for a reduction of the typical second or even microsecond precision, to precisions that correspond more with human perception and increase privacy. Tackling the excessive use of timestamps in data models is a matter of raising awareness but also of providing ready to use alternatives. In this paper, we provide and evaluate a first such framework of timestamp alternatives. In summary, we make the following contributions:

- We design more privacy-preserving alternatives for common use cases of timestamp data types as identified by prior work.

[1] Universität Hamburg, christian.burkert@uni-hamburg.de
[2] Universität Hamburg, jonathan.balack@informatik.uni-hamburg.de
[3] Universität Hamburg, hannes.federrath@uni-hamburg.de

doi:10.18420/sicherheit2022_06

- We validate the design applicability with a case study of the application *Taiga*.
- We provide an implementation for the popular web application framework Django.
- We evaluate and demonstrate the practicality of those alternatives by replacing timestamps in data model of Taiga with our alternatives and observe the effects.

The remainder of the paper is structured as follows: We firstly present related work and our adversary model, then we describe the design and implementation of our alternatives, after which we provide an evaluation.

2 Related Work

In a prior case study of the Mattermost application, we systematically analysed the usage of personally identifiable timestamps in data models [BF19]. We found that timestamps of creation, last modification and deletion are included in a majority of models. However, most user-related timestamps were found to have no programmatic use and only a small fraction is displayed on the user interface. Based on the identified functions of timestamps, we proposed design alternatives that use precision reduction and context-aware counters. Otherwise, the literature on timestamp-related privacy patterns and practical data minimisation is scarce. In 2017, a literature survey of privacy patterns by Lenhard et al. [LFH17] showed that proposals are rarely verified or even implemented. Strategies to reduce the sensitivity of timestamps have been proposed by Zhang et al. [ZBY06] for log sanitization. They discuss *time unit annihilation* as a strategy to gradually reduce precision over time.

3 Adversary Model

To contextualise privacy gain through our more data-minimal timestamp alternatives, we provide the following adversary model. It follows the established honest-but-curious (HBC) notion commonly used to assess communication protocols. Paverd et al. [PMB14] define an HBC adversary as a legitimate participant in a communication protocol, who will not deviate from the defined protocol but will attempt to learn all possible information from legitimately received messages. Following the adaption of this model to the context of application software and data models [BF19], we consider an adversary to be an entity that is in full technical and organisational control of at least one component of a software system, e.g., the application server. The adversary will not deviate from default software behaviour and its predefined configuration options, but will attempt to learn all possible information about its users from the data available in the application. This especially means that an adversary will not modify software to collect more or different data, or employ additional software to do so. However, an adversary can access all data items that are collected and recorded by the software system irrespective of their exposure via GUIs or APIs. We reason

that this adversary model fits real world scenarios, because software operators in general lack the technical abilities to modify their software systems or are unwilling to do so, to not endanger the stability of their infrastructure or to not document potentially illegal behaviour. We come back to this adversary model when we employ server-side reduction later on.

4 Design

Based on alternative concepts for timestamps in the literature, we designed three data types: a generalised date with a static precision reduction (*rough date*), a context-aware counter for chronological ordering (*ordering date*), and a generalised date with temporally progressing precision reduction (*vanishing date*). The designs are targeted as replacements for the conventional timestamp data type in the Django framework, but are using only standard database features typically available in development frameworks. The following describes the design for each alternative type.

4.1 Type 1: Rough Date

Rough date is a variation of Django's standard `DateTimeField` that truncates the date to a given precision. As such, it should offer all functionality that `DateTimeField` does and maintain the same interface, to be usable as a drop-in replacement. The desired precision is given as a mandatory argument at field initialisation, either in seconds or in the style of the `timedelta` class from Python's standard library package `datetime` [Py21] as multiples of the units minutes, hours, etc. The following creates a rough date with one hour precision: `RoughDateField(hours=1)`. We deliberately do not provide a default precision to force users to consider the necessary precision for their given use case. The date part below a given precision is truncated.

4.2 Type 2: Ordering Date

The ordering date is an alternative to using timestamps for ordering items chronologically, if absolute date references and relative distances are not actually needed. `OrderingDateField` offers ordering via context-specific auto-incremented counters. Consequently, ordering date requires that objects are inserted in chronological order. As shown in Fig. 1, a context-defining string key is given for each `OrderingDateField` at model initialisation. The context label is cryptographically hashed to a 256 bit key which then uniquely identifies its corresponding `OrderingContext` which persists the actual counter state information. This way of maintaining the relation between ordering dates and their contexts in code and not in the database is space-efficient and allows for dynamic contexts keys. And since the context label is given at initialisation, ordering contexts can be defined very flexible. For instance, a

label can comprise a username and thereby create an isolation between counter contexts of different users. This can be used to increase user privacy by avoiding an otherwise global counter context that would make instances with ordering date temporally chronologically comparable across users.

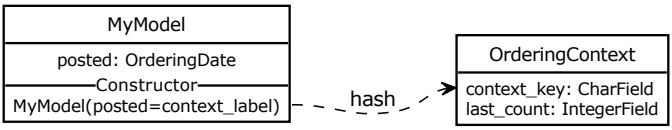


Fig. 1: Class diagram showing a user-defined model with `OrderingDateField`. The related `OrderingContext` is identified by a context label given as initial field value. The integer value of `OrderingDateField` is then set to the context's next count.

4.3 Type 3: Vanishing Date

Vanishing date implements the privacy pattern of *time unit annihilation*. This alternative offers a progressing reduction of precision according to given increments until the end precision is reached. For each step, a precision is provided like for `RoughDateField` in combination with a temporal offset, i. e., the distance from object creation that marks when the reduction step is due. A background process regularly checks for due reductions and applies them. List. 1 shows an example of a vanishing date with a three step reduction policy, the first of which is immediately on creation, whereas the second and third follow after a given time. Tab. 1 lists the resulting stored date and next reduction event for each step.

```
created_at = VanishingDateField(policy=make_policy([
  Precision(hours=1),
  Precision(days=1, after_hours=3),
  Precision(months=1, after_days=7),
]))
```

List. 1: Construction of a vanishing date with a three step reduction policy ranging from initially 1 hour to finally 1 month precision after 7 days. Helper `make_policy` ensures correct reduction progression.

Step	**Current Time***	**Stored Date**	**Next Due Date**
Creation/1st Red.	2021-11-08 15:17	2021-11-08 15:00	2021-11-08 18:00
2nd Reduction	2021-11-08 18:01	2021-11-08 00:00	2021-11-15 00:00
3rd Reduction	2021-11-15 00:03	2021-11-01 00:00	-

Tab. 1: Exemplary progression of vanishing date reduction with a 3-step policy leading to a precision of one month after seven days. (*Current times depend on the frequency and delay of periodic checks.)

As shown in Fig. 2, the resulting design of vanishing date is more complex than for rough date and requires auxiliary models to persist information about the reduction policy and the current progress within that policy. Therefore, `VanishingDateField` sets a reference to

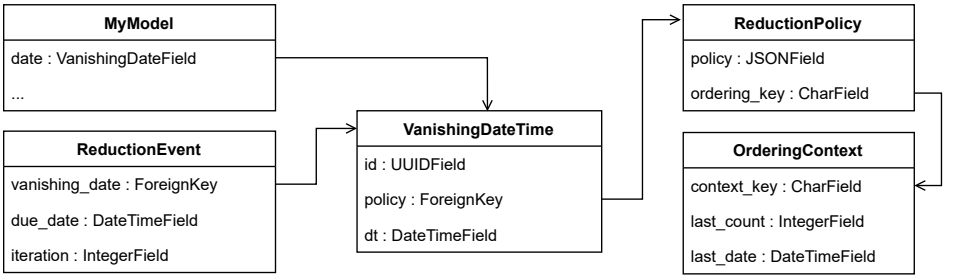


Fig. 2: Class diagram showing a custom model that uses `VanishingDateField` which references a `VanishingDateTime` object specifying the information about the reduction policy and events.

a `VanishingDateTime` class that holds the actual, gradually reduced date, a reference to a policy instance, and to a `ReductionEvent` that represents the next due reduction step. All instances of `ReductionEvent` form a queue that can be efficiently processed by the periodic due check. Since Django lacks the ability to natively trigger periodic tasks, we offer a management command for periodic reduction that can be triggered via, e. g., *Cron*.

Note that to not leave any traces of the previously truncated time information, due dates for subsequent reduction steps are calculated on the basis of the reduced step. In Tab. 1, for instance, the second reduction is due at 18:00 instead of 18:17, to not thwart the reduction to hour precision in the first step. As a result, the time periods given between each policy step are upper boundaries. In the previous example, the hour precision is available for 17 minutes less than the full three hours given in the policy. Also note, the reduction level of a step should not be larger than the offset of the following step.

Following our adversary model in Sect. 3, the application itself holds the only record of the timestamp reduced by vanishing date. This especially means, that the software operator does not keep a mirror of the information before reduction or of earlier reduction levels.

4.4 Design Validation

The purpose of design validation is to examine whether the design mainly based on previous case studies also holds for the application we selected to evaluate our implementation. As described below, we inspected its timestamp usage following the methodology of [BF19].

Taiga is a project management software that is built on the Django framework. Taiga focuses on user interaction like the creation, processing and commenting of tasks to control and document the progress of projects. Following the agile approach, interactions occur around planning elements like tasks, issues, epics, sprints and user stories. We chose Taiga for our evaluation because it is a popular app built on Django and it is focused on structuring and recording user interactions, which likely brings sufficiently complex requirements to test our

implementation. In the following, we describe our methodology, the identified timestamp use, and any necessary modifications to our design.

4.4.1 Methodology

Following [BF19], we examine the source code of Taiga's back-end component [Ta21a] for occurrences of Django's date-related model fields `DateTimeField`, `DateField`, and `TimeField`. We assess semantic and purposes of each timestamp by examining all of their uses by the back-end, their presentation in the front-end [Ta21b] and their exposure via the API. We then use the identified purposes to select from our proposed types an alternative that provides the required functionality. If none should be available, our design would need refinement. We find that the back-end REST API typically returns every attribute (model field) related to the requested object, regardless of whether the front-end uses them or not. Therefore, it is not sufficient to access timestamp usage and purpose only based on API exposure, but actual use based on inspections of the rendered front-end are necessary. To do so, we manually examined Taiga's front-end user interface cataloguing presented timestamp information. Usage in the back-end was assessed by manually inspecting all occurrences of date-related field names throughout the back-end code. These analyses were conducted on version 6.0.7 of both back-end and front-end, as released on March 8th, 2021.

Model Field	Occurrences	Used in Models	Exposed via API
DateTimeField	170	48	41
DateField	15	3	3
TimeField	5	0	0

Tab. 2: Usage of date-related Django model fields in Taiga's back-end.

4.4.2 Identified Timestamps

Tab. 2 shows the numbers of identified uses per model field. In total, we located 190 occurrences in Taiga's back-end code of which 51 are part of data model definitions. The remaining matches occurred in database migrations and serialization code. We did not further inspect the latter occurrences as they do not contribute any usage and purpose information. Almost all definitions use the `DateTimeField`. Only 3 (6 %) use the `DateField`, whereas `TimeField` is not used in current model definitions at all.

To assess the API exposure, we inspected the source code and consulted the official API documentation for information about which fields are included in a query response. We found that all but 7 timestamps (86 %) are exposed through the API. Of those 7 timestamps, 5 are also not programmatically used on the back-end. The visual inspection of the front-end UI also revealed that at least 22 (50 %) of the timestamps fetched from the back-end API are not used there. Regarding those timestamps without a detectable usage or purpose, we

can not determine a purpose-appropriate alternative. For the sake of data minimisation, they should be removed entirely. Also, not all timestamp fields are necessarily personal data. This is true for the three `DateField` uses, which are used to model due dates of planning elements (e. g., sprints) which are not directly linked to actions of users. Hence, we omit those from classification as well. For the remaining timestamps, we classified their type purpose according to their usage context in back-end and/or UI.

4.4.3 Timestamp Semantic and Purpose Classification

We classified the remaining timestamps based on the semantic given by their variable name and source code context. All models in Taiga (27) have a creation timestamp to automatically capture when a model instance was created. 17 models additionally record the time of the latest update, three the time a planning element was completed, and one when a notification was read. Regarding purpose classification, we followed [BF19] and used a bottom-up classification that inspects and labels each timestamp's programmatic use in the source code with respect the function they serve in the respective context, resulting into similar purposes: presentation for user information, sorting, and comparison. Tab. 4 in the appendix lists the identified purposes for each timestamp.

4.4.4 Design Revision

Based on the identified purposes and functional properties of our proposed alternatives, we select possible replacements for each timestamp. The selections are shown in Tab. 4 (appendix). If a timestamp is only used for sorting like the creation date of `Attachment`, the ordering date is the apparent alternative. Timestamps with a presentation or comparison purpose can equally be replaced by rough date and vanishing date. The latter should be chosen if an initial higher demand for precision exists.

We find that our proposed alternatives cover all found purposes. However, we noticed that the purpose of maintaining temporal order sometimes coincides with providing a temporal context (presentation or comparison). To replace such a timestamp, two fields are required in the initial design (e. g., `OrderingDateField` and `VanishingDateField`). Since this would both complicate usage and increase memory footprint, we decided to introduce two additional fields that combine the properties of ordering date with rough date and vanishing date respectively, which otherwise do not maintain order for dates reduced to the same value. To do so without increasing memory footprint, we use the sub-second value range available in most timestamp representations to hold the ordering counter. For a microsecond timestamp this leaves space for a 10^6 counter. The counter is incremented for all timestamps with identical values in their end-precision (Tab. 3). Note that this approach only works for timestamps that are added in chronological order, e. g., that are automatically set to the current time, otherwise the insertion order would not reflect their temporal order.

Original	**Vanishing** 1. Iteration [5 sec]	**Vanishing+Order** 1. Iteration [5 sec]	**Vanishing+Order** 2. Iteration [30 sec]
12:20:11:673320	12:20:10:000000	12:20:10:000000	12:20:00:000000
12:20:14:313406	12:20:10:000000	12:20:10:000001	12:20:00:000001
12:20:17:248323	12:20:15:000000	**12:20:15:000002**	**12:20:00:000002**
12:20:33:040852	12:20:30:000000	12:20:30:000000	12:20:30:000000
12:20:35:917632	12:20:35:000000	12:20:35:000001	12:20:30:000001

Tab. 3: Sample sequence of vanishing date with and without added support to preserve ordering. The highlighted timestamps demonstrate that counter reset is determined by the end-precision of 30 sec which defines the counter scope.

5 Implementation

We implemented our revised concept as a Django app that can be included into other Django projects to provide our date alternatives. It is available open source on GitHub [EM22]. In the following, we describe trade-offs and limitations of this implementation. Ordering date can simply build on available counter fields and is hence omitted from description.

5.1 Rough Date

To offer `RoughDateField` as a drop-in alternative for `DateTimeField`, it also has to support the options `auto_now` and `auto_now_add`, which automatically set the field to the current time at the moment when the object is saved (not initialised). To support these options, the reduction of precision has to be integrated in the saving process, since at any earlier point the value is not yet defined. To do so, we use pre-save hooks that apply the reduction. As a consequence, date values assigned to `RoughDateField` remain in full precision until saved.

5.2 Vanishing Date

As vanishing date is the most complex type, we face three main implementation challenges.

Avoiding chronology leak with UUIDs By default, Django would use an auto-incrementing integer primary key for `VanishingDateTime` if no other primary key was specified. Auto-incremented integers would however leak information about the temporal creation order of all instances of every model that uses `VanishingDateField`. For instance, an attacker could learn that user A logged in after user B posted their last comment but before user B closed the issue. To prevent such chronology leaks, we use randomized UUIDs as primary key. It should be noted that databases might still leak the temporal order by exposing the insertion order in certain queries. Future work should investigate options to, e. g., prevent users from executing such ordering queries.

Policy Reuseability As previously shown in Fig. 2, we decided to make `ReductionPolicy` a separate model to allow its reuse among vanishing dates with the same policy. We provide the helper function `make_policy` that transparently ensures policy reuse.

Model Identification with `VanishingDateMixin` An identification of all models that make use of `VanishingDateField` is required for two reasons: Firstly, to ensure a two-way cascading delete of `VanishingDateTime`, and secondly to create an initial `ReductionEvent`. To locate the relevant models, we decided to employ the common mix-in pattern, which uses inheritance to add functionality to a given class or model. Users of `VanishingDateField` have to add the `VanishingDateMixin` as a base class for their model. Thereby, we can automatically find all sub-classing models and register post-delete listeners to ensure a two-way deletion, as well as a post-safe listener to update reduction events.

6 Evaluation

In this section, we evaluate the practicality of our framework and its storage cost.

6.1 Practicality of Taiga Integration

To evaluate the practicality of our alternatives, we modified Taiga version 6.0.7 to use the timestamp replacements identified in Sect. 4.4 and detailed in the appendix. To test the correctness and impact of our modifications, we ran the modified Taiga and inspected the error output as well as the web front-end for potential negative effects. To check functional integrity, we created and modified various planning elements and compared the visible front-end behaviour before and after integrating our alternatives.

We found that all timestamps could be replaced as suggested, with few adjustments to the code base. As expected, rough date required the least effort of only replacing the field type. More effort was needed for the other alternatives: We used vanishing date and ordering date to each replace timestamps in three models. Since both replacements do not behave like a standard `DateTimeField`, all code that accessed or modified the field value had to be adjusted to either use the reference `VanishingDateTime`, or an appropriate context label instead. After these modifications, Taiga operated normally and we were able to create and modify elements as usual without functional impairments visible through UI or error log.

When it comes to presenting the replacements in the UI, rough date and vanishing date can mostly be treated like standard dates. However, to avoid confusion or wrong expectation of precision, the formatting of both should be adopted to reflect their precision. In contrast, ordering date can no longer be presented as a date in a meaningful way. But if the timestamp previously only fulfilled ordering purposes this should not be an issue. Otherwise, vanishing date might be a more fitting replacement.

6.2 Storage Cost

The following assesses storage cost compared to ordinary date and time (8 byte) using MariaDB as example [Ma19]. Rough date has the same memory footprint. Ordering date uses a 4 byte counter reducing cost by half. The cost of vanishing date is dominated by three UUIDs, which require 38 byte. Added to two 8 byte date, one foreign key and one event counter (each 4 byte), a total of 138 byte is required for vanishing date, which is 17.25 times the cost of an ordinary date and time. Additionally, usage-dependent storage cost is added for vanishing date and ordering date by their auxiliary models. The number `OrderingContext` used depends on the number of distinct context labels set by developers, and scales with the number of users if individual contexts were used to avoid unnecessary comparability. In MariaDB, each `OrderingContext` requires 44 bytes. Moreover, a variable amount of storage is required for `ReductionPolicy`, which depends on the number of defined reduction steps. To give an overall example, applying the replacements in Tab. 4 increases Taiga's average storage cost per timestamp by about 5 times (not weighted by instance frequency).

7 Conclusion

Excessive, unthought use of timestamps in software data models is a violation of the data minimisation principle and potentially harmful to user privacy. Our case study of the Taiga application not only supports the findings of prior work regarding excessive use, but also in terms of minimisation potential through the use of more-privacy preserving timestamp alternatives. We have presented a framework of alternatives for common timestamp functions and purposes. We demonstrated its practicality by implementing it as a Django app which we then used to replace timestamps in Taiga. Although demonstrated for Django, these alternatives only use standard concepts and can be implemented for other development frameworks. Our evaluation with Taiga revealed that code changes were necessary but limited to adopting changed initialisation and access methods. Additionally, the presentation of timestamp may need adjustment to convey decreased precision levels. Depending on the selection of alternatives, especially the frequency of vanishing date, storage cost might increase noticeably. Where this is nor acceptable, rough date and ordering date can be used with little to no additional storage cost, but without gradual reduction. Our integration test suggests that more privacy-preserving alternatives can be adopted with reasonably low effort. A user study to evaluate their usability with developers is left to future work.

Acknowledgements The work is supported by the German Federal Ministry of Education and Research (BMBF) as part of the project Employee Privacy in Development and Operations (EMPRI-DEVOPS) under grant 16KIS0922K.

References

[BF19] Burkert, C.; Federrath, H.: Towards Minimising Timestamp Usage In Application Software - A Case Study of the Mattermost Application. In: DPM/CBT@ESORICS. Vol. 11737. Lecture Notes in Computer Science, Springer, pp. 138–155, 2019.

[EM22] EMPRI-DEVOPS: django-privacydates, 2022, URL: https://github.com/EMPRI-DEVOPS/django-privacydates, visited on: 01/18/2022.

[LFH17] Lenhard, J.; Fritsch, L.; Herold, S.: A Literature Study on Privacy Patterns Research. In: 43rd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2017, Vienna, Austria, August 30 - Sept. 1, 2017. IEEE Computer Society, pp. 194–201, 2017.

[Ma19] MariaDB: Data Type Storage Requirements, Apr. 2019, URL: https://mariadb.com/kb/en/data-type-storage-requirements/, visited on: 06/23/2021.

[PMB14] Paverd, A.; Martin, A.; Brown, I.: Modelling and automatically analysing privacy properties for honest-but-curious adversaries, tech. rep., 2014, URL: https://www.cs.ox.ac.uk/people/andrew.paverd/casper/casper-privacy-report.pdf.

[Py21] Python Software Foundation: datetime - Basic date and time types, May 2021, URL: https://docs.python.org/3/library/datetime.html, visited on: 05/14/2021.

[Ta21a] Taiga Agile: taiga-back, Mar. 2021, URL: https://github.com/taigaio/taiga-back/releases/tag/6.0.7, visited on: 03/11/2021.

[Ta21b] Taiga Agile: taiga-front, Mar. 2021, URL: https://github.com/taigaio/taiga-front/releases/tag/6.0.7, visited on: 03/11/2021.

[ZBY06] Zhang, J.; Borisov, N.; Yurcik, W.: Outsourcing Security Analysis with Anonymized Logs. In: Second International Conference on Security and Privacy in Communication Networks and the Workshops, SecureComm 2006, Baltimore, MD, USA, August 2, 2006 - September 1, 2006. IEEE, pp. 1–9, 2006.

A Taiga Timestamp Purposes and Replacements

Model / Timestamp	Presentation	Sorting	Comparison	Replacement
Attachment				
created_date		✔		OD
Epic				
created_date	✔			RD
HistoryChangeNotification				
updated_datetime			✔	RD
HistoryEntry				
created_at	✔	✔		VD+O
delete_comment_date	✔			VD
edit_comment_date	✔			VD
Issue				
created_date	✔			RD
modified_date	✔	✔		RD
Like				
created_date			✔	RD
Task				
created_date	✔			RD
TimeLine				
created	✔	✔	✔	VD+O
User				
date_joined	✔			RD
UserStory				
created_date	✔			RD
Watched				
created_date		✔		OD
WebNotification				
created	✔	✔		VD+O
read		✔		OD
WikiPage				
created_date	✔			RD
modified_date	✔			RD

Rough Date **(RD)** Ordering Date **(OD)** Vanishing Date **(VD)**
Vanishing Date with Ordering **(VD+O)**

Tab. 4: Identified purposes and suggested replacements for used timestamps in Taiga.

On CRDTs in Byzantine Environments

Conflict-Freedom, Equivocation Tolerance, and the Matrix Replicated Data Type

Florian Jacob[1] Saskia Bayreuther[1] Hannes Hartenstein[1]

Abstract: Conflict-free Replicated Data Types (CRDTs) allow updates to be applied to different replicas independently and concurrently, without the need for a remote conflict resolution. Thus, they provide a building block for scalability and performance of fault-tolerant distributed systems. Currently, CRDTs are typically used in a crash fault setting for global scale, partition-tolerant, highly available databases or collaborative applications. In this paper, we explore the use of CRDTs in Byzantine environments. This exploration is inspired by the popular Matrix messaging system: as recently shown, the underlying Matrix Event Graph replicated data type represents a CRDT that can very well deal with Byzantine behavior. This "Byzantine Tolerance" is due to mechanisms inherent in CRDTs and in the hash-based directed acyclic graph (HashDAG) data structure used in Matrix. These mechanisms restrict Byzantine behavior. We, therefore, discuss Byzantine behavior in a context of CRDTs, and how the notion of Byzantine tolerance relates to equivocation. We show that a subclass of CRDTs is equivocation-tolerant, i.e., without equivocation detection, prevention or remediation, this subclass still fulfills the CRDT properties, which leads to Byzantine tolerance. We conjecture that an operation-based Byzantine-tolerant CRDT design supporting non-commutative operations needs to be based on a HashDAG data structure. We close the paper with thoughts on chances and limits of this data type.

Keywords: Dependable Distributed Protocols; Conflict-Free Replicated Data Types; Equivocation Tolerance; Byzantine Fault Model; Matrix Event Graph

1 Introduction

Conflict-free Replicated Data Types (CRDTs) are maintained by replicas that run on multiple processes of a distributed system.[2] To keep all replicas up to date about local changes to the CRDT, replicas repeatedly broadcast updates to all replicas. As the name suggests, CRDTs provide powerful properties: in particular, updates can be applied without further coordination of replicas, and recovery from network partitions can be done with ease. The corresponding property that CRDTs provide, namely Strong Eventual Consistency (SEC), ensures that correct replicas eventually converge to a consistent state, i.e., the same state, regardless of the order in which updates were received. Therefore, CRDTs are popular

[1] Karlsruhe Institute of Technology, KASTEL Institute of Information Security and Dependability, Am Fasanengarten 5, 76131 Karlsruhe, Germany

[2] For simplicity, we typically assume a one-to-one correspondence between replicas and processes.

doi:10.18420/sicherheit2022_07

for global-scale, partition-tolerant, highly-available databases or peer-to-peer collaborative applications like shopping lists and collaborative text editors.

However, previous CRDT-related work mainly focused on crash fault settings. We are interested in an understanding of CRDTs also in a Byzantine environment. In particular, we are interested in the characterization of Byzantine-tolerant CRDTs as well as in the relevance of these CRDTs. This interest is currently shared by various researchers [KH20, Au21] and also inspired by the success of the Matrix messaging system [Th21] that is based on the Matrix Event Graph replicated data type, a CRDT in a Byzantine environment [Ja21].

What does a Byzantine environment mean for CRDTs? Basically, we will call a CRDT Byzantine-tolerant when Byzantine processes cannot induce a violation of the CRDT properties. To achieve Byzantine tolerance, recent work on CRDTs in Byzantine environments have followed different paths. Some previous work makes use of classical assumptions of an honest two-thirds majority [Zh16] or introduce coordination mechanisms (e.g., coordinated Byzantine-tolerant causal-order broadcast [Au21]). Other recent publications study coordination-free, Sybil-resistant CRDTs using broadcast based on the happened-before relation as directed, acyclic graphs [KH20, Ja21].

In this paper, we take up the latter approach that does neither depend on additional coordination mechanisms nor on a particularly strengthened broadcast. As a motivating example, we look at the Matrix Event Graph replicated data type and its use of a hash-based directed acyclic graph data structure. We show that Byzantine behavior targeted to violate SEC essentially reduces to equivocation (and omission as a special case of equivocation) and under which conditions a subclass of crash-tolerant CRDTs is equivocation-tolerant in Byzantine environments. In particular, we show that, due to equivocation tolerance, all state-based and a subclass of operation-based CRDTs in the crash fault model tolerate any number of Byzantine processes. Further, we conjecture that the only non-trivial Byzantine-tolerant CRDT design is a grow-only HashDAG as it is done in the Matrix Event Graph.

Do Byzantine-tolerant CRDTs matter? As "safety does not guard against faulty clients" [Ca99, Section 3], even if Byzantine faults can be tolerated on a CRDT level, the application itself on top of a Byzantine-tolerant CRDT has to cope with the provided SEC guarantee and a potentially Byzantine behavior on application level. While this application-level Byzantine tolerance is out of scope of our paper and cannot be achieved in many cases, in this paper we like to start the discussion on chances and limits of Byzantine-tolerant CRDT deployments.

The structure of the paper is as follows: In Sect. 2, we provide the system model with its terminology and assumptions as well as the relationship between Byzantine tolerance and equivocation tolerance for CRDTs. In Sect. 3, we study the case of the Byzantine-tolerant Matrix Event Graph that powers the Matrix messaging system. In Sect. 4, we provide the general technical characterization of Byzantine-tolerant CRDTs as well as a conjecture. In Sect. 5, we take up the question on the relevance of Byzantine-tolerant CRDTs based on the technical characterization and address open issues.

2 System Model: Terminology and Assumptions

Replicas run on processes of a distributed system and serve data to a (distributed) application. It is, therefore, natural to consider the system under study on three layers: the network layer for the communication between processes, the CRDT layer as the data layer, and the application layer. The focus of this work is on the CRDT layer in the middle, in which replicas execute operations to query or update the state of the CRDT. The replicas rely on the network layer to provide broadcast in order to exchange information on their current state or state changes. In this section, we will first review the guarantees to the application layer that are provided by the replicas as well as their requirements for the network layer, for the case of a crash fault setting. Afterwards, we move to Byzantine environments and the notion of Byzantine-tolerant CRDTs.

Without coordination or conflict resolution between replicas, CRDTs ensure a notion of "conflict-freedom" formalized as *Strong Eventual Consistency* (SEC). SEC consists of the following properties [Sh11]:

Eventual Delivery: If an update is applied at some correct replica, it is eventually applied at every correct replica.
Termination: Every operation that is executed by a correct replica eventually terminates.
Strong Convergence: Correct replicas that applied the same set of updates maintain the same state.

CRDTs are either *state-based* or *operation-based* (see, e.g., [Sh11]). With state-based CRDTs, all states of the CRDT form a *semilattice*. A semilattice is a partially ordered set, and every possible CRDT state is one element of the set. Every pair of states has a least upper bound, which is also called the *join* of two states. An update is *valid* if it is part of the semilattice. If a replica r receives a valid state from another replica p, p's state can be directly applied by merging r's state with p's state using the join operation of the semilattice. To fulfill SEC, replicas repeatedly broadcast the current local state to the other replicas, which merge the received state with their local state. Thus, eventual delivery is required as a property of the underlying network layer.

Operation-based CRDTs differ from state-based CRDTs in the fact that replicas do not send their whole new state as updates but only the operation that lead to the new state. To fulfill SEC, the updates must either be applied in causal order or be commutative. In crash fault environments, CRDTs usually require an underlying causal order broadcast on the network layer to enforce the causal order [Sh11]. We follow a different approach (as it is done in the Matrix Event Graph) to enforce the causal order by making use of the happened-before relation that is used in the CRDT itself. We present the happened-before relation and the corresponding causal order in Sect. 3.

Replicas only apply *valid updates*. The validity of an update is defined by the specific CRDT when viewed on its own, e.g., in case of state-based CRDTs by the semilattice. An invalid update violates locally verifiable properties and cannot be applied to the CRDT. A *conflict*

would occur when two concurrent updates that may be individually *valid* but, when viewed together, violate some invariant (cf. Sect. 4 and [Sh11, Footnote 1]). However, a CRDT guarantees that a conflict on the CRDT layer will not occur, typically under the assumption of a crash fault model.

We are now interested to analyze the conditions or restrictions under which a CRDT in the crash fault model can tolerate Byzantine behavior on the CRDT layer. A Byzantine-tolerant CRDT is defined as follows:

Byzantine-tolerant CRDT: A *Byzantine-tolerant CRDT* ensures that on correct processes the SEC guarantee is provided to the application layer by tolerating Byzantine behavior regardless of the number of Byzantine processes as long as the correct processes build a connected component on the network layer.

A CRDT in the crash fault model already provides strong means against Byzantine behavior: an invalid update according to the CRDT definition will not be accepted anyhow. Whether the CRDT guarantees are sufficient for an application is a different aspect on which we comment later. Thus, a Byzantine replica cannot attack the system with invalid updates with respect to the CRDT definition, and *Equivocation* and *Omission* remain as the only options for an attack on CRDT layer. Equivocation is the act of sending different valid updates to different recipients, where a replica should have sent the same update [Ch07]. Omission can be seen as a special case of equivocation where an update is not sent to a subset of replicas or even to all other replica. In contrast to an invalid update, which is detectable when viewed on its own, equivocated updates can only be detected globally or with both equivocated updates. Equivocated updates always originate from a Byzantine replica and do not exist in the crash fault model. Please note that in the scope of this paper, equivocated updates are not necessarily conflicting, i.e., with respect to the invariants of the technical CRDT layer.

We can, therefore, check the Byzantine tolerance of a CRDT by checking whether a CRDT is *equivocation-tolerant* as defined as follows:

Equivocation-tolerant CRDT: A CRDT is *equivocation-tolerant* if it neither needs to detect, prevent, nor remedy equivocation to ensure its provided guarantees beyond what is needed to cope with omission.

Through our assumptions, the relevant Byzantine behavior is restricted to the network and CRDT layer in form of equivocation and omission, as Byzantine behavior on the application layer cannot harm SEC and is thereby out of scope. In Sect. 4, we present a characterization of subclasses of CRDTs that are equivocation-tolerant and, thus, Byzantine tolerant.

For the network layer, we assume an asynchronous network with a static set of processes participating in the system. The processes are connected with authenticated channels and all correct processes form a connected component[1] in the communication graph, so that Byzantine processes can neither forge message senders nor block communication

[1] No fully connected mesh is required.

between two sets of correct processes. Every update that is sent over a channel can be arbitrarily delayed, but is received eventually on the other side of the channel. The requested connected component can, therefore, be built based on a Best-Effort Broadcast as, e.g., defined in [CGR11] (also cf. Appendix A). In the case of state-based CRDTs, one has to require periodic broadcasts of all correct replicas to ensure 'connectedness', i.e., liveness. Correspondingly, in the case of operation-based CRDTs, we will require eventual response to a request when a replica requests information from another replica.

A different question is which applications would work reasonably well with the guarantees provided by Byzantine-tolerant CRDTs — and whether Byzantine behavior can be dealt with on application layer. We will take up this discussion in Sect. 5. In short, one would either stick to grow-only CRDTs or one has to add (policy-based) mechanisms that are based on access control (or both). As grow-only CRDTs only support 'append' operations, they are susceptible to application-layer spamming if not prevented via access control. The Matrix Event Graph, as illustrated in the following section, represents a grow-only CRDT and serves an instant messaging application.

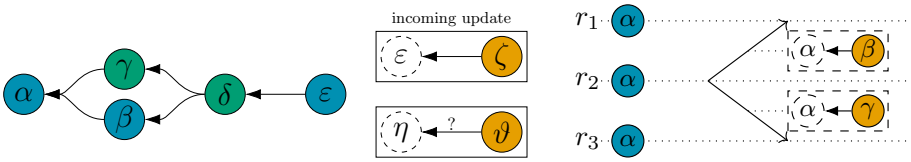
3 Matrix Event Graph

As of today, Matrix is primarily used for decentralized instant messaging, e.g., by the French public sector, by the German military forces, and as upcoming standard in the German healthcare sector [Ho21b]. As of October 2021, the public Matrix federation consists of more than 35 million users and 70 thousand servers [Ho21a].

The Matrix Event Graph (MEG) represents the CRDT at the core of Matrix. It provides a grow-only history of messages to the publish/subscribe messaging application layer. The MEG is conflict-free not only in the crash fault model, but also in the Byzantine fault model [Ja21]. Therefore, the MEG serves as our prime example for Byzantine-tolerant operation-based CRDTs in this paper. From the network layer, the MEG only requires a best-effort broadcast and a connected component of all correct processes. We present a short introduction to the MEG concept, and outline why neither Byzantine equivocation nor omission can 'hurt' the MEG.

The data structure of a MEG is a directed, acyclic graph (DAG) [Ja21]. A new message e is appended to a replica of the MEG by an update operation that takes the set L of all currently known forward extremities (intuitively: messages without 'children'), adds a unique identifier w for the operation and sends the tuple (e, L, w) to all replicas (including itself). Each (correct) replica now adds the new vertex (e, w) to the DAG as well as corresponding edges from (e, w) to all 'parents' in L, provided all vertices in L exist at this replica. If not, the replica requests information of missing vertices from other replicas. For a formalization of the CRDT definition please consult Algorithm 1 in Appendix B.

The edges represent a happened-before relation as defined by [La78], and the corresponding potentially causal order provides the causal order for the CRDT. Of course, the potentially



(a) Matrix Happened-Before Relation (b) Equivocation performed by r_2

Fig. 1: (a) MEG state of one replica with messages from itself (●) and from other replica (●). It receives two updates from a third replica (●): The update ζ can be applied immediately, since its causal parent ϵ is already known. The update ϑ cannot be applied (yet) since η is not part of the replica's state. (b) An example for an equivocation performed by replica r_2.

causal order does not refer to the actual 'real' causality of the messages, but to the potential causal order given by the selection of the set L. In the following, we simply refer to the 'causal order of the CRDT'. Fig. 1a shows a simple example of a causal history of a replica.

To be able to cope with Byzantine behavior, integrity needs to be protected: it should neither be possible to change a message's content in an undetectable manner once the message is added to the MEG, nor should it be possible to change the causal order in the MEG. For integrity protection, *content identifiers* are used based on cryptographic hashes to bind both the identity as well as the causal order of operations in a way that is locally verifiable for correct processes, but unforgeable for Byzantine processes. Let h_1 be a cryptographic hash function that provides for the content identifier $h_1(e)$ of message e. Assume that $w_1, \ldots, w_n$ are the unique identifiers of the operations that added the elements of L to the DAG. Then the content identifier (and unique identifier) for the operation that adds message e is given by $h_2(w_1, \ldots, w_n, h_1(e))$, for a cryptographic hash function h_2. Thus, content and structure of the DAG is protected using cryptographic hashes: when the content identifier for adding message β contains the hash of the content identifier for adding message α, α must have happened before β. Through these hash-based content identifiers, in analogy to a hash chain in blockchains, the MEG builds up a *Hash-based DAG* (HashDAG) data structure.

As depicted in Fig. 1a, the happened-before relation allows replicas to detect faults: If the parents of a new message are not yet known, it is not appended to the graph. Instead, other replicas can be queried for the missing operations. Through the hash-based content identifiers, replicas can verify the integrity of replies to their query even from Byzantine replicas, and will eventually receive the operation if any correct replica received it. Thus, messages with made-up parents will never be added to the graph maintained at a correct replica. As shown in Fig. 1b, a Byzantine replica (r_2) can perform equivocation: It sends an update with message β to r_1 and an update with message γ to r_2 where it should have sent the same to both r_1 and r_3. As different updates have different content identifiers, as soon as correct replicas r_1 and r_3 communicate with each other, they can reliably detect that their graphs differ by comparing content identifiers of received updates. Because the DAG

only implies a partial order, the MEG can treat and merge two equivocated updates as two causally independent, concurrent updates, and consistency is restored.

4 CRDTs: Equivocation Tolerance and Byzantine Tolerance

In this section, we analyze under which conditions CRDTs in the crash-fault model are Byzantine-tolerant as defined in and under the assumptions from Sect. 2. As presented in Sect. 2, to check Byzantine tolerance, we have to check equivocation tolerance as well as handling of omissions. Equivocation itself mainly threatens the Strong Convergence property. A Byzantine replica can equivocate using two updates trying to attack the notion of which updates are the same, or the application order of updates of correct replicas. Thus, we have to address these identity and ordering aspects of updates. As we will also see below, omission faults will not threaten the property of Termination.

4.1 State-based CRDTs

Proposition 4.1. *All state-based CRDTs in the crash fault model also trivially provide equivocation tolerance in Byzantine environments.*

Sketch of proof. State-based CRDTs are based on a defined join-semilattice of all valid states. Replicas of a state-based CRDT in the crash fault model only send their current state without metadata, which means that an update is valid if and only if it is part of the semilattice, which is locally verifiable. Due to the commutativity of the join relation and the partial order of the semilattice, any two valid updates cannot conflict with each other, as both can be merged in an arbitrary order with the same result. Thus, in case a replica wants to equivocate, all the replica can do is to send different valid updates that will not conflict, by definition of the state-based CRDT in the crash fault model. Accordingly, an equivocation consisting of d differing updates can be treated as d independent updates for which omission has occurred.

Corollary 4.2. *State-based CRDTs in the crash fault model also ensure Strong Eventual Consistency for all correct replicas and are thereby Byzantine-tolerant.*

Sketch of proof. Due to Proposition 4.1, we can treat equivocation as omission, i.e., an update was not sent to all other replicas. State-based CRDTs broadcast their current state regularly to all other replicas. With the given system model, every update that is sent to another replica is received eventually by this replica. As the replicas' current state indirectly contains all updates they have received and merged before, updates not sent directly to a specific replica will eventually reach that replica indirectly via correct replicas. Thus, also Eventual Delivery and Termination are not violated.

4.2 Operation-based CRDTs

While state-based CRDTs need no further assumptions to be Byzantine-tolerant, only certain operation-based crash-tolerant CRDTs are also Byzantine-tolerant, and they need the following two additional assumptions or invariants.

Operation-based CRDTs require that non-commutative updates are applied in causal order. A CRDT provides an *inherent ordering* when all information that a correct replica needs to apply an update in correct causal order is part of the update and cannot be equivocated. In other words, the updates are either commutative or are integrity protected as it is the case for a HashDAG as outlined in Sect. 3: when inherent ordering is implemented via hash-based content identifiers as in the MEG, Byzantine attackers cannot tamper with the happened-before relation, as hashes verifiably prove the order of the updates.

Hash-based content identifiers also provide an *inherent identity* for update operations: An update has an inherent identity when all information a correct replica needs to distinguish two updates (or to decide that two updates are identical) are part of the update and cannot be equivocated. Thereby, identical updates are not applied twice when received twice.

Proposition 4.3. *Operation-based CRDTs in the crash fault model require inherent identity of updates and inherent ordering of updates to be equivocation-tolerant in any Byzantine environment.*

Sketch of proof. Operation-based CRDTs rely on update metadata, especially on content identifiers of update operations. The uniqueness of content identifiers of update operations represents an invariant whose violation by Byzantine replicas would violate Strong Convergence, and thereby SEC. Without inherent identity, a Byzantine replica can equivocate by sending two different updates with the same identifier to different replicas. Without inherent ordering, a Byzantine replica can equivocate by sending two versions of two non-commutative updates with converse causal order. In both cases, the conflicting updates would be applied and lead to an inconsistent state, without a coordination-free way for the receiving replicas to detect or remedy. With inherent identity and inherent ordering, one of any pair of conflicting updates that would violate identifier uniqueness resp. happened-before correctness is invalid, i.e., can be locally detected and rejected by correct replicas without coordination. It follows that both inherent identity as well as inherent ordering is required for operation-based CRDTs to be equivocation-tolerant.

Without periodic broadcasting of state-based CRDTs, operation-based CRDTs need to be able to handle omissions to ensure Eventual Delivery. If a happened-before relation is included in updates, then *Omission Handling* can rely on the relation to detect missing updates. Using hash-based content identifiers, those missing updates can be requested from other replicas. Alternatively, CRDTs can periodically gossip the set of all received updates. The gossiping approach can be formalized and made more efficient through the happened-before relation and hash chaining [KH20]. We note that this approach essentially

uses a state-based set CRDT to synchronize all updates, benefiting from the Byzantine tolerance of all state-based CRDTs shown in Corollary 4.2.

Corollary 4.4. *Omission-handling, equivocation-tolerant operation-based CRDTs ensure Strong Eventual Consistency for all correct replicas and are thereby Byzantine-tolerant.*

Sketch of Proof. Proposition 4.3 allows to treat equivocation as omission. For CRDTs that use an omission handling mechanism (e.g., one of the approaches explained above), Byzantine replicas cannot prevent that updates they sent to at least one correct replica are eventually delivered to all correct replicas, i.e., they cannot harm the Eventual Delivery property through omission.

4.3 Uniqueness Conjecture

In Byzantine environments, a CRDT with non-commutative operations has to record the happened-before relation of updates in the data structure to locally ensure the causal order independently of the broadcast order. Ensuring that some update happened-before some other update in a locally verifiable way in the presence of Byzantine processes directly points to hash-chaining the corresponding updates with a cryptographic hash function guaranteeing preimage resistance. The happened-before relation being a partial order inherently leads to a directed, acyclic graph (DAG) of all updates. To efficiently ensure Eventual Delivery, one can employ the happened-before relationship recorded in the graph by requesting missing parent operations from other replica. In combination, these considerations lead to a grow-only HashDAG. The presented line of thought was independently followed in [KH20] as well as in [Th21, Ja21], which leads us to the following conjecture:

Conjecture. *The only Byzantine-tolerant operation-based CRDT design that supports non-commutative updates is a grow-only HashDAG.*

5 Discussion and Conclusion

We analyzed the reasons why and under which conditions a subclass of CRDTs is not only crash-tolerant, but also Byzantine-tolerant. For the analysis we made use of the notion of equivocation tolerance and its relation to conflict freedom of CRDTs. We showed that regardless of the number of Byzantine processes, this subclass can keep the characteristic traits of CRDTs, like efficiency, low coordination effort, and Strong Eventual Consistency. On the network layer, only a best effort broadcast, i.e., eventual delivery, is required.

We now like to take up the question of "do Byzantine-tolerant CRDTs matter?" by looking to the current practical relevance and limitations of Byzantine-tolerant CRDTs as well as to potential combinations with access control and/or further coordination mechanisms. First of all, the Matrix Event Graph with its grow-only HashDAG design proves the practical relevance of Byzantine-tolerant CRDTs for the use case of instant messaging. But is there

a relevance for other use cases than the Matrix one? The Matrix example also points to limitations: while a grow-only data structure avoids the need of handling of deletion events, it brings up the issue of garbage collection and spamming. Furthermore, one has to rethink the applicability to other application scenarios which we will discuss in the following.

State-based CRDTs are easy to deploy in Byzantine environments because of their unconditional equivocation tolerance we showed in Corollary 4.2. However, the identity of updates gets lost since only states are propagated in the system. It is not possible to reconstruct the update that led to a new state, which makes it impossible to prove which replica performed which CRDT updates and whether it was allowed to do so. Hence, access control on the different operations of state-based CRDTs, giving different permissions to different participating replicas, cannot be enforced. Therefore, state-based CRDTs might be suitable for decentralized systems in the spirit of the Newsgroup system where any user can write new articles and reply to old ones.

In contrast to state-based CRDTs, with operation-based CRDTs the original caller of an update operation can be determined and verified with authentication mechanisms like digital signatures. Therefore, operation-based CRDTs provide the necessary prerequisites for access control, which makes them easier to deploy in more demanding systems.

In general, CRDTs forfeit consensus and coordination (cf. the example of SEC but no consensus in Appendix C) in favor of availability and partition tolerance. Without Sybil countermeasures like controlled membership, the space of solvable application-layer problems is the class of invariant-convergent problems, i.e., invariants for which local, uncoordinated replica decisions are sufficient to preserve the invariants globally [KH20]. While this takes cryptocurrencies out of question, the typical CRDT use cases for which Strong Eventual Consistency suffices, e.g., collaborative applications like shopping lists, text editors or whiteboards, are also invariant-convergent and, therefore, uses cases for Byzantine-tolerant CRDTs.

When the application layer requires stronger guarantees, e.g., strong consistency, Byzantine-tolerant CRDTs can obviously only serve as part of a solution and need to be combined with other, stronger mechanisms. While the combination 'technically' inherits the stronger assumptions, from a practical deployment perspective, a hybrid mode of operations might make sense: an application can make use of the Byzantine-tolerant CRDT to collect data only requiring corresponding weak assumptions on the system. Whenever stronger assumptions like synchronous operation are fulfilled at certain phases in time, more demanding tasks like consensus can be performed, which also allows for garbage collection in the CRDT layer. Thus, the resulting hybrid system would fall into the category of partially synchronous system [DLS88], reaching agreement on previously aggregated updates.

We hope the characterization of Byzantine-tolerant CRDTs we presented in this paper provides a basis for further exploration.

A Best-Effort Broadcast

For convenience and to avoid misunderstanding, the definition of best-effort broadcast as presented in [CGR11, Section 3.2] is reproduced here. An example best-effort broadcast algorithm implementation can also be found there.

Abstraction 1 Best-effort Broadcast Interface and Properties

Events:

$\langle Broadcast, m\rangle$: Broadcasts a message m to all processes.
$\langle Deliver, p, m\rangle$: Delivers a message m broadcast by process p.

Properties:

Validity: If a correct process broadcasts a message m, then every correct process eventually delivers m.
No duplication: No message is delivered more than once.
No creation: If a process delivers a message m with sender s, then m was previously broadcast by process s.

B Conflict-free Replicated Data Types

The following formalization and example of Conflict-free Replicated Data Type (CRDT) implementations, i.e., replicated objects that belong to one of the two CRDT families, is based on the original CRDT paper [Sh11] and on the paper [Ja21] that analyzes the Matrix Event Graph. A state-based replicated object is defined as a tuple (S, s^0, q, u, m). S is the space of possible per-replica states; the replica at process p_i has state $s_i \in S$. The initial state of every replica is s^0. The query method q returns the current state of the replica. The update method u modifies the current state. The merge method m merges the state from a remote replica with the current local state. If Eventual Delivery and termination is ensured, an state-based replicated object provides SEC and thereby is a state-based CRDT if the set of possible states S with the merge function m is a join-semilattice.

An operation-based replicated object is defined as a tuple (S, s^0, q, t, u, P). Again, S is the space of possible states and s^0 the initial state. The query method q returns the current state. The update method is composed of a side-effect-free generator step t and an effector step u that performs the state change. The generator step is executed by the source replica and returns an operation that is then broadcast to all replicas. Then, every replica executes the effector step of the update, which applies the operation to their current state. The effector may contain a precondition P, which must be fulfilled before an operation is applied. If causal delivery of updates and termination is ensured, an operation-based replicated object provides SEC and thereby is a operation-based CRDT if all concurrent updates are commutative, and the delivery precondition P is satisfied by causal delivery.

As an example, we provide the definition of the Matrix Event Graph (MEG) grow-only HashDAG as operation-based replicated object in Algorithm 1, which was shown to be a Byzantine-tolerant, operation-based CRDT in [Ja21]. Each vertex is a tuple (e, w) with w

Algorithm 1 Matrix Event Graph Operation-based Replicated Object

state set $S = (V, E)$ ▷ vertices are of form $V =$ (`event` e, `uid` w), E represents edge $E \subseteq (V \times V)$
init $(\{e_0, w_0\}, \emptyset)$
query `lookup` (`uid` w) : `boolean`
  return $\exists(e', w') \in V : w' == w$
query `hasChild` (`vertex` (e, w)): `boolean`
  return $\exists((e', w') \in V) : ((e', w'), (e, w)) \in E$
query `getExtremities` () : `list of vertices`
  return $L = \bigcup_{(e,w) \in V : !\texttt{hasChild}((e,w))} \{(e, w)\}$
query `getState` () : `set`
  return S
update `append`
  generator
    let $L =$ `getExtremities()`
    let $w =$ `unique`(L, e)
    return `append`, (e, L, w)
  effector `event` e, `list of vertices` L, `uid` w
    pre $\forall(e_p, w_p) \in L :$ `lookup`(w_p)
    $V = V \cup \{(e, w)\}$
    $E = E \cup \bigcup_{(e_p,w_p) \in L} \{((e, w), (e_p, w_p))\}$

being a unique identifier and e the actual event. Edges represent the causal relationship between a child and a parent vertex. Then, the state is a DAG which is defined through vertices and edges. The types of methods are indicated with **query** and **update**. The precondition of the effector step is indicated with **pre**. When appending a new event e to the graph, the generator returns operation in the form (e, L, w), where L is the list of current forward extremities, i.e., vertices without children. Let h_1 be a cryptographic hash function that provides for the content identifier $h_1(e)$ of message e. Assume that $w_1, \ldots, w_n$ are the unique identifiers of the operations that added the elements of L to the DAG. Then the content identifier (and unique identifier) w for the operation (e, L, w) that adds message e is given by `unique`$(L, e) = h_2(w_1, \ldots, w_n, h_1(e))$, for a cryptographic hash function h_2. After validation, the operation (e, L, w) is applied by the effector which first verifies that the parent vertices from L are already part of the current state, and then adds the new vertex and the edges between the new vertex and the parent vertices from L.

C Example of an Equivocation tolerated by the MEG while being an Application-Layer Conflict

In Sect. 3, we showed that the Matrix Event Graph (MEG) is a Byzantine-tolerant CRDT, and can thereby guarantee the notion of 'conflict-freedom', defined as Strong Eventual Consistency (SEC) in Sect. 2, in face of equivocation. However, this 'conflict-freedom' on the CRDT layer does not mean that the equivocation does not also present a conflict on the application layer that might even require consensus, which is out of scope for our work. As an example, in Fig. 2, we show an equivocation that contains the classical "Attack!" and

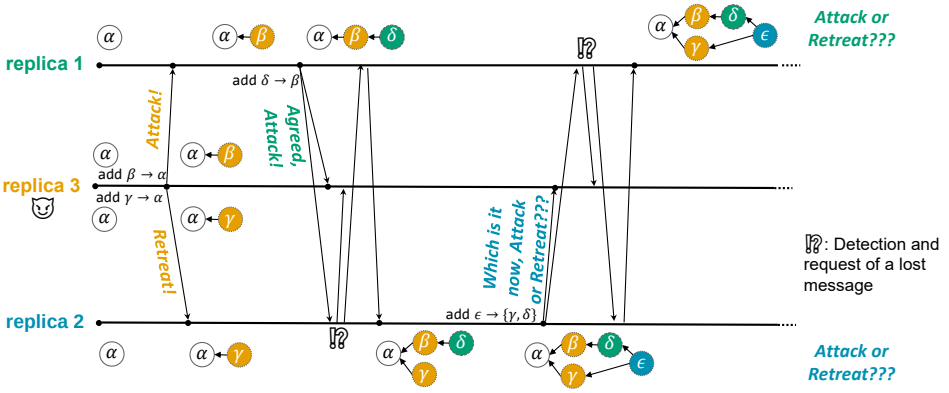


Fig. 2: Evolution of a MEG over time in face of replica 3 running on a Byzantine faulty process performing equivocation on replicas 1 and 2 running on correct processes. Messages associated to operations β and γ are an application-layer conflict, and replica 3 equivocates by sending $\beta \rightarrow \alpha$ to replica 1 but $\gamma \rightarrow \alpha$ to replica 2. The MEG, guaranteeing the SEC formalization of 'conflict-freedom' to the application layer, eventually provides converged current states of the causal history at both replica 1 and 2, and contains both equivocated operations β and γ. To solve the remaining application layer conflict based on the synchronized causal history is left to the application.

"Retreat!" conflict of the Byzantine Generals Problem [LSP82] on the application layer, while the MEG still provides SEC to the application layer.

All replicas start with initial state $s^0 = \alpha$. Replica 3 then creates the messages "Attack!" with operation β and "Retreat!" with operation γ that conflict on the application layer, and performs equivocation by sending an update attaching β to α at replica 1, while sending the different update to replica 2 that instead attaches γ to α there. The current state of replica 1 and 2 is thereby momentarily inconsistent. When replica 1 now broadcasts $\delta \rightarrow \beta$, replica 2 notices that it is missing the operation containing β, and re-requests it from replica 1 and 3. While replica 3 pretends not to know about β, replica 1 returns $\beta \rightarrow \alpha$ and thereby enables replica 2 to apply both $\delta \rightarrow \beta$ and $\beta \rightarrow \alpha$. The respective procedure repeats with $\eta \rightarrow \gamma, \delta$ broadcast by replica 2. After both replica 1 and 2 added η to their graphs, their current states are consistent again.

The example shows that while the application layer still has to deal with conflicts, it can at least rely on SEC in the way that both α = "Attack!" and β = "Retreat!" end up as parallel events in the current state of both replica 1 and 2, no later than when those current states have eventually reached consistency again.

D Acknowledgements

We like to thank the anonymous reviewers as well as our shepherd Prof. Dr. Joachim Posegga for their instructive comments that helped us a lot to improve the paper. The work of Florian Jacob and Saskia Bayreuther is supported by the JuBot project. The JuBot project is funded by the Carl-Zeiss-Foundation.

Bibliography

[Au21] Auvolat, Alex; Frey, Davide; Raynal, Michel; Taïani, François: Byzantine-tolerant causal broadcast. Theoretical Computer Science, 2021.

[Ca99] Castro, Miguel; Liskov, Barbara et al.: Practical byzantine fault tolerance. In: OSDI. volume 99, pp. 173–186, 1999.

[CGR11] Cachin, Christian; Guerraoui, Rachid; Rodrigues, Luís: Introduction to reliable and secure distributed programming. Springer Science & Business Media, 2011.

[Ch07] Chun, Byung-Gon; Maniatis, Petros; Shenker, Scott; Kubiatowicz, John: Attested append-only memory: Making adversaries stick to their word. ACM SIGOPS Operating Systems Review, 41(6):189–204, 2007.

[DLS88] Dwork, Cynthia; Lynch, Nancy; Stockmeyer, Larry: Consensus in the presence of partial synchrony. Journal of the ACM (JACM), 35(2):288–323, 1988.

[Ho21a] Hodgson, Matthew: , Element raises $30M to boost Matrix. https://matrix.org/blog/2021/07/27/element-raises-30-m-to-boost-matrix, 2021. The Matrix.org Foundation C.I.C.

[Ho21b] Hodgson, Matthew: , Germany's national healthcare system adopts Matrix! https://matrix.org/blog/2021/07/21/germanys-national-healthcare-system-adopts-matrix, 2021. The Matrix.org Foundation C.I.C.

[Ja21] Jacob, Florian; Beer, Carolin; Henze, Norbert; Hartenstein, Hannes: Analysis of the matrix event graph replicated data type. IEEE Access, 9:28317–28333, 2021.

[KH20] Kleppmann, Martin; Howard, Heidi: Byzantine eventual consistency and the fundamental limits of peer-to-peer databases. arXiv preprint arXiv:2012.00472, 2020.

[La78] Lamport, Leslie: Time, Clocks, and the Ordering of Events in a Distributed System. Communications of the ACM, 21(7):558–565, 1978.

[LSP82] Lamport, Leslie; Shostak, Robert; Pease, Marshall: The byzantine generals problem. ACM Transactions on Programming Languages and Systems, 4(3):382–401, 1982.

[Sh11] Shapiro, Marc; Preguiça, Nuno; Baquero, Carlos; Zawirski, Marek: Conflict-free replicated data types. In: Symposium on Self-Stabilizing Systems. Springer, pp. 386–400, 2011.

[Th21] The Matrix.org Foundation C.I.C.: Matrix specification v1.1. Technical report, 2021. https://spec.matrix.org/v1.1/.

[Zh16] Zhao, Wenbing: Optimistic byzantine fault tolerance. International Journal of Parallel, Emergent and Distributed Systems, 31(3):254–267, 2016.

Session 3

Recovering information from pixelized credentials

Viktor Garske[1] Andreas Noack[2]

Abstract: Pixelation is a common technique to redact sensitive information like credentials in images. In this paper, we propose a system that is able to recover information from pixelized text. Our contribution consists of a neural network as well as a generic pipeline that generates a realistic training dataset considering flexible specifications including wordlists, fonts, font sizes and letter spacings. The contributed neural network is a composition of a Convolutional Neural Network (CNN), a Recurrent Neural Network (RNN) using Long short-term memory (LSTM) and a Connectionist Temporal Classification (CTC) layer to decode sequences of characters. With our approach, we achieve a Label Error Rate (LER) under 50% when taking pixelation block sizes of up to 8×8 pixels on a 22pt font into account. Thereby, our results indicate that pixelation of sensitive data does not satisfy common privacy standards.

Keywords: Neural networks, privacy, machine learning, computer vision, password, credentials, pixelized

1 Introduction

"A picture is worth a thousand words" – this applies to many situations in everyday life. We use pictures to support our explanations because they can massively accelerate the time we need to understand complex issues. Screenshots showing a misbehavior of an application on GitHub or Stack Overflow as well as linked images on Reddit are common examples how pictures are used on the internet today. On E-commerce platforms like eBay, pictures are mainly used to draw attraction to a specific product.

Fig. 1: Pixelized credential sticker on an Access Point

In most cases, the uploaders of pictures take care of their own privacy, by applying a pixelization (see Fig. 1) on confidential information like postal addresses, public IP addresses, usernames or passwords. Image manipulation tools like *GIMP* provide functions

[1] University of Applied Sciences Stralsund, Department of Electrical Engineering and Computer Science, Zur Schwedenschanze 15, 18435 Stralsund, Germany, viktor.e.garske@hochschule-stralsund.de
[2] University of Applied Sciences Stralsund, Department of Electrical Engineering and Computer Science, Zur Schwedenschanze 15, 18435 Stralsund, Germany, andreas.noack@hochschule-stralsund.de

doi:10.18420/sicherheit2022_08

to redact information in photos or screenshots conveniently. From a security point of view, optically redacted data is in many cases less secure than uploaders think.

This paper focuses on recovering information from images of credentials that are redacted by pixelation. Our contribution is a system incorporating neural networks that allows to recover credential information from cropped redacted images. In our setup, the redacted images are allowed to have a height of only four pixels in the minimum case.

The organization of this paper is as follows: Section 2 deals with related and previous work on the topic of recovering information from redacted images. We give an overview about different types of pixelation in Section 3. Section 4 introduces our approach to gain information from pixelized character sequences using a system of neural networks. In Section 5, we demonstrate how accurate our solution is. Section 6 concludes this paper and provides an outlook.

2 Related Work

The obfuscation of credentials in images has already been investigated in the literature. For instance, McPherson et al. [MSS16] studied the influence of deep learning techniques against image obfuscation. Their work focuses on mosaicing, pixelation and blurring of specific datasets, including the MNIST dataset. Different from our work, McPherson et al. concentrate on individual MNIST digits and do not take sequences of characters, like natural words or passwords, into account.

Also in 2016, Hill et al. [Hi16] presented their in-depth study of mosaicing and blurring as tools for document redaction with respect to character sequences. They use Hidden Markov Models (HMM) to model the underlying text of pixelized images as the hidden states. The trained models are used to infer the text from other pixelized images. Hill et al. note that the effort needed to adjust the parameters like font, font size and pixelation offset is high and furthermore, a parameter mismatch would reduce the recognition accuracy substantially.

Although HMMs constitute a promising approach to handle sequences, Graves et al. [Gr06] choose a different path for training such models. They utilize Recurrent Neural Networks (RNN) and remove additional preprocessing and postprocessing for the up to that point necessary segmentation steps. Their approach using RNNs in combination with the introduced Connectionist Temporal Classification (CTC) reduces training effort and is more robust to temporal and spatial noise. The approach developed by Graves et al. [Gr06] was primarily utilized in the field of speech recognition.

Studies of information recovery in pixelized text images have also been researched from a non-privacy-motivated point of view as Optical Character Recognitions (OCR) face a similar problem. In the OCR case, it is assumed that images have a bad quality which is induced unintentionally. Gilbey et al. [GS21] cover the recognition of ultra-low-resolution printed

text images that were scanned with a resolution of as little as 60 dpi. They follow a human-vision inspired approach that takes into account how our brain deals with low-resolution images. Gaussian Blur or bicubic interpolation reduce the higher-frequency information in low-resolution images and lead to a mean character level accuracy of 99.7%. As typical for OCR, Tesseract is incorporated with a custom model to provide this functionality. Tesseract uses a special RNN architecture called Long short-term memory (LSTM) [HS97] in combination with the CTC presented by Graves et al. Our approach differs because we do not focus on English texts but rather common passwords, which may also contain special characters. Thus, transfer learning with pre-trained Tesseract language models as utilized by Gilbey et al. cannot be applied to our use case directly.

Another application for deciphering low-resolution or pixelized images is the recognition of license plates. Kaiser et al. [Ka21] used CNNs to evaluate opportunities for information recovery relating to low-quality JPEG images from a forensic perspective.

There are two related open-source projects available covering depixelation of passwords. Mellema [Me20] published a tool called *Depix* which tries to resemble the content of pixelized images optically. Schatz [Sc21] implemented the HMM-based approach of Hill et al. [Hi16] in his tool *DepixHMM*. It should be noted that the term *depixelation* refers to the recovery of information in pixelized text as the pixelation itself cannot be inverted.

3 Pixelation Techniques

Pixelation is one of several methods to obfuscate text in an image. The goal is to reduce information up to a point where no human can recognize the redacted text without raising too much visual attention. A pixelation algorithm rasterizes the image with a lower resolution and averages all pixels.

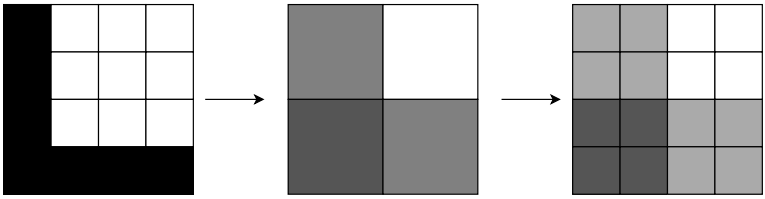


Fig. 2: Method of operation of a character pixelation with a block size of 2×2 pixels. The image with 4×4 pixels (1) will be re-rasterized and the pixel color values will be averaged (2). In the last step (3), the color values of each rectangle in (2) will be assigned to all pixels that lie in this rectangle, i.e. that contributed to the average.

Fig. 2 illustrates that averaging a rectangle of pixels is a non-reversible lossy operation which leaves certain structures behind. Another way to create a similar pixelation effect is

to scale the image down based on the block size and to scale it up using nearest neighbor interpolation again. Both methods are used in practical implementations:

- The open-source image editing software *GIMP* uses GEGL for its operations. Its *pixelize* filter implements the aforementioned averaging technique.
- *ImageMagick* or the popular *Pillow* package in Python do not support pixelation directly, so that developers are going to implement pixelation by using the double resizing technique.

In addition, a pixelation effect occurs unintentionally when a document is photographed with a too low resolution or on a too high distance.

4 Recovery of pixelized information

Pixelation is a common technique to hide credentials in photos while preserving the impression of the whole photo. It reduces the amount of information in an image but does not erase them. Our approach consists of two main components that aim to extract as much information about the underlying text of pixelized images as possible:

1. A **preprocessor** that (a) converts pixelized images to grayscale and (b) aligns the image to left and center, and crops it to a uniform height. This is an important step to increase the accuracy of the following neural network.
2. A **neural network** including a CNN and RNN with the CTC loss function builds the core of our approach.

The proposed system generates key candidates that restrict the search space in order to optimize key recovery attacks.

In the following, we begin with the description of the depixelation training pipeline, which is a generic approach that is also compatible with other solutions. Then, we describe our proposed preprocessing, and the internal structure and details about our neural network. Finally, we add some technical remarks that explain how we increase our systems performance furthermore.

4.1 Training Pipeline

In order to create a good classifier for pixelized texts, a model has to be trained. Our goal is to provide a training concept that is efficient but also fault-tolerant. Fig. 3 shows our training pipeline, which works as follows:

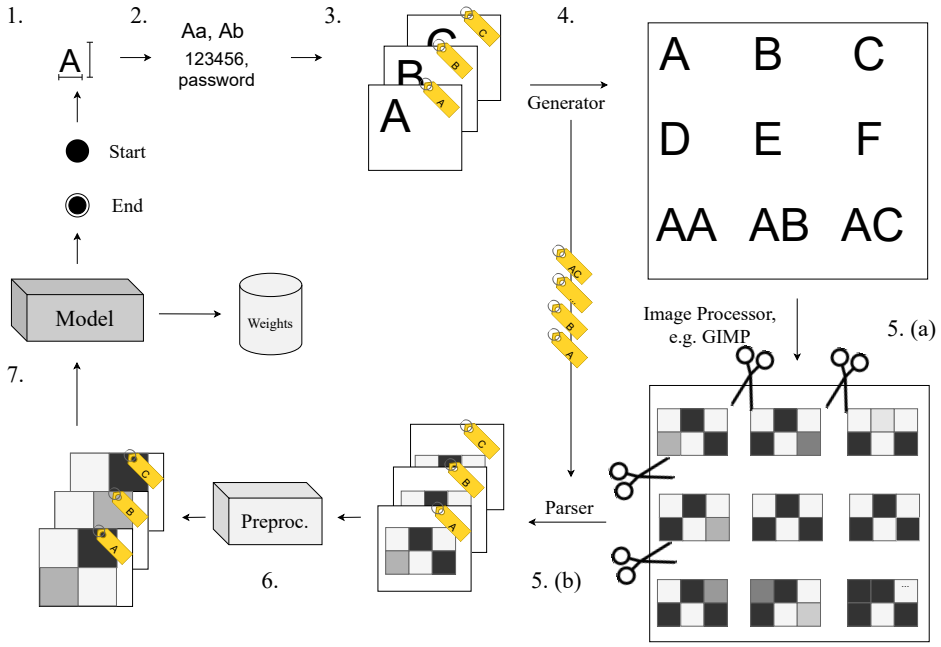


Fig. 3: Pipeline for training the model

1. **Specify parameters**. Select font, font size and pixelation block size.
2. **Generate dictionary**. Permute a predefined alphabet or extract words from a dictionary.
3. **Generate images**. Sequences of words are synthesized on a canvas with a predefined size to make it compatible to the input layer of subsequent analysis steps. The text is truncated if it exceeds the canvas size. To make the system perform more generic and fault-tolerant, noise can be added in an optional step. Further details are explained in a moment.
4. **Concatenate images**. All images of sequences are randomly shuffled and concatenated into a large collage image to allow more efficient handling.
5. **Pixelize collage**. The collage image is (a) fed to an image processor like *GIMP* or *Pillow* in order to apply the pixelation. The resulting image will be cut into single images again and (b) relabeled based on their respective preimage.
6. **Preprocess dataset**. This is an optional step that improves the quality of the pixelized images. Section 4.2 describes our contributed algorithm.
7. **Train the analysis system**. A model is trained with the pixelized images and their corresponding labels.

It is crucial to mimic the environment of the pixelized as good as possible. We assume that high-quality images like screenshots and scans act as the preimage for most pixelations. Thus, we can synthesize the training data similarly. We suggest adding Gaussian noise to train the analysis system with both full-quality and lower-quality images. The noise will be added prior to the pixelation in order to diversify the pixelation results. Additionally, any pixelation result depends on the exact position of the letters so that we also suggest several variations of letter position along the x and y axis to be included in the training set.

4.2 Preprocessing

The preprocessor is applied to pixelized images before training as well as to every test or validation image before it is fed to the neural network. Our main goal for the proposed preprocessor is to align and fit the pixelized letters in the image which are later forwarded to the neural network. First, all images are converted to grayscale with e.g. 256 discrete values. Secondly, we align and scale the image so that it fits on a canvas with a uniform height and width.

In the following, we only describe our aligning and scaling algorithms because the grayscale conversion is done by a standard function. Our algorithm takes roughly cropped pixelized images and removes empty borders first. In order to do so, it scans horizontally and vertically through the image to detect nearly empty borders around the supposed text. If any of the values in each line or column exceeds a threshold value, it will be treated as the first or, respectively, last line or column. After removing the borders, the image is scaled to the desired height using a linear standard function. To further improve the classification accuracy, the image is blurred with a Gaussian blur filter as suggested by Gilbey; Schönlieb [GS21]. Finally, the resulting image is placed on a fixed-width canvas to truncate words that exceed our character limit on the right side.

4.3 Neural Network

Neural networks (NNs) are particularly suitable for classification or regression tasks, as many publications show over the last decades [GR01][MV15][GBC16]. A typical application is image classification, where the network accepts the pixels of an image on the input layer and outputs probabilities for a given number of classifications on the output layer. Trained with a given data set, the neural network is later capable of finding classifications for unknown input data, whereby its accuracy depends on the suitability of the training data and the internal structure of the neural network.

For the depixelation use case, a neural network architecture with two convolutional and pooling layers performs reliably with images of single pixelized characters as shown by McPherson et al. [MSS16]. In the case of longer character/letter sequences, the accuracy

of the naive approach decreases dramatically. There are, however, different approaches to counter the accuracy loss:

- **Pre- and Postprocessing**. An extensive preprocessing of pixelized character sequences produces single images of pixelized characters that can then be fed to the neural network. A postprocessor combines the predicted labels of the neural network with respect to their corresponding probabilities to character sequences, again.
- **Connectionist Temporal Classification**. The CTC loss function that is used in combination with the Recurrent Neural Network (RNN) was proposed by Graves et al. [Gr06]. In the area of speech recognition, research results indicate that it outperforms the Hidden Markov Model (HMM) approach. HMM is one of the most promising approaches to deal with our use case, namely depixelation, as shown by Hill et al. [Hi16].

For this paper, we choose the Connectionist Temporal Classification (CTC) approach by Graves et al. [Gr06] and adapt it to our use case, the depixelation of credentials.

The core of a CTC-fitted recurrent neural network (RNN) is the loss function that does not only consider the raw input data but also takes the length of the input into account. The CTC-RNN implementation by Soullard et al. [SRP19] uses the edit distance, which is explained later, in order to calculate the similarity between the predicted and the real label.

After preprocessing, we proceed with one channel grayscale images in the dimension of 192×32 pixels that are fed into the CNN. The neural network architecture was inspired by ResNet [He15] and DenseNet [HLW16], and we make use of skip connections to enable more efficient training. In comparison to ResNet, we use fewer layers, again for efficiency reasons. Fig. 6 (Appendix) gives a structural overview about the complete neural network architecture. To avoid overfitting, Gaussian noise is added at the beginning of the training.

Intermediate layers prepare the output of the CNN for the following LSTM by flattening and reshaping. Two LSTM layers process the output and feed it into the CTC layer. This CTC layer combines the output of the LSTMs together with the content width of the pixelized image, and the value and length of its corresponding label sequence.

4.4 Technical Remarks

We design our solution to be as close as possible to a real world pixelation scenario by deploying standard fonts and font sizes as well as standard tools like *GIMP* for the pixelation process.

For realistic results, we use a standard wordlist from the *SecLists* project by Miessler et al. [MHg21]. The selected wordlist is named `10-million-password-list-top-10000.txt` and

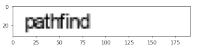
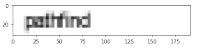
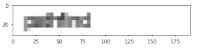
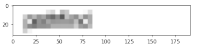
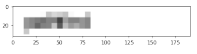
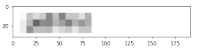
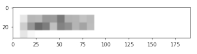
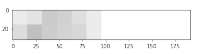
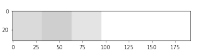
contains the top 10,000 entries of a collection compiling the 10 million most used passwords. Entries include combinations of letters, numbers and special characters in one word.

We noticed that a pixelation result does not only depend on the image but also on its position in the coordinate system, i.e. after realigning two pixelations from the same image at different positions, the result differs. To improve the prediction results, we let our image synthesizer generate text images at several positions on the canvas to cover the variations in the pixelation results.

The text image synthesis will be performed by the Python *Pillow* package [LCC21]. Furthermore, we modify the functionality of the `ImageDraw` class because we found out that it did not take letter spacings into account. In our pixelation pipeline, we call *GIMP* in batch mode using the *PNG* file format with compression level 6, whereby level 1 has the best speed and level 9 the best compression rate.

5 Results

The performance of the proposed information recovery system depends on the used font size relative to the size of the blocks that are created in the pixelation step. We use the Arial font and define a font size of 22pt for our tests and evaluate the error rate in dependency of pixel block sizes between 2×2 and 32×32 pixels, whereby our canvas has a maximum height of 32 pixels. As you can see in Tab. 1, a pixelation block size of 8×8 pixels results in a height of four blocks. The last two block sizes, 16×16 pixels and 32×32 demonstrate edge cases.

Block Size	Example Image	Label Error Rate	Sequence Error Rate
2×2		2.99%	17.97%
3×3		16.00%	57.00%
4×4		20.47%	65.63%
5×5		36.17%	87.50%
6×6		28.92%	78.90%
7×7		41.17%	90.63%
8×8		49.93%	97.66%
16×16		61.09%	99.22%
32×32		69.25%	99.22%

Tab. 1: Error rate of the proposed depixelation system, see Fig. 4 for chart illustration.

For comparability reasons, we use two established metrics, the Label Error Rate (LER) proposed by Graves et al. [Gr06] and the Sequence Error Rate (SER) by Soullard et al. [SRP19].

The Label Error Rate (LER) metric measures the **mean normalized** edit distances between classifications and preimage labels and is defined as follows:

$$LER(h, S') = \frac{1}{|S'|} \sum_{(\mathbf{x},\mathbf{z}\in S')} \frac{ED(h(\mathbf{x}), \mathbf{z})}{|\mathbf{z}|} \tag{1}$$

whereby the LER is computed for a temporal classifier h (i.e. the model) and a given test set S'. x stands for the input sequences and z for corresponding labels. Both are compared with the edit distance function $ED(\cdot, \cdot)$ which is the number of characters that must be changed to make both parameters equal. The LER results in Tab. 1 approve our assumption that the error rate increases proportional to the block size, Fig. 4 (Appendix) shows even better that there is a linear relationship.

As a second metric, we choose the Sequence Error Rate (SER) that reveals the percentage of **completely correct** predictions in the test set. Our test results show that the probability to predict a pixelized text without any errors drops dramatically with increasing information loss due to larger pixelation block sizes.

As a consequence of different approaches and data representations, a comparison to previous publications is a challenging task. When comparing to [MSS16], we have to keep in mind that this experiment focused on individual pictures of digits in the MNIST dataset and not on sequences of letters, digits and special characters, which emerges as an easier problem. We can show that a reasonable recognition rate is still realistic, even if character sequences instead of single digits are investigated.

The HMM approach by Hill et al. [Hi16] considers character sequences. Their goal is to specifically train an HMM-based recognizer for a given set of specifications. This includes finding the correct offset, as well as other parameters like font size or font type. As a result, this approach requires high effort for each image sample but outperforms more-general approaches like ours, especially at high mosaic grid sizes. Unfortunately, we could not examine and compare our results in detail as their code and data set for the claimed accuracy is not public. Our results indicate that our system is slightly more robust to offset variations than the HMM approach as presented in Fig. 5 (Appendix).

It becomes clear that the probability for a successful information recovery depends on the block size and training. Higher block sizes result in higher LERs which in turn lead to higher privacy probabilities for the redacted text. Proper training and dataset selection, however, is vital to enable low LERs. Consequently, our network design leads to suitable results in less than 15 epochs of training using a dataset of about 25,000 samples. Nonetheless, training has to be monitored with respect to loss metrics to ensure a high quality.

Predictions of the network have to be assessed by a human operator in order to evaluate the plausibility in respect to the supposed content. We examined our neural network structure for superfluous elements and could observe an essential role of the CNN part. When removed or replaced by more dense layers, the LER rises up to 50 percentage points.

6 Conclusion and Future Work

In this paper we have analyzed the privacy of pixelation whereby our research is especially focused on the obfuscation of credentials like passwords. We make two major contributions, firstly, (1) a NN-based information recovery system for pixelized text in images with a competitive accuracy compared to previous HMM-based approaches and secondly, (2) a generic and flexible training pipeline for depixelation systems.

The proposed system consists of a combination of Convolutional Neural Network (CNN) and Long short-term memory (LSTM) network together with a Connectionist Temporal Classification (CTC) layer. Thereby, we reach a mean error rate, namely the Label Error Rate (LER), better than 50% with pixelation block sizes up to 8×8 pixels. Our results indicate that the proposed system still works even when humans are no longer able to depixelize images anymore. This leads to a substantial privacy impact on the security of pixelation algorithms used in standard software. However, the performance of our approach decreases if sufficient high block sizes are used. We discovered that the label error rate rises above 70% if a block is at least as big as the pixelized character.

Furthermore, we developed a complete training pipeline for obfuscation problems that comprises a generic approach for various depixelation concepts, e.g. based on HMMs or NNs, and is able to employ standard software as subroutines. The training pipeline allows efficient and automated training, while it is flexible in terms of parameter variations.

In the future, we plan to investigate several techniques like ensemble methods or the separation of image classification and content recognition to further improve the error rate of our system. Additionally, our network architecture is going to be revised with the idea to find alternatives for the CNNs or LSTMs in order to either decrease the training effort or improve the accuracy.

References

[GBC16] Goodfellow, I.; Bengio, Y.; Courville, A.: Deep learning. MIT press, 2016.

[GR01] Giacinto, G.; Roli, F.: Design of effective neural network ensembles for image classification purposes. Image and Vision Computing 19/9-10, pp. 699–707, 2001.

[Gr06] Graves, A.; Fernández, S.; Gomez, F.; Schmidhuber, J.: Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In: Proceedings of the 23rd International Conference on Machine Learning. ICML '06, Association for Computing Machinery, Pittsburgh, Pennsylvania, USA, pp. 369–376, 2006, ISBN: 1595933832, URL: https://doi.org/10.1145/1143844.1143891.

[GS21] Gilbey, J.; Schönlieb, C.-B.: An end-to-end Optical Character Recognition approach for ultra-low-resolution printed text images. arXiv preprint arXiv:2105.04515/, 2021.

[He15] He, K.; Zhang, X.; Ren, S.; Sun, J.: Deep Residual Learning for Image Recognition. CoRR abs/1512.03385/, 2015, arXiv: 1512.03385, URL: http://arxiv.org/abs/1512.03385.

[Hi16] Hill, S.; Zhou, Z.; Saul, L.; Shacham, H.: On the (in) effectiveness of mosaicing and blurring as tools for document redaction. Proceedings on Privacy Enhancing Technologies 2016/4, pp. 403–417, 2016.

[HLW16] Huang, G.; Liu, Z.; Weinberger, K. Q.: Densely Connected Convolutional Networks. CoRR abs/1608.06993/, 2016, arXiv: 1608.06993, URL: http://arxiv.org/abs/1608.06993.

[HS97] Hochreiter, S.; Schmidhuber, J.: Long Short-term Memory. Neural computation 9/, pp. 1735–80, Dec. 1997.

[Ka21] Kaiser, P.; Schirrmacher, F.; Lorch, B.; Rieß, C.: Learning to Decipher License Plates in Severely Degraded Images. In: MultiMedia FORensics in the WILD - accepted for publication. Jan. 11–11, 2021, URL: https://faui1-files.cs.fau.de/public/publications/mmsec/2021-Schirrmacher-LTD.pdf.

[LCC21] Lundh, F.; Clark, A.; Contributors: Pillow: The friendly PIL fork (Python Imaging Library) repository, Version 8.3.1, Apr. 1, 2021, URL: https://github.com/python-pillow/Pillow/tree/8.3.1.

[Me20] Mellema, S.: Depix, 2020, URL: https://github.com/beurtschipper/Depix.

[MHg21] Miessler, D.; Haddix, J.; g0tmi1k: SecLists, Mar. 1, 2021, URL: https://github.com/danielmiessler/SecLists.

[MSS16] McPherson, R.; Shokri, R.; Shmatikov, V.: Defeating Image Obfuscation with Deep Learning, 2016, arXiv: 1609.00408 [cs.CR].

[MV15] McDonnell, M. D.; Vladusich, T.: Enhanced image classification with a fast-learning shallow convolutional neural network. In: 2015 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–7, 2015.

[Sc21] Schatz, J.: DepixHMM, 2021, URL: https://github.com/JonasSchatz/DepixHMM.

[SRP19] Soullard, Y.; Ruffino, C.; Paquet, T.: CTCModel: a Keras Model for Connectionist Temporal Classification, 2019, arXiv: 1901.07957 [cs.LG].

A Additional Information

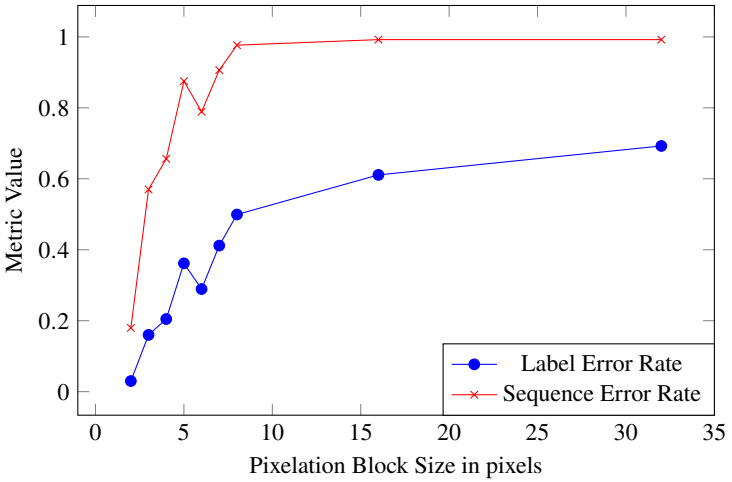


Fig. 4: Error rates in relation to the pixelation block size

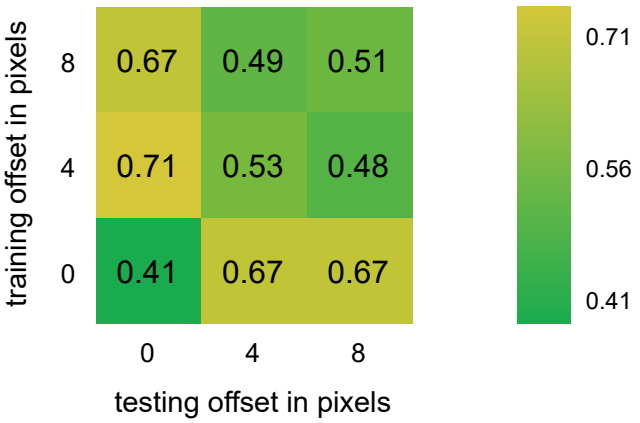
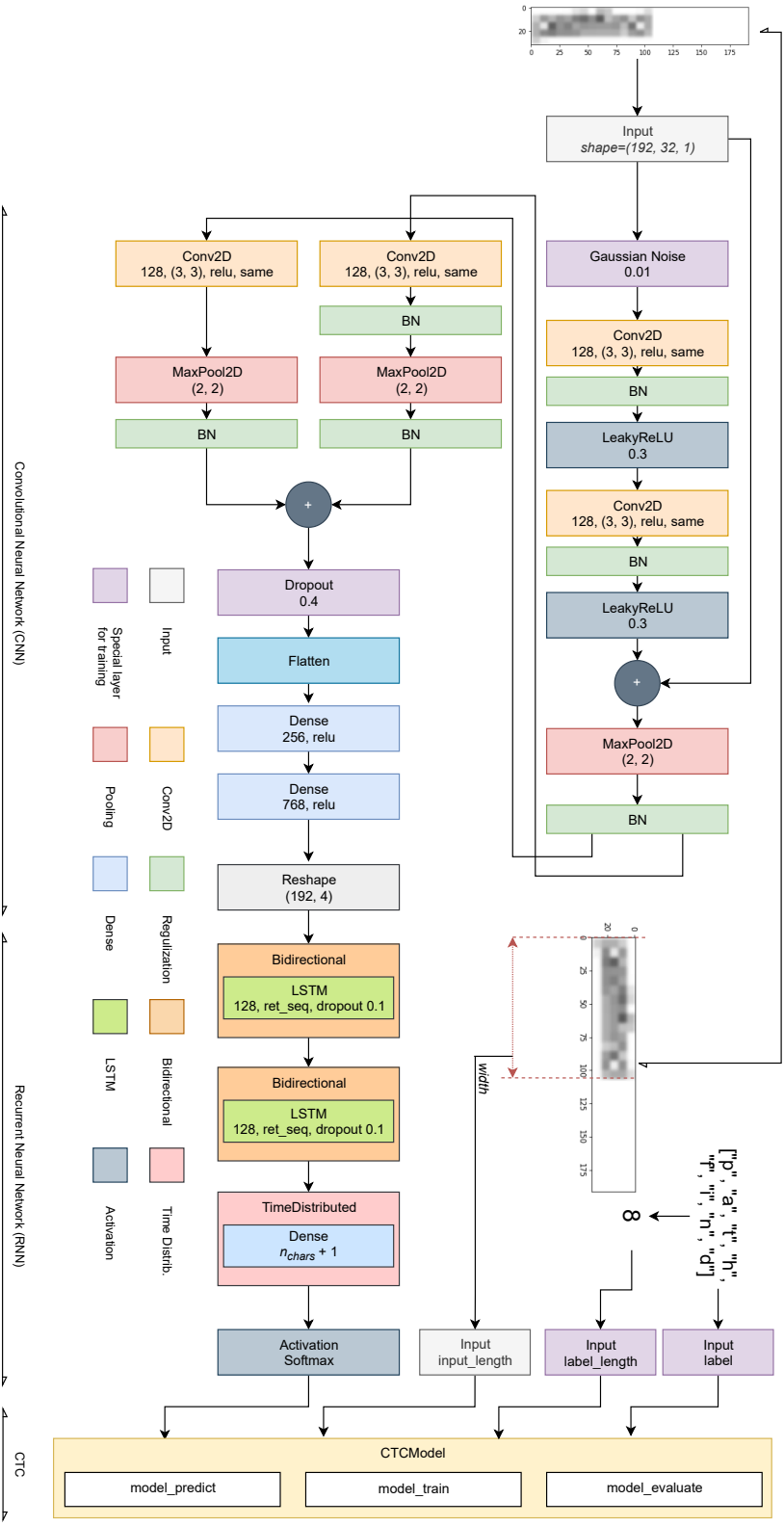


Fig. 5: A LER confusion matrix for offset variations when using a pixelation block size of 6×6 pixels


Input
shape=(192, 32, 1)
Conv2D
128, (3, 3), relu, same
Conv2D
128, (3, 3), relu, same
BN
MaxPool2D
(2, 2)
MaxPool2D
(2, 2)
BN
BN
Gaussian Noise
0.01
Conv2D
128, (3, 3), relu, same
BN
LeakyReLU
0.3
Conv2D
128, (3, 3), relu, same
BN
LeakyReLU
0.3
MaxPool2D
(2, 2)
BN
Dropout
0.4
Flatten
Dense
256, relu
Dense
768, relu
Reshape
(192, 4)
Bidirectional
LSTM
128, ret_seq, dropout 0.1
Bidirectional
LSTM
128, ret_seq, dropout 0.1
TimeDistributed
Dense
n_{chars} + 1
Activation
Softmax
Input
input_length
Input
label_length
Input
label
width
8
["p", "a", "t", "h", "n", "p"]
Convolutional Neural Network (CNN)
Recurrent Neural Network (RNN)
CTC
Special layer for training
Input
Pooling
Conv2D
Dense
Regulzation
LSTM
Bidirectional
Activation
Time Distrib.
CTCModel
model_predict
model_train
model_evaluate

Sicherheit medizintechnischer Protokolle im Krankenhaus

Christoph Saatjohann[1] Fabian Ising[1] Matthias Gierlings[2] Dominik Noss[2] Sascha Schimmler[3] Alexander Klemm[4] Leif Grundmann[5] Tilman Frosch[3] Sebastian Schinzel[1]

Abstract: Medizinische Einrichtungen waren in den letzten Jahren immer wieder von Cyber-Angriffen betroffen. Auch wenn sich diese Angriffe derzeit auf die Office-IT-Infrastruktur der Einrichtungen konzentrieren, existiert mit medizinischen Systemen und Kommunikationsprotokollen eine weitere wenig beachtete Angriffsoberfläche.

In diesem Beitrag analysieren wir die weit verbreiteten medizintechnischen Kommunikations-Protokolle DICOM und HL7 sowie Protokoll-Implementierungen auf ihre IT-Sicherheit. Dafür präsentieren wir die Ergebnisse der Sicherheitsanalyse der DICOM- und HL7-Standards, einen Fuzzer "MedFUZZ" für diese Protokolle sowie einen Schwachstellenscanner "MedVAS", der Schwachstellen in medizintechnischen Produktivumgebungen auffinden kann.

Keywords: DICOM; HL7; TLS; Netzwerksegmentierung; Scanning; Fuzzing; Telematikinfrastruktur

1 Einleitung

Kürzlich bekannt gewordene Angriffe gegen medizinische Einrichtungen, z. B. gegen das Uniklinikum Düsseldorf, zielten auf die generische Office-IT-Infrastruktur ab. Jedoch sind auch medizinische Systeme, wie etwa bildgebende Systeme und Bildarchive, nicht vor Angriffen gefeit. So zeigte 2019 zum Beispiel die Firma Greenbone, dass zahlreiche DICOM-Systeme öffentlich zugänglich im Internet verfügbar waren und dort auch reale Patientendaten von Dritten heruntergeladen werden konnten [Gr19]. Obwohl diese Schwachstellen bekannt und medial aufbereitet wurden, konnten 2020 immer noch DICOM-Systeme sowie weitere Praxen- und Krankenhaus-Systeme in internetweiten Scans nachgewiesen werden [STB20].

In diesem Beitrag analysieren wir die gängigen medizintechnischen Protokolle DICOM und HL7 sowie zwei Implementierungen mit Fokus auf IT-Sicherheit.

Wissenschaftlicher Beitrag In Abschnitt 3 und 4 präsentieren wir Sicherheitsanalysen der DICOM- und HL7-Standards und beleuchten dabei die Aspekte Authentifizierung, Vertraulichkeit, Authentizität und Integrität. Wir betrachten außerdem Sicherheitslücken, die wir durch Fuzzing einer DICOM- und einer HL7-Implementierung aufgedeckt haben. Außerdem geben wir einen Überblick über in der Realität eingesetzte Sicherheitsmaßnahmen bei DICOM-Systemen. In Abschnitt 5 präsentieren wir unsere Plattform zum Testen

[1] FH Münster
[2] Ruhr-Universität Bochum
[3] G Data Advanced Analytics GmbH
[4] radprax Holding GmbH & Co.KG
[5] MedEcon Ruhr GmbH

doi:10.18420/sicherheit2022_09

medizinischer Software und Systeme. Hierbei liegt ein besonderes Augenmerk auf den beiden Hauptkomponenten, dem Fuzzer "MedFUZZ" und dem Scanner "MedVAS". Wir beschreiben insbesondere auch Erkenntnisse, die wir mit dem Scanner in realen Krankenhausumgebungen gemacht haben. In Abschnitt 6 fassen wir unsere Ergebnisse zusammen. Aufbauend geben wir in Anhang A konkrete Handlungsempfehlungen für Nutzende und Hersteller medizinischer Systeme und Vorschläge für die Weiterentwicklung der Standards.

Verwandte Arbeiten Klick et al. analysierten 2021 [KKB21] aus dem öffentlichen Internet aus erreichbare IP-Adressen deutscher Krankenhäuser und scannten dafür knapp 100 Ports pro IP-Adresse auf bekannte Services und Schwachstellen. Diese externen Scans zeigten bei 36% der Krankenhäuser bekannte Schwachstellen.

Eichelberg et al. untersuchten 2020 in einer Reihe von Papieren die Herausforderungen der IT-Sicherheit für die bildgebende Medizin [EKK20a], den aktuellen Literaturstand zur IT-Sicherheit in diesem Bereich [EKK20b] und die praktischen Probleme bei der Implementierung von IT-Sicherheit im Krankenhaus [EKK20c].

2018 befassten sich Dameff et al. [Da18] mit der Sicherheit von HL7. Die Autoren zeigten, dass es durch einfache TCP-MitM-Angriffe möglich ist, die Integrität von Patientendaten zu brechen. Anirudh Duggal zeigte 2017 ähnliche Angriffe [Du17]. Neben Denial-of-Service-Angriffen untersuchte der Autor hierbei auch mögliche Implementierungsfehler. 2019 untersuchten Wang et al. Fuzzing zum Testen von DICOM-Implementierungen [Wa19].

Danksagungen Dieser Beitrag wurde im Rahmen des Forschungsprojektes MITSicherheit.NRW (EFRE-0801419) vom Europäischen Fonds für regionale Entwicklung Nordrhein-Westfalen (EFRE.NRW) gefördert.

2 Grundlagen

DICOM Der Digital Imaging and Communications in Medicine (DICOM) Standard [Na21] definiert ein binäres Datenformat zur Speicherung und ein Kommunikationsprotokoll zum Austausch von medizinischen Bilddaten. Durch DICOM wird eine weltweite, herstellerunabhängige Interoperabilität der Daten von bilderzeugenden Geräten (Modalitäten, z.B. Computertomograph, Magnetresonanztomograph) und bildverarbeitenden, beziehungsweise -speichernden Systemen, z.B. dem Picture Archiving and Communication System (PACS), erreicht. Dadurch stellt es den De-facto-Standard für Bilddatenmanagement in der Medizin dar. Im Rahmen dieses Beitrags ist insbesondere das DICOM-Kommunikationsprotokoll von Interesse. Das DICOM Message Service Element (DIMSE)-Protokoll spezifiziert dabei den Austausch von Bilddaten und zugehörigen Daten. Nachrichten bestehen aus einem sogenannten "Command Set" und einem (optionalem) "Data Set". Einige relevante Kommandos des DIMSE-Protokolls sind in Tabelle 1 aufgelistet.

Kommando	Beschreibung
A-ASSOCIATE-RQ	Verbindungsanfrage
C-STORE	Ablage von Informationen
C-GET	Abruf von Informationen
C-ECHO	Echo-Funktionalität für Verbindungstests

Tab. 1: Liste der für diesen Beitrag relevanten DICOM-Kommandos, nach [Na21, Part 7].

Bei der Kommunikation identifizieren sich verschiedene DICOM-Systeme durch die Nutzung von Application Entity Titles (AETs). Dies sind maximal 16 Zeichen lange, netzwerkweit eindeutige Kennungen, wobei ein System auch mehrere AETs für verschiedene Komponenten nutzen kann [Na21, Part 5, Part 8]. Während des DICOM-Verbindungsaufbaus werden durch zwei Felder Informationen über die verwendete Implementierung ausgetauscht: "Implementation Class UID" (notwendig) und "Implementation Version Name" (optional).

HL7 Health Level 7 (HL7) International beschreibt mit HL7 eine Menge von Standards für den Austausch klinischer und administrativer Daten zwischen medizinischen Anwendungen. Obwohl seit 2005 HL7 Version 3 [He17] als neuer Standard verfügbar ist, ist Version 2 immer noch deutlich verbreiteter [Jo18]. Auch die Bereitstellung von HL7 über Webschnittstellen, genannt Fast Healthcare Interoperability Resources (FHIR), ist erst wenig verbreitet.

Die Nachrichtenkodierung in HL7 V2 ist prinzipiell mit XML möglich, meist wird aber das ASCII-basierte, menschenlesbare Electronic Data Interchange (EDI) Datenformat verwendet (siehe Beispiel in List. 1). Neben den fest definierten Nachrichtenelementen unterstützt HL7 applikationsspezifische Segmente und verschiedene Nachrichtenprofile. Durch diese Flexibilität im Standard haben sich verschiedene, teilweise inkompatible, Profile und Formate in der Praxis durchgesetzt. Für diesen Beitrag relevante Felder sind die `Sending Facility` und `Sending Application` in welchen Informationen über die sendende Anwendung und Institution angegeben werden können. Der HL7 V2 Standard wurde seit der Version 2.0 im Jahre 1989 stetig weiterentwickelt bis zum jetzigen Standard V2.9 [He19].

Telematikinfrastruktur-Konnektor Seit 2018 sind alle Arzt- und Zahnarztpraxen verpflichtet, einen *Konnektor* in der Praxis zu installieren [Bu15]. In weiteren Ausbaustufen der Telematikinfrastruktur (TI) wurden, und werden, schrittweise weitere Institutionen, wie Apotheken und Krankenhäuser, mit in die Anschlusspflicht genommen. Dieser Konnektor verbindet, unter anderem, das Praxisnetzwerk über einen VPN-Tunnel mit dem zentralen TI-Netz. Damit Clients aus dem internen Netzwerk mit dem Konnektor kommunizieren können, wird auf der internen LAN-Schnittstelle per HTTP eine Konnektorbeschreibungsdatei, *connector.sds*, angeboten, welche Informationen über Versionsstand, Konfiguration bestimmter Parameter und verfügbare Dienste bereitstellt.

```
MSH|^~\&||<sender>|||<dateTime>||<messageType>|<messageID>|
     <processingStatus>|<syntaxVersion>
PID|||<patientID>^^^<source>^<IDType>||<familyName>^<givenName>||
     <dateOfBirth>|<sex>|||<address>
PV1|||<patientLocation>|||||<patientsGP>
OBR|||<accessionNumber>|<testCode>^<testName>^<codeType>|||
     <specimenDate>||||||||<specimenSource>|<requester>
OBX||<valueType>|<observableCode>^<observableName>|<subID>|<valueCode>^
     <valueText>^<valueCodeType>||||||<abnormalFlag>
```

List. 1: Aufbau einer HL7 Version 2 Nachricht: Jede Zeile definiert ein Segment der HL7 Nachricht. Die einzelnen Felder werden mit | getrennt, ^ trennt Unterfelder. Beim ersten Feld handelt es sich jeweils um den Segment Typ. Felder in <> enthalten Patientendaten. Quelle: [BG16]

Schwachstellen-Scanning Schwachstellen-Scanner werden genutzt, um in Netzwerken Schwachstellen in Software und Geräten aufzudecken. Dabei werden sowohl Fehlkonfigurationen als auch, gegebenenfalls veraltete, Software mit Sicherheitslücken entdeckt. Ein freier und weitverbreiteter Schwachstellen-Scanner ist der Open Vulnerability Assessment Scanner (OpenVAS) [Gr]. Scanning ist ein wichtiger Teil des Schwachstellenmanagements [Bu].

Schwachstellen-Scanner können üblicherweise nur bereits bekannte Schwachstellen finden, sind aber nicht in der Lage neue Sicherheitslücken aufzudecken. So müssen Scanner zum Beispiel für das Scannen medizinischer Produkte zuvor um Module für die speziellen Netzwerkprotokolle und damit verbundene bekannte Sicherheitslücken erweitert werden.

Fuzzing Als Fuzzing oder Fuzz-Testing wird eine Technik zum automatisierten Auffinden von Sicherheitslücken in Software bezeichnet. Hierbei wird die zu testende Software mit, für diese, unerwarteten Eingaben getestet. Dieser Prozess wird dabei automatisch in hoher Frequenz durchgeführt. Ein Fuzzer generiert dabei die Eingabedaten üblicherweise aus existierenden Daten (mutation-based fuzzing) oder komplett neu auf Basis des verwendeten Protokolls oder Dateiformats oder komplett zufällig (generation-based fuzzing). [SGA07]

Es wird beim Fuzzing allgemein zwischen dem Blackbox-Fuzzing, bei dem eine als Binärdatei vorliegende Software ohne zusätzliche Informationen gefuzzt wird, und dem Guided Fuzzing unterschieden. Beim Guided Fuzzing werden zusätzliche Informationen genutzt, um bessere Eingabedaten zu generieren, die einen größeren Bereich der Software testen. Diese zusätzlichen Informationen werden üblicherweise durch Instrumentierung der Binärdatei, häufig während des Kompilierens, gewonnen. Diese Instrumentierung erlaubt etwa nachzuvollziehen, welche Pfade im Programm genommen werden. [Ma19]

Klassischerweise deckt Fuzzing Speicherprobleme und undefiniertes Verhalten in nicht speichersicheren Sprachen (zum Beispiel C oder C++) auf. Es kann aber auch nach anderen Fehlern, etwa nach unauthentifizierten Datenzugriffen oder Exceptions, gesucht werden.

3 Sicherheitsanalyse DICOM

Bei der Sicherheitsanalyse des DICOM-Standards betrachten wir die Sicherheitseigenschaften Authentifizierung, Verschlüsselung, Authentizität und Sicherheit ruhender Daten. Zuletzt legen wir dar, woran die Nutzung von Sicherheitsmechanismen in der Praxis scheitert.

Authentifizierung Die "User Identity Negotiation" [Na21, Part 7] erlaubt die Authentifizierung von Benutzenden mittels Benutzername, Benutzername und Passwort, Kerberos Tickets, SAML oder JSON Web Tokens (JWTs). Die Authentifizierung wird hierbei beim Aufbau einer Verbindung mit der `A-ASSOCIATE-RQ`-Nachricht durchgeführt. Hierbei ist ausschließlich eine Authentifizierung des Anfragenden, nicht des Servers vorgesehen.

Auch wenn AETs nicht explizit für die Authentifizierung vorgesehen sind, so erlauben viele reale Systeme nur den Zugriff mit freigegebenen AETs. Da diese aber nur maximal 16 Zeichen lang sind, häufig sprechende Namen wie Abteilungskürzel haben, und üblicherweise im Klartext ausgetauscht werden, stellt dies keine effektive Authentifizierung dar.

Transportverschlüsselung Der DICOM-Standard definiert die Kommunikation über TLS-verschlüsselte Verbindungen im optionalen "BCP 195 TLS Secure Transport Connection Profile" [Na21, Part 15], das den Vorgaben der Empfehlungen von BCP 195 [SHSA15] folgt. Dadurch wird die Verwendung von TLS 1.2 und 1.3 empfohlen, vor TLS 1.1 und 1.0 wird gewarnt. Strikter ist hier das optionale "Non-Downgrading BCP 195 TLS Secure Transport Connection Profile", das die Verwendung von TLS 1.0 und 1.1 verbietet und Ciphersuites mit kurzlebigen Schlüsseln vorschreibt. Kritisch ist hier, dass das DICOM Komitee die Verteilung und Generierung von Zertifikaten als außerhalb des Geltungsbereichs definiert, was die Implementierung in der Praxis stark erschwert. Gespräche mit IT-Verantwortlichen medizinischer Einrichtungen ergaben, dass TLS bei DICOM praktisch nicht eingesetzt wird.

Authentizität Um die Authentizitätsprüfung von DICOM-Daten zu ermöglichen, definiert der DICOM-Standard "Digital Signature Profiles" [Na21, Part 15], die beschreiben, wie ein Hash ausgewählter DICOM Daten signiert werden kann.[6] Dabei wird ausschließlich die Signatur mithilfe von EMSA-PKCS1-v1_5 [KS98] und auch explizit nur die Verwendung von RSA-Schlüsseln definiert. Die weiteren Signaturprofile definieren dabei, welche Datenelemente spezifischer DICOM-Objekte signiert werden sollten.

Bei den Profilen ist es problematisch, dass neben akzeptablen Hashalgorithmen wie SHA256, SHA384 und SHA512 und RIPEMD160 auch noch MD5 und SHA1 erlaubt sind, welche aufgrund praktischer Angriffe auf ihre Kollisionsresistenz als unsicher gelten [WY05, St17].

[6] Kurioserweise spricht der Standard hier von "MAC" (Message Authentication Code), obwohl hier mutmaßlich nur normale Hashes gemeint sind.

Weiterhin ist auch bei der Definition von TLS für Signaturen die Verteilung von Zertifikaten nicht näher spezifiziert, sondern wird als "site-specific" definiert. Auch die Einschränkung auf RSA-Zertifikate ist nicht mehr zeitgemäß. Am kritischsten ist jedoch, dass Signaturen ohne weiteres durch einen Angreifer entfernt werden können, ohne dass dies erkennbar wäre. Nur wenn ein System die Verwendung von Signaturen erzwingt, würde dies auffallen.

Sicherheit ruhender Daten Neben den Sicherheitsprofilen für den Austausch von DICOM-Daten, definiert der DICOM-Standard auch Sicherheitsprofile für ruhende Daten, die "Media Storage Security Profiles" [Na21, Part 15]. Darunter ist ein spezifisches Profil für die Kapselung von DICOM Dateien in "Secure DICOM Files" definiert. Hierbei werden die Daten mit AES oder Triple-DES verschlüsselt, und können entweder mit einer RSA-Signatur oder einem Hash geschützt werden. Die DICOM-Objekte werden dazu mithilfe der Cryptographic Message Syntax (CMS) verpackt. Zum Schutz des Content Encryption Keys kann entweder RSA oder die PBKDF2 genutzt werden.

Fragwürdig ist hierbei, dass der als unsicher geltende [BL16] Triple DES Algorithmus noch immer erlaubt ist, von dessen Nutzung bereits seit einigen Jahren abgeraten wird [NI17]. Die verwendbaren Hashalgorithmen sind das unsichere SHA1 und die aktuell als sicher eingestuften SHA256, SHA384, SHA512. Für Signaturen wird erneut nur RSA erlaubt.

DICOM in der Praxis In der Praxis scheitert die Implementierung der Sicherheitsmechanismen, neben fehlender Unterstützung durch die Produkte, an dem Fehlen von Zertifikaten. Da der Standard keine Vorgaben zur Erstellung und Verteilung von Zertifikaten macht und in medizinischen Praxen keine Strukturen dafür bestehen, werden nach unserer Erfahrung weder TLS noch digitale Signaturen eingesetzt. Auch Eichelberg et al. stellten 2020 das Zertifikatsmanagement als eine zentrale Herausforderung in diesem Bereich dar [EKK20c].

4 Sicherheitsanalyse HL7

Im aktuellen HL7 2.9-Standard liegt die Datensicherheit, konkret genannt werden die Verschlüsselung und Authentizität von Medizindaten, explizit nicht im Geltungsbereich des Standards [He19, Kapitel 1.8.2]. Die Verantwortung für die rechtliche und technische Einschätzung wird explizit den HL7-Implementierenden überlassen.

Während der Standard weder Sicherheitsziele noch -maßnahmen vorgibt, wurde 1999 ein sogenannter *non-balloted Standard Guide* veröffentlicht [Bl99], welcher technisch sehr detailliert Sicherheitsziele wie Vertraulichkeit, Authentizität und Integrität erklärt. Für die Erreichung dieser Ziele wird der Einsatz von TLS zwischen den Kommunikationspartnern empfohlen. Zu kritisieren ist, dass der Einsatz von TLS nur optional ist, und auch bis heute nicht vom Standard verpflichtend vorgegeben wurde. Auch der Standard Guide wurde seit Veröffentlichung nicht aktualisiert und enthält entsprechend veraltete Empfehlungen.

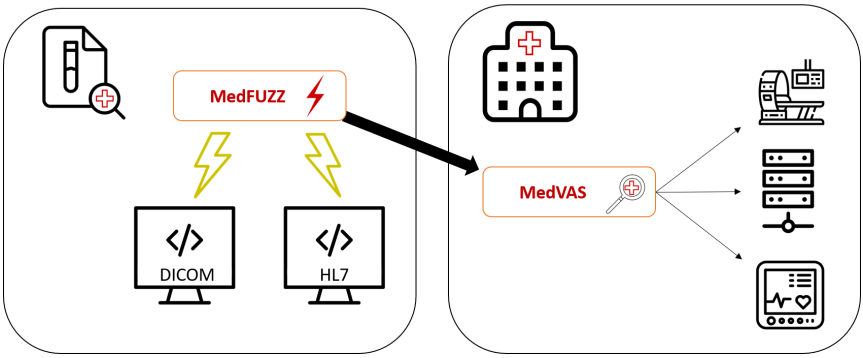


Abb. 1: Plattform zum Testen medizinischer Software und Systeme.

Beispielsweise werden im Zusammenhang mit TLS die unsicheren Algorithmen SHA1 und MD5 als “secure hash functions” genannt. Da es zum Veröffentlichungszeitpunkt noch keinen Advanced Encryption Standard (AES) gab, enthalten die Auflistungen der zu nutzenden Verschlüsselungen weitere unsichere Algorithmen wie RC2, RC4 oder DES.

Eine HL7-Implementierung, die sich allein auf diesen Leitfaden verlässt, und nicht eigenständig eine Aktualisierung auf den Stand der Technik durchführt, wird als Folge dessen mit großer Wahrscheinlichkeit verwundbar gegen bekannte Angriffe sein.

Das HL7 HCS Dokument aus dem Jahr 2014 [Se14] beschreibt ein interoperables Format für Sicherheits-Klassifizierung von HL7-Objekten. Während der Standard fünf verschiedene Sicherheits-Einstufungen definiert, bleibt die konkrete Umsetzung der Sicherheitsstufen den jeweiligen Softwareherstellern und Systembetreibern überlassen. Da keine weiteren technischen Erläuterungen verfügbar sind, und auch die verlinkten Pilotprojekte nicht mehr existieren, gehen wir davon aus, dass dieser Standard in der Praxis wenig relevant ist.

5 Softwareplattform zum Testen medizinischer Software und Systeme

Unsere Plattform zum Testen medizinischer Software und Systeme besteht aus zwei Komponenten, die in Abbildung 1 dargestellt sind. In einem ersten Schritt wurden in einer Laborumgebung Implementierungsfehler in DICOM- und HL7-Software durch den Fuzzer “MedFUZZ” identifiziert. Informationen über Software mit diesen Fehlern wurden in den auf OpenVAS basierenden Scanner “MedVAS” integriert, sodass dieser neben den üblichen Schwachstellen, z.B. in der Office-Infrastruktur, auch verwundbare medizinische Software erkennen kann. Im Folgenden beschreiben wir die beiden Komponenten genauer.

5.1 MedFUZZ - Fuzzer für medizinische Software

Zum Testen von DICOM- und HL7-Implementierungen haben wir den Fuzzer **MedFUZZ** auf AFL- [Za] und SharpFuzz-Basis [Mi19] entwickelt, mit dem mehrere Probleme aufgedeckt werden konnten. Neben der Durchführung des Fuzzings unterstützt die Fuzzing-Plattform Nutzende auch bei der Einrichtung der Umgebung, dem Bau der zu testenden Software und der Ergebnisauswertung (insb. Deduplizierung) mit Hilfe von Skripten. Außerdem enthalten sind Werkzeuge zur Übertragung der Ergebnisse auf Closed-Source-Software.

Als DICOM-Implementierung haben wir das weit verbreitete, und oft als Basis für kommerzielle Software genutzte, Open-Source, in ANSI C und C++ geschriebene DICOM-Toolkit (DCMTK) [Of] untersucht. In unser Testumgebung wird als Zielanwendung der DCMTK DIMSE-Kommunikation MedFUZZ genutzt, welcher statt standardkonformer DICOM-Nachrichten Testvektoren für das Fuzzing sendet. Die Kommunikation erfolgt aus Performancegründen nicht über ein physisches Netzwerk, sondern über ein Loopback-Interface.

Für die Untersuchung von HL7-Implementierungen wurde die ebenfalls weit verbreitete .NET Bibliothek NHapi [Co] ausgewählt. Als Einstiegspunkt wurde die Parsingroutine `PipeParser` gewählt, die für die Verarbeitung jeder eingehenden HL7-Nachricht aufgerufen werden muss, und in unserer Umgebung direkt MedFUZZ-Testvektoren entgegennimmt.

Wir haben zum Fuzzing ein zweistufiges Verfahren gewählt: Zuerst wurde ein Durchlauf ohne Instrumentierung zur Suche von Speicherfehlern durchgeführt, um möglichst schnell Testvektoren mit hoher Testabdeckung zu generieren. In einem zweiten Durchlauf haben wir die generierten Testvektoren als Basiseingaben (Seed) für den Fuzzer genutzt, um damit in den instrumentierten Binärdateien Speicherfehler aufzudecken. Durch dieses Verfahren kann trotz langsamerer instrumentierter Binärdateien schnell eine hohe Testabdeckung erreicht werden, ohne dabei Fehler in schwer zu erreichenden Code-Pfaden zu übersehen.

Zur tieferen Analyse der Testvektoren, die zum Absturz bzw. Einfrieren der Applikation geführt haben, wurden diese manuell in einem Debug-Build geprüft und die zugrundeliegenden Fehler identifiziert und an die Entwickler gemeldet. So wurden mehrere Memory Leaks und zwei Out-of-Bound-Reads im DCMTK aufgedeckt. Weiterhin wurden 62 Probleme in NHapi aufgedeckt, die unter anderem zu einem Denial-of-Service führen können.

Durch eine Kooperation konnten wir die DICOM Testvektoren auf einem DCMTK-basierten Closed-Source PACS evaluieren und die beiden gefundenen Bugs auch hier bestätigen.

5.2 MedVAS - Scanner für medizinische Systeme

Das Scannen von digitaler Krankenhausinfrastruktur ist zum einen durch die heterogene IT-Landschaft, und zum anderen durch die hohe Kritikalität risikoreicher als Schwachstellen-Scans in reinen Office-IT-Umgebungen. Für **MedVAS** (Medical Vulnerability Assess-

ment Scanner) haben wir OpenVAS als Basis genommen und für den Einsatz im laufenden Krankenhausbetrieb erweitert. Ein wichtiger Aspekt ist die Reduzierung der Scan-Geschwindigkeit, genauer die vom Scanner pro Sekunde verschickten Pakete. Diese Änderung ist notwendig da einige Medizingeräte äußerst fragil auf aggressive Scans reagieren [Ag19]. In eigenen Versuchen mit verschiedenen netzwerkfähigen Medizingeräten haben wir festgestellt, dass mindestens ein produktiv eingesetztes Gerät eine, vom Hersteller bestätigte, Intrusion-Detection-Funktion hat, welche bei einem Portscan die Funktion komplett einstellt und erst nach einem Werksreset wieder konfiguriert und genutzt werden kann.

Um neben OpenVAS bekannten Geräten auch Medizingeräte in Krankenhausinfrastrukturen zu identifizieren haben wir folgende Module entwickelt und in den Scanner integriert:

- DICOM-Server-Identifizierung
- HL7-Server-Identifizierung
- TI-Konnektor-Identifizierung
- TLS-Scanner-Integration

Das DICOM-Modul schickt an die zu testende IP-Adresse und Port eine DICOM-`C-ECHO`-Nachricht und wertet die ggf. empfangene Antwortnachricht aus. Hiermit kann neben der Existenz des DICOM-Servers in den meisten Fällen anhand der Antwort die eingesetzte Software samt Versionsnummer aus den Protokollfeldern `Implementation Version Name` und `Implementation Class UID` identifiziert werden.

Das HL7-Identifizierungs-Modul schickt, analog zum DICOM-Modul, eine HL7-Nachricht und überprüft die empfangene Nachricht auf HL7-Konformität und gibt, falls mitgeschickt, die `Sending Facility` und die `Sending Application` als Identifizierungsmerkmal mit an.

Das dritte Modul, die TI-Konnektor-Identifizierung, versucht von dem zu scannenden Gerät die XML-basierte Konnektor-Beschreibungsdatei `connector.sds` vom Port 80 und 443 mittels HTTP GET zu laden. Heruntergeladene Dateien werden auf Übereinstimmung mit der Konnektor-Spezifikation überprüft. Im Falle einer Übereinstimmung, werden die Konfigurationsfelder `ANCL_TLS_MANDATORY` und `ANCL_CAUT_MANDATORY` ausgegeben. Diese beiden Konfigurationsparameter sollten für einen sicheren Betrieb, und für die Nutzung zukünftiger Anwendungen wie beispielsweise der Komfortsignatur, auf `TRUE` stehen um die Verwendung der TLS-Transportverschlüsselung zwischen dem Konnektor und den Clients sowie die Authentifizierung beim Konnektor zu erzwingen [ge21].

Um gefundene TLS-unterstützte Systeme zu evaluieren wurde der TLS-Scanner [So16] in den Scanner integriert. Dieser evaluiert alle unterstützten TLS-Konfigurationen der gefundenen Server auf bekannte TLS-Schwachstellen.

Für MedVAS wurde auf das Client-Server-Konzept zurückgegriffen. Dadurch kann ein Dienstleister remote Überprüfungen verfolgen und auswerten. Der Scan-Client wird dazu in einem dafür vorgesehenen Netz mit den erforderlichen Zugriffsrechten installiert. Der Dienstleister kann den Client über den Server starten und die Ergebnisse auslesen und auswerten. So kann die krankenhausinterne IT-Abteilung sinnvoll entlastet werden, gerade falls keine ausreichenden Ressourcen für regelmäßige Schwachstellenscans vorhanden sind.

Evaluation im Krankenhaus Für die praktische Evaluation wurde MedVAS in einem großen radiologischen Versorgungszentrum, sowie in zwei Krankenhäusern während des Produktivbetriebs getestet. Die zu scannenden Netze wurden von der dortigen IT-Abteilung vorgegeben. Durch die defensive Scangeschwindigkeit von MedVAS dauerten die Scans teils über mehrere Tage. In diesen stichprobenartigen Tests funktionierte der Scanner zuverlässig und beeinflusste den medizinischen Betrieb nicht. Die häufigsten gefundenen Probleme waren Software außerhalb des Supportzeitraums sowie unsichere Standardpasswörter. In einem der Krankenhäuser erkannte MedVAS einen MIRTH-HL7-Server in einem Netzbereich in dem, laut Dokumentation, keine HL7-Server betrieben werden. Die vollständigen Berichte wurden den Krankenhäusern für die weitere Bearbeitung zur Verfügung gestellt.

In der Evaluation in der echten Krankenhausinfrastruktur haben wir mehrere Grenzen des Scanners festgestellt. Ein mögliches Hindernis für eine erfolgreiche Ausführung ist, dass der Scan nicht komplett vollautomatisch funktioniert. Das heißt insbesondere, dass die jeweilige IT-Abteilung eine gut gepflegte Dokumentation der genutzten IP-Adressen und Ports benötigt. Gerade bei Protokollen, welche in der Praxis keine standardisierten TCP-Ports verwenden, beispielsweise HL7, ist die Konfiguration der zu scannenden Ports essenziell für einen aussagekräftigen Bericht.

In unseren Praxistests konnten wir keine DICOM-Server identifizieren. Nach Rücksprache mit den IT-Abteilungen konnten wir dies darauf zurückführen, dass die PACSs nur mit zuvor fest konfigurierten IP-Adressen kommunizieren. Aus IT-Sicherheits-Sicht ist diese Konfiguration grundsätzlich positiv zu bewerten. Allerdings kann unser Scanner dadurch keine veraltete, oder sogar verwundbare, DICOM-Software identifizieren. Falls ein solcher Scanner dauerhaft genutzt werden soll, wäre die Vergabe einer entsprechend konfigurierten IP-Adresse eine Möglichkeit auch DICOM-Systeme zu identifizieren.

6 Fazit

Mit der immer weitergehenden Digitalisierung in der Gesundheitsbranche steigen auch die IT-Sicherheits-Risiken. Wir haben gezeigt, dass die aktuell gängigen Protokolle im klinischen Alltag einen unzureichenden Schutz gegen IT-Angriffe bieten und langfristig gegen moderne Verfahren ausgetauscht werden sollten.

Da solche Migrationen im stark regulierten Krankenhausumfeld langwierige Prozesse bedeuten, zeigen wir in Anhang A Handlungsempfehlungen auf, welche von Krankenhäusern und Medizingeräteherstellern kurzfristig in Erwägung gezogen werden sollten, sowie für die Erstellung von neuen Medizinstandards evaluiert werden sollten.

Des Weiteren wurde ein nicht-invasiver Schwachstellen-Scanner implementiert und evaluiert, welche die besonderen Umstände medizinischer Infrastruktur mit einbezieht und im produktiven laufenden Umfeld genutzt werden kann.

Literaturverzeichnis

[Ag19] Agnew, Joe: Medical Device Security, Part 1: How to Scan Devices Without Letting Safety Flatline. Bericht, Rapid7, April 2019. https://www.rapid7.com/blog/post/2019/04/29/medical-device-security-how-to-scan-devices-without-letting-safety-flatline.

[BG16] Benson, Tim; Grieve, Graham: Principles of Health Interoperability. Springer Verlag London, 3. Auflage, 2016.

[Bl99] Blobel, Bernd; Spiegel, Volker; Pharow, Peter; Engel, Kjeld; Krohn, Rolf: Standard Guide for EDI (HL7) Communication Security. 1999. https://www.hl7.org/implement/standards/product_brief.cfm?product_id=238.

[BL16] Bhargavan, Karthikeyan; Leurent, Gaëtan: On the Practical (In-)Security of 64-Bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. CCS '16, Association for Computing Machinery, New York, NY, USA, S. 456–467, 2016.

[Bu] Bundesamt für Sicherheit in der Informationstechnik (BSI): Open Vulnerability Assessment System (OpenVAS). https://www.bsi.bund.de/EN/Topics/Industry_CI/ICS/Tools/OpenVAS/OpenVAS_node.html, abgerufen am 20.09.2021.

[Bu15] Bundesgesetzblatt Jahrgang 2015: Gesetz für sichere digitale Kommunikation und Anwendungen im Gesundheitswesen sowie zur Änderung weiterer Gesetze. Bundesanzeiger Verlag, Kapitel Teil 1, Nr. 54, S. 2408–2423, Dezember 2015. http://www.bgbl.de/xaver/bgbl/start.xav?startbk=Bundesanzeiger_BGBl&jumpTo=bgbl115s2408.pdf.

[Co] NHapi. http://nhapi.sourceforge.net/home.php, abgerufen am: 09.11.2021.

[Da18] Dameff, Christian; Bland, Maxwell; Levchenko, Kirill; Tully, Jeff: Pestilential Protocol: How Unsecure HL7 Messages Threaten Patient Lives. In: Blackhat USA 2018. August 2018. https://i.blackhat.com/us-18/Thu-August-9/us-18-Dameff-Pestilential-Protocol-How-Unsecure-HL7-Messages-Threaten-Patient-Lives-wp.pdf.

[Du17] Duggal, Anirudh: HL7 2.X Security. In: HITBSecConf 2017. April 2017. https://paper.bobylive.com/Meeting_Papers/HITB/2017/D2T2---Anirudh-Duggal---Hacking-Medical-Devices-and-Healthcare-Infrastructure.pdf.

[EKK20a] Eichelberg, Marco; Kleber, Klaus; Kämmerer, Marc: Cybersecurity Challenges for PACS and Medical Imaging. Academic Radiology, 27(8):1126–1139, 2020.

[EKK20b] Eichelberg, Marco; Kleber, Klaus; Kämmerer, Marc: Cybersecurity in PACS and Medical Imaging: an Overview. Journal of Digital Imaging, 2020.

[EKK20c] Eichelberg, Marco; Kleber, Klaus; Kämmerer, Marc: Cybersecurity Protection for PACS and Medical Imaging: Deployment Considerations and Practical Problems. Academic Radiology, 2020.

[ge21] gematik: Spezifikation Konnektor, Version 5.14.0. September 2021. https://fachportal.gematik.de/fachportal-import/files/gemSpec_Kon_V5.14.0.pdf.

[Gr] OpenVAS – Open Vulnerability Assessment Scanner. https://openvas.org, abgerufen am 09.11.2021.

[Gr19] Greenbone Networks GmbH: Sicherheitsbericht - Ungeschützte Patientendaten im Internet. September 2019. https://www.greenbone.net/wp-content/uploads/CyberResilienceReport_DE.pdf.

[He17] Health Level Seven International: Health Level Seven Version 3. Januar 2017. https://www.hl7.org/implement/standards/product_brief.cfm?product_id=186.

[He19] Health Level Seven International: HL7 Messaging Standard Version 2.9. Dezember 2019. https://www.hl7.org/implement/standards/product_brief.cfm?product_id=516.

[Jo18] Joyia, Gulraiz Javaid; Akram, Muhammad Usman; Akbar, Chaudary Naeem; Maqsood, Muhammad Furqan: Evolution of Health Level-7: A Survey. In: Proceedings of the 2018 International Conference on Software Engineering and Information Management. ICSIM2018, Association for Computing Machinery, New York, NY, USA, S. 118–123, 2018.

[KKB21] Klick, Johannes; Koch, Robert; Brandstetter, Thomas: Epidemic? The Attack Surface of German Hospitals during the COVID-19 Pandemic. In: 2021 13th International Conference on Cyber Conflict (CyCon). S. 73–94, 2021.

[KS98] Kaliski, B.; Staddon, J.: PKCS #1: RSA Cryptography Specifications Version 2.0. Internet Engineering Task Force (IETF), Oktober 1998. https://datatracker.ietf.org/doc/html/rfc2437.

[Ma19] Manès, Valentin Jean Marie; Han, HyungSeok; Han, Choongwoo; Cha, Sang Kil; Egele, Manuel; Schwartz, Edward J.; Woo, Maverick: The Art, Science, and Engineering of Fuzzing: A Survey. IEEE Transactions on Software Engineering, S. 1–1, 2019.

[Mi19] SharpFuzz: Bringing the power of afl-fuzz to .NET platform. https://mijailovic.net/2019/01/03/sharpfuzz/, abgerufen am: 09.11.2021.

[Na21] National Electrical Manufacturers Association (NEMA): , The DICOM Standard. http://dicom.nema.org/medical/dicom/2021d/, 2021. Stand 2021d, abgerufen am 05.11.2021.

[NI17] NIST: Update to Current Use and Deprecation of TDEA. Juli 2017. https://csrc.nist.gov/News/2017/Update-to-Current-Use-and-Deprecation-of-TDEA.

[Of] DICOM-Toolkit (DCMTK). https://dcmtk.org/, abgerufen am: 09.11.2021.

[Se14] Security Work Group: HL7 Healthcare Privacy and Security Classification System (HCS). Bericht, HL7 International, August 2014. https://www.hl7.org/implement/standards/product_brief.cfm?product_id=345.

[SGA07] Sutton, Michael; Greene, Adam; Amini, Pedram: Fuzzing - Brute Force Vulnerability Discovery. Pearson Education Inc., 2007.

[SHSA15] Sheffer, Y.; Holz, R.; Saint-Andre, P.: Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). Internet Engineering Task Force (IETF), Mai 2015. https://tools.ietf.org/pdf/bcp195.pdf.

[So16] Somorovsky, Juraj: Systematic Fuzzing and Testing of TLS Libraries. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. CCS '16, Association for Computing Machinery, New York, NY, USA, S. 1492–1504, 2016.

[St17] Stevens, Marc; Bursztein, Elie; Karpman, Pierre; Albertini, Ange; Markov, Yarik: The first collision for full SHA-1. In: Annual international cryptology conference. Springer, S. 570–596, 2017.

[STB20] Saatjohann, Christoph; Tschirsich, Martin; Brodowski, Christian: Tut mal kurz weh – Neues aus der Gesundheits-IT. In: Remote Chaos Experience (rC3). Dezember 2020. https://media.ccc.de/v/rc3-11342-tut_mal_kurz_weh_neues_aus_der_gesundheits-it.

[Wa19] Wang, Zhiqiang; Li, Quanqi; Wang, Yazhe; Liu, Biao; Zhang, Jianyi; Liu, Qixu: Medical Protocol Security: DICOM Vulnerability Mining Based on Fuzzing Technology. In: The 2019 ACM SIGSAC Conference. S. 2549–2551, 11 2019.

[WY05] Wang, Xiaoyun; Yu, Hongbo: How to Break MD5 and Other Hash Functions. In (Cramer, Ronald, Hrsg.): Advances in Cryptology – EUROCRYPT 2005. Springer Berlin Heidelberg, Berlin, Heidelberg, S. 19–35, 2005.

[Za] american fuzzy lop. https://lcamtuf.coredump.cx/afl/, abgerufen am: 09.11.2021.

A Handlungsempfehlungen

Basierend auf unseren Ergebnissen geben wir kurzfristig umsetzbare Handlungsempfehlungen für Krankenhäuser und Hersteller digitaler Medizintechnik.

A.1 Krankenhäuser

Für Krankenhäuser gibt es neben den allgemein gültigen IT-Sicherheits-Empfehlungen spezielle Empfehlungen und Leitfäden. Für Krankenhäuser welchen den Vorgaben der Kritische Infrastrukturen (KRITIS) unterliegen, in der Regel ab 30.000 vollstationären Fällen pro Jahr, ist der Branchenspezifische Sicherheitsstandard (B3S) für die Gesundheitsversorgung im Krankenhaus verpflichtend umzusetzen. Auch kleinere Häuser sind durch §75c im SGB Vverpflichtet IT-Sicherheitsmaßnahmen nach dem *Stand der Technik* zu implementieren. Die hier empfohlenen Maßnahmen können die Sicherheit im Krankenhaus unmittelbar wirksam erhöhen, sie ersetzen allerdings nicht eine langfristige IT-Sicherheits-Strategie, welche beispielhafte Themen wie Backup-Systeme, Asset- und Patch-Management sowie Incident-Response-Prozesse betrifft.

Netzwerk- und Sicherheitsarchitektur Unsere internen Scans in realen Krankenhäusern und externe Scans haben gezeigt, dass oftmals Firewall- und Server-Konfigurationen nicht sicher umgesetzt sind. Nicht benötigte Portweiterleitungen und -freigaben für Anwendungen sollten entfernt werden, das betrifft insbesondere erlaubte Verbindungen aus dem Internet auf interne Systeme wie DICOM- oder HL7-Server. Es sollten generell nur Dienste freigegeben werden, die tatsächlich verwendet werden.

Weiterhin sollte das Netzwerk so segmentiert werden, dass lediglich Administratoren auf sicherheitskritische Systeme und Dienste zugreifen können. Dazu bietet sich die bereits vorhandene Unterteilung des Netzwerks in VLANs an. Es sollte evaluiert werden, ob die Implementierung einer Network Access Control (NAC)-Lösung sinnvoll ist, um unbekannten Geräten den Zugriff auf das Netzwerk zu verwehren.

Nicht verwendete Netzwerk-Ports innerhalb des Gebäudes sollten zudem deaktiviert werden, um sicherzustellen, dass frei zugängliche Ports nicht verwendet werden können.

Nutzung vorhandener Sicherheitsmechanismen Sicherheitsmaßnahmen, welche von den eingesetzten Geräten von Haus aus unterstützt werden, sollten standardmäßig genutzt werden. Das umfasst beispielsweise sowohl die verpflichtende Authentifizierung und Nutzung des TLS-Protokolls bei TI-Konnektoren als auch SSH-basierte Remoteverbindungen für Administrationszwecke, welche ausschließlich über eine Public-Key-Authentifizierung mit einem passwortgeschützten Schlüssel gewährt werden sollten.

Unsere Evaluation hat gezeigt, dass einige DICOM-basierte Systeme zwar TLS-Unterstützung anbieten, es sich im praktischen Betrieb allerdings schwierig gestaltet diese zu nutzen, da nicht alle im Krankenhaus genutzten Geräte TLS bzw. eine interoperable Konfiguration unterstützen. Auch das benötigte Zertifikatsmanagement ist oftmals nicht im Krankenhaus vorhanden. Aus dieses Grund empfehlen wir für DICOM-basierte Systeme als Basisschutz einen verpflichtenden individuellen und nicht erratbaren AET einzusetzen, und die erlaubten Verbindungen durch Nutzung einer *Allowlist*, welche nur explizit benötigte Verbindungen erlaubt, einzuschränken. Da diese Konfiguration keinem echten Passwortschutz entspricht, und viele System ein Durchprobieren des AET erlauben, kann der Schutz durch Netzwerk-Logging und automatisches Blocken bei zu vielen neuen Verbindungen auf die DICOM-Systeme weiter erhöht werden.

Regelmäßige Audits Es sollten regelmäßige IT-Sicherheitsüberprüfungen (z.B. Penetrationstests) der internen und externen IT-Infrastruktur durchgeführt werden. Dadurch können Sicherheitslücken und Schwachstellen aus technischer und organisatorischer Sicht identifiziert und behoben werden. Hierbei kann insbesondere MedVAS eingesetzt werden um automatisiert die interne Netzwerkinfrastruktur zu überprüfen. Regelmäßige Tests schärfen zudem das Bewusstsein der IT-Mitarbeiter für typische Fallstricke, was die IT-Sicherheit auch nachhaltig verbessert.

Prozesse Neben den technischen Aspekten sollten alle Prozesse in die IT-Sicherheitsüberlegungen mit einbezogen werden. Diese fangen bei dem Einkauf neuer Medizingeräte an, in welchen IT-Sicherheitsfunktionalitäten wie bspw. die Verwendung von Zwei-Faktor-Authentisierung (2FA) oder die Verwendung von Transportverschlüsselung als verpflichtend betrachtet werden sollten.

A.2 Hersteller und Standardisierungsorganisationen

Forcierung von IT-Sicherheitsfunktionalitäten in Geräten und Standards Wie wir in Abschnitt 3 gezeigt haben, ist es in der Praxis oftmals nicht möglich vom Standard unterstützte Sicherheitsmerkmale zu nutzen. Hier liegt es bei den Herstellern und deren Verbänden einheitliche interoperable Konfigurationen festzulegen welche regelmäßig auf sogenannten *Plugfests* evaluiert werden.

Neue Standardversionen sollten Verfahren, welche nicht mehr dem Stand der Technik entsprechen, konsequent als obsolet kennzeichnen und nach einer Karenzzeit als nicht mehr standardkonform behandeln. Weiterhin sollten Neu- und Weiterentwicklung von medizinischen Standards und Protokollen eine verpflichtende Nutzung von aktuellen Sicherheitsmaßnahmen wie Verschlüsselung, Authentizität und Integrität der Daten vorsehen.

Sichere Entwicklungs-Prozesse Durch unsere Fuzz-Tests auf DICOM- und HL7-Software haben wir mehrere sicherheitskritische Bugs gefunden und den Herstellern mitgeteilt. Fuzz-Tests werden in der modernen Softwareentwicklung immer mehr zum Standard und sollten dementsprechend auch in der Entwicklung von Medizinsoftware eingesetzt werden.

Analyzing the Software Patch Discipline Across Different Industries and Countries

Robin Müller[1], Julius Ruppert[1], Katharina Will[1], Lukas Wüsteney[1], Tobias Heer [1]

Abstract:

In view of recent cyberattacks and new regulatory requirements, companies in different industries and countries are forced to implement additional IT security measures. Nevertheless, a large number of services with vulnerable or outdated software can be found on the Internet. In this work, we investigate whether industry-specific differences exist in the maintenance and use of outdated Internet-facing software. For this purpose, we combine results from Internet-wide port scans with product and version information as well as information of companies listed at stock markets in different countries. We show that different industries have more or less up-to-date software for different services like remote access tools, databases, webservers and file servers. With this approach, we discovered surprising amounts of outdated and even unsupported software in use across many industries and countries.

Keywords: Internet Scanning, Vulnerability Management, Patch Management

1 Introduction

The number and severity of cyberattacks increased dramatically within the last years. In many cases, attacks exploit new software vulnerabilities, or use already known software vulnerabilities in older software. Hence, companies should have an intrinsic interest in keeping their exposed and Internet-facing software up to date. Moreover, cybersecurity regulations in different industry sectors and in different countries have created an uneven playing field for companies. Some industry sectors in some countries are tightly regulated while only few regulations exist elsewhere. Examples of mandatory or voluntary regulations are NERC CIP in the US [No21], the NIS Directive in the EU [Eu21] and the PCI regulation in the financial sector [pc21].

Assuming that forced or voluntary cybersecurity standards raise the general cybersecurity maturity of a company, we evaluate whether or not differences in the application of patch management and the use of outdated or current software exist across industrial sectors and geographical areas. In order to achieve this goal, we conducted Internet-wide scans and correlated the results with product data as well as with company information and Internet network allocations. While the Internet-facing services of a company may not necessarily be part of the regulated IT infrastructure, we aim to evaluate whether a higher maturity and

[1] Hochschule Esslingen, Flandernstraße 101, 73732 Esslingen, Germany
{romuit04, juruit03, kawiit01, lukas.wuesteney, tobias.heer} @hs-esslingen.de

doi:10.18420/sicherheit2022_10

proficiency in cybersecurity might carry over to other less critical areas of an enterprise. An example of this would be critical infrastructure where the secure operation of an electrical plant may be tightly regulated while the Internet-facing customer portal may not be regulated in the same way. However, it still may be under the same IT security governance. In our evaluation, we focus on larger companies since these companies are more likely to fall under sectoral or local regulations regarding cybersecurity.

Within our research, we created a dataset containing a list of available services for stock listed companies. In addition to scanning, we collect available metadata (e.g., software-version, product name or sector of the company) as well. Our dataset reveals information about potential vulnerabilities, present on each host and a mapping between hosts and companies. To the best of our knowledge, we are the first presenting the approach to generate this structured dataset. With this dataset, we are analyzing our hypothesis and research questions. Specifically, we analyze (a) if different local regulations impact the occurrence of outdated software versions, and (b) if critical sectors, like finance and healthcare, are more up to date with their software.

Our contribution is threefold: (a) Develop and present a pipeline for Internet-wide version tracking [Mü22]. (b) Evaluate a large scan of stock market companies and their accessible services. (c) Visualize the software versions in use based on the location and the industrial sector of the companies for 14 different services.

This paper is structured as follows: First we summarize the concept of our scanning and evaluation approach in Section 2. In Section 3, we describe the path from our hypothesis towards a complete dataset for data analysis. Afterwards, we analyze our data to evaluate our hypothesis in Section 4. Next, we show the relation of our work to previous literature in Section 5. Finally, in Section 6 we draw our conclusion and discuss ethical considerations in the Appendix.

2 Analysis Pipeline Overview

In this section, we explain our approach on collecting and structuring data. We introduce all relevant data-sources and the path through our pipeline. Many services (e.g., SSH-Servers, HTTP-Servers, Mail-Servers, etc.) announce their identity via so-called banners upon connection establishment. Banners are service descriptions or information that can be visualized to users if they connect to a service. Often, these banners do not only contain the vendor of a product, but also contain product names and versions.

In order to acquire a list of active services on the Internet, we scan the IPv4 address space and collect the service banners. Service banners are retrieved during the connection phase with a service or protocol. They contain information which is either auto-generated or customized by the user (e.g., the administrator changes the name of the service). For each IP address, we only scan well-known ports for the protocols we intend to analyze. Our goal is to evaluate the landscape of software versions which are available on the Internet. For this

purpose, we create product identifiers in the form of *Common Platform Enumerations* (CPE) [Na11] for all collected banners. A CPE is used to identify a particular software or hardware product or operating system including the version number. The CPE is based on service information (e.g., the service announces the software and its version in an error response), or it can be derived from the banner (e.g., the service banner contains additional information about other software like the operating system). This leaves us with a rather unorganized set of IP addresses and identified products and versions.

In the next step, we identify whether a software version is up-to-date or not. To this end, we map the identified versions to vendor-specific lists in order to identify versions that are up-to-date, in extended support or have reached their end-of-life date (EOL). Without an adequate data source for such information, this matching involves manual compilation of version lists, support dates and end-of-life dates.

Making statements about the prevalence of older software in specific sectors requires a mapping from an IP address to a network to a company which in turn belongs to a certain sector or geographic region. In order to establish this mapping, we match and combine data from several sources. To map a scanned host to a network, we use IP address block information provided by ipinfo.io [ip21]. This enables us to group individual IP addresses so we can identify all services belonging to a company. Next, we match the names of the companies from ipinfo.io to sectors and countries by using publicly available stock market information from *Yahoo Finance*. In addition to identifying the industry sector, in which a company is active, we also retrieve geographic information about all companies based on their stock market registration. Using this legal information rather than IP-based geographic matching provides a more stable picture because load balancing services such as Content Delivery Networks (CDN) or geographic hosting can introduce a skew to IP geo-information based on the location of a scanner. So, depending on the scanner location, we will receive a response by different servers of an CDN. Yet, the fidelity of our IP information is still limited since companies may operate services at different branches in different countries. Hence, measurements that include geographic data must be considered with such limitations in mind.

3 Analysis Pipeline Implementation

In this section we discuss the pipeline, presented in Section 2, in more detail. First, we explain the process of creating the list of IP addresses to scan. Next, we describe the essential steps in the scanning process of our target hosts and the improvements we made to it. Finally, we conclude this section with our categorization and classification of the software discovered in the scanning process. The overview of the complete pipeline is depicted in Figure 1. Each step is explained and discussed in the following sections.

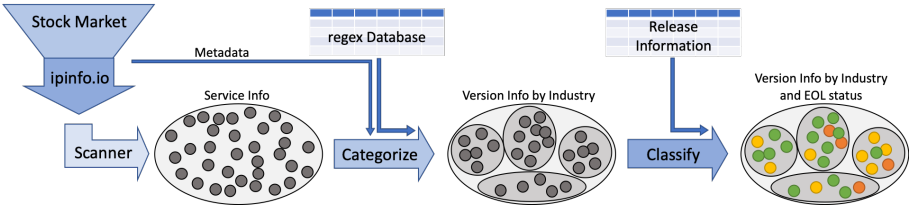


Abb. 1: Overview of Scanning, Categorization and Classification Pipeline

3.1 Gathering IP Addresses and Services

We identified stock market information as a useful source of company information. To be more efficient than scanning the complete Internet, we compiled a database from publicly available company data and map IP addresses to companies as discussed in Section 2. This list of IP ranges is used as basis for our IP and port scans. For each protocol, the port scanner *Zmap* can identify if the default port is open. In addition, *Zgrab2* can extract meta information, related to the service (e.g., banners).

Due to insufficient banner information or a low number of results for the protocol in general, we are not able to generate meaningful results for *BACnet*, *Niagara Fox*, *IPP*, *SMB* and *NTP*. Therefore, we stopped scanning for these protocols. The remaining scanned protocols are therefore: *HTTP*, *IMAP*, *POP3*, *SMTP*, *FTP*, *SSH*, *Telnet* and database services *MariaDB*, *Microsoft SQL Server*, *MongoDB*, *MySQL*, *Oracle Database* and *Redis*. Our evaluation is based on a single scan of the available IP addresses and protocols and therefore does not contain duplicate data for any host.

3.2 Scanning the Targets

In the context of scanning hosts, it is important to remember that companies might protect their infrastructure from malicious scanning and Denial-of-Service (DoS) attacks. Scanning from a single IP address leads to easily identifiable traffic. To avoid getting blocked based on the type and amount of traffic, there are two guidelines to follow: (a) As already presented by [Wa20], the scanning rate itself should be selected to be about 100,000 probes per second. (b) Additionally, IP addresses should not be scanned sequentially, but randomly. This spreads the scans across different address ranges and makes source-based identification over time more difficult. These steps help to achieve higher scan response rates, and also respect the service hosters by not posing a threat to the availability of their services. All of our scanning activities originated from a single source in the United States. Therefore, our data might not be complete as some services are not available from this location, or we might already be on blacklists, as our provider may be known for scanning activities.

Figure 2 visualizes the distribution of successful scans per sector. A successful scan means the port of the service is open and we receive additional information via a banner. We collect the information on industry sectors and companies from the data in the stock market

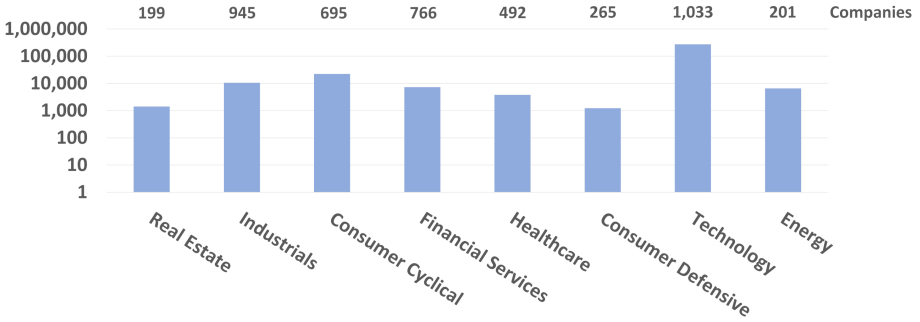


Abb. 2: Successful Scans per Sector

database. Figure 2 shows that the sector "Technology" clearly is the largest sector within our scans. However, the diagram shows a distribution of services across all sectors, which makes the comparisons between sectors possible. The sectors are as follows: Basic Materials, Communication Services, Consumer Cyclical, Consumer Defensive, Consumer Goods, Energy, Financial, Financial Services, Healthcare, Industrial Goods, Industrials, Real Estate, Services, Technology and Utilities. Within all of our diagrams we only present the sectors with the most CPEs.

3.3 Analyzing Banners

At this stage in our pipeline, we possess information about available hosts and services per company by scanning for service banners. In order to express statements other than "in sector ABC, the average company runs 100 services on 20 hosts", we need three more steps. Unfortunately, two of them require a lot of manual work (i.e., extracting data from the banners and identifying whether versions are still supported by their vendors).

As a first step, we extract information from all collected banners that are relevant to our research. This is a challenge since the banners vary in structure and content. While some banners might provide us with full information about the used product, software and even operating systems, others might only reveal the product itself. We use regular expressions (regex) to identify the parts of the banner that indicate product and version information. A regex defines a desired structure of a string and extracts substrings (e.g., a version in the format of aa.bb.cc with aa as major, bb as minor and cc as bugfix number). If a regex did not match a defined format, our tool continues with the next regex. The creation of the regex definitions requires manual analysis of the collected banners.

Depending on the banner structure, we can extract information about the product itself, its version, build dates and information about the operators or owners of the service. The combination of product name, vendor and version can be represented in a CPE. The quality of the gathered CPEs is highly dependent on our regular expressions. We visualize the distribution of the information we gain from the banners in Figure 3. Some protocols need a greater amount of scans to retrieve a meaningful amount of CPEs. For instance, from the

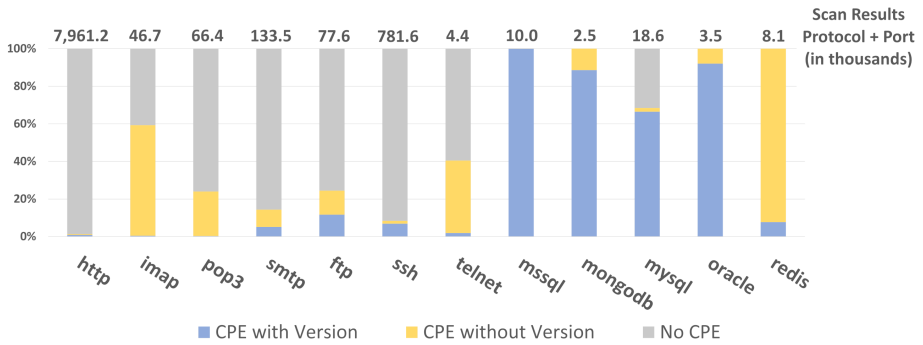


Abb. 3: Distribution of CPE information quality throughout our scans

HTTP protocol scans, only a very small percentage produce CPEs with product and version information, but the overall number of scans makes up for that fact. A CPE allows us to evaluate whether outdated and/or unsupported software is used.

In a second step, we compile a lookup table to decide which versions of the products that were revealed from the banners, are still maintained. This information is typically present on the vendor specific product page. After standard product support has ended, the vendor may still offer security fixes for an additional fee. This means we essentially have four cases per product: (1) still maintained, (2) maintained in certain cases, (3) end-of-life, and (4) no information. Case (4) represents all banners that revealed the product but not the version.

As a last step, we summarize all products of one category (e.g., *MySQL* and *MariaDB*, which are both database services, or *Apache HTTP Server* and *ngnix*, which are both webservers). In this way, we obtain an overview of the software landscape within a certain area on the Internet, which can be grouped by geographic location, industry sector or other metrics.

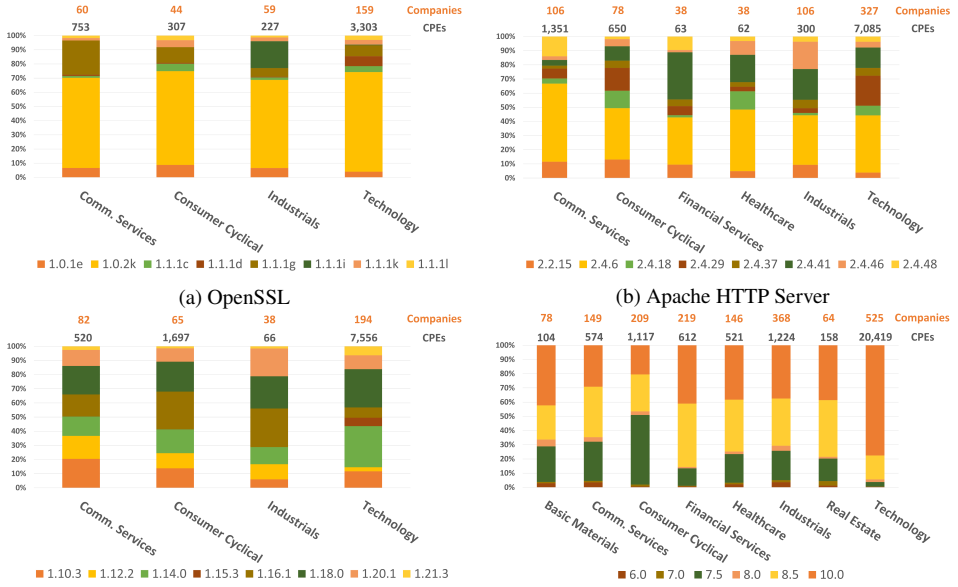
4 Evaluation

In this section we evaluate the collected dataset. We also present the findings about the software versions analyzed by region and sector.

4.1 Classification of Software Versions

In Figure 4, we compare services, commonly scanned, by the deployed software version. For a better overview, we present only sectors containing more than 30 companies and versions with more than 50 CPEs. We apply both filters for each diagram independently and limit the number of visualized sectors to eight. Hereby, each bar has a balanced distribution. In general, the figure shows that there are no obvious differences between the sectors since the distribution of versions is similar.

Unsupported *OpenSSL* versions such as 1.0.2 and below are still widely used across multiple sectors as we can see in Figure 4a. Especially version 1.0.2k is still very dominant in our



(a) OpenSSL

(b) Apache HTTP Server

(c) Nginx

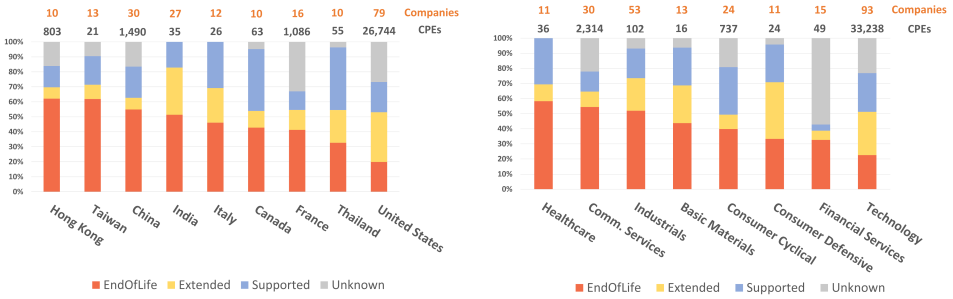
(d) Microsoft IIS

Abb. 4: Distribution of software versions per sector

scans. However, it must be noted that the *OpenSSL Software Foundation* does offer extended support for this version. As shown in Figure 4b there are still *Apache HTTP Server* with version 2.2.x online. This version has reached its end-of-life in 2018 and was found about 550 times spread over 115 companies. The analysis of *nginx* webservers in Figure 4c shows that the majority of companies run software which is no longer supported (version 1.18 and below). All odd nginx versions represent the mainline branch, which receives updates most frequently. Figure 4c shows that the technology sector uses a mainline branch comparatively more often. For the other three sectors this is not the case. For *IIS*, the versions 7.5 and below are no longer maintained with security updates since January 2020. However, it is possible to get extended support for three years. *IIS* 6.0 is already end-of-life since 2015. The technology sector seems to keep this software relatively up to date, with versions 10 and 8.5 making up the bulk of Figure 4d.

4.2 Insights into Database Services

We compiled all our information related to databases into an aggregated view grouped by country and industry sector. For this evaluation, we excluded all sectors or countries that contained less than 10 companies to produce a more meaningful result. Our view on a per country basis (see Figure 5a) reveals that many old databases are still in use. This could be due to the fact that upgrading to a newer version is in many cases accompanied by a lot of work for migration. We might take the view that countries with rigid regulation bodies, like Thailand with a Cybersecurity Strategy [Of17] and the United States, are more



(a) Software support aggregated by country

(b) Software support aggregated by business sector

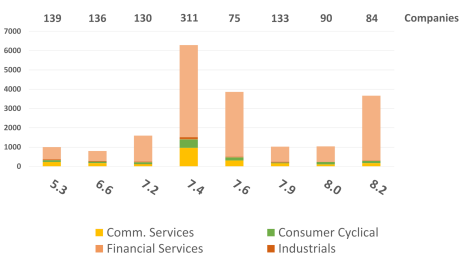
Abb. 5: The software support status of scanned database services (*MySQL*, *MariaDB*, *Oracle Database*, *Microsoft SQL Server* and *Redis*). Presented in a stacked diagram and visualizing the relation between identified end-of-life, possibly extended support, supported and unknown software versions.

likely to keep their software versions up to date. In the US, the FCC and NIST regulate on a federal level, while states have additional requirements like [Ca18] in California. For Thailand, personal data storages (often implemented with databases) security measures must be updated every three months [Of17]. Thailand also has a multitude of regulation bodies for cybersecurity, like the Organization of Critical Information Infrastructure (CIIO) and National Cybersecurity Committee (NCC). The data suggests that these enforcement measures may contribute to a better sense of security and therefore to a greater discipline when it comes to keeping versions up to date. Our literature research does not reveal similar strict regulations for the other countries, nevertheless such regulations may exist.

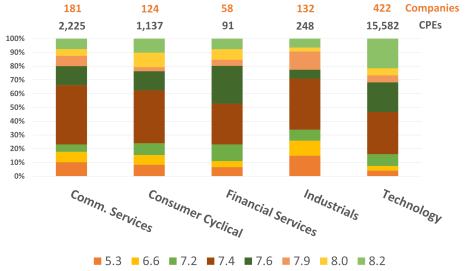
Figure 5b presents the view per sector. The technology sector seems to be faster when it comes to updating its services. This could be due to its proximity to the subject matter. In contrast, sectors with a completely different core business like Healthcare and Industrials operate a lot of older database versions. For some sectors, like financial services, our data contains too much noise in the form of unknown software configurations. This means, that we cannot make meaningful assumptions. Another consideration that needs to be made is that by focusing on the stock market data we only capture a relatively small number of the companies in a country or sector. Our assumptions can only be transferred to this limited amount of companies.

4.3 Insights into OpenSSH

We have taken a closer look at SSH, as it is a frequently used and widespread protocol. SSH is a security critical protocol, because it sets up remote shells with up to root privileges. In our scans, OpenSSH is the most common SSH server we discovered. Moreover, the banner analysis performed well for this protocol. We were able to assign 22,671 CPEs with version information and identify a total of 51 different versions. Figure 6a represents a filtered view, as all versions with less than 200 occurrences are filtered out. We see that the versions 7.4, 7.6 and 8.2 are widely used. We evaluated that there is a connection between the



(a) Distribution of matched OpenSSH CPEs per version. Only visualizing all versions with more than 800 hits



(b) Distribution of matched OpenSSH CPEs per company sector

Abb. 6: Deep Dive in OpenSSH

high numbers of these versions and LTS versions of widely used Linux operating systems. In our dataset 99% of OpenSSH versions 8.2 and 7.6 are represented by Ubuntu. These versions ship with Ubuntu LTS 20.04 and 18.04. For version 7.4 we cannot make a clear determination of the corresponding operating system. A major update of the OpenSSH stack often requires a full upgrade of the operating system, as many dependencies exist. This might be a reason why many outdated versions are used as well. However, old versions are not necessarily vulnerable as Operating System manufacturers implement security fixes into their provided OpenSSH fork.

Analysis of version distribution across the sectors (see Figure 6b) shows no clear trends. The same applies to the version distribution by country and therefore was not visualized. All sectors have roughly equal numbers of old and new software versions of OpenSSH in use. However, Technology and Financial Services seem to run the least old OpenSSH versions.

4.4 Scanning Performance and Matching Products and Versions to CPEs

Since we are scanning at a rate of 100,000 probes per second the regex matching needs to be able to keep up with the inflow of scans. The system we designed continually collects new scans from our database and matches them via the regular expressions. We can process 263,000 scans per second with a single core of an *AMD Ryzen™ 9 5900X*. Since there are no dependencies between scans for the CPE assignment process, this work can be parallelized. This process can further be optimized since scans that do not result in any banner data can be discarded without any applied processing time.

In our scan, we generated 198,205 CPEs containing vendor and product information from which 102,874 CPEs contained version information.

5 Related Work

In this section, we present related work in the area of scanning on the Internet, vulnerability identification and CPE matching. Several authors performed Internet scans with goals that

differ from ours. Dahlmanns et al. scanned the entire IPv4 address space to analyze OPC UA appliances [Da20]. OPC UA is a protocol for secure industrial communication, hence it is applicable for one industry sector. In our work, we try to compare services in different industries and match versions and CPEs to identify dated software. Internet-wide scans were used by Durumeric to identify new vulnerabilities [Du17]. Therefore, software versions which might already be outdated, are not within his focus. We use outdated product versions to identify bad patching discipline instead. Na et al. suggest identifying CPEs and their vulnerabilities automatically based on grabbed banners [NKK18]. We decided to match CPEs manually with regex, as this is more reliable and stable, especially since we need to identify versions as well.

Morishita et al. used Internet-wide scanning to identify honeypots [Mo19]. The authors identified these honeypots in research networks, in commercial hosting and access networks. We did not analyze, if responses to our scan can be matched to honeypots. However, the numbers presented in [Mo19] do not distort our large scan results.

Another point of view was taken by Wan et al. by researching the impact of the scanner's geographical location on the scan results and how inaccurate Internet-wide scans are [Wa20]. Their results also show that there is no optimal location to scan from. The authors show the different factors that can further impact results such as blocked scans based on source addresses. Compared to the large number of results collected, we argue that the effect of these blocked scans is marginal.

6 Conclusion

Within our IP scan of 9,838 stock-listed companies we have received a total of 368,953 successful protocol replies of operational services on the Internet. Across all 198,205 successfully identified CPEs, we can show a wide range of outdated software versions for all industry sectors.

We could not identify sectors that are more up-to-date in general (i.e., we have seen an even distribution for SSH). However, for specific protocols (i.e., all database CPEs combined) a difference between sectors and regions is noticeable. Based on local regulations for critical infrastructure, we assume certain software is updated more regularly. For database software, we have also presented a detailed analysis on the supported or outdated software versions in use. We found an astonishing amount of old services still operated by stock exchange listed companies. The technology sector showed the best patch discipline and the most up-to-date versions. This result is surprising because other sectors (e.g., healthcare, and financial services) are more tightly regulated in many countries.

Some sectors are not well represented in our dataset. Therefore, we are not able to draw deeper conclusions about their state. To improve statements made based on industry sectors and Internet-wide scanning, further investigation is needed so less ambiguous information can be extracted.

Appendix: Ethical Considerations

Starting with broad scanning processes requires a few things to remember and guidelines to follow. One always needs to respect certain parties not being happy about scans and should exclude them once notified. We implemented a webserver on our scan server with information on this project and added the according abuse fields in our *whois* entry. With this information, every target can contact us, even with auto-generated emails, and can get excluded from future scans.

Literatur

[Ca18] California Legislative Information: Bill Information. In: Assembly Bill No. 1906. 2018, URL: https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill%5C_id=201720180AB1906, Stand: 11. 11. 2021.

[Da20] Dahlmanns, M.; Lohmöller, J.; Fink, I. B.; Pennekamp, J.; Wehrle, K.; Henze, M.: Easing the Conscience with OPC UA: An Internet-Wide Study on Insecure Deployments. In: IMC '20: Proceedings of the ACM Internet Measurement Conference. S. 101–110, 2020.

[Du17] Durumeric, Z.: Fast Internet-Wide Scanning: A New Security Perspective, Diss., University of Michigan, 2017.

[Eu21] European Union Agency for Cycbersecurity: enisa. In: NIS Directive. 2021, URL: https://www.enisa.europa.eu/topics/nis-directive, Stand: 11. 11. 2021.

[ip21] ipinfo.io: IP Ranges API, 2021, URL: https://ipinfo.io/developers/ranges, Stand: 11. 11. 2021.

[Mo19] Morishita, S.; Hoizumi, T.; Ueno, W.; Tanabe, R.; Gañán, C.; van Eeten, M. J.; Yoshioka, K.; Matsumoto, T.: Detect Me If You... Oh Wait. An Internet-Wide View of Self-Revealing Honeypots. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). 2019.

[Mü22] Müller, R.; Ruppert, J.; Will, K.; Wüsteney, L.; Heer, T.: HSES-Patchwatch Project Documentation, 2022, URL: https://hs-esslingen-it-security.github.io/hses-patchwatch/, Stand: 19. 01. 2022.

[Na11] National Institute of Standards and Technology (NIST) - U.S. Department of Commerce: Common Platform Enumeration: Naming Specification Version 2.3, 2011, URL: https://www.govinfo.gov/content/pkg/GOVPUB-C13-c213837a04c3bcc778ebfd420c6a3f2a/pdf/GOVPUB-C13-c213837a04c3bcc778ebfd420c6a3f2a.pdf, Stand: 11. 11. 2021.

[NKK18] Na, S.; Kim, T.; Kim, H.: Service Identification of Internet-Connected Devices Based on Common Platform Enumeration. In: Journal of Information Processing Systems. Bd. 14, S. 740–750, 2018.

[No21] North American Electric Reliability Corporation: NERC. In: CIP Standards. 2021, URL: https://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx, Stand: 11. 11. 2021.

[Of17] Office of the National Security Council: Thailand. In: National Cybersecurity Strategy 2017-2021. 2017, URL: http://www.nsc.go.th/wp-content/uploads/2018/08/strategyit60-64-1.pdf, Stand: 11. 11. 2021.

[pc21] pci Security Standards Council: pci. In: Document Library. 2021, URL: https://www.pcisecuritystandards.org/document_library, Stand: 11. 11. 2021.

[Wa20] Wan, G.; Izhikevich, L.; Adrian, D.; Yoshioka, K.; Holz, R.; Rossow, C.; Durumeric, Z.: On the Origin of Scanning: The Impact of Location on Internet-Wide Scans. In: IMC '20: Proceedings of the ACM Internet Measurement Conference. S. 662–679, 2020.

Short Papers

Short Paper: Untersuchung des Gender-gaps bei Cybersecurity-Publikationen

Nico Mayer[1], Steffen Wendzel[2], Jörg Keller[3]

Abstract: Im Bereich der Informatik konnte bereits aufgezeigt werden, dass es eine geringere Anzahl an weiblichen Autoren von wissenschaftlichen Publikationen gibt. Wir untersuchen die Frage, ob es ein ähnliches Verhältnis bei Publikationen im Teilbereich Cybersecurity gibt, ob Frauen seltener zitiert werden als Männer und ob ein Trend in den letzten 10 Jahren erkannt werden kann. Zur Beantwortung der Frage untersuchen wir ausgewählte Journale und Tagungen auf deren Zitierungsanzahl und die Geschlechtsverteilung der Autor:innen. Wir stellen *keinen* Gender-gap in Form einer Benachteiligung in der Zitierungsanzahl fest, allerdings liegt ein Gender-gap bei der Publikationszahl vor, der jedoch erwartbar ist und zudem in Cybersecurity weniger ausgeprägt ist als in der Informatik als Ganzes.

Keywords: Cybersecurity; IT-Sicherheit; Gender-gap; Bibliometrie; Szientometrie; Zitate

1 Einleitung

Es gibt auch heute noch Berufsfelder mit vorwiegend männlichen Beschäftigten, etwa die MINT-Fächer [IW21; WD17]. Mattauch et al. haben bei der Untersuchung des Gender-Gaps (unter dem wir jeglichen Unterschied zwischen Frauen und Männern verstehen) ausgewählter Informatik-Publikationen herausgefunden, dass im Durchschnitt weniger als 10% aller Publikationen von Frauen beigesteuert wurden [Ma20]. In dieser Arbeit wird ergänzend die Fragestellung aufgegriffen, ob es einen solchen Gender-Gap bei wissenschaftlichen Publikationen im Teilbereich Cybersecurity gibt, d.h. ob der Anteil der Cybersecurity-Publikationen von Frauen geringer ist als 50%, und ob er mindestens ihrem Anteil an Informatik-Publikationen insgesamt entspricht.. Neben der Frauenquote wird zusätzlich untersucht, ob weibliche Autoren eine niedrigere Zitierungsanzahl als ihre männlichen Kollegen erhalten und ob sich ein Trend in den letzten 10 Jahren aufzeigt.

Der Rest dieses Artikels gliedert sich wie folgt. In Kap. 2 werden der aktuelle Forschungsstand zum Gender-gap und Methoden zur Geschlechtszuordnung aufgezeigt. In Kap. 3 wird die Vorgehensweise dargestellt, mit der der Gender-gap untersucht wurde. Anschließend werden die Ergebnisse in Kap. 4 dargestellt, und in Kap. 5 ein Fazit gezogen.

[1] FernUniversität in Hagen, Fakultät für Mathematik und Informatik, 58084 Hagen, Germany mayer.nico95@gmail.com
[2] Hochschule Worms, Zentrum für Technologie und Transfer (ZTT) / FernUniversität in Hagen, Fakultät für Mathematik und Informatik wendzel@hs-worms.de
[3] FernUniversität in Hagen, Fakultät für Mathematik und Informatik, 58084 Hagen, Germany joerg.keller@fernuni-hagen.de

doi:10.18420/sicherheit2022_11

2 Forschungsstand

Um den Gender-gap analysieren zu können, werden Methoden und Techniken zur Geschlechtszuweisung benötigt. Santamaría und Mihaljević haben dazu verschiedene Methoden (Gender API, Gender-guesser, genderize.io, NameAPI und NamSor) miteinander verglichen und Benchmarks aufgestellt. Durch Anpassung der Parameter konnte die Rate der fehlerhaften Gender-Klassifikationen bei NamSor auf unter 2% gesenkt werden, während weiterhin 75% der Namen klassifiziert werden konnten. Das Python-Paket Gender-guesser weist ohne Anpassung der Parameter mit 2,6% die geringste Rate an fehlerhaften Klassifikationen auf (NamSor 4,29%). Aufgrund der geringen Datenbasis von Gender-guesser geht dieses Ergebnis jedoch mit einer hohen Rate an nicht klassifizierten Datensätzen einher [SM18]. Abazi-Bexheti et al. haben 2019 den Gender-Gap im Bereich der Informatik untersucht. Die Publikationsdaten mit den zugehörigen Autorennamen wurden aus der dblp-Datenbank extrahiert und anschließend den gewonnenen Namen mittels Gender-guesser ein Geschlecht zugewiesen. Der Gap der Frauenquote unter den Autoren hat demnach in den letzten 40 Jahren bei Journalen, Konferenzen und Büchern stark zugenommen [AKA19]. Auch Mattauch et al. haben den Gender-Gap im Bereich Informatik untersucht. Die Datengrundlage bilden die Namen der Autor:innen, die in ausgewählten Konferenzen in einem Zeitraum von 6 Jahren publiziert haben. Die Namen der Autor:innen wurden per Skript von dblp computer science bibliography extrahiert und anschließend mit NamSor einem Geschlecht zugewiesen. Die Daten zeigten einen eindeutigen Gender-Gap. Der höchste Frauenanteil einer untersuchten Konferenz betrug 11,73% [Ma20]. Zitationen im Bereich Cybersecurity wurden bereits von Wendzel et al. [WLC20] untersucht, doch haben die Autoren dabei keinerlei Unterscheidung hinsichtlich Gender getroffen. Ein für unseren Beitrag wichtiges Ergebnis der Arbeit ist, dass eine höhere Autorenzahl nur in bestimmten Fällen zu signifikant zu mehr Zitationen pro Publikation führt und diese zudem abhängig von der Teildisziplin (Kryptographie, Netzwerksicherheit etc.) ist.

3 Methodik

Um zu untersuchen, ob ein Gender-Gap bei Cybersecurity Publikationen existiert, werden zuerst Metadaten von wissenschaftlichen Publikationen sowie Zitierungsdaten gesammelt. Die Metadaten der Publikationen enthalten keine Angaben zum Geschlecht von Autor:innen. Aus diesem Grund wird die Klassifizierung des Autoren-Geschlechts über den Namen vorgenommen, wozu verschiedene Services evaluiert werden.

Datenauswahl und -quellen Unsere Datenbasis bilden Publikationsdaten von ausgewählten Journalen und Tagungen. Dieser Ansatz wurde auch von Mattauch et al. eingesetzt [Ma20] und hat den Vorteil, dass die Qualität der Publikationen im Hinblick auf den gewollten wissenschaftlichen Standard bestimmt werden kann. Bei der Auswahl der Journale und Tagungen (s. Tab. 1) wurde die Google Ranking-Liste der Top Publikationen (`https://scholar.google.es/citations?view_op=top_venues&vq=eng_computersecuritycryptography`) sowie das CORE-Ranking (`https://www.core.edu.au`) verwendet.

Titel	Typ	Core Ranking	Abkürzung
IEEE Transactions on Information Forensics and Security	Journal	A*	tifs
Computers and Security	Journal	B	compsec
IEEE Transactions on Dependable and Secure Computing	Journal	A	tdsc
IEEE Security and Privacy Magazine	Journal	B	ieeesp
ACM Transactions on Privacy and Security	Journal	A	tissec
IEEE Symposium on Security and Privacy	Tagung	A*	sp
ACM Conference on Computer and Communications Security	Tagung	A*	ccs
International Conference on Availability, Reliability and Security	Tagung	B	IEEEares
Asia Conference on Information, Computer and Communications Security	Tagung	B	asiaccs
European Symposium On Research In Computer Security	Tagung	A	esorics

Tab. 1: Übersicht der ausgewählten Journale und Tagungen.

Datenquelle	Google Scholar	Microsoft Academic	Scopus	WoS	Open Citations COCI
Abdeckung der Zitierungen nach Martín-Martín et al. im Bereich Computer Science	88%	63%	61%	48%	30%
API vorhanden	Nein	Ja	Ja	Ja	Ja
Daten zu zitierenden Publikationen einsehbar (DOI, Id, Autornennamen o.Ä.)	Ja	Ja	Ja	Ja	Ja

Tab. 2: Vergleich der Quellen für Zitierungsdaten [Cl21; Ei20; El21; Go21; MT+21; Op21]

Es gibt mehrere Quellen, die diese Daten bereitstellen. Analog Mattauch et al. wurden die Daten über die kostenfreie API der dblp computer science bibliography bezogen. Im Gegensatz zu den APIs der einzelnen Verlage (etwa IEEE und ACM) müssen bei dblp nicht mehrere unterschiedliche Schnittstellen angefragt werden. Zur Beantwortung der Forschungsfragen werden die gesammelten Informationen noch um Zitierungsdaten mit möglichst hoher Abdeckung im Bereich der Informatik erweitert.

Selbstzitierungen King et al. haben aufgezeigt, dass Männer sich häufiger als Frauen selbst zitieren [Ki17]. Bei Selbstzitierungen kann die eigene Mitarbeit der Autor:in an der jeweiligen Publikation im Vordergrund stehen. Bei der Untersuchung des Gender-gaps wird daher ein Vergleich der Zitierungen zwischen Männern und Frauen aufgrund der unterschiedlich hohen Anteile an Selbstzitierungen verfälscht. Um die Qualität des Vergleichs zu erhöhen, müssen die darin enthaltenen Selbstzitierungen erkannt und ausgeschlossen werden. Mit diesen Anforderungen wurden die Dienste Crossref, Microsoft Academic, OpenCitations und GoogleScholar näher untersucht, s. Tab. 2. Im Bereich der Informatik besitzt Google Scholar zwar die höchste Abdeckung, bietet jedoch keine öffentliche API an. Darüber hinaus wird in den Nutzungsbedingungen automatisierten Suchanfragen widersprochen. Die zweithöchste Abdeckung hat Microsoft Academic mit 63% [MT+21]. Neben der Suche über die Webseite wird eine API mit der Obergrenze von 10.000 kostenlosen Requests/Monat angeboten, weshalb Microsoft Academic als Quelle verwendet wird.

Gewertete Zitate King et al. haben die Zitierungen bei wissenschaftlichen Publikationen in 'author-to-author' und 'paper-to-paper' eingeteilt [Ki17]. Die paper-to-paper Zitierungen beziehen sich auf die gesamte Publikation und finden häufig Anwendung bei der Anzeige der Anzahl an Zitierungen bei Publikationsdatenbanken. Die Zahl sagt aus, wie häufig die Publikation A von anderen Publikationen referenziert/zitiert wurde. Die zweite Zitierungsart ist die author-to-author Zitierung. Dabei werden die Autor:innen der jeweiligen Publikationen miteinander verglichen. Jede Autor:in aus Publikation A wird mit jeder Autor:in aus Publikation B verbunden. Die daraus resultierenden Verbindungen stellen dann jeweils ein author-to-author Paar dar. Zwei Publikationen mit jeweils 2 Autor:innen haben demnach 4 author-to-author Verbindungen [Ki17]. Jede dieser Verbindungen kann als Selbstzitierung markiert werden, wenn es sich bei den beiden verbundenen Autor:innen um die gleiche Person handelt. Die Berechnung der Zitierungen kann mit author-to-author Selbstzitierungen individuell auf die einzelne Autor:in zugeschnitten werden. Da die einzelnen Autor:innen der Publikationen Gegenstand dieser Analyse sind, haben wir uns dafür entschieden, die author-to-author Selbstzitierungen für die Berechnung der gewerteten Zitate zu verwenden. Die verbleibenden Zitierungen müssen noch auf die Jahre aufgeteilt werden, die eine Publikation bereits veröffentlich ist, um den Altersunterschied der Publikationen auszugleichen. Der Hintergrund dafür ist, dass ältere Publikationen aufgrund ihres Alters von einer größeren Anzahl an Publikationen hätte zitiert werden können. Final erfolgt die Berechnung der gewerteten Zitate mit folgender Formel: $Zitate = \frac{Zitierungen - Selbstzitierungen}{AktuellesJahr - Publikationsjahr}$, wobei immer galt: *Aktuelles Jahr* (2021) >*Publikationsjahr* (max. 2020).

Geschlechtszuweisung In den Metadaten der dblp sind keine Informationen über das Geschlecht der Autor:innen enthalten. Daher haben wir die Zuordnung des Geschlechts anhand des Namens durchgeführt. Für einen kleinen Teil der Autor:innen (ca. 2.000) konnte außerdem der geografische Kontext über ORCID gesammelt und zur Verbesserung der Zuweisung verwendet werden. Für die verbleibenden Autor:innen lag entweder keine ORCID-ID in dblp vor oder die Landesinformationen in ORCID waren nicht öffentlich verfügbar. Es gibt einige Dienste, wie NamSor und Gender-guesser, die Namen einem Geschlecht zuordnen. Dabei werden unterschiedliche Methoden eingesetzt, die von Namenslisten bis hin zu maschinellem Lernen reichen. Santamaría und Mihaljević haben verschiedene Dienste zur Geschlechtsermittlung miteinander verglichen und Benchmarks ausgeführt [SM18]. Die Ergebnisse der Benchmarks wurden von Menéndez im Jahr 2020 ebenfalls validiert. Dabei konnten Verbesserungen der Dienste NamSor, genderize.io und NameApi festgestellt werden [Me20]. Im Folgenden werden die Ergebnisse von Santamaría und Mihaljević aufgeführt und die Benchmarks von 5 Diensten miteinander verglichen, s. Tab. 3.

Gender API hat in diesem Benchmark zwar die höchste Genauigkeit nach Parametertuning, jedoch sind nur 500 Anfragen/Monat kostenlos möglich. Aufgrund der unlimitiert möglichen kostenlosen Anfragen bei gender-guesser und der damit verbundenen hohen Genauigkeit wird gender-guesser für die erste Klassifizierung in dieser Arbeit ausgewählt. Um die hohe Rate an nicht klassifizierbaren Namen von gender-guesser zu kompensieren, werden in einer zweiten Runde alle noch nicht klassifizierten Namen mit NamSor klassifiziert.

Dienst	NamSor	gender-guesser	genderize.io	NameApi	Gender API
Genauigkeit ohne Parametertuning ohne nicht klassifizierte Namen	95,71%	97,36%	94,98%	96,58%	94,97%
Nicht klassifizierte Namen ohne Parameterturning	15,04%	20,12%	9,74%	15,04%	3,01%
Genauigkeit mit max. 25% nicht klassifizierten Namen	98,61%	97,71%	98,26%	96,98%	99,12%
kostenlose Requests	5.000 Credits/Monat	unlimitiert	1.000 Requests/Tag	10.000 Credits/Monat 3 Credits/Anfrage für Geschlechtszuordnung	500 Credits/Monat 1 Credit/Anfrage für Geschlechtszuordnung

Tab. 3: Vergleich der Dienste zur Geschlechtsermittlung [Ge21a; Ge21b; Na21a; Na21b; Pé16; SM18]

Typ	Journal					Tagung				
Kürzel	tifs	compsec	tdsc	ieeesp	tissec	ccs	sp	IEEEares	asiaccs	esorics
Anzahl der Publikationen	2023	1410	624	891	192	2921	895	1224	447	976

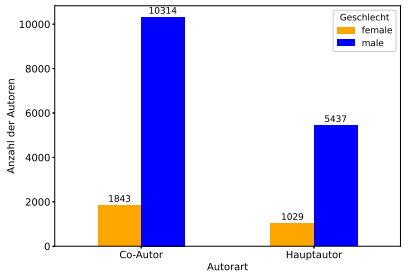
Tab. 4: Anzahl der Publikationen pro Journal bzw. Tagung

NamSor wird aufgrund der hohen Anzahl an kostenlosen Requests/Monat und nur minimal geringeren Genauigkeit als Gender API gewählt. Die hohe Genauigkeit von NamSor wurde von Santamaría und Mihaljević mithilfe von Parametertuning erreicht. Carsenat empfiehlt weiteres Parametertuning [Ca19, S. 11], weshalb wir die Konfiguration ebenfalls verwenden.

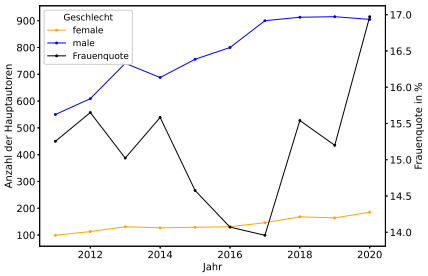
4 Ergebnisse

Von unseren 11.602 analysierten Publikationen wurden 5.140 in einem Journal und 6.463 in Tagungsbänden veröffentlicht, s. Tab. 4. Diese Publikationen sind 41.603 Autor:innen zugeordnet, wobei diese Anzahl einige Autor:innen mehrfach beinhaltet, da sie an mehreren Publikationen mitgearbeitet haben (insg. 21.024 unterschiedl. Autor:innen).

Autor:innen In den Daten befinden sich 11.603 *Haupt*autor:innen, worunter wir gemäß den Usancen in Informatik-Publikationen Erstautor:innen verstehen. Diese entsprechen gruppiert noch 8.227 individuellen Autor:innen. Neben den Hauptautor:innen sind 30.000 Co-Autor:innen enthalten. Ohne Duplikate entspricht dies 15.663 einzelnen Co-Autor:innen. Es konnte insg. 6.466 Haupt- und 12.157 Co-Autor:innen ein Geschlecht mit dem hybriden Ansatz (gender-guesser und NamSor) zugewiesen werden. Beide Zahlen enthalten Autor:innen, die jeweils als Haupt- und als Co-Autor:in publiziert haben und somit in beiden Mengen enthalten sind. Die Frauenquoten in Abb. 1a der Haupt- und Co-Autor:innen ergeben ähnliche Werte. Von den Co- und Hauptautor:innen wurden 15,16% bzw. 15,91% als Frauen klassifiziert. Diese Quoten bleiben bei den Hauptautor:innen auch über den Analysezeitraum von 10 Jahren relativ konstant. Abb. 1b stellt die Anzahl der Hauptautor:innen aufgeteilt nach Jahr und Geschlecht dar. Die Frauenquote befindet sich in diesem Analysezeitraum bei 13%–16%. Wir sehen also einen Gender-Gap, da die Frauenquote weniger als 50% beträgt, allerdings fällt dieser geringer aus als bei Informatik-Publikationen insgesamt [Ma20].



(a) Haupt- und Co-Autoren nach klassifiziertem Geschlecht



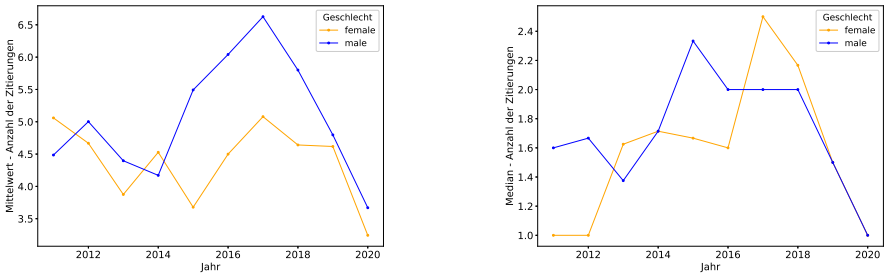
(b) Hauptautoren nach Geschlecht pro Jahr

Abb. 1: Übersicht Haupt- und Co-Autoren

Zitierungsdaten In unseren Daten befinden sich insgesamt 5.466.434 Zitationseinträge. Davon referenzieren 309.995 Einträge auf eine der 11.603 Publikationen, die als Grundlage für die Analyse der Zitierungsdaten verwendet werden. Von diesen enthalten 6.727 Publikationen Selbstzitierungen (paper-to-paper). Das entspricht fast 58% der Publikationen (mit durchschnittlich etwas mehr als einer Selbstzitierung). Die Analyse der author-to-author Selbstzitierungen nach der von King et al. vorgestellten Methode hat ergeben, dass sich männliche Autoren in den uns vorliegenden Daten 1,136x häufiger selbst zitiert haben.

Analyse Zunächst wird analysiert, ob Publikationen mehr Zitate erhalten, wenn die Hauptautoren männlich bzw. weiblich sind, und ob es dabei einen Unterschied zwischen Journalen und Tagungen gibt. Weitergehend wird untersucht, ob es in der Anzahl dieser Zitierungen einen Trend in den letzten 10 Jahren gegeben hat. In Abb. 2a wird die zu untersuchende Anzahl der Zitierungen dargestellt. Erkennbar ist, dass die Zitierungsanzahl der Männer von 2015 bis 2017 stark angestiegen ist. Anschließend fällt diese Kurve wieder ab, was unter anderem dem Publikationsdatum geschuldet sein könnte. Es wird jedoch deutlich, dass bis auf diese zwei Jahre die Werte bei männlichen Hauptautoren höher liegen. Der berechnete Mittelwert ist jedoch anfällig für einzelne Werte die wesentlich höher sind als die restlichen. Daher wird in Abb. 2b noch der Median dargestellt. Im Jahr 2017 haben männliche Autoren die höchste Anzahl an Zitierungen erhalten. Im Vergleich dagegen liegen beim Median die weiblichen Autoren vor den männlichen. Diese hohe Anzahl ist ebenfalls einigen wenigen Publikationen geschuldet. Eine Recherche gegen die Datenbasis hat ergeben, dass aus den 900 Publikationen mit männlichen Hauptautoren aus dem Jahr 2017 die obersten 10 über 33% aller Zitierungen erhalten haben. Unter Berücksichtigung von Median und Mittelwert kann kein wesentlicher Unterschied festgestellt werden. Alle größeren Abweichungen scheinen nur einige wenige Publikationen als Ursache zu haben.

Gini-Koeffizient Deshalb analysieren wir mit dem Gini-Koeffizient [Se73], ob eine Ungleichverteilung bei der Zitierungsanzahl besteht, d.h. ob wenige Publikationen einen



(a) Mittelwert.

(b) Median

Abb. 2: Vergleich Anzahl der Zitierungen zwischen männlichen und weiblichen Hauptautoren pro Jahr

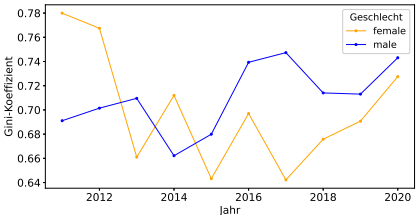


Abb. 3: Gini-Koeffizient für die Anzahl der Zitierungen zwischen Männern und Frauen pro Jahr

Großteil aller Zitierungen erhalten. Der Gini-Koeffizient für alle Publikationen entspricht 0,73 und deutet somit auf eine Ungleichverteilung der Zitierungen hin. In Abb. 3 wird der Gini-Koeffizient für die einzelnen Jahre dargestellt. Dabei wurde analog den vorherigen Diagrammen jedes Jahr noch einmal in das Geschlecht der Hauptautoren unterteilt; der Wert schwankt zwischen 0,64 und 0,78. Die hohen Werte des Gini-Koeffizienten untermauern, dass wenige Publikationen den größten Teil aller Zitierungen erhalten. Diese Ungleichverteilung existiert bei Publikationen mit männlichen und weiblichen Hauptautoren gleichermaßen.

Zitierungen pro Journal/Tagung Um die Analyse zu verfeinern, wurde in Abb. 4 jedes Journal bzw. jede Tagung separat dargestellt. Bei den Journalen ist auch nach der Aufteilung, bis auf einzelne Ausreißer, kein allgemeiner Trend ersichtlich. Alle Linien kreuzen sich mehrfach und liegen fast immer auf einem ähnlichen Niveau. Bei Tagungen wird die Abweichung zwischen 2015 und 2018 vor allem durch sp und ccs verursacht. Final ergibt sich kein ausschlaggebender Unterschied zwischen den analysierten Journalen und Tagungen bei der Anzahl der Zitierungen für Männer und Frauen. Die größten Unterschiede konnten jeweils auf einzelne Publikationen zurückgeführt werden.

Frauenquote der Haupt- bzw. Co-Autoren In Abb. 5 wird die Frauenqote aus den letzten 10 Jahren für jedes Journal und jede Tagung einzeln visalisiert. Die dargestellten

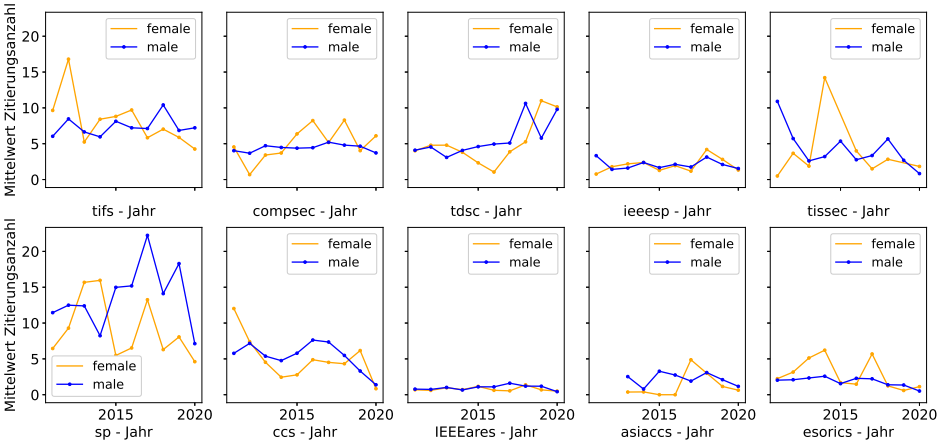


Abb. 4: Vergleich der Zitierungsanzahl zwischen Männern und Frauen pro Jahr pro Journal/Tagung

Frauenquoten bei den Hauptautoren befinden sich überwiegend im Bereich zwischen 10% und 20%, was mit der anfänglich berechneten Frauenquote von 15% übereinstimmt. Die hohen Schwankungen der Quote sind zudem häufig verbunden mit der Menge der Publikationen des jeweiligen Journals bzw. Tagung. Als Beispiel sei tissec genannt, bei dem die Frauenquote zwischen 0 und fast 30% variiert. Die allgemeine Aussage, dass weibliche Autoren unterpräsent sind, trifft auf alle Journale bzw. Tagungen zu. Im Abgleich mit den Hauptautoren (Abb. 1b) ist bei den Co-Autoren ein nahezu identisches Bild zu erkennen, weshalb die Frauenquote der Co-Autoren nicht gesondert dargestellt wird. Journale oder Tagungen, bei denen diese Quote maßgeblich geringer ausfällt, konnten nicht ermittelt werden. Für den Bereich Cybersecurity konnte über alle Publikationen hinweg der Trend erkannt werden, dass die Gesamtzahl der Autor:innen ansteigt. In den analysierten Jahren ist die Anzahl der Männer und Frauen in einem nahezu gleichbleibenden Verhältnis angestiegen.

Zusätzlich wurde untersucht, ob double blind review (Reviewern ist der Name und damit das Geschlecht der Autor:innen unbekannt) einen Einfluss auf die Frauenquote hat. Aktuell führen gemäß ihren Webseiten alle Konferenzen und Journals bis auf compsec, tissec und esorics double-blind review durch. Betrachtet man die über die Jahre gemittelten Frauenquoten aus Abb. 5, so ist esorics die Könferenz mit der zweithöchsten mittleren Frauenquote (17%, Range von 11% bis 18%), bei den Journals erreichen compsec und tissec die höchste bzw. niedrigste mittlere Frauenquote (18% bzw. 10%). Hieraus lässt sich nicht ableiten, dass eine Benachteiligung im Review aufgrund des Gender auftritt.

5 Fazit

Unsere Ergebnisse untermauern die bisher nur vermutete Tatsache, dass Autor*innen* bei wissenschaftlichen Publikationen im Bereich Cybersecurity mit einer Quote von 15%

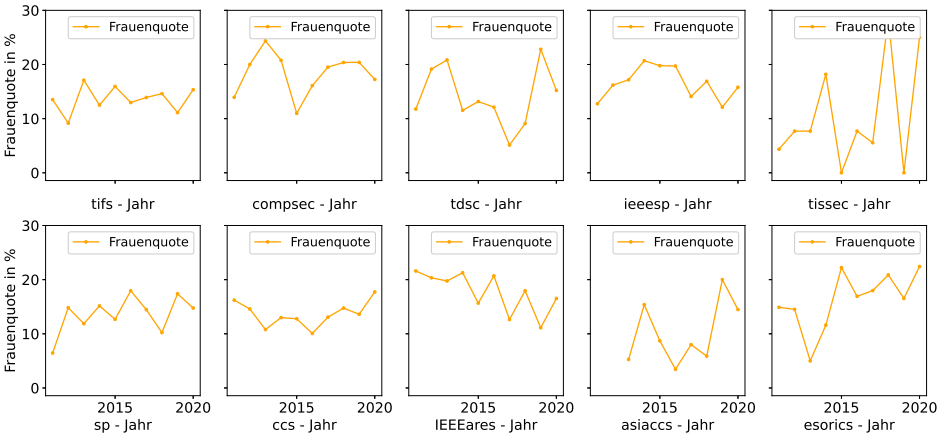


Abb. 5: Frauenquote der Hauptautoren aufgeteilt in die einzelnen Journale und Tagungen pro Jahr

weniger präsent sind als ihre männlichen Kollegen. Es konnte somit ein Gender-gap in Form einer niedrigen Frauenquote der Haupt- als auch Co-Autoren nachgewiesen werden. Im Zusammenhang mit der Anzahl der Zitierungen konnte hingegen keine Benachteiligung von Autor:innen festgestellt werden (weder bei den selektierten Top-Tagungen noch bei Top-Journalen). Sowohl bei Autoren als auch bei Autorinnen zeigt sich, dass einige wenige Publikationen einen Großteil der Zitate erhalten. Gesamtgesellschaftlich wäre es wünschenswert, wenn insg. mehr Frauen im Bereich Cybersecurity publizieren (und tätig) würden, was auch hinsichtlich der Nutzung des personellen Potenzials im Kontext des Fachkräftemangels bedeutungsvoll wäre. Die Ableitung politischer Handlungsempfehlungen ist jedoch kein Ziel *dieses* Beitrags. Eine Weiterführung der Untersuchungen ist u.a. hinsichtlich der Fragestellung denkbar, ob die Frauenquote abhängig von A, B und C Journalen/Tagungen variiert oder ob Männer/Frauen bevorzugt Männer/Frauen zitieren. Aufgrund von Lizenzrechten können wir unsere Daten ausschließlich auf Anfrage bereitstellen.

Literatur

[AKA19] Abazi-Bexheti, L.; Kadriu, A.; Apostolova, M.: Investigating Gender Gap in Computer Science Research Community. In: 42nd Int. Conv. Inform. and Commun. Techn., Electr. and Microelectr. (MIPRO). S. 737–741, Mai 2019.

[Ca19] Carsenat, E.: Inferring gender from names in any region, language, or alphabet, Preprint, Researchgate, Nov. 2019.

[Cl21] Clarivate: Clarivate Developer Portal - Swagger UI, 2021, URL: `https://api.clarivate.com/swagger-ui/`, Stand: 05. 07. 2021.

[Ei20] Eide, D.: What is Project Academic Knowledge? - Microsoft Academic Services, 2020.

[El21] Elsevier: Abstract Retrieval API, 2021, URL: https://dev.elsevier.com/documentation/AbstractRetrievalAPI.wadl, Stand: 05. 07. 2021.

[Ge21a] Gender API: Gender API About, de, 2021, URL: https://gender-api.com/de/pricing, Stand: 11. 07. 2021.

[Ge21b] Genderize.io: Genderize.io | Determine the gender of a name, 2021, URL: https://genderize.io/, Stand: 11. 07. 2021.

[Go21] Google: Automated queries, 2021, URL: https://developers.google.com/search/docs/advanced/guidelines/automated-queries, Stand: 25. 05. 2021.

[IW21] IW Köln: MINT-Berufe: Frauenanteil in Deutschland bis 2018, de, 2021, URL: https://de.statista.com/statistik/daten/studie/946493/umfrage/frauenanteil-in-mint-berufen-in-deutschland/, Stand: 17. 07. 2021.

[Ki17] King, M. M.; Bergstrom, C. T.; Correll, S. J.; Jacquet, J.; West, J. D.: Men Set Their Own Cites High: Gender and Self-citation across Fields and over Time. Socius: Sociological Research for a Dynamic World 3/, 2017.

[Ma20] Mattauch, S.; Lohmann, K.; Hannig, F.; Lohmann, D.; Teich, J.: A bibliometric approach for detecting the gender gap in computer science. Communications of the ACM 63/5, S. 74–80, Apr. 2020.

[Me20] Menéndez, D. A.: Damegender: Writing and Comparing Gender Detection Tools, Preprint 5763, EasyChair, 2020, URL: https://easychair.org/publications/preprint/g6hG, Stand: 12. 10. 2021.

[MT+21] Martín-Martín, A.; Thelwall, M. et al.: Google Scholar, Microsoft Academic, Scopus, Dimensions, Web of Science, and OpenCitations' COCI: a multidisciplinary comparison of coverage via citations. Scientometrics 126/1, 2021.

[Na21a] NameAPI: NameAPI - Quotas, 2021, URL: https://www.nameapi.org/en/account/quotas/, Stand: 11. 07. 2021.

[Na21b] Namsor: Name Ethnicity and Gender Classifier API - NamSor, Apr. 2021, URL: https://www.namsor.com/, Stand: 17. 04. 2021.

[Op21] OpenCitations: OpenCitations - Querying Data, 2021, URL: https://opencitations.net/querying, Stand: 05. 07. 2021.

[Pé16] Pérez, I. S.: gender-guesser: Get the gender from first name. 2016, URL: https://github.com/lead-ratings/gender-guesser, Stand: 25. 04. 2021.

[Se73] Sen, A.: On Economic Inequality. Oxford University Press, 1973.

[SM18] Santamaría, L.; Mihaljević, H.: Comparison and benchmark of name-to-gender inference services. PeerJ Computer Science 4/, e156, Juli 2018.

[WD17] Wang, M.-T.; Degol, J. L.: Gender Gap in Science, Technology, Engineering, and Mathematics (STEM). Educational Psychology Review 29/1, 2017.

[WLC20] Wendzel, S.; Lévy-Bencheton, C.; Caviglione, L.: Not all areas are equal: analysis of citations in information security research. Scientometrics 122/, S. 267–286, 2020.

Short Paper: Debating Ethics *with* Cybersecurity Students

Jan Breig[1] Dirk Westhoff[2]

Abstract: We aim to debate and eventually be able to carefully judge how realistic the following statement of a young computer scientist is: "*I would like to become an ethical correctly acting offensive cybersecurity expert*". The objective of this article is not to judge what is good and what is wrong behavior nor to present an overall solution to ethical dilemmas. Instead, the goal is to become aware of the various personal moral dilemmas a security expert may face during his work life. For this, a total of 14 cybersecurity students from HS Offenburg were asked to evaluate several case studies according to different ethical frameworks. The results and particularities are discussed, considering different ethical frameworks. We emphasize, that different ethical frameworks can lead to different preferred actions and that the moral understanding of the frameworks may differ even from student to student.

Keywords: Cybersecurity; ethical frameworks; offensive security techniques

1 Introduction

From our viewpoint it is essential for the well-being of a modern digitized society to educate people with a strong knowledge not only in computer science, but in particular in IT-security respectively cybersecurity. Over the recent years, a plethora of new job descriptions have emerged in this field, e. g. malware analyst, threat hunter, threat intelligence analyst, pentester etc.. This also includes to some degree offensive techniques, aiming to successfully attack systems. However, it is of similar or even more importance that these people also possess a strong moral compass. Just because a security expert has the capability to exploit security vulnerabilities, it is highly desirable that he/she can discipline himself/herself not to perform such actions in the wild. On the other hand, companies, authorities and countries require persons with such technical IT-capabilities to better understand how to defend their systems, economy and society from massively increasing number of attacks via the Internet.
To debate this and to build up a common understanding is not only beneficial for the individual security expert himself, but also for the society. Use cases have been offered for debating ethical questions according to the daily work of the general computer science community [GI]. In this article, we would like to debate more specific ethical questions for the increasing and demanding subgroup of cybersecurity experts. Recently Macnish and van der Ham have identified the missing of a proper ethical education at an undergraduate and postgraduate level [Mv20] for computer scientists. They recommend that ethics should be taught in far greater depth on computer science courses than it is currently the case.

[1] Hochschule Offenburg, Badstraße 24, 77652 Offenburg, Germany jan.breig@hs-offenburg.de
[2] Hochschule Offenburg, Badstraße 24, 77652 Offenburg, Germany dirk.westhoff@hs-offenburg.de

doi:10.18420/sicherheit2022_12

Moreover, they point out that typically Research Ethics Boards (REB) fail to propose reasonable ethical proposals for cybersecurity experts due to the lack of their own expertise. Exactly this is the strength of the work at hand since it involves upcoming cybersecurity experts in the judgment of various case studies after they attended the ethics lecture at HS Offenburg.
Let us for the moment forget that by pointing out that such a concrete and specific professional wish "*I would like to become an ethical correctly acting offensive cybersecurity expert*" is rather seldomly the case. However if it turns out, that on the one side the wish to always act ethically correct and on the other side the profession to be an offensive cybersecurity expert is an unsolvable dilemma, does this mean that the society ends up in a situation that most human beings which would like to act ethically correct will decide to not become a cybersecurity expert? Moreover, would this also mean that an over-proportional number of actually active offensive cybersecurity experts does not really care about moral or ethical acting since those who are aware of this dilemma have not started such a carrier in the first place or did just quit the job? Where exactly is the borderline of techniques and toolsets to be used between a cybersecurity expert and an offensively acting cybersecurity expert e. g. what about port scanning and other elementary techniques to spy out a victim to prepare an active attack? Can an *arp -a* or a *ping* yet be considered as belonging to the first phase of a cyber-kill-chain [Lo]? Is it sufficient to argue that *white hat hackers* have their own code of ethics e. g. ground rules are established with the defender regarding targets and in what is off-limits and that *certified ethical hackers* always have undergone a rigorous moral and law inspection? Does this end the overall discussion? Obviously, we cannot answer all these questions within such an article. However, we would like to open the discussion with respect to a *correct ethical acting or at least reflected ethical acting for offensive cybersecurity experts*.

2 Philosophical Ethical Frameworks

Over the centuries, the philosophical prophets of their epoch established several directions like *Virtue Ethics* (Aristoteles (384-322 B. C.)), *Ethics of duty* (Immanuel Kant (1724-1804)), *Utilitarianism* (John Stuart Mill (1806-1873)), *Ethics of responsibility* (Max Weber (1864-1920)) or *Ethos ethics* (Albert Schweitzer (1875-1965)) to name a few. Will the fundamental rules of what acting is considered to be good or evil, right or wrong, or virtue and vice argued in some of these ethical frameworks indeed support the insights of our offensive cybersecurity expert 'in spe' even if these ethical frameworks definitively could not have considered dilemmas that arose due to the appearance of the cyberspace?
According to three predominant different philosophical ethical frameworks, Virtue, Utilitarian and Deontological Ethics, a human being, in our case a cybersecurity expert [Ma18, p.49], may like to ask the following questions to decide what is ethically justifiably:

Virtue Ethics (VE): Which position best expresses my value and character? If I choose this, can I live with myself?

Utilitarian Ethics (UE): Which position will give the greatest positive utility and produce the fewest negative consequences? What costs respectively benefits are associated with each outcome?
Deontological (DE): Who will be affected by this decision? Am I treating others as a means or an end in themselves? If my actions became a rule and I myself was subject to that rule, would I accept it and view it as ethical?

In [Ma18, p.49] Manjikian summarizes the pros and cons for the usage of such philosophical frameworks in today's decision processes. Here we aggregate some of them: The strong point of Aristotle's Virtue Ethics approach is, that it creates consistent ethical positions across issues and it emphasizes the character of decision making. As a downside Manjikian argues, that the framework is traditional and therefore perhaps outdated in new environments. The calculations of the utilitarian ethics approach are "clean and often value free". It is universally valid and not based on the value of a particular culture. As a clear downside, the approach expects to adopting an instrumental view of human beings as means to an end. Human being own rights are fully ignored. Kant's deontological approach is judged by Manjikian such that the focus is on those affected by decisions. Its clear benefit is the reciprocal character forcing the acting party to see himself as both decider and subject of decision. As the downside with Kant's approach, she argues that it "overemphasizes duty to individuals over duty to produce the best possible outcome".

3 Acting in Contexts - Professional Life

Moreover, our security expert 'in spe' may also come to a point where he/she asks himself/herself whether it is morally justifiably to ethically act and behave differently in different contexts e. g. at home, in the public space, as an online user and, finally, as a professional actor. For the latter, does it make a difference to ask: "How should I act ethically as an employee? ...as a computer scientist? ...as cybersecurity expert? ... as an offensive cybersecurity expert?" With respect to an ethically correct behavior does it make a difference for an offensive cybersecurity expert who will be the employer. Eventually a security expert will receive job offers from companies, governmental and/or intelligence agencies, public authorities or the military.
Here, belonging to one or another employer does surely make a difference. Similar can be stated with respect to the concrete circumstances which in some cases might even allow for public authorities to use tools like e. g. *Staatstrojaner* or *Pegasus*. Sometimes, acting in conformity to law may still be unethical. Obviously, the opposite can also be the case ending for some people in a personal moral dilemma which frequently cannot be resolved over the long run.[3] Of course there are also many situations which are solvable without moral dilemma e. g. pentests carried out with previous agreement or situations where many if not all individuals would agree that a certain behavior is ethically un/justifiable.[4]

[3] Note that in this article we are not debating if a person is acting in conformity to law.
[4] However since the goal is to become aware of and discuss potential moral dilemmas, these situations are not taken into account in this article.

4 Ethical Frameworks for Cyber-communities

Moreover, we can observe that shortly after the appearance of the Internet era more specific ethical frameworks were either implicitly or explicitly established [Ma18, p.63].

Old Hacker Ethic: "Information wants to be free", laws do not apply in cyberspace, transparency is more important than privacy.
New Hacker Ethic: Community should govern itself, privacy is important, don't freeload.
Professional Hacker Ethic: Professional organization sets behavior norms, theft of personal information is not ethically justifiable, nations can regulate cyberspace.

We can also mention the work of *Research Ethic Boards* who are installed to judge the activities within various research directions in academia. However, as pointed out in [Mv20], the members of such boards rarely have the competence or technical background to judge cybersecurity related questions. The GI's ethical guidelines for computer scientists, encouraging people to critical thinking, also rarely help when it comes to the sometimes very specific situations a cybersecurity expert may face during his work. Due to this, we consired the outcome of our discussions about ethics *with* cybersecurity students to be of interest for the overall IT-security community. .[5]

5 Individual's Moral Compass and Community's Code of Conduct

The moral compass in philosophical ethics, recommending concepts of right and wrong conduct is indeed envisioned to be a fundamental part of each individual human being. But also groups, communities or the society as a whole may either implicitly or explicitly follow an ethical framework or (at least) a code of conduct as another agreed norm of right behavior besides the pure law. The fundamental problems applied ethics has to deal with has been verbally illustrated by the trolley problem [Fo78] in which a person must decide between killing one person or letting several persons die. What should be emphasized is that the trolley problem points out a decisional dilemma thus, the recommendation of right and wrong conduct is not always such clear. We can also observe from the trolley problem, that in some cases simply not acting does mean having chosen implicitly and thus also having to life with the consequences.
Independently of any companies', government agencies', or even intelligence agencies' written or silently agreed code of conduct, *whistleblowing* may frequently start with an ethical dilemma, where a human being belonging to a group or community recognizes a conflict of his own values with the values of the unit he is working for. Or, speaking in the notion of the trolley problem: The employee values the survival of multiple persons higher than the survival of one person (or vice versa). Since both sides, the whistleblower as well as the group/community, may act ethically correct within 'their' chosen ethical frameworks, we observe that indeed, in specific situations, different ethical frameworks may contradict in what is considered to be good and wrong respectively.

[5] In contrast to discuss ethics *for* cybersecurity experts without considering their view.

6 Applying Ethical Frameworks

Next we describe three exemplary case studies along the supply-chain of vulnerabilities and exploits of IT-ecosystems. We will see, that some of these case studies do not necessarily involve a cybersecurity expert. However even here either a significant amount of personal data is leaked, or later exploitation would not be possible without a previous situation like this. So when we talk about ethical behavior along the supply-chain of vulnerabilities and exploits of IT-ecosystems, ethical behavior is not only required by cybersecurity experts. Instead, when we want to mitigate security vulnerabilities we also have to talk about morally sound behavior of other actors.
The tables within the following subsections have been anonymously filled out by 14 international students of a cybersecurity master program when attending the ethics seminar. These students are ideal to be interviewed for such a little study since they are yet on their career path to become a cybersecurity expert. Obviously such a small data set is not representative. Nevertheless, it may serve to better understand arising personal dilemma of cybersecurity experts. Students have first been requested to i) fill out what *they believe* would be the outcome from the application of the three philosophical ethical frameworks to various cybersecurity related case studies (VE, UE, DE). Secondly, ii) they should judge what *they believe* would be the outcome when applying different ethical frameworks for cyber-communities. Finally, iii) they should provide *their own personal judgment*.[6]
The notation **+x**, **−x** and **0x** in tabulars 1, 2 and 3 means, that **x** students believe the concrete action is justifiable, unjustifiable or undecidable respectively within the given ethical framework. We aggregated the outcome by purely listing the majority of all votes (out of 14 possible votes). Thus a value of +10 (ten students voted justifiable) also means that four students either voted **0** (unjustifiable) or **–** (undecidable).

6.1 Case study 1: App development

Before we start debating ethical dilemmas cybersecurity experts may face, let us for the moment consider the work of an App programmer. This person would never claim to be an offensive security expert, since his knowledge is limited to Java programming, in particular the development of mobile Apps.

Careless App-Developer: May tend to implement the app asking for more permissions such that at the end the running software is over-privileged e. g. to assure that the internet connection is not blocked by too strict access rights.

Intentional App-Developer: Is aware that due to the old/new Android permission system the App can perform silently by far more actions than the user of the App becomes aware of [Er21]. Although his App is promoted to only provide service X, he has implemented more and more functionalities (internet connection, camera, SMS, phone, BT). At some

[6] One of the students did not fill out the category personal judgment.

point in time he decided to obfuscate the code such that it passes the static/dynamic code analysis check to be available in the App store just due to curiosity. In particular he obfuscated the internet connection transmitting the harvested personal data to a server with a concrete IP-address. However, at some point in time he decides to provide his App 'Give me everything' (of course he gave it another name say X) to the Appstore. Luckily, the app passes the security checks and thus is available in the store. Meanwhile it has been downloaded and used more than 100.000 times worldwide which means that the storage capacity of his servers are definitely too small. Maybe he should reconsider selling his App to a company that is more powerful and which could set up a suitable storage-system in the back-end for such an amount of user data. The students' answers for this use case are aggregated in tabular 1 according to the notation introduced in section 6.

	VE	**UE**	**DE**	**Old Hacker Ethic**	**New Hacker Ethic**	**Prof. Hacker Ethic**	**Personal Judgment**
Provide service X	+11	+12	+10	+11	+12	+11	+10
Additional functionality	+7	+10	–6	+12	+10	+8	–8
Obfuscation	–10	–7	–13	+11	–10	–12	–11
Send data to company	–10	+8	–12	+10	–11	–12	–10

Tab. 1: Students moral understanding according to different ethical frameworks and own personal judgment for case study on careless/intentional mobile App development.

6.2 Case study 2: Storage corruption vulnerabilities

The next person would definitively claim himself as cybersecurity expert. He build-up significant knowledge with respect to the Windows Memory protection means like Safe CRT, GS-Cookies, ASLR/DEP, Safe-SHE) and how to circumvent them. Naturally such knowledge is extremely valuable for a number of actors in the field of IT-security. In parallel to his bachelor studies he educated himself with respect to classical storage corruption vulnerabilities (StoCV) and early mitigation techniques, first by reading articles and blogs with respect to this issue and subsequently by building a PoC to demonstrate the attack for older Windows operating systems. However, he was also interested why for actual Windows versions such PoCs are not successful anymore. Thus, he also studied the details of actual in use storage-corruptions means in depth which mitigate the vulnerability of storage-corruption. During this study – in the spare-time to his bachelor studies – he got an idea how to corrupt also the storage of current operating system versions. To check whether his idea indeed succeeds, he built a PoC also here and yes it works! Up to this moment he had never the intention to make use of his knowledge. However, due to an unforeseeable and unpleasant personal economic situation – due to the COVID pandemic situation he lost his job in a restaurant – he considered to anonymously sell the PoC. Since he never used his PoC to attack a concrete person, nor does he know if the party which he sold the PoC is actually using it, respectively for which concrete destinations it will be used, he judged his

decision to be morally justifiable. The students' answers for this use case are aggregated in tabular 2.

	VE	UE	DE	Old Hacker Ethic	New Hacker Ethic	Prof. Hacker Ethic	Personal Judgment
Build up StoCV knowledge	+8	+12	+9	+10	+12	+11	+10
PoC for classical StoCV	+9	+13	+11	+10	+12	+10	+9
Get familiar with protection	+11	+10	+10	+11	+13	+10	+11
PoC to bypass protection	+9	+7	+9	+10	+10	–8	–6
Anonymously sell the PoC	–9	–7	–11	–7	–7	–10	–12

Tab. 2: Students moral understanding according to different ethical frameworks and own personal judgment for case study on storage corruption vulnerabilities.

6.3 Case study 3: Cyber-Kill-Chain

Offensive cybersecurity experts are required to be experienced in attacking systems, spying out individuals, installing malware, controlling victims remotely etc.. Lockheed Martin [Lo] developed the cyber-kill-chain with the objective to structure such a process and address advanced persistent threats (APT). The five phases of the cyber-kill-chain according to [Lo] are: 1. Reconnaissance (harvesting email addresses, conference information), 2. Weaponization (coupling exploit with backdoor into deliverable payload), 3. Delivery (deliver weaponized bundle to the victim via email, web, USB, etc.), 4. Exploitation (exploiting a vulnerability to execute code on victim's system) and 5. Actions on Objective (accomplishing original goals with 'hands on keyboard'). The students' answers for this use case are aggregated in tabular 3.

	VE	UE	DE	Old Hacker Ethic	New Hacker Ethic	Prof. Hacker Ethic	Personal Judgment
Reconnaissance	–8	+10	–8	+10	+8	–9	–6
Weaponization	–9	+7	–12	+9	+8	–11	–8
Delivery	–9	+7	–11	+9	–7	–10	–11
Exploitation	–10	+8	–12	+9	–7	–10	–11
Installation	–9	+7	–10	+9	–8	–10	–11
Command and Control	–11	–8	–12	+9	–7	–10	–11
Actions on Objective	–10	+8	–12	+9	+7	–10	–11

Tab. 3: Students moral understanding according to different ethical frameworks and own personal judgment for case study on cyber-kill-chain.

6.4 Remarks on Case Studies

As yet said, with such a small data set the results are not representative. Any form of statement made here can therefore not be generalized. Nevertheless, we want to share some of our observations. Not surprisingly, there is a tremendous difference what students believe what actions are ethical and unethically with respect to the different ethical frameworks. This holds for all debated case studies. The students' personal judgment matched different ethical frameworks for the different case studies.[7] Recall that with respect to their personal judgment the students did not explicitly follow a specific ethical framework, instead they decided rather intuitively. Generally, they were also quite restrictive in evaluating actions as ethically justifiable.

Remark 1: In particular phase 1 of the cyber-kill-chain seems to be debatable. Everyone would agree, that normally searching for someone's e-mail address over the public Internet is by far not an unethical act. However, if this yet is done with the intention to later attack this person, even phase 1 can be considered as highly unethical according to the classical ethical frameworks VE and DE.

Remark 2: It is frequently the case that those parties attacking a victim according to the various steps listed in the cyber-kill-chain do not implement exploits with respect to specific vulnerabilities on their own. Instead they buy such exploits from security companies or individuals which sell them over some platforms. How can the behavior of those security experts be judged that do on the one side implement the exploits, do never perform the cyber-kill-chain with respect to a concrete victim on their own, but (anonymously) sell it to actors which from that point on can and will apply the cyber-kill-chain to potentially every possible victim. Is the argumentation of those persons justifiable, who argue, since they do neither know the victim nor perform the concrete attack they do not act unethically?

Remark 3: In particular with respect to the case study three we expected some hints regarding the whistleblower's dilemma[8]: In particular the hacker ethics have been elaborated from communities and have also been adapted to the needs of these communities. Naturally, there should be significant conflicts with the moral compass of the classical ethical frameworks, otherwise, we assumed, there would not have been the need to establish them. However, amazingly at least from the students' votes we could not validate this. The judgments for VE, DE as well as the professional hacker ethics are almost equivalent.

Remark 4: Over all case studies, the old hacker ethic as well as UE evaluate most actions as ethical. The evaluations for VE, DE, new and professional hacker ethic were also rather similar. It turns out, that overall the traditional ethical framework UE has a better mapping to the old hacker ethic than with other traditional frameworks.

[7] Ethical frameworks with similar evaluation are marked with gray color in the tables.
[8] In 2019 the European Institution has presented a common statement for a whistleblowing guideline.

Remark 5: An individual, over the long run, may, either intuitively or reflected, orientate his/her own moral compass more with respect to one of the classical philosophical ethical frameworks. When the orientation is in line with VE, we expected that this may almost certainly result in a personal dilemma and conflict for the employee. The situation is getting even more problematic if he/she has to switch from one ethical framework to another one according to the various contexts he/she currently is involved in (private, public, professional). In particular following the framework of virtue ethics does not allow dividing one's actions e. g. public life, private life and professional life. We expected that, if the employee in the field of cybersecurity is dedicated to professional hacker ethic, the personal moral dilemma of such person is almost foreseeable. As said, the students' output does not justify this. Instead, VE and professional hacker ethic are evaluated similarly by them.

7 Discussion

The society as a whole should have an increasing interest, that in particular young people with a strong moral compass will choose a cybersecurity career. Citizens, companies, towns, regions and countries are increasingly dependent from digital processes and critical infrastructures. To secure and defend such infrastructures it is essential to understand how concrete attacks are performed. Thus, a security expert always needs to be educated also in offensive security means. With this powerful knowledge it is almost self-explanatory that such people have to be educated such that they are able to build up a sustainable strong moral compass. Such strong characters are essential for almost every level of a hierarchy within a team acting in the area of cybersecurity and should thus be carefully balanced with respect to the overall demand for loyalty within the group. This is obviously predominant for positions in authorities, military, but also 'common' cybersecurity companies, since the latter frequently equip the aforementioned with technology (See remark 2). A first, however indeed very small step towards this direction, is to educate these people in cybersecurity ethics and pinpoint to the dilemma of controversial ethical frameworks. However, much more activities are surely required here. Applying the various ethical frameworks results in different recommendations how to morally act with respect to concrete cybersecurity case study steps. This was an outcome of our small cybersecurity study with international students. We assume that this may sometimes result in ethical trade-offs of employee vs. employer. Consequently, a good balance of tolerating different ethical cybersecurity viewpoint versus loyalty within the cybersecurity team may be required here. Our assumption is that cybersecurity related companies and authorities tend more to be in line with community driven ethical frameworks like professional hacker ethic, human being decisions however over the long run may tend to be VE, or DE dominated. We also observe that the vulnerability and exploit supply-chain is never purely influenced by cybersecurity actors. Also, careless developers and users have a significant portion here. Moreover, one proposal we could make is that also cyber-experts who are in particular oriented towards a VE or DE driven ethical framework *have to* become part of a security team since we believe that a good mix is required to provide better results and acceptability for overall society. An open issue is how to ethical act as a cybersecurity

expert with respect to real-time challenges (e. g. trolley problem) where different acting options due to different ethical frameworks cannot be debated since one has to decide and act immediately. This observation encourages our believe that teams of people with mixed ethical background are in particular good when formulating the code-of-conduct of a team. However, the more it comes to real-time responsive decision making, the less time is available to find a consensus between different and maybe contradicting ethical frameworks of the group.

8 Conclusion

We discussed potential ethical dilemma security experts may face during their professional career. Although we could not provide the solution how to resolve such dilemma, we feel it is yet an important step forward to transparently discuss these issues. This is why we decided to publish an aggregated summation of the debates with young upcoming cybersecurity experts within our ethics seminar to share with a broader audience. We conclude with an enhanced and more mature statement: "*I would like to become an ethical dilemma aware offensive cybersecurity expert*".

9 Acknowledgments

We want to thank the anonymous reviewers, in particular our shepard Bernhard Hämmerli.

References

[Er21] Erden, Ç.: Give Me Everything: Analyzing leakage of information in the Android platform with respect to wireless communication and personally identifiably information, MA thesis, Hochschule Offenburg, Aug. 2021.

[Fo78] Foot, P.: Virtues and Vices and Other Essays in Moral Philosophy. Clarendon Press, Oxford, 1978.

[GI] GI-Fachgruppe Informatik und Ethik: Gewissensbits, Fallbeispiele zu Informatik und Ethik, URL: https://gewissensbits.gi.de/, visited on: 09/09/2021.

[Lo] Lockheed Martin Rotary and Mission Systems: Cyber Kill Chain, URL: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html, visited on: 09/15/2021.

[Ma18] Manjikian, M.: Cybersecurity ethics: an introduction. Routledge, 2018.

[Mv20] Macnish, K.; van der Ham, J.: Ethics in cybersecurity research and practice. Technology in Society 63/C, 2020.

Short Paper: Business Chat ist verwirrt. Es hat sich vor Verwirrung selbst verletzt!

Moritz Gruber[1], Christian Höfig[1], Matteo Große-Kampmann[1], Tobias Urban[2], Norbert Pohlmann[3]

Abstract: In dieser Arbeit wird eine ganzheitliche Bedrohung für Business-Chat-Anwendungen aufgezeigt und bewertet: *Chishing* – Phishing über Business-Chats. Die Bedrohung hat ihren Ursprung in den Anfängen der heutigen vernetzten Welt und das zugrunde liegende Problem wird in seiner einfachsten Form als *Spoofing* bezeichnet. In vier von sechs Business-Chat-Tools, die in dieser Arbeit analysiert werden, ist es möglich, Anzeigenamen, Profilbilder und weitere persönliche Informationen erfolgreich zu fälschen. Dies stellt eine Bedrohung für Unternehmen dar, da es Insider-Bedrohungen Vorschub leistet und unter Umständen auch externen Entitäten dazu einlädt, sich als interne Mitarbeiterin auszugeben.

Keywords: Phishing; Chishing; Social Engineering; Business Chats

1 Einleitung

Seit den Anfängen des Internets haben Cyberkriminelle die Gelegenheit ergriffen Internetnutzer zu täuschen und sich so zu bereichern. Eine Möglichkeit, Verwirrung zu stiften, ist das sogenannte *Spoofing*. Spoofing beschreibt im Allgemeinen den Akt der Täuschung, indem versucht wird, eine gefälschte Entität so darzustellen, als sei sie das Original. Heutzutage wird der Begriff *Spoofing* für Angriffstechniken verwendet, bei denen Authentifizierungs- oder Identifizierungsverfahren umgangen werden und somit die Vertrauenswürdigkeit von Systemen nicht mehr sichergestellt werden kann, beispielsweise bei GPS- oder Gesichtserkennungssystemen [BKH16; Ti11]. *Spoofing* ist besonders kritisch, da die meisten Unterhaltungen im Internet heutzutage per E-Mail oder Chat stattfinden.

Diese Arbeit diskutiert eine Bedrohung für verschiedene Business-Chat-Programme. Es wird gezeigt, wie eine interne oder externe Angreiferin in verschiedenen Business-Chat-Anwendungen Identitäten fälschen kann. Analog zu *Phishing* über SMS, welches als *Smishing*[4] definiert wird, führten die Autoren die Wortschöpfung *Chishing* ein. Dieser Be-

[1] AWARE7 GmbH, Munscheidstraße 14, 45886 Gelsenkirchen, Deutschland, vorname@aware7.de
[2] Institut für Internet-Sicherheit – if(is), Westfälische Hochschule und secunet Security Networks AG, urban@internet-sicherheit.de
[3] Institut für Internet-Sicherheit – if(is), Westfälische Hochschule, Neidenburgerstraße 43, 45877 Gelsenkirchen, Deutschland, pohlmann@internet-sicherheit.de
[4] https://www.bsi.bund.de/DE/Service-Navi/Presse/Alle-Meldungen-News/Meldungen/Smishing_SMS-Phishing_141021.html

doi:10.18420/sicherheit2022_13

griff wurde durch die Autoren erstmalig in einem nicht zitierfähigen Preprint veröffentlicht[5]. Dieses Preprint sorgte bereits für eine kontroverse Diskussion in einem Online-Forum[6]. Nichtsdestotrotz ist der Begriff Chishing korrekt gewählt, da er Phishing über einen neuen Delivery-Vektor beschreibt. Gerade der Delivery-Vektor spielt bei der konkreten Beschreibung einer Bedrohung eine essentielle Rolle. Ähnlich wie bei Injection-Schwachstellen wie *SQL-*, *NoSQL-* oder *GraphQL-Injections* dient der im Namen der Schwachstelle enthaltene Delivery-Vektor der Abgrenzung und Präzisierung eines Angriffs. Zusammengefasst leistet der Artikel die folgenden Beiträge:

- Diese Arbeit zeigt auf, wie eine interne oder externe Angreiferin unter bestimmten Bedingungen in verschiedenen Business-Chat-Programmen Identitäten fälschen und somit eine Bedrohung für die Informationssicherheit darstellen kann.
- Die Bedrohung wird anhand einer relevanten Teilmenge der verfügbaren Anwendungen in diesem Bereich evaluiert.

2 Hintergrund

Traditionell hat die E-Mail eine wesentliche Rolle in der digitalen Kommunikation auf der ganzen Welt eingenommen [Cr82]. Heutzutage gewinnen aber auch andere Kommunikationskanäle an Bedeutung. Diese Arbeit konzentriert sich auf den Übermittlungsvektor Chat und befasst sich mit möglichen Schwierigkeiten und Bedrohungen, die sich aus dem *Spoofing* bestimmter Aspekte in diesen Chats ergeben.

Social Engineering: Social Engineering spielt in der heutigen Bedrohungslage eine bedeutende Rolle. Besonders kritisch bei dieser Bedrohung ist die Tatsache, dass es immer schwieriger wird, digitale Identitäten zwischen „beruflichem“ und „privatem“ Leben zu trennen [Br16]. Beim Social Engineering geht es der Angreiferin darum, ein Opfer erfolgreich von etwas zu überzeugen. Wenn hinter einem Social-Engineering-Szenario eine vertrauenswürdige Identität oder eine gefälschte Identität steht, können die Ergebnisse für die Informationssicherheit kritisch sein.

Phishing: Phishing wurde erstmals 1996 im Zusammenhang mit dem Diebstahl von America Online (AOL)-Konten verwendet [Ol04]. Falsche und betrügerische Nachrichten, die dem Opfer Geld versprechen, gehen bis ins 19. Jahrhundert zurück [Ne]. In der Computersicherheit beschreibt *Phishing* den Vorgang, bei dem sich eine Angreiferin als vertrauenswürdige Instanz ausgibt, um Benutzer dazu zu bringen, persönliche Daten preiszugeben oder über bösartige Anhänge oder Links Malware zu verbreiten [Pa15]. Dies kann per E-Mail, Chat oder über jedes andere Kommunikationsmedium geschehen. Solange es mindestens zwei Kommunikationspartner gibt, besteht immer eine gewisse Wahrscheinlichkeit, dass eine der beiden Personen böswillig handelt. *Phishing* wird von Angreifern immer wieder angewendet,

[5] https://www.researchgate.net/publication/355916867_Business_Chat_is_Confused_It_Hurt_Itself_in_its_Confusion_-_Chishing

[6] https://web.archive.org/web/20220111181830/https://www.reddit.com/r/cybersecurity/comments/qn9rv2/chishing_an_emerging_threat_for_business_chat/

weil es oft einfacher ist, Menschen auszunutzen – vor allem im Vergleich zu einem Einbruch in ein gehärtetes technisches System [FV18].

Spoofing: Spoofing ist eine weit verbreitete Bedrohung im Internet und beschreibt eine Technik bei welcher eine Angreiferin sich als eine vertrauenswürdige Entität ausgibt. Wenn eine Angreiferin sich als rechtmäßige Benutzerin des Systems ausgeben kann, versagen die Schutzmechanismen des Systems, da das System die Angreiferin als legitime Benutzerin einschätzt. Es wurden technische Gegenmaßnahmen eingeführt, um Spoofing-Angriffe zu verhindern, welche ihre Grenzen und Nebenwirkungen besitzen [Sh21].

3 Bedrohungsmodell

Dieser Abschnitt beschreibt die Bedrohungsmodelle, die für den Angriff berücksichtigt werden. Der Spoofing-Angriff gilt für die folgenden Angriffstypen.

Interne Angreiferin: Interne Angriffe gelten als große Bedrohung, da diese schwer zu erkennen sind [Pa21]. So kann eine interne Angreiferin, beispielsweise eine verärgerte Mitarbeiterin, das gesamte Unternehmen oder zumindest jeden, der am Unternehmens-Chat teilnimmt, angreifen. In dem Szenario dieser Arbeit kann die interne Angreiferin sich als beliebige andere Mitarbeitende ausgeben und in ihrem Namen Aktionen durchführen und Nachrichten verschicken.

Externe Angreiferin: Eine externe Angreiferin hingegen muss zunächst eine Einladung erhalten, damit ein Angriff erfolgreich sein kann. Die meisten Chat-Plattformen bieten die Möglichkeit, externe Personen in ihr Chat-System einzuladen. Dies stellt eine potenzielle Bedrohung dar, denn wenn eine externe Person ihren Namen im Chat ändern kann, eröffnet dies einen Phishing-Vektor. Die Auswirkungen des Phishing-Angriffs ändern sich jedoch, da die Angreiferin in diesem Experimenten nur Kontakte kontaktieren kann, die bestimmte Kommunikationsräume mit der Angreiferin teilen, und nicht die gesamte Belegschaft des Unternehmens.

4 Methodik

Der Schwerpunkt dieser Arbeit liegt auf der Analyse, welche geschäftlichen Chats es einer potenziellen internen und externen Angreiferin ermöglichen, Phishing-Nachrichten durch gezieltes Spoofing zu versenden. In diesem Kapitel wird zunächst erläutert, wie die untersuchten Business-Chats ausgewählt wurden. Anschließend wird definiert, welche Analyseschritte bei der Untersuchung der Anwendungen durchgeführt wurden.

4.1 Geprüfte Business-Chats

Insgesamt wurden in dieser Arbeit sechs verschiedene Business-Chat-Anwendungen untersucht. Hierbei handelt es sich um *Microsoft Teams*, *Element.io*, *Google Chat*, *Mattermost*,

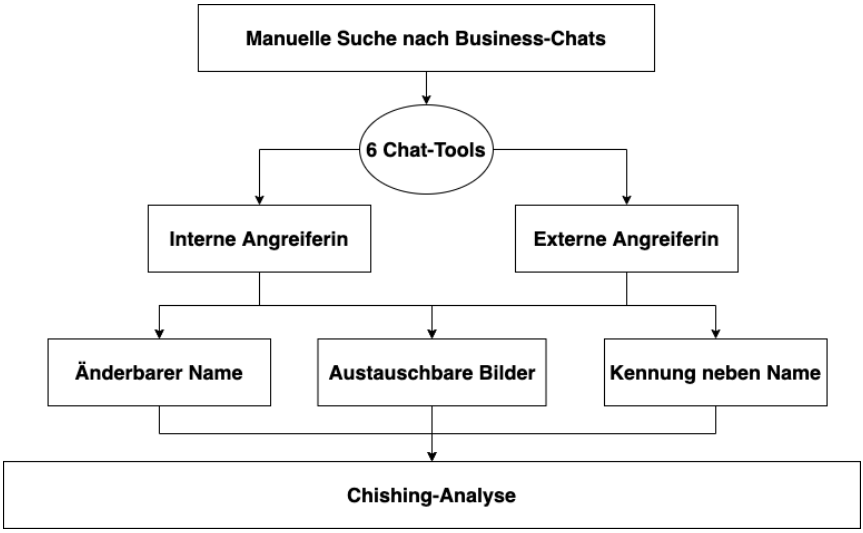


Abb. 1: Analyse-Pipeline zur Bewertung der Business-Chats

Slack und *WebEx Teams*. Mit Microsoft Teams und Slack wurden die Marktführer im Bereich Business Chat ausgewählt und um weitere vier Anwendungen ergänzt [Ke].

4.2 Analyse-Pipeline

Alle getesteten Anwendungen wurden wie in Abbildung 1 dargestellt untersucht. Zunächst wurde eine Benutzerin erstellt, die dem Angreifermodell entspricht. Dazu wurden für jede Business-Chat-Anwendung eine Gastbenutzerin und eine neue interne Benutzerin angelegt. Die folgenden Tests wurden anschließend mit diesen beiden Konten durchgeführt.

Zunächst wurde für jedes der beiden Konten geprüft, ob eine Benutzerin oder Gastbenutzerin den Namen ändern kann. Weiterhin wurde evaluiert, ob das Profilbild der Nutzerin für alle Anwendungen änderbar ist. Ziel dieser beiden Änderungen war es, das Profil einer anderen, bereits existierenden Benutzerin zu imitieren. Wenn der Name und das Profilbild einer Benutzerin geändert werden konnte, galt die Anwendung als fälschbar.

Es wurde außerdem die Anzahl der Klicks ausgewertet, die zur Identifizierung der Benutzerin erforderlich sind. Dadurch kann festgestellt werden, wie einfach es für das Opfer einer Chat-Phishing-Kampagne ist, die wahre Identität zu erkennen. Im Falle der externen Angreiferin wurde in diesem Schritt auch bewertet, wie eine Gastnutzerin identifiziert wurde.

Plattform	Externe Angreiferin möglich?	Interne Angreiferin möglich?
Microsoft Teams	Nein	Nein
Element.io	Nein	Nein
Google Chat	Ja	Ja
Mattermost	Ja	Ja
Slack	Ja	Ja
WebEx Teams	-	Ja

Tab. 1: Analysierte Plattformen und Ergebnisse

Plattform	Profilbild änderbar?	Benutzername änderbar?	Gast erkennbar?
Microsoft Teams	Ja	Nein	Ja
Element.io	Ja	Ja	Ja
Google Chat	Ja	Ja	Ja
Mattermost	Ja	Ja	Ja
Slack	Ja	Ja	Nein
WebEx Teams	Ja	Ja	Nein

Tab. 2: Übersicht, für welche Anwendungen Namens- und Profilbildänderungen möglich sind und ob eine Gastbenutzerin hervorgehoben wird

5 Ergebnisse

In Tabelle 1 sind die die Ergebnisse unserer Analyse für die sechs verschiedene Business-Chat-Tools dargestellt, bezüglich der Bedrohungsmodelle, die in Kapitel 3 definiert wurden. Insgesamt wurde festgestellt, dass es in allen Anwendungen möglich ist, das Profilbild ohne Überprüfung durch die Arbeitgeberin zu ändern und in den meisten der untersuchten Anwendungen den Anzeigenamen zu bearbeiten.

Wie in Tabelle 2 gezeigt, ist es in allen Business-Chats möglich, das Profilbild zu ändern. Der Name der eigenen Profils konnte in fünf von sechs der untersuchten Chat-Anwendungen geändert werden, was zu potenziellen *Chishing*-Angriffsvektoren führt. Bei zwei der Anwendungen ist es unter bestimmten Bedingungen nicht direkt möglich, eine externe von einer internen Angreiferin zu unterscheiden.

Abbildung 2 zeigt, dass Chat-Anwendungen identifiziert wurden, die der Benutzerin direkt die Identität (Benutzer-ID oder E-Mail-Adresse) der Angreiferin anzeigen. Bei Messengern wie *Slack* oder *WebEx* Teams muss das Opfer erst auf das Profil der Angreiferin klicken

und dann erneut klicken, um das vollständige Profil zu sehen (insgesamt zwei Klicks). In einer stressigen Situation stellt dies bereits ein erhebliches Hindernis bei der Erkennung dar. Geschäftliche Chat-Anwendungen wie *Google Chat* oder *Mattermost* zeigen der Nutzerin die Identität (E-Mail) der Angreiferin nach einem Klick auf das Profil (ein Klick) an. In

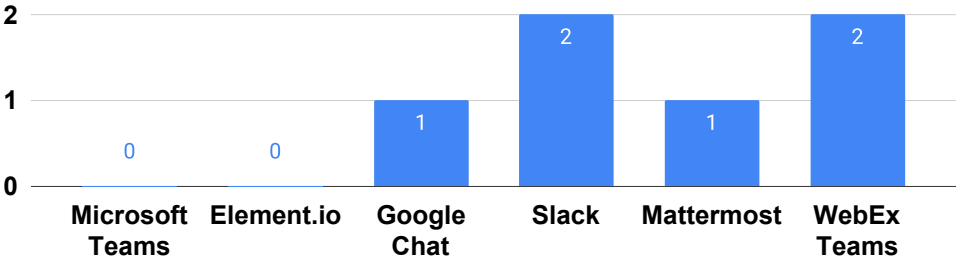


Abb. 2: Überblick über die Anzahl der Klicks, die erforderlich sind, um die Identität der Angreiferin aufzudecken

den folgenden Abschnitten werden die verschiedenen Auswirkungen und Bedrohungen beschrieben, welche die Ergebnisse für die jeweiligen Plattformen haben.

5.1 Microsoft Teams

Weder internes noch externes Spoofing ist bei Microsoft Teams möglich, wie in Tabelle 1 zu sehen ist. Bei MS Teams gehört eine Benutzerin immer zu einer Organisation, wobei Benutzerin und zugehörige Organisation von verschiedenen Administratoren verwaltet werden. Daher ist es nicht möglich, als normale Angestellte das eigene Profil zu bearbeiten. Somit ist das Bedrohungsmodell des Spoofing als interne und externe Angreiferin für Microsoft Teams hinfällig.

5.2 Element.io

Bei der Verwendung von *Element.io* als Business-Chat ist es nicht möglich, ein Profil so zu ändern, dass eine Angreiferin eine Benutzerin erfolgreich fälschen kann. Ändert eine Benutzerin ihren Namen, werden alle Mitglieder eines Kommunikationskanals benachrichtigt. Hinter dem Anzeigenamen wird außerdem immer eine eindeutige Kennzeichnung angezeigt, die für jede Benutzerin festgelegt wird und nicht geändert werden kann.

5.3 Google Chat

In Google Chat ist es relativ einfach, den angezeigten Namen zu ändern[7]. Dies ist sowohl für interne als auch für externe Angreifer möglich. Da Google Chat in Google Workplace eingebettet ist, führt die Umbenennung durch eine interne Angreiferin zu weiterer Verwirrung in anderen Google-Tools z. B. Google Kalender (Abbildung 4).

[7] `https://support.google.com/mail/answer/8158?hl=de`

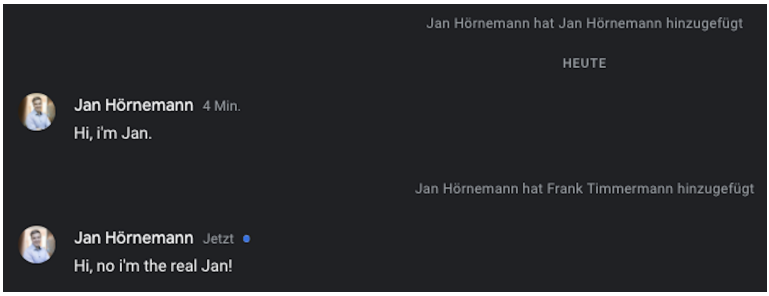


Abb. 3: Chishing-Angriff in der Google Chat-Webanwendung

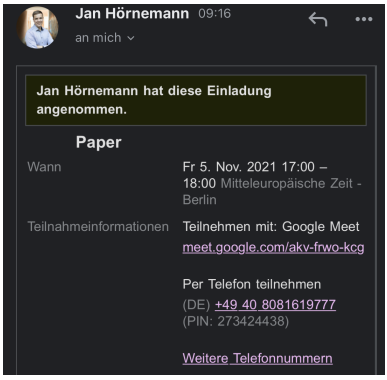


Abb. 4: Das gefälschte Konto nimmt Einladungen als jemand anderes an

Interne Angreiferin: Abbildung 3 zeigt, wie ein Chishing-Angriff in Google Chat aussehen könnte. Welches Profil des Benutzers Jan „echt“ ist, lässt sich erst erkennen, wenn sich die Nutzerin das Profil genauer ansieht und somit die unveränderbare E-Mail-Adresse angezeigt bekommt. Dies ermöglicht es internen Angreifern, eine legitime Benutzerin zu täuschen, was eine Bedrohung für Unternehmen darstellt. Das gleiche Problem besteht auch in der mobilen Version von Google Chat.

Auch beim Erstellen eines Meetings gibt es Probleme: Die E-Mail-Adresse der Benutzerin stimmt zwar nicht mit dem Namen überein, aber das wird leicht übersehen, da das Profilbild und der Anzeigename gut sichtbar sind.

Externe Angreiferin: Auch eine externe Angreiferin hat bei Google Chat mit einem Chishing-Angriff Erfolg. Ein oranger, visueller Marker in der oberen Ecke signalisiert, dass der Kanal mit einer Angreiferin geteilt wird.

5.4 Slack

Interne und externe Angreiferinnen können in Slack bestimmte Verhaltensweisen ausnutzen, um erfolgreich *Chishing*-Angriffe durchzuführen

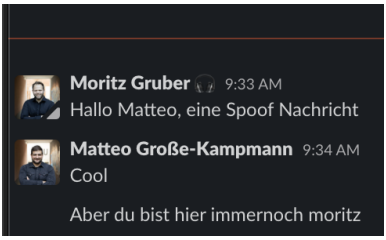


Abb. 5: Chishing in Slack - Bei dem gespooften Account handelt es sich um den Account „Moriz Gruber“

Interne Angreiferin: Als interne Angreiferin ist ein *Chishing*-Angriff umsetzbar. Wie in Abbildung 5 zu sehen ist, bemerkt die andere Person nicht, wer hinter dem Profil steckt. Selbst in der mobilen Version von Slack ist nicht erkennbar, dass es sich um einen möglichen Spoofing-Angriff handeln könnte. Der Benutzer „Matteo“ ist der legitime Benutzer und „Moritz“ ist eine gefälschte Identität von Mitarbeiter „Frank“. Dies ist jedoch für „Matteo“ in der Chat-Oberfläche nicht zu erkennen, sondern wird erst nach zwei Klicks auf den Profilbereich des gefälschten Moritz erkennbar.

Externe Angreiferin: Das externe Bedrohungsmodell kann auch in Slack umgesetzt werden. Selbst in einem Chat, in dem die echte Person und das gefälschte Profil miteinander kommunizieren, ist auf den ersten Blick kein Unterschied zu erkennen. Ein kleines graues Dreieck am unteren Rand des Profilbildes markiert eine Gast-Benutzerin.

5.5 Mattermost

Mattermost wurde als Open-Source-Alternative zu Slack entwickelt. Der Dienst ermöglicht Chats mit Einzelpersonen sowie in Gruppen. In Mattermost ist es möglich, die kompletten Benutzerdaten zu ändern und sich als eine andere Person auszugeben [8].

Interne Angreiferin: Eine interne Angreiferin kann Chishing in Mattermost durchführen. Innerhalb eines Chats ist nicht zu erkennen, dass die Gesprächspartnerin ein gefälschtes Profil sein könnte.

Externe Angreiferin: Eine externe Angreiferin kann in Mattermost auch erfolgreich ihr Profil fälschen, wenn diese zu einer geschlossenen Unterhaltung eingeladen wird. Eine normale Benutzerin kann sehen, dass die externe Angreiferin ein Gast ist, aber auch hier gibt es keine andere Möglichkeit zu erkennen, dass es sich nicht um das echte Profil handelt.

5.6 WebEx Teams

Wie bei anderen Chat-Tools, können auch bei WebEx Teams neue Chat-Bereiche innerhalb des Unternehmens erstellt oder privat mit anderen Mitarbeitern gechattet werden. Da WebEx Teams nicht die Funktion bietet, Gäste in Chaträume einzuladen, sondern nur 1:1-Gespräche zulässt, trifft das Bedrohungsmodell einer externen Angreiferin hier nicht zu.

[8] `https://docs.mattermost.com/messaging/manage-profile-settings.html?highlight=username`

Interne Angreiferin: Als interne Angreiferin war es möglich, das eigene Profil so zu bearbeiten, dass es wie ein gültiges Profil einer anderen Person aussieht [9]. In diesem Beispiel ist es fast unmöglich zu erkennen, wer das echte Profil hat und wer einen Spoofing-Angriff durchführt. Selbst in einem privaten Chat ist nicht erkennbar, ob es sich um ein gefälschtes Profil handelt. Es wird nur die Endung der E-Mail-Adresse angegeben, die aber nicht aussagekräftig genug ist, um eine Person genau zu identifizieren.

6 Fazit

Insgesamt kann festgestellt werden, dass *Chishing* in der Mehrheit der untersuchten Business-Chats nicht behandelt wird und unsere Forschungsarbeit der erste Versuch ist, diese aufkommende Bedrohung zu untersuchen. In fünf der sechs untersuchten Business-Chat-Anwendungen war es möglich, den Namen der Benutzerin nach Belieben für eine interne oder eingeladene, externe Angreiferin zu ändern. Dies stellt eine große Bedrohung für die sichere Nutzung von Chats im geschäftlichen Kontext dar. Insbesondere Chats dienen dem schnellen und permanenten Austausch von Informationen, was eine Angreiferin mit gefälschten Benutzernamen ausnutzen kann. Außerdem ist das Vertrauen in Chat-Nachrichten in der Regel höher als in E-Mails [BAB+21].

Um dieses Problem zu beheben, können Business-Chat-Anwendungen verschiedene Maßnahmen ergreifen. So wäre es zum Beispiel möglich, die Identität der Nutzerin neben dem Nutzernamen anzuzeigen, wie es Element.io implementiert hat. Eine andere Möglichkeit wäre, ähnlich wie bei Microsoft Teams, die Benutzer über die Administratorin daran zu hindern, deren Namen serverseitig ändern zu können. Insgesamt ist festzuhalten, dass neben den klassischen Phishing-Methoden *Chishing* ein weiteres großes Bedrohungspotenzial ermöglicht. Gerade in der modernen, Home-Office-getriebenen Büroumgebung müssen Aufklärungs- und Sensibilisierungskampagnen zum Schutz vor Chishing-Angriffen durchgeführt werden. Business-Chat-Plattformen müssen Maßnahmen ergreifen, um ihre Benutzer vor dieser Art von Angriffen zu schützen und Unternehmen sollten diese Bedrohung in ihrem Risikomanagementprozess diskutieren.

Literatur

[BAB+21] Blancaflor, E. B.; Alfonso, A. B.; Banganay, K. N. U. et al.: Let's Go Phishing: A Phishing Awareness Campaign Using Smishing, Email Phishing, and Social Media Phishing Tools. Proceedings of the International Conference on Industrial Engineering and Operations Management/, 2021.

[BKH16] Boulkenafet, Z.; Komulainen, J.; Hadid, A.: Face Spoofing Detection Using Colour Texture Analysis. IEEE Transactions on Information Forensics and Security 11/8, S. 1818–1830, 2016.

[Br16] Broadbent, S.: Intimacy at Work: How Digital Media Bring Private Life to the Workplace. Routledge, Walnut Creek, CA, 2016.

[9] https://help.webex.com/en-US/article/nmig1kcb/Edit-your-Webex-Meetings-profile

[Cr82] Crocker, D.: RFC#822: Standard for the format of ARPA Internet text messages, 1982.

[FV18] Ferreira, A.; Vieira-Marques, P.: Phishing Through Time: A Ten Year Story based on Abstracts. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy. ICISSP '18, INSTICC, SciTePress, Setúbal, Portugal, S. 225–232, 2018, ISBN: 978-989-758-282-0.

[Ke] Kent, D.: Slack vs Microsoft Teams 2021 – The Enterprise Messaging Wars Continue, Digital, https://web.archive.org/web/20220109224334/https://dispatch.m.io/enterprise-messaging-wars-slack-microsoft-teams/, Datum des letzten Abrufs: 8. März 2022.

[Ne] New York Times: An Old Swindle Revived, Print, https://web.archive.org/web/20201112040016/https://www.nytimes.com/1898/03/20/archives/an-old-swindle-revived-the-spanish-prisoner-and-buried-treasure.html, Datum des letzten Abrufs: 8. März 2022.

[Ol04] Ollmann, G.: The Phishing Guide–Understanding & Preventing Phishing Attacks. NGS Software Insight Security Research/, 2004.

[Pa15] Parsons, K.; McCormac, A.; Pattinson, M.; Butavicius, M.; Jerram, C.: The Design of Phishing Studies: The design of phishing studies: Challenges for researchers. Computers & Security 52/C, S. 194–206, 2015, ISSN: 0167-4048.

[Pa21] Padayachee, K.: A Theoretical Underpinning for Examining Insider Attacks Leveraging the Fraud Pentagon. In: International Symposium on Human Aspects of Information Security and Assurance. Springer, S. 179–188, 2021.

[Sh21] Shen, K.; Wang, C.; Guo, M.; Zheng, X.; Lu, C.; Liu, B.; Zhao, Y.; Hao, S.; Duan, H.; Pan, Q. et al.: Weak Links in Authentication Chains: A Large-scale Analysis of Email Sender Spoofing Attacks. In: 30th USENIX Security Symposium (USENIX Security 21). 2021.

[Ti11] Tippenhauer, N. O.; Pöpper, C.; Rasmussen, K. B.; Capkun, S.: On the Requirements for Successful GPS Spoofing Attacks. In: Proceedings of the 18th ACM conference on Computer and communications security. S. 75–86, 2011.

Practitioners Track

Cyber-Defense "Gemessen, Bewertet und Ausgerichtet" - Ein Praxisbericht

Fabian Lochmann[1], Sebastian Schmerl[2]

Abstract: Bei dem hier vorgestellten Cyber-Defense-Maturity-Assessment handelt es sich um eine Methodik zur Erfassung und Bestimmung der Cyber- MITRE ATT&CK™ Framework. Es wird gezeigt, wie sich effizient und praxisorientiert die aktuelle Bedrohungslage für ein Unternehmen inklusive des Branchen- und Geo-Fokus von Abwehr-Fähigkeiten eines Unternehmens. Die Methodik basiert dabei auf dem freien und weltweit anerkannten Angreifer-Gruppen mit Hilfe präventiver sowie auf Erkennung gestützte Security-Controls erfasst und das erforderliche Schutzniveau bestimmen lässt.

Keywords: Cyber-Defense, MITRE ATT&CK™, Bewertung, Angreifergruppen, Reifegradbestimmung

1 Einleitung

Ob Script-Kiddies, finanziell motivierte Cyber-Kriminelle oder politisch motivierte Akteure: Unternehmen werden heutzutage von sehr unterschiedlichen Klassen von Angreifern attackiert – und jede Angreiferklasse arbeitet typischerweise mit ihren spezifischen Angriffstechniken.

Um den immer ausgefeilteren Angriffen zu begegnen, müssen Unternehmen diese Angriffstechniken kennen und sich auf diese vorbereiten. Nur so können Gefahren abgewehrt, Vorfälle erkannt und Reaktionen zielgerichtet ausgerichtet werden. In der Praxis können dies aber nur die wenigsten Unternehmen. Dazu kommt, dass es für fast alle Organisationen schwierig ist, ihr erforderliches Schutzniveau zu bestimmen und im Anschluss auch zu halten, denn es gibt eine Vielzahl von Methoden der Risikobeurteilung wie Kreativitätstechniken, Szenario-Analysen, Indikatoren-Analysen, Gefährdungsanalysen oder statistische Analysen (vgl. [Kö17]). Aber alle diese Methoden haben die gleichen Nachteile: Sie sind oft papierbasiert, erfordern eine zeit- und ressourcenaufwändige Erfassung des Ist-Zustandes und sind oft sehr weit entfernt von der Technik und damit von den realen Anforderungen in großen Unternehmen. Mit der vorgestellten Methodik lassen sich erfahrungsgemäß mittelständische Unternehmen mit ca. 1000 IT-User in etwa 2-3 Arbeitstagen (je 8h) erfassen und bewerten.

Der hier beschriebene Cyber-Defense-Maturity-Ansatz basiert auf MITRE ATT&CK™ (vgl. [MI]) und ist eine einfach umsetzbare Methodik, um den Reifegrad der Cyber-Defense

[1] Arctic Wolf Networks Germany Fabian.Lochmann@arcticwolf.com
[2] Arctic Wolf Networks Germany Sebastian.Schmerl@arcticwolf.com

doi:10.18420/sicherheit2022_14

zu erfassen und das weitere Vorgehen mit Hilfe des MITRE ATT&CK™ Frameworks abzubilden. Dabei werden die typischen Dimensionen „Menschen, Prozesse und Technik“ mit bewertet, um ein aussagekräftiges Bild über die gesamte Abwehrfähigkeit des Unternehmens zu erstellen. Die Erfassung des Ist-Zustandes wird mit Hilfe von alltäglichen Angriffen mit unterschiedlichem Skill-Niveau von Angreifern (von leicht zu komplex und anspruchsvoll) und einer typischen Abstraktion einer IT-Konfiguration bezogen auf die Unternehmensgröße und Eigenschaft erstellt. Das Vorgehen stellt dabei keinen Anspruch auf Vollständigkeit, sondern definiert einen zuvor festgelegten Umfang, um zu einer Aussage innerhalb weniger Wochen zu kommen.

2 Vorgehensweise

Mit einem **Cyber-Defense-Maturity-Assessment** lässt sich Klarheit schaffen. Im Rahmen des Assessments werden folgende Fragen beantwortet: Welche Angreifer haben Interesse an meinem Unternehmen? Welche Angriffstechniken verwenden diese Angreifer? Wie gut verhindern bzw. erkennen meine Schutzmaßnahmen diese Techniken? Welche Angriffstechniken erkennen meine Schutzmaßnahmen nicht? Welche meiner Schutzmaßnahmen sind inzwischen obsolet, da redundant oder veraltet? Wie lassen sich Lücken sinnvoll schließen?

Dazu haben wir eine Methodik entwickelt, um die Widerstandsfähigkeit von IT-Infrastrukturen zu bewerten. Basis hierfür ist die stets aktuelle MITRE ATT&CK™-Wissensdatenbank, die alle bekannten Angriffstechniken, Angriffsvektoren und typischen Angreiferklassen anhand der Phasen einer Cyber-Attacke zusammenführt.

Mit einem Cyber-Defense-Maturity-Assessment bewerten wir anhand von vielen hunderten Angriffstechniken des MITRE-Katalogs individuell für jedes Unternehmen, wie wirksam die bestehenden Schutzmaßnahmen gegen diese sind. Konkret betrachten wir Maßnahmen für Prävention & Detektion sowie die Response-Fähigkeiten der Unternehmen gegenüber den unterschiedlich motivierten und technisch versierten Angreiferklassen.

3 Phase 1 – Erfassung des Schutzbedarfs

In der ersten Phase werden auf Grundlage des Schutzbedarfs der unternehmensspezifischen Bedrohungslage sowie Branchen- und Geo-Fokus von Angreifergruppen in unterschiedlichen Fähigkeitskategorien erfasst. Die Festlegung des Schutzbedarfes (von typischen alltäglichen Angriffen bis zu Akteuren aus den Regierungskreisen) erfolgt hierbei auf einer hohen Abstraktionsebene durch Entscheidungsträger der Unternehmensführung. Diese Entscheidung kann ggf. durch evtl. vorhandene IT-Security-Experten unterstützt werden. In letzter Instanz die Unternehmensführung für Cyber-Risiken verantwortlich. Technische Details werden in dieser Phase vorerst nicht benötigt.

Aus den Ergebnissen der ersten Phase entsteht eine priorisierte Liste der relevanten Angreifer-Methoden. Diese Methoden bestehend aus 188 MITRE ATT&CK™ Techniken und 379

Sub-Techniken und repräsentieren die zu bewertenden Bedrohungen und Angriffsvektoren, die bei Angreifern typischerweise zum Einsatz kommen.

4 Phase 2 – Erfassung des derzeitigen Schutzes und der Grad der Abdeckung

In der zweiten Phase erfolgt die eigentliche Bewertung der identifizierten Bedrohungen und der zugehörigen Angriffstechniken. Hierfür gehen wir in drei Schritten vor:

1. Erfassung des Ist-Zustandes hinsichtlich präventiver sowie für jede identifizierte Schutzmaßnahmen pro ATT&CK^{TM}-Angreifertechnik.
2. Herleitung des tatsächlichen Schutzes gegen die ATT&CK^{TM}-Angriffstechniken durch Kombination von Schutz-Effektivität und Abdeckung der einzelnen Angriffstechniken.
3. Feststellen des Abdeckungsgrades der Schutzmaßnahmen je Angriffstechnik in der bestehenden IT-Landschaft (Welche Infrastrukturteile werden durch die zugehörigen Schutzmaßnahmen abgedeckt und zu welchem Prozentsatz?).

Die Abschnitt 4 zeigt die Auswertung exemplarisch für die MITRE ATT&CK^{TM}-Angriffstechnik T1059.001. Es resultiert eine Erkennungsrate von 56,2% für Angriffstechnik T1059.001 (basierend aus 75% Schutzeffektivität und 75% Abdeckung der Schutzmaßnahmen in der IT-Landschaft) und 12,5% bzgl. Detektion. Die Schutzeffektivität von vorhandenen Security-Controls wird hierbei aus der typischen öffentlich dokumentierten Detektions- und Präventionsrate (z.B. MITRE ATT&CK^{TM} Evaluations) der Control-Kategorie (AV, Proxy, IPS, . . .) bzgl. Angriffstechniken abgeleitet.

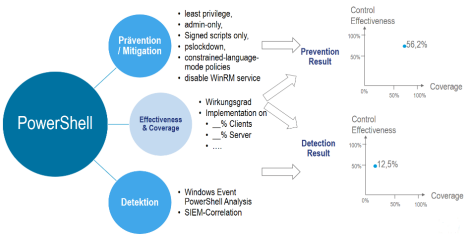


Abb. 1: Auswertung am Beispiel der Angriffstechnik: PowerShell (T1059.001)

Diese Phase 2 kann bei Bedarf durch eine automatisierte Erfassung mittels “Automatic Red-Team-Test” unterstützt werden.

5 Phase 3 – Ergebnispräsentation & Ableitung von Verbesserungen

In der dritten Phase werden die Ergebnisse aus der Korrelation der Erkenntnisse aus Phase eins und zwei zu einem Schaubild über die Attacker-Kill-Chain (vgl. [Lo20]) vereint, welches das Schutzniveau (grün) und Schutzdefizite (rot) pro Kill-Chain-Phase darstellt.

Im Weiteren werden für die IT-Infrastruktur des Unternehmens Schutzmaßnahmen priorisiert, welche die Schutzdefizite bestmöglich abdecken. Hierfür wird eine Zuordnung von typischen Schutzmaßnahmen auf Abdeckung von ATT&CK™ Angriffstechniken verwendet. Dadurch ist eine wirtschaftliche und optimale Erhöhung des Sicherheitslevels möglich. Das Cyber-Defense-Maturity-Assessment erlaubt nachvollziehbare, priorisierte Entscheidungen für oder gegen Sicherheitsmaßnahmen und veranschaulicht deren Einfluss auf das Sicherheitslevel bzgl. der verhindert- oder detektierbaren Angriffstechniken und Angreiferklassen.

Literatur

[Kö17] Königs, H.-P.: Methoden und Werkzeuge für das Informations-Risikomanagement, 2017.

[Lo20] Lockheed Martin: The Cyber kill chain, `https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html`, 2020.

[MI] MITRE Corporation: MITRE Att&ck®, `https://attack.MITRE.org/`.

Studie über das Gefahrenpotential und Gegenmaßnahmen zu Angriffen auf das DNS Protokoll durch IP-Fragmentierung

Carsten Strotmann[1], Patrick Ben Koetter[2], Roland van Rijswijk-Deij[3], Markus de Brün[4], Anders Kölligan[5]

Abstract: Caches von DNS-Resolvern können mit fragmentierten IP-Paketen kompromittiert werden. Die BSI-Studie „IP Fragmentation and Measures against DNS Cache Poisoning (Frag-DNS)“ ist der Frage nachgegangen ob und wie häufig IP-Fragmentierung auf „natürliche Weise“ im Internet vorkommt. Sie hat mögliche Gegenmaßnahmen getestet und empfiehlt einige, die mit wenig Veränderung Abhilfe schaffen. Die Studie wird unter `https://bsi.bund.de/dok/frag-dns` publiziert.

Keywords: IP-Fragmentierung, DNS Cache Poisoning, DNS-Resolver, DNSSEC

1 Inhaltsbeschreibung

Das Domain Name System (DNS) ist eines der wichtigsten Protokolle des Internets. Nahezu jede Aktion im Internet beginnt mit einer DNS-Anfrage. Kernaufgabe des DNS ist es, Domain-Namen an IP-Adressen zu binden. Inzwischen wird es jedoch auch verwendet, um Konfigurations- und Authentifizierungsanweisungen sowie Policies auszugeben.

Aufgrund seiner zentralen und kritischen Bedeutung für das Internet ist DNS zu einem lohnenswerten Ziel für Angreifer geworden. Zahlreiche auf das DNS gerichtete Angriffsmethoden existieren und für die meisten dieser Angriffe sind Gegenmaßnahmen bekannt und wurden umgesetzt. Eine relativ neue Angriffsmethode, die durch Shulman et al [6] vorgestellt wurde, nutzt die IP-Fragmentierung zur Umgehung einiger jener Sicherheitsfunktionen, die derzeit in die DNS-Software eingebaut sind.

Frühere Studien[7] konnten zeigen, dass es möglich ist, fragmentierte DNS-Antworten zu verwenden, um gefälschte oder manipulierte Daten in den Cache eines DNS-Resolvers einzuschleusen. Dieser Prozess wird als „Cache Poisoning“ bezeichnet. Im Rahmen der, von sys4 und NLnet Labs, durchgeführten BSI-Studie „IP Fragmentation and Measures

[1] sys4 AG cs@sys4.de
[2] sys4 AG p@sys4.de
[3] NLnet Labs
[4] Bundesamt für Sicherheit in der Informationstechnik (BSI)
[5] Bundesamt für Sicherheit in der Informationstechnik (BSI)
[6] Fragmentation Considered Poisonous, Amir Herzberg, Haya Shulman, `https://arxiv.org/abs/1205.4011`
[7] IP fragmentation attack on DNS, Tomas Hlavacek, RIPE 67, 16.10.2013, `https://ripe67.ripe.net/presentations/240-ipfragattack.pdf`

doi:10.18420/sicherheit2022_15

against DNS Cache Poisoning (Frag-DNS)“ wurde untersucht, ob und wenn wie häufig die notwendigen Voraussetzungen für Cache Poisoning im Internet tatsächlich gegeben sind. Somit wurde festgestellt, ob dieser Angriffsvektor überhaupt eine reale und relevante Bedrohung darstellt. Zudem wurde untersucht, wie effektiv etwaige Gegenmaßnahmen sind und welche Nebenwirkungen diese gegebenenfalls auf den DNS Betrieb hätten.

Die in diesem Beitrag vorgestellten Forschungsarbeiten untersuchten die Voraussetzungen aus zwei Gesichtspunkten:

- Aus der Perspektive des autoritativen DNS-Servers: In der Studie wurden Millionen von autoritativen DNS-Servern im Internet getestet, um festzustellen, ob diese Server relativ große DNS-Antwort-Pakete senden, welche tendenziell fragmentiert werden. Die Domänen, von denen fragmentierte Antworten eingingen, wurden mit der Tranco Liste[8] der beliebtesten 1 Million Domänen im Internet abgeglichen. Diese Messung ergab, dass es im Internet in seltenen Fällen natürlich auftretende Fragmentierung von DNS-Verkehr gibt. Obwohl sie selten auftreten, sind diese Fälle dennoch relevant, denn sie betreffen auch Domains sehr populärer Internet-Dienste.
- Aus der Sicht des DNS-Resolvers: Der DNS-Verkehr eines großen Internet Service Providers (ISP) in Deutschland wurde über einen Zeitraum von 24 Stunden beobachtet, um festzustellen, wie viele fragmentierte DNS-Antworten im realen Internetverkehr auftreten. Diese Messung bestätigt, dass DNS-Fragmentierung tatsächlich in produktiven Umgebungen vorkommt.

Da Fragmentierungs-Angriffe meist auf DNS-Verkehr abzielen, der über das zustandslose UDP-Protokoll gesendet wird, besteht eine der vorgeschlagene Gegenmaßnahmen darin, die DNS-Auflösung von UDP auf TCP umzustellen. Zur Bewertung des Potenzials dieser Gegenmaßnahme untersuchte die Studie die Anzahl autoritativer DNS-Servern im Internet, welche DNS über TCP unterstützen. Während DNS über TCP bereits seit mehr als 10 Jahren von den Internet Protocol Standards vorgeschrieben ist (siehe RFC 5966 und RFC 7766), zeigen die Messungen, dass eine beträchtliche Anzahl von DNS-Servern immer noch keine TCP Verbindungen für DNS anbietet.

Als Nebenprodukt haben die oben erwähnten Messungen ergeben, dass eine beträchtliche Anzahl autoritativer DNS-Server im Internet auf vergleichsweise alten Linux-Betriebssystemen läuft. “Long Term Support” Linux-Systeme werden zwar offiziell von ihren Herstellern gewartet und so werden Sicherheits-Patches angeboten, aber diese Updates ändern nicht notwendigerweise Standardeinstellungen in den Linux-Kerneln. Diese Standardeinstellungen könnten einige erfolgreich DNS-Fragmentierungs-Angriffe verhindern. Die Analyse ergab somit, dass der Betrieb von “Enterprise”-Linux-Systemen das Risiko von Sicherheitsproblemen erhöhen kann, selbst wenn diese Systeme vollständig gepatcht werden.

[8] https://tranco-list.eu/

Da DNS-Fragmentierungs-Angriffe eine reale Bedrohung im Internet darstellen, wurden in der Studie weitere mögliche Abhilfemaßnahmen geprüft. Da diese Maßnahmen negative Auswirkungen auf den Betrieb des DNS oder die Leistung der DNS- Namensauflösung haben könnten, wurde eine mit der DNS-Infrastruktur des Internet vergleichbare Laborumgebung aufgebaut und die Gegenmaßnahmen in dieser Umgebung getestet.

In der Studie wurden die folgenden Gegenmaßnahmen getestet und bewertet:

- **Der Betrieb eines reinen DNS-over-TCP Resolver-Dienstes:** 40-44% Performance Verlust gegenüber DNS-over-UDP
- **DNS über TLSv1.3 zwischen DNS Resolver und autoritativen DNS Servern:** Vergleichbarer Performance-Verlust wie bei DNS-over-TCP. Die TLS Verschlüsselung selbst erzeugt nur einen geringen Performance-Verlust (ca. 4%)
- **"opportunistisches TCP" (TCP bevorzugen, Rückfall auf UDP wenn notwendig):** Bei Benutzung von TCP der Performance-Verlust wie bei DNS-over-TCP, jedoch über 70% Performance-Verlust (brutto) bei Servern, welche kein DNS-over-TCP unterstützen
- **Vermeidung von DNS Fragmentierung durch Verringerung der erlaubten Antwortgröße durch EDNS:** Bei Verringerung der möglichen DNS-Antwort-Grösse auf 1232 Byte (Minimum IPv6 MTU von 1280 Byte abzüglich IP-Header und UDP-Header) geringe Performance Verluste (~ 5%) bis leichte Performance-Gewinne (3.2%) je nach DNS-Resolver-Software
- **Verwerfen alle fragmentierter UDP Pakete durch eine Firewall vor dem DNS Resolver:** Nur geringe Performance Verluste oder leichte Performance Gewinne, je nach eingesetzter DNS-Resolver-Software, da die Software bei ausbleibenden Antworten dynamisch die EDNS Antwort-Grösse anpasst (wie vorherige Gegenmassnahme)
- textbfVerwerfen von fragmentierten UDP Paketen, welche unter der durch EDNS signalisierten Antwortgröße fällt: DNS-Performance vergleichbar mit vorheriger Gegenmassnahme, jedoch komplexere Firewall-Regeln, welche mehr Last auf dem System erzeugen
- **Ignorieren der "Additional Section" in DNS Antworten:** Nicht umsetzbar, da die "Additional Section" für DNS-Delegations-Verweise benötigt werden
- **Absicherung der Kommunikation zwischen DNS Resolver und autoritativen Servern durch Transaction Signatures (TSIG):** Performance Vergleichbar mit klassischem DNS-over-UDP, muss jedoch sowohl auf DNS-Resolvern als auch auf autoritativen Servern aktiviert sein und ist bisher nicht in allen populären DNS-Resolvern implementiert

Das Ausschalten von MTU-Path-Discovery hilft gegen ICMP-Spoofing-Angriffe welche IP-Fragmentierung von DNS Paketen durch künstliches Absenken der Path-MTU erzeugen.

In der Studie haben wir jedoch auch eine Reihe von DNS-Anworten von populären Domains gefunden, welche die Ethernet-MTU von 1.500 Byte überschreiten. Daher würde die Gegenmassnahme in diesen Fällen nicht helfen.

Auf der Grundlage der Messungen und Tests zu den möglichen Abhilfestrategien empfehlen wir eine Kombination aus zwei effizienten Maßnahmen zur Abschwächung von DNS-Fragmentierungs-Angriffen:

- Verhinderung der DNS-Fragmentierung durch Herabsetzung der maximalen EDNS-Nachrichtengröße auf 1.232 Bytes. Diese Gegenmassnahme kann unilateral von Betreibern von DNS-Resolvern und autoritativen DNS-Servern konfiguriert werden und hat nur sehr geringen Einfluss auf die Performance.
- Blockieren bzw. Verwerfen aller fragmentierten DNS-Nachrichten auf der Firewall-Ebene. Wird die DNS-Antwort-Grösse auf 1.232 Byte eingeschränkt dann kann es keine “natürlichen” DNS-Anworten mit IP-Fragmentierung geben.

Zusätzlich zu diesen beiden Hauptempfehlungen führen wir weitere Konfigurationsänderungen und bewährte Verfahren zur Verhinderung von DNS- Fragmentierung auf, welche das Risiko erfolgreicher DNS-Fragmentierungs-Angriffe verringern.

Die empfohlenen Änderungen in der Konfiguration von DNS-Servern mildern zwar die Auswirkungen des Cache-Poisoning Angriffes ab, beseitigen die Ursache jedoch nicht. Zur Behebung des DNS-Cache-Poisoning Problems - und vieler anderer Angriffe auf die DNS Infrastruktur - muss die DNSSEC-Signierung und -Validierung flächendeckend eingesetzt werden.

Recent Developments in the Context of Online Elections and Digital Polls in Germany

Bernhard Beckert[1], Jurlind Budurushi[2], Armin Grunwald[3], Robert Krimmer[4], Oksana Kulyk[5], Ralf Küsters[6], Andreas Mayer[7], Jörn Müller-Quade[8], Stephan Neumann[9], Melanie Volkamer[10]

Abstract: The paper summarizes the technical report [Be21] which was published in 2021. The aim of the paper is to summarize and critically discuss the situation in Germany concerning electronic voting.

1 Introduction and Motivation

During the COVID-19 pandemic many organizations (including unions, companies and public institutions) have been faced with the issue of conducting their elections and secret polls without jeopardizing the health of their voters and poll workers. Several election organizers have decided for conducting online voting elections. At the same time, there were hardly any experiences with online voting in Germany before the pandemic, as the topic was barely discussed due to the Federal Constitutional Court Decision on voting machines in 2009[11]. After a year of the pandemic, however, the situation has changed: in the meanwhile, several of elections and polls took place online. However, the systems used often do not correspond to the state of the art in research. Therefore, for the future usage of online elections and digital polls (especially after the pandemic) it is important that election organizers, candidates and voters understand the risks that the currently used systems may imply and how they can be addressed in the context of online elections. Only in this way

[1] Karlsruhe Institute of Technology, https://formal.iti.kit.edu/beckert/index.phtml
[2] Cloudical Deutschland GmbH, https://jurlindbudurushi.com
[3] Karlsruhe Institute of Technology, https://www.itas.kit.edu/kollegium_grunwald_armin.php
[4] University of Tartu, https://www.etis.ee/CV/Robert_Krimmer/est?lang=ENG
[5] IT University of Copenhagen, https://okskulyk.github.io/
[6] University of Stuttgart, https://sec.uni-stuttgart.de
[7] Hochschule Heilbronn, https://www.hs-heilbronn.de/andreas.mayer
[8] Karlsruhe Institute of Technology, https://crypto.iti.kit.edu/head_of_institute.php
[9] https://www.stephanneumann.it
[10] Karlsruhe Institute of Technology, https://secuso.aifb.kit.edu/Team_Volkamer.php
[11] Decision: Order of March 3, 2009 - 2 BvC 3/07, 2009. http://www.bundesverfassungsgericht.de/SharedDocs/Entscheidungen/EN/2009/03/cs20090303_2bvc000307en.html

doi:10.18420/sicherheit2022_16

informed decisions can be made regarding whether the risks are appropriate and manageable for a particular system to be used in a specific use case.

A number of legal and technical requirements have been proposed regarding the security of online voting[12]. A significant part of these requirements are related to vote privacy, as well as public nature of elections, transparency and verifiability for the individual cast votes as well as for the tallied results.

In our work, we therefore set the goal to support election organizers to make an informed decision regarding choosing the approach and the implemented system for conducting online elections. We aim to draw attention to which aspects are important for the general risk assessment as well as to the potential threats and misconceptions related to already conducted online elections. More specifically, we have looked at several real-world elections and studied the security and the risks of the underlying election systems. This document also contains information and discussions about technical guidelines of the Federal Office for Information Security (BSI) on online elections. Here, we briefly discuss three such elections. We kindly refer the reader to the full version of our paper [Be21] for all details.

2 Christian Democratic Union (CDU) Party Conference

Due to the COVID-19 pandemic, the annual party conference of the CDU took place online in 2021. One particular novelty of this party conference was that a total of 1001 delegates have cast their vote online for the first time to elect the future leader of CDU. In conducting this vote, the party relied on Polyas, a company that specializes in providing services for online elections[13].

In order to ensure that the vote will be cast and stored in the digital ballot box as intended by the voter, the delegates receive a so-called verification code after casting their vote. The verification code is meant to serve as the anonymous proof that one's voting choice has been correctly recorded, in that after the party conference the CDU made all the anonymous verification codes together with the corresponding vote available. The use of such codes has been already discussed in academic publications. In particular, previous research warned about the possibility of so-called "clash-attacks" (see e.g. [KTV]): simply expressed, the adversary would issue the same verification code to several delegates who voted for the same candidate, and assign the remaining votes to other candidates. The information about such an attack has been published and is known to the developers of the voting system. However, this case shows that the users of online voting systems need to be made aware about possible risks and vulnerabilities of a system, so that informed decisions can be made. Furthermore, in case of the CDU party conference, postal voting was offered as an alternative voting channel, thus requiring a clear and transparent process in case of possible discrepancies.

[12] See e.g. the requirements on the European level: https://rm.coe.int/0900001680726f6f

[13] https://www.polyas.de/blog/de/allgemein-de/der-cdu-bundesparteitag-und-polyas (Last visited: 23.08.2021)

3 Online election at the German Informatics (GI) Society

Since 2004, the GI has been conducting their annual elections (Präsidiums- and Vorstandswahlen) online. The members can also cast their vote via postal voting upon request. The access credentials for the online voting system consist of the GI member number and a password which gets sent to the voters via post. For the election, different online voting systems, provided by the company Polyas, have been used. For many years a so-called black box system by Polyas was in use, and it was decided to switch to an end-to-end verifiable system in 2018 [Be19]. The new system has step-by-step been developed towards being end-to-end verifiable. The development is currently still in progress. The voting systems used by the GI so far hence contain a number of risks, such as (1) manipulations by Polyas as system provider cannot fully be detected by the GI, the voters or the candidates, (2) detection of manipulations by administrators of the computing centers or by cyber criminals is in place but still a bit limited, and (3) malware on the end devices of the voters can change the vote without being detected. The responsible persons at the GI are aware of these risks and consider them to be acceptable. The main reason is that the GI members are trusted to keep their devices appropriately secured. Moreover, it is assumed that neither the candidates nor external entities are interested in manipulating the election, since the elected people are already engaged in the organization and the position is offered on a honorary basis. Finally, it was decided that Polyas can be trusted not to engage in any manipulations.

The GI is aware that many of the election organizers are looking at them as an example of which systems to use. The risk of the deployed systems, however, does not only depend on the security background knowledge among the voters. Therefore, it is important for every election organizer to decide themselves whether the risks and attack possibilities that may exist in specific online voting scenarios are acceptable for their election or not.

4 Shareholder elections

The shareholders' meetings in companies have also been conducted mostly online in 2020 due to the pandemic, including the needed voting. The basis for conducting virtual shareholders' meetings is the Legislation for Mitigation of Consequences of COVID-19 Pandemic in Civic, Bancrupcy and Criminal Law (Gesetz zur Abmilderung der Folgen der COVID-19-Pandemie im Zivil-, Insolvenz- und Strafverfahrensrecht)[14] that was adopted under the emergency procedure by the Federal Assembly. According to the Justice Ministers in the Federal States, the virtual shareholders' meetings should be adopted as an equal alternative to meetings in person in the long term[15].

[14] https://www.bmjv.de/SharedDocs/Gesetzgebungsverfahren/Dokumente/Bgbl_Corona-Pandemie.pdf (Last visited: 23.08.2021)

[15] https://www.zeit.de/news/2021-06/17/justizminister-wollen-dauerhaft-virtuelle-hauptversammlung (Last visited: 23.08.2021)

A study published at the 17. German IT-Security Congress of the Federal Office for Information Security in February 2021 analyzed the security of eight shareholders' meetings web platforms [Ma21]. These web platforms were used in the time period from 28. April 2020 to 31. December 2020 to conduct 584 virtual shareholders' meetings in German companies. The results show that almost 72% of the investigated virtual shareholders' meetings contained critical vulnerabilities that could in particular lead to undetected manipulation of the vote, complete takeover of a shareholder's account by the adversary, targeted prevention of conducting the meeting or leaking personal data of shareholders.

It is worth noting, that the analysis only included well-known web-based attacks from the OWASP Top 10 list[16]. Some of the possible attack vectors were thus not investigated, in particular, attacks by other actors such as malicious server administrators, election organizers or service providers. Additionally, the study showed that further methods for ensuring the principles of public nature of elections, transparency and verifiability of individual votes as well as the tallied results are not being offered by any of the platforms. Especially, it is unclear how a notary can ensure the integrity of the voting system and how they can check the results of the voting at the end of the meeting against errors and manipulations.

5 Summary

The choice of an online voting system is not trivial for a variety of reasons. The security of such systems is often determined by small details. There is no single approach or implemented system that is optimal for all kinds of elections, and the service providers of such systems understandably advertise the advantages and security of their own product. Therefore, the following questions should be answered before conducting online elections: (a) Under which adversary model and under which assumptions does the system fulfill the fundamental election principles? Are these assumptions realistic? (b) Can manipulations via cyber-attacks, malicious administrators and election organizers be detected, or does one have to trust that all the involved entities are honest and the possibility of cyber-attacks is excluded? (c) On which information and assumptions about the system do security properties and attacker model rely on? Finally, we want to mention that extensive research has been done on the topic of security in online voting. This research covers in particular numerous cryptographic protocols for different forms of voting as well as studies on the usability of the verifiability techniques. On top of that, other countries such as Switzerland and Estonia already offer – other than the elections being conducted in Germany – verifiability.

Acknowledgement. This work was supported by funding of the Helmholtz Association (HGF) through the POF subtopic 46.23.01 called 'Methods for Engineering Secure Systems'.

[16] https://owasp.org/www-project-top-ten/

Bibliography

[Be19] Beckert, B; Brelle, A; Grimm, R; Huber, N; Kirsten, M; Küsters, R; Müller-Quade, J; Noppel, M; Reinhard, K; Schwab, J; Schwerdt, R; Truderung, T; Volkamer, M; Winter, C: GI Elections with POLYAS: a Road to End-to-End Verifiable Elections. In: E-Vote-ID. Gesellschaft für Informatik (GI), p. 293–294, 2019.

[Be21] Beckert, B; Budurushi, J; Grunwald, A; Krimmer, R; Kulyk, O; Küsters, R; Mayer, A; Müller-Quade, J; Neumann, S; Volkamer, M: Aktuelle Entwicklungen im Kontext von Online-Wahlen und digitalen Abstimmungen. Technical report, 2021. 46.23.01; LK 01.

[KTV] Küsters, Ralf; Truderung, Tomasz; Vogt, Andreas: Clash Attacks on the Verifiability of E-Voting Systems. In: 2012 IEEE Symposium on Security and Privacy. pp. 395–409.

[Ma21] Mayer, Andreas: Virtuelle Hauptversammlungen: Ein sicherer Ersatz für Präsenzveranstaltungen? Tagungsband zum 17. Deutschen IT-Sicherheitskongress, 2021.

Ongoing Automated Data Set Generation for Vulnerability Prediction from Github Data

Torge Hinrichs[1]

Abstract: This paper describes the development of a continuous github repository analysis pipeline with the focus on creating a data set for vulnerability prediction in source code. Currently, used data sets consist only of source code functions or methods without additional meta information. This paper assumes that the surrounding code of vulnerable functions can be beneficial to the detection rate. In order to test this assumption, large data sets are needed that can be created using the proposed pipeline. Although the pipeline requires some improvements, in a first test run 1.5 million repositories could be analyzed and evaluated. The resulting data set will be published in the future.

Keywords: Vulnerability Prediction; Vulnerability Detection; Machine Learning; data set generation

1 Introduction

Detecting vulnerabilities in source code has been one of the most active fields in security research during the last years. Due to the latest advances in machine learning and data science more and more researchers are using these techniques for vulnerability detection / prediction. All approaches have one thing in common. They require large data sets of non- and vulnerable source code to train their models. Finding source code repositories for security analysis, analyzing and labeling the data is a tedious task that can not be performed by hand at a large scale. Even though some projects like "Project KB"[Po19] tried to craft such a data set manually, they "only" managed to analyze 205 distinct open source-projects leading to a remarkable 1282 commit entries of data. However, the amount of data gathered is minimal in comparison of other traditional machine learning approaches which use millions of data samples to train their networks. Automatically gathered data sets like "Draper Muse Data set"[Ru18] often consists of individual source code methods or functions with a label indicating the type of vulnerability or a binary classification without additional meta information. Even though this approach is beneficial for training machine learning models with remarkable results, it limits the scope of detection since the surrounding of the function can not be taken into consideration. In addition, creating data sets for vulnerability detection suffers from the "needle in the haystack" problem[SW13]. Vulnerabilities are less frequent and hard to detect, therefore the data sets contain large quantities of non-vulnerable in comparison to vulnerable data (scarce data)[BS19], which influences the overall performance of machine learning approaches. To tackle the lack of data sets for source code analysis with machine learning that also take the surrounding of a finding into consideration, this paper proposes an ongoing data analysis of public github.com

[1] HAW Hamburg, Berliner Tor 5, 20099 Hamburg, Germany torge.hinrichs@haw-hamburg.de

doi:10.18420/sicherheit2022_17

repositories to automatically identify and classify vulnerabilities with static analysis, saving the vulnerable lines and the commit id. This reduces the “needle in the haystack” problem in the long run and enables context analysis for future work. The following section provides a brief overview of the approach and methods used, followed by first analysis results and finally a discussion with future work.

2 Approach and Methods

The hypothesis of this contribution is that a static analysis pipeline is capable of performing all previously mentioned steps autonomously and can be scaled to a larger computing setup if needed. The following figure 1 shows a brief overview of the implemented components. Each block shown in the figure is a separate component that can be scaled if needed to

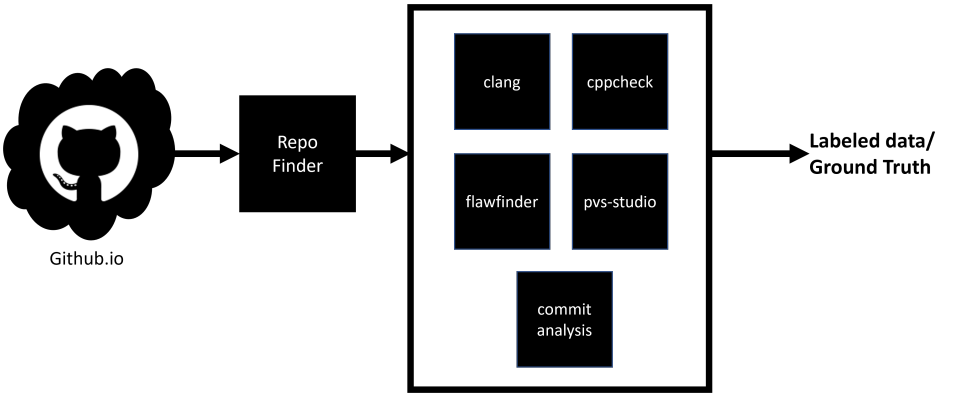


Fig. 1: Overview: Static analysis pipeline

prevent bottlenecks. Creating this approach can be split in 4 main tasks: selecting and providing the repository, analyzing with static analysis and labeling the findings. For the first task, a github interface was created that is capable of searching the repository stream provided by their API. By default, the stream is ordered by the repository id that increases for each repository created on the platform. The github interface also selects repositories that match the following constrains: The repository consists mainly of C/C++ code, due to the fact that most source code vulnerabilities occur in this language according a report by white source[Wh21]. In addition, the repository shall have a minimum size of at least 100 lines of C/C++ code. This assumes that repositories smaller than this do not contain enough complexity for context analysis. This assumption is based on manual reviews of small github repositories mostly containing snippets or small examples not suitable for machine learning. Repositories matching these criteria will automatically be cloned to a storage server and scheduled for analysis and a meta entry is created in a mongoDB database. During the analysis task, the pipeline automatically schedules an analysis block for each repository. The analysis consists of four state-of-the-art static analysis tools: clang, cppcheck, flawfinder, and pvs-studio. The selection is based on the open availability of these tools. They are also widely used in the creation of other data sets like Draper Muse. In

addition, a commit text analysis was developed that scans commit messages in order to find security relevant keywords based on the BSI-glossary [Bu21]. The last step in the pipeline is to map the reports of the different analysis tools to a common label that is associated with the corresponding Common Weakness Enumeration (CWE). This is achieved by a look up table for each tool.

3 Results

The pipeline was deployed and ran for three months as a first test. During this period 1.5 million repositories where analyzed with an overall yield of 2.3% which results in 35.000 eligible repositories with at least one hit in the analysis tools. The performance of the repository selection was constrained to about 2000 repositories per hour. This is caused by API restrictions on github. The overall performance of each analysis block, however, was measured and is shown in figure 2.

Tool	Avg. Evaluation time [s / Repository]
clang	30
cppcheck	70
flawfinder	10
pvs-studio	30
commit message	60

Fig. 2: Analysis time comparison

The deviation in analysis effort is caused by the varying feature set of the tools used. Each tool was configured to use all available checks. Therefore, cppcheck had to perform the most amount of checks and this, in conclusion, results in the longest analysis time per repository. In addition, the execution time of the individual analysis had large variations. This time is not only determined by the files or lines of code in a repository, but also by the amount of commits. For example the Linux kernel with its over one million commits took more than six hours to scan on the hardware used.

However, evaluating the performance of the commit text analysis showed some issues of this approach. The selection of keywords only based on the BSI-Glossary led to a high number of false positives as keywords like “stackoverflow” are often used in URL to the equal named developer help website “stackoverflow.com”. Overall, the execution of this analysis was rather slow in comparison to its low benefit.

The labeling method, in comparison, has no mentionable issues. It was created independently based on the documentation of each individual tool used. The label for each sample is determined by the analysis result. If one tools detects a possible vulnerability the sample is labeled accordingly, also the number of tools that back this label is stored.

The resulting data samples are stored in a mongoDB database. In hindsight, this has to be changed to a more storage friendly solution. A No-SQL database worked well to store

various different shaped data fields and allows for querying the database, but takes up a lot of disk space. This can easily be improved by using an optimized or compressed storage solution like “hdf5”. This looses the ability to query, but not necessary in a machine learning application anyway.

4 Discussion

In this paper, a concept for creating an automated data set generation for vulnerability data from github repositories was described. This concept was implemented and a first test phase over a three-month period was performed. During that time 1.5 million repositories were analyzed. An evaluation of the pipeline showed that some analysis blocks are more effective than others compared to their throughput and overall effectiveness. In addition, the implementation showed that the concept can be used to generate security relevant data samples continuously. However, this pipeline can still be improved in future work. First, the pipeline should to be set up production ready so an ongoing analysis can be performed. In addition, the overall performance of the analysis should be increased by analyzing the individual blocks and optimize the configuration of each tool. Next, up-scaling the pipeline to a larger infrastructure would also be beneficial to increase the throughput. Finally, storage should be more suitable for machine learning applications. The benefits of having a query-able database is not needed for learning, instead the handling of large data sets in less disk space should be focused on. This can be achieved by transforming a database into another format like “.hdf5”. After the improvements of the pipeline, the final data set will be released to the public.

Bibliography

[BS19] Babbar, Rohit; Schölkopf, Bernhard: Data scarcity, robustness and extreme multi-label classification. 108(8):1329–1351, 2019.

[Bu21] Bundesamt für Sicherheit in der Informationstechnik: , Glossar der Cyber-Sicherheit, 2021. access 2021-11-12, https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Glossar-der-Cyber-Sicherheit/Functions/glossar.html.

[Po19] Ponta, Serena E.; Plate, Henrik; Sabetta, Antonino; Bezzi, Michele; Dangremont, C´edric: A Manually-Curated Dataset of Fixes to Vulnerabilities of Open-Source Software. In: Proceedings of the 16th International Conference on Mining Software Repositories. May 2019.

[Ru18] Russell, Rebecca L.; Kim, Louis; Hamilton, Lei H.; Lazovich, Tomo; Harer, Jacob A.; Ozdemir, Onur; Ellingwood, Paul M.; McConley, Marc W.: , Automated Vulnerability Detection in Source Code Using Deep Representation Learning, 2018.

[SW13] Shin, Yonghee; Williams, Laurie: Can traditional fault prediction models be used for vulnerability prediction? 18(1):25–59, 2013.

[Wh21] WhiteSource: The State of Open Source Vulnerabilities 2021. Technical report, WhiteSource, 2021.

Doktorand·innenforum

Your website has been hijacked: Raising awareness for an invisible problem

Anne Hennig[1]

Abstract: Running a business without having a website is nearly impossible nowadays. Content management systems (CMS) provide features which make it easy for laypersons to create sophisticated websites. But those can pose security risks and provide vulnerabilities for manipulations. With vulnerability notifications, website owners are notified about security risks. The work of this doctoral thesis is divided into two main parts: At first it is necessary to identify common themes with respect to vulnerability notifications and provide more information on how to improve future vulnerability notifications. The second main part is to develop and evaluate suitable awareness materials.

Keywords: web security; vulnerabilities; security notification; security awareness

1 Introduction

Information about services are mainly retrieved from online resources nowadays [Eu21]. Therefore, running a business without having a website is nearly impossible. Especially for small businesses, individuals or non-profit associations, content management systems (CMS) provide features to create and maintain sophisticated websites. But CMS's frameworks, plugins, and templates also provide vulnerabilities for manipulations. Conţu et al. [Co16] describe four main threats for open-source content management systems, from which different kinds of attacks can result.

Some attacks aim at the search results of the original website. In case of search engine Spam (SEO Spam) or Pharma Hacks, an attacker deploys code on a website to redirect to fake web shops [Ma20; Ma21b]. In the search engine results, these sites appear as shops selling illegal or banned drugs and medicines, or luxurious brand-name clothing for cheap. But the manipulation is not visible on the genuine website and the malicious code is often hidden in the CSS files of a website where it cannot be easily found [Ma21b]. Website owners, therefore, have to rely on vulnerability notifications to be informed about the problem.

The work of this doctoral thesis is divided into two main parts: At first, a suitable way to contact the affected website owners will be established. The second main part is to develop and evaluate suitable awareness materials for website owners. A description of the different parts and the current status of each part is described in the following sections.

[1] Karlsruhe Insitute of Technology, Institute of Applied Informatics and Formal Description Methods, Kaiserstraße 89, 76133 Karlsruhe, Germany anne.hennig@kit.edu

doi:10.18420/sicherheit2022_18

2 Related Work

Since the problem is not easy to detect, most website owners have to rely on vulnerability notifications by the security community to be informed about the manipulation. Within the project as well as in the literature it could be shown, that there is no easy way to notify affected website owners in case of a vulnerability. The basic finding is that notifying a website owner about the problem increases remediation rates (i.a. [Çe16; Çe17; Du14; St18; VM12]). Many of these studies, however, report low remediation rates and problems to reach out to the recipients of their vulnerability notifications.

2.1 Raising awareness by providing sufficient information and establishing trust

According to Stock et al. [St18], three key factors that lead to a successful notification campaign are the e-mail reading rate, awareness rising factors and the aware-to-fix rate. So far, different aspects of these factors have been researched.

Experimental studies with modifications of **sender, message framing, subject, and language** showed that although some senders or message types work better than others, there was no statistically significant difference in the remediation rate between the treatment groups. This indicates that factors like sender reputation or framing alone do not drive remediation rates [Çe16; Ma21a; St18; Ze19].

Aspects that establish **trust** in vulnerability notifications have been identified with a quantitative survey: **formal aspects** (e.g. sender or correct spelling), **content-related aspects** (e.g. description of the problem and a motivation that is not attached to financial demands), and **verifiability aspects** (providing verification possibilities, like contact information) [Ma21a]. Furthermore, previous research has shown that providing **detailed information** were more effective with respect to remediation rates [Çe16; Li16; VM12].

Nevertheless, even if a vulnerability notification is deemed trustworthy, remediation rates are still low. Therefore, further factors need to be included for notifications to be successful.

2.2 Increasing remediation beyond establishing trust

Diligence of website operators, **popularity** of a website, as well as the **severity** of an issue, and the **media attention** the issue gets, seem to influence remediation rates as well [VM12; Ze19]. But besides intense media coverage, a significant amount of vulnerable websites was left unfixed in one study [Du14].

It has been proposed that **external incentives** also increase remediation rates [Ze19]. Recent studies showed that legal consequences and fines [Ma21a], as well as technical consequences imposed by the hosting provider [Çe19] increase remediation rates. Other authors proposed that loss of reputation can serve as an external incentive, too [VM12].

3 Identifying a suitable communication channel and message content

3.1 Interviews with affected website owners

As stated above, website owners' reactions to vulnerability notifications are mainly retrieved from experimental studies or quantitative surveys. Only a few studies have used qualitative approaches so far, but all of them were dedicated to a different target group or to answer different questions [Di18; Je20; Kr17; Li19]. But because qualitative approaches are more applicable for *understanding* opinions, we directly talked to affected website owners to assess their opinions and to identify common themes about vulnerability notification.

An interview guideline was developed that covered three main topics: notification of a previous incident, possible notifications in case of future incidents, and channels, which the persons use to actively inform themselves about security incidents. Socio-demographic data that were found to have an impact on the remediation rate were also collected [Çe16; St18].

With this we would like to answer the following research questions: (1) How did website owners receive previous web vulnerability notifications? (2) What are suitable senders and communication channels that the website owners deem trustworthy? (3) What aspects should we consider in future notifications to be deemed trustworthy? (4) What – if any – channels do website owners use to actively inform themselves about security incidents?

Between November 2020 and March 2021, our project partner compiled a list of domains that were affected by a Pharma Hack or a related SEO spam infection. In April 2021, the complete list was sent to associated project partners or the State Offices of Criminal Investigations, which were supposed to inform the website owners about the vulnerability.

Between July and September 2021, we contacted 65 website owners from this list via e-mail, and asked, if we could call them for an interview. We used the contact information given on their websites. In total, we conducted 25 qualitative interviews with website owners (response rate: 39 %) and could confirm that distrust in unexpected notifications is high. We further found that verifiability is the most important factor to establish trust in notifications. We also endorse the findings that raising awareness for the severity and the complexity of the problems is crucial to increase remediation rates.

3.2 Evaluating the effectiveness of a vulnerability notification

As described above, designing a message content that is not only trustworthy, but also provides *convincing* incentives, like legal [Ma21a], reputational [VM12], or technical [Çe19] sanctions, is currently an open area of research. It has also not been investigated whether incentives that are tied to a sender that has the executive power to enforce the proposed consequences can increase the aware-to-fix rate even more. Since this is up to this point an open area of research, we expect these findings to be a major contribution.

We plan an experimental study to answer the following research questions: (1) Which sender has which impact on the remediation rate? (2) Which framing of the message has which impact on the remediation rate? (3) Do sender and framing of a message correlate with respect to the remediation rate?

We used the results of our interview study to design a vulnerability notification that includes the major trust-promoting factors like verification possibilities, a clear description of the problem and a plausible motivation. These basic information are either combined with a framing that contains technical, one that contains reputational, or one that contains no incentives. We also identified senders which were deemed plausible by our interviewees and which match one of the three framings with respect to the consequences they can impose.

4 Awareness as prevention

As the results of our interview study and the current research show, raising awareness for vulnerability notifications in general and especially for the severity of this vulnerability is still an open area of research. The development of effective awareness materials that are tailored to certain target groups is therefore deemed crucial to increase the aware-to-fix rate. It is planned to include the website owners, hosting provider, industry branches, and other intermediaries like internal (web) administrators or CISOs, into this process. Figure 1 gives an overview of the timeline of this doctoral thesis until the end of the research project. It is possible to further evaluate and improve materials, as well as to extent the reach of the project (for example by contacting website owners in non-German speaking countries) beyond the duration of the project, if funding is provided.

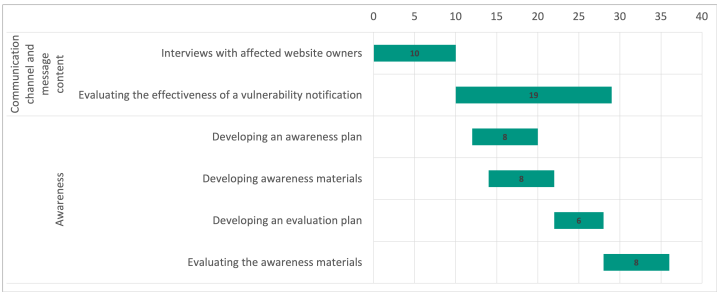


Fig. 1: Timeline for the doctoral thesis in month until the end of the research project

References

[Çe16] Çetin, O.; Jhaveri, M. H.; Gañán, C.; Eeten, M. v.; Moore, T.: Understanding the role of sender reputation in abuse reporting and cleanup. Journal of Cybersecurity 2/1, pp. 83–98, 2016, ISSN: 2057-2085, URL: https://academic.oup.com/cybersecurity/article/2/1/83/2629556.

[Çe17] Çetin, F. O.; Ganan, C. H.; Korczynski, M. T.; Eeten, M. J. G. v.: Make notifications great again: learning how to notify in the age of large-scale vulnerability scanning. In. 16th Workshop on the Economics of Information Security (WEIS 2017), San Diego, pp. 1–23, 2017, URL: http://resolver.tudelft.nl/uuid:621f4a4f-e5d9-4f04-abc4-46252f9db3db.

[Çe19] Çetin, O.; Gañán, C.; Altena, L.; Tajalizadehkhoob, S.; Eeten, M. v.: Tell Me You Fixed It: Evaluating Vulnerability Notifications via Quarantine Network. 2019 IEEE European Symposium on Security and Privacy (EuroS&P) 00/, pp. 326–339, 2019, URL: https://www.readcube.com/library/42cf3704-a798-4336-b319-363ceea244b9:55032f97-a10f-4272-baa8-31cb68b24da4.

[Co16] Conţu, C. A.; Popovici, E. C.; Fratu, O.; Berceanu, M. G.: Security issues in most popular content management systems. In: 2016 International Conference on Communications (COMM). Pp. 277–280, 2016, URL: https://ieeexplore.ieee.org/document/7528327.

[Di18] Dietrich, C.; Krombholz, K.; Borgolte, K.; Fiebig, T.: Investigating System Operators' Perspective on Security Misconfigurations. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security/, pp. 1272–1289, 2018, URL: https://www.researchgate.net/publication/328329898%5C_Investigating%5C_System%5C_Operators%5C%27%5C_Perspective%5C_on%5C_Security%5C_Misconfigurations.

[Du14] Durumeric, Z.; Li, F.; Kasten, J.; Amann, J.; Beekman, J.; Payer, M.; Weaver, N.; Adrian, D.; Paxson, V.; Bailey, M.; Halderman, J. A.: The Matter of Heartbleed. In: Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14. IMC '14, Association for Computing Machinery, Vancouver, BC, Canada, pp. 475–488, 2014, ISBN: 9781450332132, URL: https://doi.org/10.1145/2663716.2663755.

[Eu21] Eurostat: Anteil der Personen, die das Internet zur Suche nach Informationen über Waren und Dienstleistungen genutzt haben, in den Ländern der Europäischen Union (EU-28) im Jahr 2020, Mar. 2021, URL: https://de.statista.com/statistik/daten/studie/806662/umfrage/internetsuche-nach-informationen-ueber-waren-und-dienstleistungen-in-der-eu/, visited on: 10/28/2021.

[Je20] Jenkins, A.; Kalligeros, P.; Vaniea, K.; Wolters, M. K.: "Anyone Else Seeing this Error?": Community, System Administrators, and Patch Information. 2020 IEEE European Symposium on Security and Privacy (EuroS&P) 00/, pp. 105–119, 2020, URL: https://www.readcube.com/library/42cf3704-a798-4336-b319-363ceea244b9:a0e7c73b-8876-4cd4-81a7-f25b6b36240e.

[Kr17] Krombholz, K.; Mayer, W.; Schmiedecker, M.; Weippl, E.: Ï Have No Idea What I'm Doing On the Usability of Deploying HTTPS. In: 26th USENIX Security Symposium (USENIX Security 17). USENIX Association, Vancouver, BC, pp. 1339–1356, 2017, ISBN: 978-1-931971-40-9, URL: https://www.usenix.

org/conference/usenixsecurity17/technical-sessions/presentation/krombholz.

[Li16] Li, F.; Durumeric, Z.; Czyz, J.; Karami, M.; Bailey, M.; McCoy, D.; Savage, S.; Paxson, V.: You've Got Vulnerability: Exploring Effective Vulnerability Notifications. In: 25th USENIX Security Symposium (USENIX Security 16). Pp. 1033–1050, 2016, ISBN: 978-1-931971-32-4, URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/li.

[Li19] Li, F.; Rogers, L.; Mathur, A.; Malkin, N.; Chetty, M.: Keepers of the Machines: Examining How System Administrators Manage Software Updates. In: Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019). USENIX Association, Santa Clara, CA, pp. 273–288, 2019, ISBN: 978-1-939133-05-2, URL: https://www.usenix.org/conference/soups2019/presentation/li.

[Ma20] Martori, A.: Spamdexing: What is SEO Spam and How to Remove It, Feb. 2020, URL: https://blog.sucuri.net/2020/02/spamdexing-seo-spam.html, visited on: 10/28/2021.

[Ma21a] Maass, M.; Stöver, A.; Pridöhl, H.; Bretthauer, S.; Herrmann, D.; Hollick, M.; Spiecker, I.: Effective notification campaigns on the web: A matter of Trust, Framing, and Support. In: 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, pp. 2489–2506, 2021, ISBN: 978-1-939133-24-3, URL: https://www.usenix.org/conference/usenixsecurity21/presentation/maass.

[Ma21b] Malcare: What is WordPress Pharma Hack & How to clean it?, Jan. 2021, URL: https://www.malcare.com/blog/what-is-pharma-hack-how-to-clean-it/.

[St18] Stock, B.; Pellegrino, G.; Li, F.; Backes, M.; Rossow, C.: Didn't You Hear Me? - Towards More Successful Web Vulnerability Notifications. Proceedings 2018 Network and Distributed System Security Symposium/, pp. 1–15, 2018, URL: https://swag.cispa.saarland/papers/stock2018notification.pdf.

[VM12] Vasek, M.; Moore, T.: Do Malware Reports Expedite Cleanup? An Experimental Study. In: 5th Workshop on Cyber Security Experimentation and Test, CSET '12, Bellevue, WA, USA, August 6, 2012. USENIX Association, pp. 1–8, 2012, URL: https://www.usenix.org/conference/cset12/workshop-program/presentation/vasek.

[Ze19] Zeng, E.; Li, F.; Stark, E.; Felt, A. P.; Tabriz, P.: Fixing HTTPS Misconfigurations at Scale: An Experiment with Security Notifications. In: The 2019 Workshop on the Economics of Information Security (2019). Boston, MA, pp. 1–19, 2019, URL: https://www.semanticscholar.org/paper/Fixing-HTTPS-Misconfigurations-at-Scale%5C%3A-An-with-Zeng-Li/b22c522c6201f8545e1626deaf6ca43db52444d7.

Automated Monitoring of Operational Technology Security and Compliance for Power Grids

Enhancing Trust by Continuous Security Configuration Monitoring

Bastian Fraune [1]

Abstract: IT security standards can increase trust in a system or component if compliance to the standard can be proven to third parties. Those standards usually specify requirements for security features, which then lead to a certain configuration of an industrial control system. Continuous monitoring of IT security configurations on intelligent electronic devices is difficult because there is no standardised way to query the security configurations of those devices. The objective of this PhD project is to enable automatic querying of security settings from industrial control system in the use case of the power grid infrastructure for remote monitoring. This opens up the possibility of automatically comparing the actual security state on the device against the defined IT security standard configurations. In such cases, industrial control systems that do not comply with defined security standards can thus be identified directly by monitoring systems in the control centre.

Keywords: ICT-Monitoring; IT-Security; OT-Security; ICS; Smart Grid; Trust; Compliance; Trusted Computing; Power Grid

1 Introduction and Motivation

Regarding the energy domain the general increase in digitization is also affecting the power grid, which is undergoing increasingly strong transition. This is characterized in particular by the restructured energy generation, which is moving from centralized generation to decentralized generation. This refers to wind turbines and wind farms, large solar farms and small solar installations on private buildings. Likewise, a variety of flexible loads and generators need to be controlled; electric cars can feed in or take out energy through their batteries and thus help to keep the physical power grid in a stable equilibrium. In order to implement such scenarios, modern communication technologies are required and inevitably the attack surfaces on the supervisory, control, and data acquisition (SCADA) systems there are increased, since (future) loads and generators can no longer communicate only via isolated communication channels.

Possible threats for the presented scenario are, for example, high-wattage botnets. In such a scenario, due to weak security configurations operational technology (OT) devices as

[1] City University of Applied Sciences Bremen, Institute of Computer Science and Automation, Flughafenallee 10, DE-28199 Bremen, Germany, bastian.fraune@hs-bremen.de

doi:10.18420/sicherheit2022_19

part of the power grid have been manipulated. After this manipulation they now operate as part of a botnet that then can be controlled remotely by the attackers. Through this hostile takeover, big loads can be manipulated and can thus be used to make profits in the electricity market [Sh21]. According to the Federal Office for Information Security Germany (BSI) security compendium, the present security issues of industrial control system (ICS) devices are, among others, insecure configuration, inadequate monitoring and detection procedures, insufficient access protection as well as inappropriate integration into organizations security [Bu21].

One general, systematic approach to counteract those general issues is to apply security measures. An international standard for information security management system is defined in the ISO 27001 series [IS17]. It describes a management process with the aim to establish and implement as well as operating and improving information security [Ke16]. To underline the importance of such standards, in Germany its application is forced by law to all operators of critical infrastructures (see §11 section 1b German „Energiewirtschaftsgesetz“[2]). On the other hand, in contrast to organisational standards, technical IT security standards describe the security requirements for OT ICS in much more concrete technical terms and defined properties. An example in the energy domain is the standard IEC 62351-7:2017 [In17]. It describes data models for network and system management and targets all network devices. Such devices shall be able to .g. detect unauthorised access, detect resource overload and detect invalid protocols.

It becomes challenging when an organisational standard is provided for monitoring the configurations, but the technical possibilities of such devices do not offer the possibility to do so. As stated in [KL15] a large portion of the critical configuration information is on the device itself, which run on proprietary or closed operating systems, which makes it more difficult to query configuration data from devices.

2 Related Work

Montesino et al. have examined the automation of information security in [MF11]. Their focus was set on the requirements of ISO 27001 and National Institute of Standards and Technolgy (NIST) SP800-53. They identified 133 security controls in annex A of ISO 27001 and 198 security control in NIST SP800-53. Two criteria were defined for the assessment of the automation capability: 1. A security control can be automated if the operation of the control can be done without human intervention, which means that it requires only machine-readable and processable resources. 2. The control can be implemented partially or completely by at least one security application. The listed security applications include, among others, Microsoft Systems Management Server, nCircle IP360 configuration and compliance server, AlienVault and PfSense Firwall. Based on the analysis, in the ISO 27001 standard 37 controls can be automated, which is 27.8% of the identified security controls.

[2] German Energy Industry Act

The analysis of the NIST SP800-53 has shown 62 controls that can be automated, making 31.3% of the 198 security controls. Despite a tabular list by category including examples, it remains unclear which controls exactly can be automated. It also looks like the settings that can be automated primarily affect the (office-) IT area.

The Security Content Automation Protocol (SCAP) is a project of NIST and emerged from the NIST IT security automation agenda. SCAP is not a protocol after all and is described on the website as "[...] a synthesis of inter operable specifications derived from community ideas"(see [Na18]). Therefore it is rather a suite of applications for exchanging security automation content, to be able to assess configuration compliance and it supports the detection of present vulnerabilities. The suite contains, among others, a vulnerability scanner, describes asset reporting formats as well as asset identification and other description languages. In 2018 NIST has announced version 2 of SCAP with a much broader focus on the overall architecture and is currently in Internet-Draft status under the Internet Engineering Task Force (IETF) as *Security Automation and Continuous Monitoring (SACM) Architecture* [WFM18]. SCAP v1 targets primarily the typical office IT domain.

Automated standard compliance for Industry 4.0 has been investigated by Bicaku et al. with the aim to express compliance in accordance to standards automatically [Bi18]. They have developed the *Monitoring and Standard Compliance Verification Framework* (MSVP), which considers security and safety standards as well as organisational indicators. For the evaluation, they queried the needed data from devices by using custom agents that are able to communicate with their monitoring system. The monitoring system then provides the collected data to their MSVP framework. The data from the devices has been collected by using custom scripts in their devices, which were located in the cloud. As data collection was not their focus, they mentioned that "[...] measurable metrics should be imposed by standardised bodies [...]"[Bi18, p. 751]. In the context of this PhD project, this underlines how important it is, to provide measurable security configurations in an integrated way.

In [Ch18] a framework for continuous compliance monitoring is presented. The authors propose a process which maps requirements from standards to an ontology and extracts the requirements from the documents using natural language processing (NLP). Their framework then links requirements with custom scripts to query data from systems. Those scripts have only been created for Powershell and demonstration purposes. It seems to be a good approach to gather information from different standards and extract requirements with NLP. Since their scripts have been made for Windows environments, they cannot be used for OT devices. But, the authors note, that it is important that the "[...] scripts have to be as atomic as possible in order to be reusable[...]". This also shows the need for a common information model that can be used to query security configuration from OT devices.

A very widely used protocol for monitoring information and communication technology (ICT) network components is the simple network management protocol (SNMP). It is standardised by RFC 3410-3418 and its original purpose was the management and configuration of network devices. Despite version 1 and 2 having security issues in their

design, it's still broadly ches and firewalls. Enhanced security was integrated in version 3, but the protocol still is mainly used for monitoring purposes [DH08].

3 Research Questions

As described, there is a gap in the ability to continuously monitor security configurations on OT devices. The associated risk is that this can lead to an unnoticed weakening of the security measures. At the same time, monitoring of configurations is mentioned as a requirement in organisational standards such as ISO 27001 [IS17]. Since the implementation of security configurations itself is a technical requirement, it should be possible to monitor them by technical means, if available [GHS17]. This is not the case for continuous monitoring of security configurations in OT devices. For this reason, this PhD project addresses the following research questions:

"How can automated monitoring of technical security configurations, for SCADA components in the energy distribution domain, be enabled?"

To be able to give an appropriate answer, sub-questions have been defined, as they allow to break the research question down and thus allow to approach it in a structured way.

Which security features are relevant from the view of existing domain standards? — It must be investigated first, which technical security requirements and features actually exist for those domains, especially OT devices in power substations. As already stated in [RU13] and [GHS17] many security standards related to smart grids exist. In order to extract the correct requirements, it is important to analyse which requirements are applicable in which context and thus necessary for this PhD project. As it is expected that typical security requirements are also included in industry standards such as the IEC 62443 series, those requirements will be considered, too.

How can existing information models from ICT component monitoring be considered? — This question aims to explore the extent, to which existing protocols for monitoring can be incorporated. The TC 57 is a technical committee of the International Electrotechnical Commission (IEC). TC 57 prepares and proposes standards for "Power systems management and associated information exchange"[In]. Many standards have been released and are currently in use. To be able to integrate the idea of the research project into the power grid, their standards have to be considered. Therefore an investigation driven by use cases is necessary in order to identify the required and applicable protocols. Suitable protocols shall be able to transport the monitoring data from the process level up to the operation and enterprise level.

The third research question is about *How can concepts of trusted computing support this?* — To enhance the trust of such monitorable security configurations an integration of hardware based trust anchors will be investigated. Trusted Computing concepts which allow to enable measured boot and remote attestation in combination with a hardware security module

(HSM) are part of the research. The HSM specified by the Trusted Computing Group (TCG) is the Trusted Platform Module 2.0 (TPM2). Such an integration into the monitoring of security configuration allows to ensure that the devices authenticity can be proven.

4 Methodology

To approach the described problem, different artefacts will be developed, which are derived from the research sub-questions. The first artefact aims at an information model suitable for querying security configurations from OT devices in the power grid. To achieve this, first the necessary or applicable security standards for the power grid are reviewed by a literature research. In the next step the needed security requirements and configurations from the identified standards are extracted. The resulting requirements then form the basis for an information model of the security configurations. In order to be able to evaluate the model, this is to be implemented iteratively on a virtual intelligent electronic device (IED). The primary development objective is the information model, that represents the required security configuration information. Thus it is the basis for further implementations and information exchange in the energy domain.

The second artefact aims to identify suitable communication protocols and extend them in order to be able to support the data from the information model. In the domain of the power grid, it must be taken into account that a large number of specific communication protocols already exist. Therefore, existing protocols are to be examined to see whether they can transmit the information model from the first artefact. Since this will probably not be the case, the most suitable protocol should be identified and then enabled to do so. The implementation is first carried out by analysing the capabilities of the communication standards of the TC 57 Group. The focus will be on the communication protocols between the sub-station and the control centre. The evaluation of those protocols will be done against the information model from the first artefact. A prototypical implementation will be used to represent a continuous evaluation of the research.

5 Conclusion

In order to get closer to a possible solution for the outlined research question, the primary research question and derived sub-questions were presented, as well the methodology to approach a solution. The answers to these questions should lead to the objective of automated verification of security configurations of OT devices. With such new possibilities, security audits for compliance and automated monitoring of OT devices can be enabled. Automatic proof of compliance with relevant security standards also contributes to increase trust in the system or the queried device. In the next steps, the use cases will be defined more precisely and more in-depth in order to be able to address the research questions in a more targeted way.

Bibliography

[Bi18] Bicaku, Ani; Schmittner, Christoph; Tauber, Markus; Delsing, Jerker: Monitoring Industry 4.0 applications for security and safety standard compliance. In: 2018 IEEE Industrial Cyber-Physical Systems (ICPS). IEEE, pp. 749–754, 2018.

[Bu21] Bundesamt für Sicherheit in der Informationstechnik (BSI): , IT-Grundschutz-Kompendium, 2021.

[Ch18] Cheng, Danny C.; B., Jod; Cu, Gregory; Rose, Nathalie: Towards end-to-end Continuous Monitoring of Compliance Status Across Multiple Requirements. International Journal of Advanced Computer Science and Applications, 9(12):456–466, 2018.

[DH08] Dinger, Jochen; Hartenstein, Hannes: Netzwerk- und IT-Sicherheitsmanagement - Eine Einführung. Universitätsverlag Karlsruhe, 2008.

[GHS17] Genzel, Carl-Heinz; Hoffmann, Olav; Sethmann, Richard: Zusammenfassung relevanter Informationssicherheitsstandards für deutsche Verteilungsnetzbetreiber. Technical report, Hochschule Bremen - FRI, 2017.

[In] International Electrotechnical Commission (IEC): , IEC - TC 57 Scope.

[In17] International Electrotechnical Commission: , Power Systems Management And Associated Information Exchange - Data And Communications Security – Part 7: Network And System Management (NSM) Data Object Models (IEC 62351-7:2017) (English Version), 2017.

[IS17] ISO Central Secretary: Information technology - Security techniques - Information security management systems - Requirements. Standard ISO/IEC 27001:2017, International Organization for Standardization, Geneva, CH, 2017.

[Ke16] Kersten, Heinrich; Klett, Gerhard; Reuter, Jürgen; Schröder, Klaus-Werner: IT-Sicherheitsmanagement nach der neuen ISO 27001. Springer Fachmedien Wiesbaden, Wiesbaden, 2016.

[KL15] Knapp, Eric D.; Langill, Joel Thomas: Security Monitoring of Industrial Control Systems. In: Industrial Network Security, pp. 351–386. Elsevier, 2015.

[MF11] Montesino, Raydel; Fenz, Stefan: Information Security Automation: How Far Can We Go? In: 2011 Sixth International Conference on Availability, Reliability and Security. IEEE, pp. 280–285, aug 2011.

[Na18] National Institute of Standardization and Technology (NIST): , Security Content Automation Protocol - Project Overview, 2018.

[RU13] Rosinger, Christine; Uslar, Mathias: Smart Grid Security: IEC 62351 and Other Relevant Standards. In: Power Systems, volume 71 of Power Systems, pp. 129–146. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[Sh21] Shekari, Tohid; Irvene, Celine; Cardenas, Alvaro A.; Beyah, Raheem: MaMIoT: Manipulation of Energy Market Leveraging High Wattage IoT Botnets. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. ACM, New York, NY, USA, pp. 1338–1356, nov 2021.

[WFM18] Waltermire, David; Fitzgerald-McKay, Jessica: Transitioning to the Security Content Automation Protocol (SCAP) Version 2. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, Sep 2018.

Reinforcement Learning-Controlled Mitigation of Volumetric DDoS Attacks

Hauke Heseding[1]

Abstract:

This work introduces a novel approach to combine *hierarchical heavy hitter* algorithms with *reinforcement learning* to mitigate evolving volumetric distributed denial of service attacks. The goal is to alleviate the strain on the network infrastructure through early ingress filtering based on compact filter rule sets that are evaluated by fast ternary content-addressable memory. The reinforcement learning agents task is to maintain effectiveness of established filter rules even in dynamic traffic scenarios while preserving limited memory resources. Preliminary results based on synthesized traffic scenarios modelling dynamic attack patterns indicate the feasibility of our approach.

Keywords: Distributed denial of service; software defined networks; hierarchical heavy hitters; reinforcement learning

1 Introduction

Distributed denial of service (DDoS) attacks pose a constant and severe threat to communication infrastructures. Particularly, volumetric DDoS attacks, which congest bottleneck links near a target system with unsolicited traffic, have become increasingly popular. To mitigate such attacks, we seek to achieve early attack traffic removal directly in the data plane. For this, we apply ingress filter rules that can be evaluated by ternary content-addressable memory (TCAM) in a single clock cycle enabling traffic filtering at line speed. This alleviates the strain on the network and protects downstream systems (including systems conducting further mitigation steps). Furthermore, we address the following question: How to handle dynamic traffic scenarios, where filter rules may become outdated, and how to balance filter rule effectiveness against TCAM utilization? Intelligent attackers may evade outdated rules by altering attack traffic composition. Also, unnecessarily fine-granular rules are undesirable since TCAM capacity is limited by high monetary costs and energy consumption.

To keep filter rules up to date, we monitor the data stream passing the ingress filter with hierarchical heavy hitter (HHH) algorithms that enable detection of suspicious IP subnets sending excessive traffic in volumetric DDoS scenarios. Recent advances enable direct integration of HHH algorithms into the data plane (e. g., [PAM17, Si17, Be20, Zh21]). To

[1] Karlsruhe Institute of Technology (KIT), KASTEL, Institute of Telematics, Kaiserstraße 40, 76133 Karlsruhe, Germany hauke.heseding@kit.edu

doi:10.18420/sicherheit2022_20

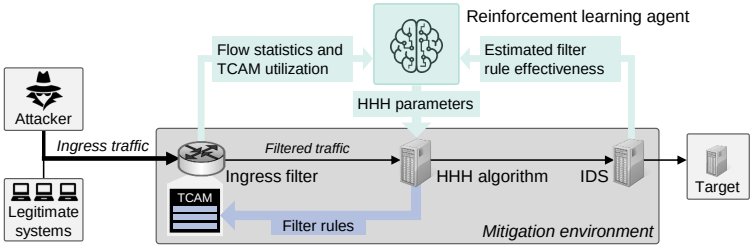


Fig. 1: Architecture components and workflow

achieve adaptivity to counteract intelligent attacks we leverage reinforcement learning (RL) to distinguish between highly distributed and densely clustered attack sources. This allows adjusting filter rule granularity accordingly (via parameterization of HHH algorithms) to avoid unnecessarily fine-granular rules. In essence, the RL agent serves to maintain the balance between filter rule effectiveness and TCAM utilization when traffic patterns change.

Threat model This work focuses on dynamic, volumetric DDoS attacks. To address intelligent attacker behavior, we first consider different volumetric attacks: direct botnet attacks (generating elephant flows) and amplification attacks (DNS, NTP, or SSDP reflection). Distribution of attack traffic sources and traffic intensity depend on the chosen attack vector. While bots typically have more compact distributions, reflectors yield higher per-node attack traffic volume. Second, attackers have the ability to change attack traffic composition by employing various attack vectors at different times during an ongoing attack.

Mitigation objectives The overall goal is to achieve fast and early attack traffic removal (by leveraging TCAM capabilities). We also seek to counteract evolving attacks by keeping filter rules up to date, preventing intelligent attackers from circumventing defenses. In essence, filter rules should maintain high precision (to preserve legitimate traffic) and sensitivity (to capture attack traffic) in dynamic traffic situations. Adaptation to evolving volumetric DDoS scenarios was recently studied in related work, focusing on programmable data plane (PDP) technologies [Zh20, Li21] as well as leveraging RL [MK15, SRP20]. In contrast to previous work, we apply rule-based filtering based on aggregated traffic features immediately at the ingress, to alleviate the strain on network infrastructure and downstream systems. This avoids resource-intensive state-keeping of per-flow mitigation, offers more fine-grained control than per-router throttling and does not require sophisticated PDP technologies.

2 Adaptive Ingress Filtering

Our strategy to keep filter rules up to date is twofold: (a) continuous traffic monitoring via an HHH algorithm and (b) adjusting filter rule granularity through RL. For this, our

mitigation system comprises four core components: a TCAM-based ingress filter, an HHH algorithm instance, a downstream IDS, and an RL agent (see Fig. 1). The agent interacts with the mitigation environment at discrete time steps to apply chosen parameters when the HHH algorithm is queried for filter rules (query time).

TCAM-based ingress filter The ingress filter is placed on a switch with TCAM resources, positioned upstream at an ingress point of a larger network (e. g., a backbone network). It applies a set of filter rules $\mathcal{R}$ obtained from the HHH algorithm. Each rule specifies an IP subnet. Any packet whose IP source address matches a subnet contained in $\mathcal{R}$ is removed from the ingress data stream. Due to TCAM technology, evaluating the entire set $\mathcal{R}$ requires only a single clock cycle. The second task of the ingress filter is to keep track of the number of removed packets for subsequent estimation of filter rule effectiveness.

HHH algorithm Packets passing the ingress filter are subsequently monitored by an HHH algorithm (executed on a server in proximity to the IDS) The HHH algorithm tracks the hierarchical distribution of source IP addresses to identify IP subnets sending excessive amounts of traffic. At query time a *frequency threshold* ϕ is applied during HHH computation indicating the minimum number of packets necessary to classify an IP subnet as potentially malicious. To avoid excessive hierarchical aggregation during HHH computation we restrict the maximum size of IP subnets accepted as filter rules since HHHs become less expressive with increasing aggregation. Indiscriminately adopting large subnets would lead to unwarranted removal of significant portions of the ingress traffic. To prevent this, we introduce a *hierarchy threshold* $H^{\max}$ limiting filtered subnet size to no more than $2^{H^{\max}}$. Together, the frequency and hierarchy thresholds ϕ and $H^{\max}$ govern filter rule granularity and can be adjusted to match evolving attack traffic patterns.

Reinforcement learning agent We employ deep reinforcement learning to learn the complex, non-linear relationship between HHH parameters, traffic characteristics, generated filter rules and rule effectiveness in a model-free fashion. For this, we train a Deep Q-Network (DQN) on simulated dynamic traffic scenarios to select effective thresholds ϕ and $H^{\max}$. DQNs use deep neural networks to approximate an optimal action-value function Q^* [Mn15], i. e., to learn a policy that maximizes cumulative reward. In our case, the agents objective is to achieve high precision and recall, while minimizing false positive ratio (FPR) and the number of generated filter rules. At query time t, the agent acts by selecting values for ϕ and $H^{\max}$ from its (discrete) action space. The choice is based on the observed mitigation environment state $s^{(t)}$, which comprises indicators for TCAM utilization, filter rule distribution and granularity, as well as filter rule effectiveness. Tab. 1 provides an overview of the (most important) elements of the state and action spaces. The mitigation objectives are conveyed by a reward function $\mathfrak{r} = \mathfrak{r}_\mathrm{p} \cdot \mathfrak{r}_\mathrm{s} \cdot \mathfrak{r}_\mathrm{f} \cdot \mathfrak{r}_\mathrm{r}$, where each partial function $\mathfrak{r}_\mathrm{p}, \mathfrak{r}_\mathrm{s}, \mathfrak{r}_\mathrm{f}, \mathfrak{r}_\mathrm{r}$ constitutes a weighted mapping of precision, sensitivity, FPR,

State $s^{(t)}$	Meaning
$s_{\mathrm{l}}^{(t)} \in \mathbb{N}^{32}$	Number of HHHs detected at different hierarchy levels
$s_{\mathrm{d}}^{(t)} \in [0,1]$	Indicator for distribution and size of filtered IP regions
$s_{\mathrm{p}}^{(t)} \in [0,1]$	Estimated filter precision
$s_{\mathrm{s}}^{(t)} \in [0,1]$	Estimated filter sensitivity
$s_{\mathrm{f}}^{(t)} \in [0,1]$	Estimated FPR
$s_{\mathrm{r}}^{(t)} \in \mathbb{N}$	Number of generated filter rules

Action space	ϕ, $H^{\max}$-values
$A = \bigcup_{1 \le i \le 25, 16 \le j \le 24} (\phi_i, H_j^{\max})$	$\phi = i \cdot 0.01, H_j^{\max} = j$

Partial reward function	Weighting
$\mathfrak{r}_{\mathrm{p}}(x) : [0,1] \to [0,1]$	$x \mapsto x^1$
$\mathfrak{r}_{\mathrm{s}}(x) : [0,1] \to [0,1]$	$x \mapsto x^{1.5}$
$\mathfrak{r}_{\mathrm{f}}(x) : [0,1] \to [0,1]$	$x \mapsto (1-x)^{2.0}$
$\mathfrak{r}_{\mathrm{r}}(x) : \mathbb{N} \to \mathbb{R}$	$x \mapsto 1 - 0.04 \cdot \log_2(x)^{0.2}$

Tab. 1: Excerpt of state, action space, and reward function parameters

and number of generated filter rules (resp.) to scalar values within comparable ranges. By tuning the partial reward functions (see Tab. 1), the agent can learn to realize different trade-offs (e. g., emphasize high precision over small filter rule sets or vice versa).

Downstream IDS In order to provide the DQN agent with feedback on achieved filter precision, sensitivity and FPR (see state $s^{(t)}$ in Tab. 1), we currently apply an oracle IDS that serves to reflect capabilities of a traditional IDS (distinguishing attack and legitimate traffic). Precision, sensitivity, and FPR are estimated from ingress filter statistics (number of removed packets), traffic passing the ingress filter as well as sampled traffic.

Sampled traffic Since early traffic removal prevents downstream systems from monitoring discarded traffic, it hinders their ability to determine traffic distribution and filter rule effectiveness. To address this issue, the ingress filter excludes a fraction of the ingress traffic from filter rule application through sampling. This allows downstream systems to estimate traffic distribution (HHH algorithm) and filter rule effectiveness (IDS) based on sampled traffic. Furthermore, it prevents oscillation of filter rules, since the HHH algorithm would otherwise exclude traffic filtered at the ingress from its estimation of the traffic distribution.

3 Preliminary Results

We evaluate the ability of the RL agent to learn effective HHH parameters based on synthetic, dynamic traffic scenarios. Each scenario constitutes a training episode that models and randomizes the activity of legitimate and attack traffic sources represented by IPv4 addresses over 300 discrete time indices. The activity of legitimate traffic sources is uniformly distributed over time and normally distributed over an address space of size 2^{16}. Attack traffic sources use more dynamic patterns. An episode is divided into four (partially overlapping) phases, each selecting active attack traffic sources from different subnets to distinguish activity of densely clustered sources (e. g., a high number of bots located in an IPv4 subnet) and sparse but widely distributed sources (such as reflectors). Phase one and three use densely clustered normally distributed traffic sources, phase two changes the traffic

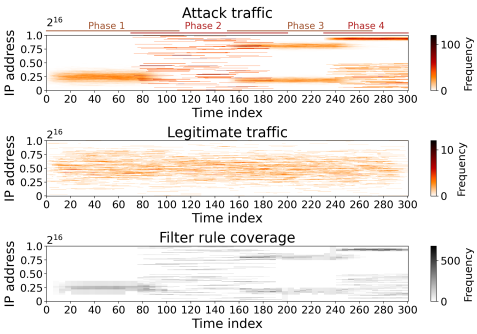


Fig. 2: Snapshot of traffic source activity and generated filter rule coverage during a traffic scenario.

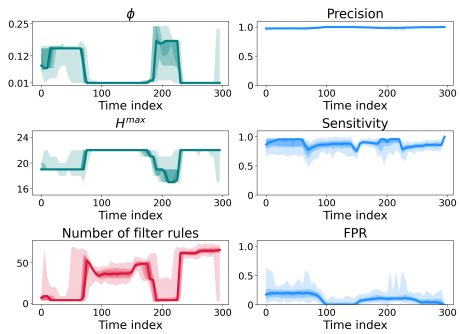


Fig. 3: Parameters (ϕ, $H^{\max}$), precision, sensitivity, FPR, and filter rule count (last 250 episodes).

pattern to few high-intensity sources uniformly distributed over the entire address space, while phase four combines densely clustered sources with sparsely distributed high-intensity sources. During a training episode, an RL agent adapts frequency and hierarchy thresholds (ϕ, $H^{\max}$) every five time indices, while the entire training process consists of 80000 adaptation steps. Fig. 2 provides an example snapshot of a synthesized scenario and corresponding filter rule coverage from the end of RL training. The distribution of choices for frequency and hierarchy thresholds, number of generated filter rules, precision, sensitivity, and FPR during the last 250 randomized training episodes is outlined in Fig. 3 (min-max, 10%-90%, 25%-75%, 40%-60% quantiles, and median depicted with increasing shading).

In phase one, attack traffic sources are clustered in a small region in the lower IP address range. Consequently, the agent adapts thresholds ϕ and $H^{\max}$ to generate fewer coarse-grained rules that are sufficient to cover the corresponding address range (see Fig. 2 and Fig. 3). After transitioning to fewer, widely distributed high-intensity attack traffic sources (phase two), the agent emphasizes fine-grained rules to account for the sparse distribution. It reduces frequency threshold ϕ and increases hierarchy threshold $H^{\max}$ to maintain high precision and sensitivity as well as low FPR (time index 100-150). Hence, the agent manages to distinguish between distributions and adjusts filter rules to match attack traffic patterns.

During phase three, ϕ and $H^{\max}$ are again chosen to emphasize fewer rules that are sufficient to capture the two coherent regions with active attack traffic sources (time index 190-225). Finally, the agent chooses low ϕ and high $H^{\max}$ in phase four to apply fine-grained filter rules to the sparsely distributed attack traffic sources (lower half of the address space) as well as the densely clustered sources (upper address space). By emphasizing fine-granular rules in this hybrid phase, the agent maintains low FPR at the cost of more rules. The same applies when transitioning between different phases. Accepting more filter rules in these cases is in-line with the distribution of attack traffic sources and the mitigation goals conveyed by the reward function, since coarse-grained filter rules would necessarily have a strong negative impact on FPR during these periods and, thus, yield lower overall reward.

Acknowledgment

This work was supported by funding of the Helmholtz Association (HGF) through the Competence Center for Applied Security Technology (KASTEL) (POF structure 46.23.01: Methods for Engineering Secure Systems).

Bibliography

[Be20] Ben Basat, Ran; Chen, Xiaoqi; Einziger, Gil; Rottenstreich, Ori: Designing Heavy-Hitter Detection Algorithms for Programmable Switches. IEEE/ACM Transactions on Networking, 28(3):1172–1185, 2020.

[Li21] Liu, Zaoxing; Namkung, Hun; Nikolaidis, Georgios; Lee, Jeongkeun; Kim, Changhoon; Jin, Xin; Braverman, Vladimir; Yu, Minlan; Sekar, Vyas: Jaqen: A High-Performance Switch-Native Approach for Detecting and Mitigating Volumetric DDoS Attacks with Programmable Switches. In: USENIX Security Symposium. 2021.

[MK15] Malialis, Kleanthis; Kudenko, Daniel: Distributed Response to Network Intrusions Using Multiagent Reinforcement Learning. Eng. Appl. Artif. Intell., 41(C):270–284, May 2015.

[Mn15] Mnih, Volodymyr; Kavukcuoglu, Koray; Silver, David; Rusu, Andrei A; Veness, Joel; Bellemare, Marc G; Graves, Alex; Riedmiller, Martin; Fidjeland, Andreas K; Ostrovski, Georg et al.: Human-level control through deep reinforcement learning. nature, 518(7540):529–533, 2015.

[PAM17] Popescu, Diana Andreea; Antichi, Gianni; Moore, Andrew W.: Enabling Fast Hierarchical Heavy Hitter Detection Using Programmable Data Planes. In: Proceedings of the Symposium on SDN Research. SOSR '17, Association for Computing Machinery, New York, NY, USA, p. 191–192, 2017.

[Si17] Sivaraman, Vibhaalakshmi; Narayana, Srinivas; Rottenstreich, Ori; Muthukrishnan, S.; Rexford, Jennifer: Heavy-Hitter Detection Entirely in the Data Plane. In: Proceedings of the Symposium on SDN Research. SOSR '17, Association for Computing Machinery, New York, NY, USA, p. 164–176, 2017.

[SRP20] Simpson, Kyle A.; Rogers, Simon; Pezaros, Dimitrios P.: Per-Host DDoS Mitigation by Direct-Control Reinforcement Learning. IEEE Transactions on Network and Service Management, 17(1):103–117, 2020.

[Zh20] Zhang, Menghao; Li, G.; Wang, Shicheng; Liu, Chang; Chen, Ang; Hu, Hongxin; Gu, Guofei; Li, Qi; Xu, Mingwei; Wu, Jianping: Poseidon: Mitigating Volumetric DDoS Attacks with Programmable Switches. In: NDSS. 2020.

[Zh21] Zhang, Yinda; Liu, Zaoxing; Wang, Ruixin; Yang, Tong; Li, Jizhou; Miao, Ruijie; Liu, Peng; Zhang, Ruwen; Jiang, Junchen: CocoSketch: High-Performance Sketch-Based Measurement over Arbitrary Partial Key Query. In: Proceedings of the 2021 ACM SIGCOMM 2021 Conference. SIGCOMM '21, Association for Computing Machinery, New York, NY, USA, p. 207–222, 2021.

Differential Testing: How to find differences between programs that mostly behave identically?

Jonas Möller[1]

Abstract: Differences between programs based on the same specification might lead to vulnerabilities that can not be detected by conventional testing. Differential testing is able to find these discrepancies by executing multiple programs on the same input and comparing their output. In this work, we discuss the fundamentals of differential testing and outline a general scheme for differential testing methods which is used to categorize and analyze current research approaches. Based on this, we formulate our own research questions which focus on how machine learning can aid differential testing.

Keywords: Differential Testing; Fuzzing

1 Introduction

An aspect that is often overlooked in software engineering is that although we expect systems of the same kind to behave identically they might exhibit minor discrepancies. Even when components adhere to the same specification, irregularities like underspecified functionality, misconceptions about API usage, or programming errors can lead to slightly different outputs for the same input. In the case of underspecified functionality, the discrepancies between programs are not defects per se, since each one implements the specification correctly; thus, they can not be caught by conventional test suites or even verification efforts. If these components are used in conjunction, their interplay might introduce vulnerabilities stemming from a different handling of the same data.

An example for such a vulnerability can be crafted using HTTP parameter pollution [Ba11]: If an HTTP request contains multiple values for the same parameter, a web application firewall and a server behind this firewall might process different values for the same parameter; this can be used to pass a benign value to the firewall and a malicious one to the server. Another example is the evasion of malware detectors. These need to process incoming files to inspect them (e.g. extract a zip archive) and thus have to recreate parts of the target application logic. Even for a single file type the detector has to recreate the functionality of multiple competing implementations which result in abusable discrepancies between detector and user application [JS12]. Other examples include file type inference [BCS09], SSL/TLS certificate verification [Br14; CS15] and HTTP request smuggling [Ja21]. Thus, finding differences between programs is not only relevant for interoperability,

[1] TU Braunschweig, Institut für Systemsicherheit, Mühlenpfordtstraße 23, 38106 Braunschweig, Deutschland, jo.moeller@tu-braunschweig.de

doi:10.18420/sicherheit2022_21

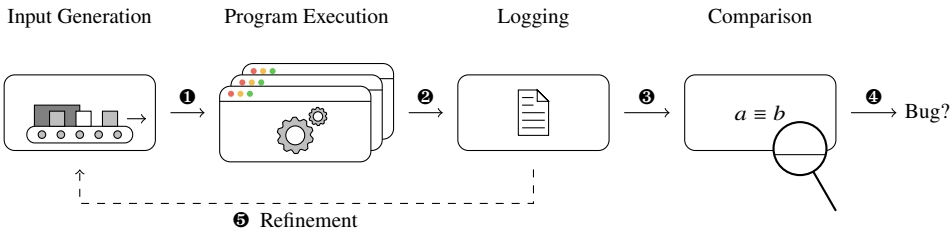


Fig. 1: The differential testing pipeline: Input is generated (❶), passed to multiple programs (❷), run time information is collected (❸) and the output is checked for equivalence (❹). If the input generation is guided, the run time information is used for refinement (❺).

but is also a security-critical task. This is a hard task, since, in essence, we want to verify the equivalence of two programs, which is an undecidable problem.

One approach to systematically check programs for their equivalence is *differential testing*. By feeding the same inputs to multiple programs, we can use them as cross-referencing oracles, compare their outputs and detect subtle differences in their implementation. Although this does not guarantee program equivalence, it increases the trust in the correctness of programs—similar to conventional software testing.

2 Differential Testing

The basic idea of differential testing is pretty simple: If we feed the same input to multiple programs, the programs should return equivalent output. Differential testing approaches can usually be described by the pipeline shown in Figure 1: In each iteration, an input is generated and passed to the programs under test. Then, the outputs of the programs are logged and the results are compared to find differences. In the following, we describe the individual steps of the pipeline in more detail and—to give an overview of the current research literature—we discuss relevant aspects of differential testing papers for each step.

2.1 Input Generation

In the first step, the inputs for the programs are generated. Depending on the type of input expected by the program and the availability of source code, different input generation strategies can be used. In general, a good strategy creates inputs that execute unexplored program paths where uncommon edge-cases make it more likely to find differences. There are three aspects that characterize an input generation strategy: First, a strategy can be either guided or unguided. Guided methods use feedback from the program evaluations to refine the next inputs, whereas in unguided methods each iteration is independent of the previous iterations. For guided methods, the programs can be instrumented during compilation, if

the source code is available, to gather run time information like coverage measurements. Second, the inputs exhibit various degrees of structure: The simplest inputs are based on a random sequence of bytes while more sophisticated methods can employ a context-free grammar or modify an initial corpus. While unstructured, semi-random input can be easily generated, semantically sound input reaches deeper into program logic. Lastly, the input generation ranges from generative to transformative. The former creates inputs from a set of rules (e.g. a context-free grammar) while the latter applies transformations to a set of known inputs; hybrid approaches can combine both types. Since differential testing is based on fuzzing, the large body of research in this direction can be leveraged to improve discrepancy detection.

In the literature, Brubaker et al. [Br14] use an unguided, transformative approach by combining and randomly altering SSL/TLS certificates. This approach is improved upon by Chen; Su [CS15] who use coverage-guided Markov chain Monte Carlo (MCMC) methods which randomly select transformations. In the domain of testing Java Virtual Machine implementations, MCMC-based sampling of input transformations is also used by Chen et al. [Ch16]. For a generative approach, McKeeman [Mc98] discusses and implements seven different input tiers ranging from "Random sequence of ASCII characters" to "Model-conforming C programs" to differentially test C compilers at a greater depth. This work is improved upon in *CSmith* which adds further restrictions to dismiss all programs that include any undefined behaviour [Ya11]. Lastly, Holler et al. [HHZ12] and Jabiyev et al. [Ja21] combine generative with transformative aspects for unguided fuzzing of language interpreters and HTTP servers respectively.

2.2 Program Execution

Executing the programs is largely an engineering problem: Each instance is given the input for the current iteration and is monitored to collect run time information as described in the logging phase. There exist two obstacles that need to be addressed when setting up a differential testing pipeline: First, passing input to a program is not always straightforward, so it is common to create a test harness, which acts as an adapter between the fuzzer and the fuzzing target and handles additional issues (e.g. if an application uses a GUI). Second, the functionality of programs may be influenced by external inputs like environment variables, configuration files or command line arguments. These either need to be semantically equivalent for each program under test or need to be treated as part of the input.

A rather sophisticated program execution stage is employed by Jabiyev et al. [Ja21] who construct a controlled network environment of server-, proxy- and CDN-technologies. In this network, HTTP requests are used as inputs for the instances and HTTP messages between servers are monitored. This setup is used to find differences between server implementations which can be leveraged for HTTP request smuggling.

2.3 Logging

During its execution, a program is monitored and its results are logged for further analysis. The logging phase consists of two parts: First, the output of the programs is logged and used for the comparison phase. This is not trivial since the output of the programs should adequately reflect their functionality, but still needs to be comparable. Second, if the input generation uses a guided input generation approach, run time information for the refinement phase is collected. This can include coverage, standard output, program exit status or more detailed results from program analysis. Which analyses are feasible depends on the scope of the testing, e.g. if the source code of the program is available, the code can be instrumented to achieve more fine-grained run time information. Although more detailed analysis of the program leads to better refinement, it also comes at a run time cost. This trade-off between throughput and analysis depth is a standard problem of fuzzing-based systems [Ma21].

In the literature, a wide variety of output choices is used, such as: discrete output states [Br14; Ch16; CS15], program return values [Pe17], standard output [Pe17], simulation states [Hu21] and—for comparing compilers—the program state of the compiled programs at the end of their execution [Ya11]. Although Srivastava et al. [Sr11] employ static analysis instead of differential testing, their logging approach is worth mentioning: To find differences between Java Class Library implementations, they create a set of security-sensitive events and security checks for each function, which are compared between the implementations.

2.4 Comparison

In the comparison phase, the normalized outputs are compared to each other. If any outputs between programs differ, we found a discrepancy and thus a potential bug. Note the semantic gap here: A discrepancy between program outputs does not necessarily constitute a bug or even a vulnerability. This step can result in a lot of low-grade differences that can be regarded as noise. One example we found in our experiments was when comparing JSON parsers some parsers directly stored the substrings containing floating point numbers while others converted the strings to the data type “float”. During the casting, some information can be lost due to rounding and thus minor differences in the output can be observed. This problem is not new in the fuzzing community and different approaches exist that try to deduplicate crashes [Ma21], distinguish benign and malicious crashes [Bl20] and cluster crashes based on a root-cause analysis [Ji21]. For differential testing, Chen et al. [Ch16] reduce a discrepancy-triggering input to a simplified form.

2.5 Refinement

If the input generation is guided, the last step uses the run time information of the programs to refine the inputs. For fuzzing-based methods, the coverage information is used to guide the inputs towards unexplored paths. An interesting example is given by Petsios et al. [Pe17] who propose a coverage-guided domain-independent framework where the coverage metric named δ-diversity measures asymmetries between program execution traces.

3 Research Questions

Input generation frameworks like CSmith [Ya11], LangFuzz [HHZ12] and T-Reqs [Ja21] try to accommodate generic fuzzing applications and thus work in an unguided fashion to not require the availability of source code. However, unguided methods discard valuable coverage information that could be collected to refine the input generation and find discrepancies more efficiently. Our first research question is therefore: Can unguided, structured input generation methods be improved with coverage-guided refinement? And further: Can δ-diversity [Pe17] be used as a coverage metric to improve the results?

Another important aspect of differential testing is the comparison of program outputs. Currently, programs need to generate normalized output for them to be comparable. We want to explore whether discrepancies between programs can be found by just examining their internal states. Our intuition is that, for some programs, inputs can be grouped into classes that trigger the same functionality and arrive at similar states. If statistical patterns emerge for internal states during program executions, they can be analyzed using machine learning techniques. Similar to how sentences can be summarized by a thought vector in natural language processing, the state of a program might be encoded using the execution trace. This vector representation of a program state could be used in the δ-diversity metric to learn a model of normality across programs. Deviations, i.e. anomalous behaviour, could then indicate a program discrepancy. Another similar approach is to learn to predict the program states directly from the input. If machine learning models can predict program behaviour, we can examine whether it is possible to translate between internal states of multiple programs. This could be used to search for discrepancies by predicting the state for a program, translating it to another program and then comparing the expected behaviour with the actual behaviour.

The underlying assumption in these approaches is the emergence of statistical patterns for program behaviour. Thus, our next research question is: Can statistical patterns be found in program behaviour? And lastly: Can machine learning techniques use these patterns to find discrepancies?

References

[Ba11] Balduzzi, M.; Gimenez, C. T.; Balzarotti, D.; Kirda, E.: Automated Discovery of Parameter Pollution Vulnerabilities in Web Applications. In: NDSS. Citeseer, 2011.

[BCS09] Barth, A.; Caballero, J.; Song, D.: Secure content sniffing for web browsers, or how to stop papers from reviewing themselves. In: 2009 30th IEEE Symposium on Security and Privacy. IEEE, 2009.

[Bl20] Blazytko, T.; Schlögel, M.; Aschermann, C.; Abbasi, A.; Frank, J.; Wörner, S.; Holz, T.: {AURORA}: Statistical Crash Analysis for Automated Root Cause Explanation. In: USENIX Security Symposium. 2020.

[Br14] Brubaker, C.; Jana, S.; Ray, B.; Khurshid, S.; Shmatikov, V.: Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations. In: IEEE Symposium on Security and Privacy. IEEE, 2014.

[Ch16] Chen, Y.; Su, T.; Sun, C.; Su, Z.; Zhao, J.: Coverage-directed differential testing of JVM implementations. In: proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation. 2016.

[CS15] Chen, Y.; Su, Z.: Guided differential testing of certificate validation in SSL/TLS implementations. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. 2015.

[HHZ12] Holler, C.; Herzig, K.; Zeller, A.: Fuzzing with code fragments. In: USENIX Security Symposium. 2012.

[Hu21] Hur, J.; Song, S.; Kwon, D.; Baek, E.; Kim, J.; Lee, B.: DifuzzRTL: Differential Fuzz Testing to Find CPU Bugs. In: IEEE Symposium on Security and Privacy. IEEE, 2021.

[Ja21] Jabiyev, B.; Sprecher, S.; Onarlioglu, K.; Kirda, E.: T-Reqs: HTTP Request Smuggling with Differential Fuzzing. In: ACM Conference on Computer and Communications Security. 2021.

[Ji21] Jiang, Z.; Jiang, X.; Hazimeh, A.; Tang, C.; Zhang, C.; Payer, M.: Igor: Crash Deduplication Through Root-Cause Clustering. In: Proc. of the ACM Conference on Computer and Communications Security. 2021.

[JS12] Jana, S.; Shmatikov, V.: Abusing file processing in malware detectors for fun and profit. In: IEEE Symposium on Security and Privacy. IEEE, 2012.

[Ma21] Manès, V. J. M.; Han, H.; Han, C.; Cha, S. K.; Egele, M.; Schwartz, E. J.; Woo, M.: The Art, Science, and Engineering of Fuzzing: A Survey. IEEE Trans. Software Eng. 47/11, 2021.

[Mc98] McKeeman, W. M.: Differential testing for software. Digital Technical Journal 10/1, 1998.

[Pe17] Petsios, T.; Tang, A.; Stolfo, S.; Keromytis, A. D.; Jana, S.: Nezha: Efficient domain-independent differential testing. In: 2017 IEEE Symposium on security and privacy. IEEE, 2017.

[Sr11] Srivastava, V.; Bond, M. D.; McKinley, K. S.; Shmatikov, V.: A security policy oracle: Detecting security holes using multiple API implementations. ACM SIGPLAN Notices 46/6, 2011.

[Ya11] Yang, X.; Chen, Y.; Eide, E.; Regehr, J.: Finding and understanding bugs in C compilers. In: Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation. 2011.

Autorenverzeichnis

B
Balack, J., 101
Bayreuther, S., 113
Beckert, B., 213
Boes, F., 35
Breig, J., 183
Brün, M. de, 209
Budurushi, J., 213
Burkert, C., 101

D
Demir, N., 19
Doerner, D., 83

F
Federrath, H., 101
Fraune, B., 231
Freiling, F.C., 49
Frosch, T., 143

G
Garske, V., 129
Gierlings, M., 143
Große-Kampmann, M., 193
Gruber, J., 49
Gruber, M., 193
Grundmann, L., 143
Grunwald, A., 213
Güdemann, M., 67

H
Hartenstein, H., 113
Heer, T., 159
Hennig, A., 225
Heseding, H., 237
Hinrichs, T., 219
Höfig, C., 193

I
Ising, F., 143

J
Jacob, F., 113

K
Keller, J., 173
Kempf, L., 35
Klemm, A., 143
Koetter, P., 209
Kölligan, A., 209
Krimmer, R., 213
Kulyk, O., 213
Küsters, R., 213

L
Lochmann, F., 205

M
Mayer, A., 213
Mayer, N., 173
Mechler, J., 83
Meier, M., 35
Möller, J., 243
Müller, R., 159
Müller-Quade, J., 83, 213

N
Neumann, S., 213
Noack, A., 129
Noss, D., 143

O
Ohm, M., 35

P
Pohlmann, N., 19, 193

R
Rijswijk-Deij, R. van, 209
Ruppert, J., 159

S
Saatjohann, C., 143
Schimmler, S., 143
Schinzel, S., 143
Schmerl, S., 205
Strotmann, C., 209

T
Theis, D., 19

U
Urban, T., 19, 193

V
Volkamer, M., 213

W
Wendzel, S., 173
Westhoff, D., 183
Will, K., 159
Wüsteney, L., 159