Gesellschaft für Informatik (GI)

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

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

The volumes are published in German or English

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

ISSN 1617-5468
ISBN 978-3-88579-208-6

The proceedings of the IMF 2007 include international scientific contributions of an annual conference of the Special Interest Group on Security Intrusion Detection and Response of the Gesellschaft für Informatik (GI).The conference took place in Stuttgart, September 11 -13, 2007.Within three days the advances of research in IT-Incident Management and IT-Forensics with focus on theory and applications where presented and discussed by researchers and professionals from industry.

S. Frings, O. Göbel, D. Günther, H. G. Hase, J. Nedon, D. Schadt, A. Brömme (Eds.):
IMF 2007 – IT-Incident Management & IT-Forensics

114

GI-Edition

Lecture Notes in Informatics

Sandra Frings, Oliver Göbel, Detlef Günther, Hardo G. Hase, Jens Nedon, Dirk Schadt, Arslan Brömme (Eds.)

IMF 2007 IT-Incident Management & IT-Forensics

Proceedings of the 3rd International Conference on IT-Incident Management & IT-Forensics

September 11 – 13, 2007, Stuttgart, Germany

Proceedings

Sandra Frings, Oliver Göbel, Detlef Günther,
Hardo G. Hase, Jens Nedon, Dirk Schadt,
Arslan Brömme (Eds.)

IMF 2007

IT-Incident Management & IT-Forensics

**Proceedings of the 3rd International Conference on
IT-Incident Management & IT-Forensics
September 11 – 13, 2007
Stuttgart, Germany**

Gesellschaft für Informatik e.V. (GI)

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

Volume P-114

ISBN 978-3-88579-208-6
ISSN 1617-5468

Volume Editors

Sandra Frings
Fraunhofer Institut für Arbeitwirtschaft und Organisation (IAO),
*Nobelstrasse 12, D-70569 Stuttgart, Email: sandra.frings@iao.fraunhofer.*de
Oliver Göbel
Stabsstelle DV-Sicherheit der Universität Stuttgart (RUS-CERT)
Breitscheidstraße 2, D-70197 Stuttgart, Email: goebel@cert.unistuttgart.de
Detlef Günther
CERT-VW, Volkswagen AG
D-38436 Wolfsburg, Email: detlef.guenther@volkswagen.de
Hardo G. Hase
IT-Security Expert,
Postach 10 50, D-66441 Bexbach, Email: hardo.hase@hase-bexbach.de
Jens Nedon
ConSecur GmbH – security & consulting,
Schulze-Delitzsch-Strasse 2, D-49716 Meppen, Email: nedon@consecur.de
Dirk Schadt
SPOT Consulting
Zehntweg 21, D-52078 Aachen, Email: dirk.schadt@spot.net
Arslan Brömme
ConSecur GmbH – security & consulting
Schulze-Delitzsch-Strasse 2, D-49716 Meppen, Email: broemme@consecur.de

printed by Köllen Druck+Verlag GmbH, Bonn

Preface

Information technology has become crucial to almost every part of society. IT infrastructures are critical to the world-wide economy, the financial sector, the health sector, the government's administration, the military, and the educational sector. Due to its importance the disruption or loss of IT capabilities results in a massive reduction of operability.

Hence, IT security is continously gaining importance and has become technically essential to IT infrastructures.

Although security usually gets integrated into the design process of IT systems nowadays, the process of maintaining security in IT infrastructure operation still lacks the appropriate attendance in most cases.

Especially the capability to manage and respond to IT security incidents and their forensic analysis is established in the rarest cases. The quickly rising number of security incidents worldwide makes the implementation of incident management capabilities, targeting the mitigation of immediate consequences to the own infrastructure, essential.

Also, the need of subsequent forensic analysis of selected cases to gather evidence on the incident's details and work up the information for law suits or to avert unwarranted liability claims of aggrieved third parties is constantly growing.

In order to advance the fields of IT-Incident Management and IT-Forensics the special interest group *Security - Intrusion Detection and Response* (SIDAR) of the German Informatics Society (GI) organises the annual *International Conference on IT-Incident Management and IT-Forensics* (IMF), creating a platform for experts from throughout the world, to discuss state of the art in these areas. IMF promotes collaboration and exchange of ideas between industry, academia, law-enforcement and other government bodies.

IMF 2007 is supported with keynotes by the Chairman of the International Federation for Information Processing (IFIP) and the Vice President Security Solutions Group of the Bell Laboratories (USA).

The organising committee would like to thank all persons who helped in realising the conference, especially the authors whose papers and presentations make the essence of the conference, the members of the program committee who reviewed and evaluated the papers submitted and whose professional competence ensures the scientific quality of the program, as well as the sponsors who supported the conference.

Oliver Göbel, General Chair

General Chair

Oliver Göbel
RUS-CERT, Germany

Program Chair

Sandra Frings
Fraunhofer IAO, Germany

Co-Chairs

Detlef Günther, *CERT-VW, Volkswagen AG, Germany*
Hardo G. Hase, *IT-Security Expert, Germany*
Jens Nedon, *ConSecur GmbH – security & consulting, Germany*
Dirk Schadt, *SPOT Consulting, Germany*
Arslan Brömme, *ConSecur GmbH – security & consulting, Germany*

Program Committee

Susan Brenner, University of Dayton - School of Law, USA
Klaus Brunnstein, University of Hamburg, Germany
Brian Carrier, Basis Technology, USA
Jack Cole, US Army Research Laboratory, USA
Andrew Cormack, UKERNA, UK
Ralf Dörrie, Telekom-CERT, Germany
Sandra Frings, Fraunhofer IAO, Germany
Oliver Göbel, RUS-CERT, Germany
Dieter Gollmann, TU Hamburg-Harburg, Germany
Detlef Günther, CERT-VW, Volkswagen AG, Germany
Bernhard Hämmerli, ACRIS GmbH, Switzerland
Hardo G. Hase, IT-Security Expert, Germany
Alexander Herrigel, Secude, Switzerland
Klaus-Peter Kossakowski, DFN-CERT, Germany
Thorsten Lieb, Avocado Rechtsanwälte, Germany
Jim Lyle, NIST CFTT, USA
Robert A. Martin, MITRE Corporation, USA
Neil Mitchison, European Commission
Jens Nedon, Consecur GmbH, Germany
Bernhard Otupal, Interpol Jason Rafail CERT/CC, USA
Dirk Schadt, SPOT Consulting, Germany
Carlos Solari, Bell Laboratories, USA
Marco Thorbrügge, ENISA, EU
Stephen Wolthusen, Gjovik University College, Norway
Steven Wood, Alste-Tech GmbH, Germany

Hosts

Special Interest Group on Security Intrusion Detection and Response (SIDAR) of the Gesellschaft für Informatik (GI) e.V.
http://www.gi-fg-sidar.de

Stabsstelle DV-Sicherheit der Universität Stuttgart (RUS-CERT)
http://cert.uni-stuttgart.de/

Fraunhofer Institut für Arbeitwirtschaft und Organisation (IAO)
http://www.iao.fraunhofer.de/

Cooperation Partners

Institute of Electrical and Electronics Engineers, Inc.
http://www.ieee.org

IEEE Computer Society
http://www.computer.org

Sponsors

ConSecur GmbH – security & consulting
http://www.consecur.de

PRESECURE Consulting GmbH
http://www.pre-secure.de

Supporter

European Network and Information Security Agency
http://enisa.europa.eu/

Table of Contents

IMF 2007 Keynotes **11**

About the Role of IT Security in The Information Society
Klaus Brunnstein 13

The Security Landscape in a Converged IP World
Carlos C. Solari 15

IMF 2007 – Tuesday, September 11th, 2007 **17**

A Common Process Model for Incident Response and Computer Forensics
Felix C. Freiling and Bastian Schwittay 19

IT Incident Management and Structured Documentation
Sandra Frings 41

Proposal Of A System For Computer-Based Case And Evidence Management
Fritjof Haft, Pascal Hassenpflug, and Hans Lecker 51

Information-Sharing System for Vulnerability Information Dissemination in Large-Scale Organization
Tohru Sato and Jumpei Watase 67

IMF 2007 – Wednesday, September 12th, 2007 **81**

A Case Study on Constructing a Security Event Management System
Vijay K. Gurbani, Debra L. Cook, Lawrence E. Menten, and Thomas B. Reddington 83

Taxonomy of Anti-Computer Forensics Threats
Joseph C. Sremack and Alexandre V. Antonov 103

Testing Forensic Hash Tools on Sparse Files
Harish Daiya, Maximilian Dornseif, and Felix C. Freiling 113

Towards Reliable Rootkit Detection in Live Response
Felix C. Freiling and Bastian Schwittay 125

IMF 2007 Workshops – Thursday, September 13th , 2007 **145**

IMF 2007 Keynotes

Keynote:

About the Role of IT Security in the Information Society

Klaus Brunnstein

Faculty for Informatics, University of Hamburg, Germany
International Federation for Information Processing (IFIP), (Past) President
brunnstein@informatik.uni-hamburg.de

Paper presented at the **3rd International Conference on IT-Incident Management & IT-Forensics (IMF 2007) Stuttgart, Germany (September 11 - 13, 2007)**

Abstract: The development of economies, national and international organisations as well as of many aspects of social and individual lives are growingly determined by developments of Information and Communication Technologies (ICTs). Consequently, institutions, governments and many scientists refer to the actual phase of human history as shaping an **"Information Society" (IS)** (or "Knowledge Society", as named by UNESCO and other UN organisations). Developments are much faster than in the **"Industrial Society",** but there are surprising similarities which may best be described in analogy to the theory of **economic cycles** of the Industrial Society, developed by Schumpeter and Kondratieff. Locating the start of the Information Society about 1940 (with first computers such as Zuse Z3 and ENIAC), the **first "IS cycle"** – comparable to steam engines following James Watt´s invention, is determined by single computers with initially very low capacities but fast improvement in speed and storage together with reduction in space (micro-miniaturisation). Comparable to the development of traffic networks (railways etc) in the Industrial Society, single computers are interconnected in the **second IS cycle** to form networks of growing complexity, with software technologies developing autonomous agents to support system and application work.

As experienced also in the Industrial Society, **InSecurity** governs the first and second phase of the Information Society, and networks (esp. The Internet) and computer systems (whether clients or servers) are as "Insecure and Unsafe at any speed" (quoting Ralph Nader) as industrial products in related phases. Besides many beneficial effects of IT for enterprises, organisations and individual, many incidents have contributed to significant loss and damage. Even pubertarian boys succeed easily in attacking important IT systems and produce significant damage to systems, users and customers.

Causes for IT InSecurity are manifold, ranging from paradigmatic to practical aspects. Among several reasons, including basic design of technologies, IT experts do not care sufficiently for the consequences of their design, products and usage. As technical improvements of contemporary IT systems will - for a foreseeable period - only partly help to overcome basic causes of InSecurity, education of IT engineers to safer and more secure design and implementation of their products may help to reduce IT risks. While some professional organisations such have suggested some rules for ethical behaviour of their members, contemporary curricula fail to include Ethics into the education of IT experts esp. including System and Software Engineering. Consequently, reduction of InSecurity must address all related aspects.

"Good Practice" becomes more important with growing dependency of enterprises, organisations, governments and individuals on vulnerable, interconnected IT systems, IT (through improved methods and protocols) and legal experts (also through internationally accepted rules) must find **ways to reduce contemporary InSecurity**.

Keynote:

The Security Landscape in a Converged IP World

Carlos C. Solari

Bell Laboratories

Abstract: The move to an all Internet Protocol (IP) convergence of media is potentially one of the moments where a technology has historical impact; meaning a technology that changes our social structures in beneficial ways.

Technology can also be turned against society. The rise of the Internet is well suited to this discussion. We should not have been surprised that a tool meant to connect us for the good of society could be targeted to take advantage of the vulnerable as with phishing scams and identity theft.

But we were surprised. Securing this technology was someone else's problem: the individual consumer, business, and government. So what about the security, why is it more important than ever, and how can we take a better approach than the approach taken to date of reactive, after-the-fact security patching?

It matters now more than ever that we ask these questions and implement the right answers as the move to convergence to all- IP means that our dependency to IP-based systems will be complete. The future is near when the data of what television show we are watching, when we watched it, for how long we watched is collected to provide the consumer with "personalized marketing". This future day is one consumer's "cool technology" and another's privacy "nightmare" if that data is not protected and if we don't provide the consumer with the choices to opt in or out of that personalized marketing.

The time to develop that level of security and privacy protection is now when the new IP television system is still in design - not later, when the retrofit can only serve to remediate a problem with a patchwork of temporary fixes. Clearly, no single approach is going to change the dynamics of the market to change its ways and design security in before the delivery of the service.

The presenter will discuss key elements needed to set a strategy for security in the new landscape of convergence.

IMF 2007

Tuesday, September 11th, 2007

A Common Process Model for Incident Response and Computer Forensics

Felix C. Freiling
Laboratory for Dependable Distributed Systems
University of Mannheim, Germany
freiling@informatik.uni-mannheim.de

Bastian Schwittay
Symantec (Deutschland) GmbH, Germany
Bastian_Schwittay@symantec.com

Abstract: Incident Response and Computer Forensics are two areas with similar goals but distinct process models. While in both cases the goal is to investigate computer security incidents and contain their effects, Incident Response focusses more on restoration of normal service and Computer Forensics on the provision of evidence that can be used in a court of law. In this paper we present a common model for both Incident Response and Computer Forensics processes which combines their advantages in a flexible way: It allows for a management oriented approach in digital investigations while retaining the possibility of a rigorous forensics investigation.

1 Introduction

In the field of computer security, new threats such as bot networks and modular malicious code have emerged, while well-known attack tools such as rootkits and trojans remain dangerous by constantly using new and improved techniques. Since not all of these attacks can be prevented, *Incident Response* (IR) has become an important component of IT security management, because it provides procedures for detecting and containing computer security incidents to restore normal computing services as quickly as possible. Many attacks nowadays are already motivated by profit, attempting to achieve financial gain by identity theft, extortion, fraud, or theft of confidential information. With criminal acts becoming more common in computer related incidents, the need for valid evidence for these crimes also rises. The field of *Computer Forensics* (CF) aims at providing such evidence.

IR and CF are two highly related topics, and it is questionable whether a strict separation makes sense at all. Certainly, both investigate a large number of different computer security incidents or offenses, sometimes use identical tools and methods and also share some of the key phases of the investigation, although they may set different priorities. However, the specific view of the investigative process in IR and CF respectively may sometimes be too narrow to achieve optimal results in either case. On the one hand, a CF investigation may lack the proper management that is enforced in an IR investigation, where the investigative efforts are coordinated with all parts of an organization, e.g. legal counsel, Human Resources and business executives. Without these, the "bigger picture" of an incident might not be seen. On the other hand, the scientific standards of a CF investigation can

give benefit to an IR procedure, as it promotes objectivity throughout the whole process and a precise and well-documented analysis.

In this paper, we take a unifying view of IR and CF. We take a step back from the two process models of IR and CF and develop a common process model for Incident Response and Computer Forensics which combines the advantages of both models in a flexible way: It allows for a management oriented approach in digital investigations while retaining the possibility of a rigorous CF investigation. At first sight one can argue that this approach seems to be "proposing the obvious" or that no experimental evidence is presented to show the superiority of the new approach. However, we feel that our Common Model (1) structures the phases of a digital investigation in a more logical way, (2) can help unify terminology and methods from different communities, and (3) offers further insight into the strenghts and weaknesses of different approaches. For example, because the large effort to conduct a full-scale forensic investigation may not be necessary in every case, the Common Model will identify certain parameters that determine the extent of an investigation. These parameters are *attacker threat level* and *potential damage* of an incident. We show how to evaluate them in order to develop a sensible response strategy.

The paper is structured as follows: We begin with a brief survey of the processes of Incident Response and Computer Forensics in Sect. 2. In Sect. 3, a Common Model for Incident Response and Computer Forensics is developed, which integrates forensic analysis practices into an Incident Response procedure. We argue in Sect. 4 that the Common Model is flexible enough to handle both IR and CF by providing the necessary criteria to parametrise the Common Model in a flexible way.

2 Background and Definitions

This section briefly recapitulates some important definitions and then gives an overview over IR and CF based on the standard literature by Mandia et al. [5] and Casey [3].

2.1 Definitions

A *computer security incident* can be defined as "a violation or imminent threat of violation of computer security policies, acceptable use policies, or standard security practices" [4]. Incident Response deals with computer security incidents in a well-defined manner to detect incidents, minimize the damage done to the organization, fix the weaknesses that were exploited and return to normal operations.

Computer Forensics is a scientific discipline which is concerned with the collection, analysis and interpretation of digital data connected to a computer security incident; it is sometimes also called *digital forensics*. Since any device, data or other resource subject to a forensic examination may be subsequently used as evidence in a court of law, Computer Forensics puts special emphasis on the correct treatment of potential evidence to prevent it from being altered or tampered with. Because of the same reason, any technique or tool

used during an investigation has to meet strict standards, and any conclusions drawn must be scientifically reasonable. It is important to note that an Incident Response procedure can sometimes *include* a full forensic investigation.

2.2 Incident Response

As mentioned in the introduction, Incident Reponse is an organization's reaction to unlawful or unacceptable actions involving a computer or network component. Instead of being caught unprepared and starting a chaotic and possibly devastating response, a systematic and well-organized approach should be used to react. Therefore incidents are usually handled by a so-called *Computer Security Incident Response Team*, or *CSIRT*, which is comprised of personnel with different qualifications that are needed during the response process, in particular people with legal and technical expertise. The CSIRT will then coordinate the appropriate response to an incident, effectively providing a *Computer Security Incident Response Capability (CSIRC)*.

Some of the key benefits that an organized CSIRC offers are for example a confirmation whether an incident actually occured or quick detection, containment and recovery. Achieving all this can seem like an overwhelming task, especially when one considers that often incidents put a lot of pressure on the team responsible for the resolution of the problem, which could for example threaten the very existence of a company. Another characteristic of computer security incidents is their wide variety, which includes, but is not limited to denial of service attacks or theft of confidential information. To be able to manage these different kinds of incidents and their unique complications regarding public law, business operations or even a company's reputation, the process of Incident Response has traditionally been broken down into a number of logical steps. The following model has been proposed in Mandia et al. [5], which is a major reference in the field of Incident Response.

The methodology describes the process of Incident Response using seven different phases, which are also illustrated in Figure 1. We now give a short summary of each step.

2.2.1 Pre-incident Preparation

Pre-incident preparation is an ongoing phase, and it is actually the only one that takes place even before an incident occurs. Its purpose is to prepare the organization and the CSIRT for a possible incident. Preparing the organizion may involve implementing host- and network-based security measures, e.g. an Intrusion Detection System. To make sure that a CSIRT is able to handle an incident well, all hardware and software needed have to be provided. Appropriate training for the members of the CSIRT is mandatory to maintain an effective Incident Response Capability.

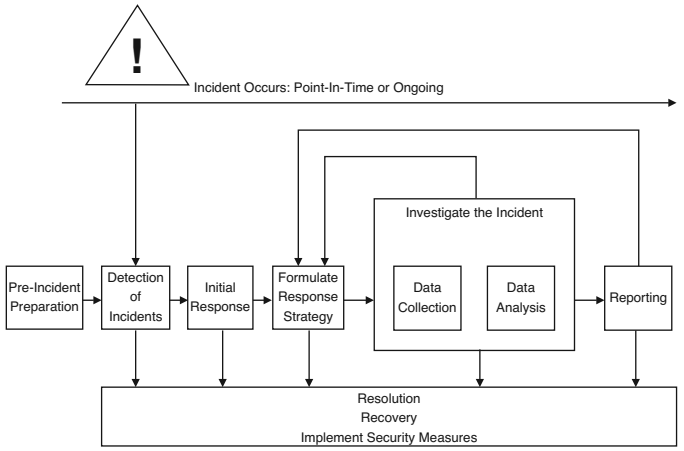

Figure 1: The Incident Response Process [5].

2.2.2 Detection of Incidents

Since obviously no incident can be responded to until it is detected first, detection of incidents is a very important part of IR. Generally speaking, anyone from a system administrator to a regular employee with no technical background could detect an incident. That is why it is important to have clear guidelines for all members of an organization explaining how to react to suspicions that something might be wrong. Most importantly it should be known who should be notified and what details have to be recorded, so that the CSIRT may take over the IR process as quickly as possible to further investigate the potential incident.

2.2.3 Initial Response

During Initial Response, the collection of information concerning the incident that has started in the previous phase continues. The goal is to gather enough information to allow formulation of an adequate response strategy in the next step. Typically, the data that is collected in this step consists of interviews of any persons involved in reporting the suspected incident, and available network surveillance logs or IDS reports, that may indicate that an incident occured. At the end of this phase, the CSIRT should have confirmed that an incident actually occured, should have identified the type of incident and the affected hosts and users, and should be able to assess the potential impact or damage. This will allow to make an informed decision about a viable response strategy.

2.2.4 Formulation of Response Strategy

The goal of this phase is to formulate a response strategy which best fits the situation, thus "considering the totality of the circumstances" [5] that surround the incident. These cir-

cumstances include the criticality of the affected systems or data, what kind of attacker is suspected and what the overall damage might amount to. Also, an organization's *response posture*, which defines its policy regarding the response to computer security incidents, may have a large impact on the choice of a response strategy. Because an incident might also induce legal or administrative action, the development of the strategy has to involve people responsible for the respective areas, which means e.g. upper management executives or the Human Resources department.

2.2.5 Investigation of the Incident

During the investigation of the incident, different types of evidence relevant to the incident, e.g. host- or network-based evidence, are collected in order to reconstruct the events that comprise the computer security incident. This reconstruction should provide explanations for what happened, when, how or why it happened and who is responsible. To achieve this, an investigation is typically divided into two steps: *Data Collection* and *Data Analysis*.

Examples of data collected during the Data Collection are host-based information retrieved from a live system, a duplication of a compromised host's harddrive or network surveillance logs. While collecting these information it is advisable to use forensically sound methods; this will be detailed later in this paper.

In the Data Analysis step, the previously collected information is reviewed to uncover the details behind the incident. It should be noted that often the collection and analysis of relevant data may take turns, effectively turning the investigation into a series of feedback loops until the final result is obtained.

2.2.6 Reporting

After the investigation of a computer security incident is finished, all the findings and results have to be documented in a written report. In this report, all investigative activities have to be documented, and all conclusions drawn have to be explained. The report should be written in a concise, yet understandable way, so that even non technical readers can follow. Because the results of the report might be used as evidence during a lawsuit, the report should be able to hold up against legal scrutiny.

2.2.7 Resolution

The purpose of the Resolution phase is to take the right measures to contain an incident, solve the underlying problems that caused the incident and to take care that a similar incident will not occur again. All the necessary steps performed should be taken and their progress supervised to verify that they are effective. It is important that changes to the affected systems are only done after collecting possible evidence, otherwise that evidence might be lost. After the resolution of the incident is complete, it may be necessary to update security policies or the IR procedures, if the response to the incident exposed a weakness in current practices.

2.3 Computer Forensics

Computer Forensics, or Digital Forensics, is a forensic science that deals with obtaining, analyzing and presenting *digital evidence*, which can be defined as "any data stored or transmitted using a computer that support or refute a theory of how an offense occured or that address critical elements of the offense such as intent or alibi" [3]. By employing accepted and proven techniques and principles, which are also applied in other forensic sciences, admissibility in a court of law and credibility of the evidence is achieved. This includes for example a high level of objectivity in every step of an investigation, and the use of reliable, repeatable and well-documented methods throughout the examination.

The Investigative Process Model [3] (see Fig. 2) which will be described now is in fact a more general approach to investigation of a computer-related offense, as it also includes steps which normally tend to fall into the responsibility of law enforcement personnel rather than the forensic analyst. We now briefly describe each step in the process model and already point out some similarities and overlaps with the steps of the IR process model.

Figure 2: The Investigative Process Model [3].

2.3.1 Incident Alerts or Accusation

The CF process starts with an incident alert or an accusation which could be an automated alert from an IDS or a concerned citizen reporting possible criminal activity. At this point, an initial assessment of the reliability of the source of the alert is done, and some facts surrounding the suspected incident are gathered to get an idea of what the investigator is dealing with. Notice that there is no Pre-Incident Preparation Phase because an investigator is not necessarily working for the organization that reported the incident, but rather contacted when an incident is suspected.

2.3.2 Assessment of Worth

During this phase, it is decided whether a detailed investigation should take place or if the suspicion of an incident that raised the alarm can be discarded. Generally this depends on how severe the problem appears to be, i.e. what the potential damage might be or whether the incident can be contained easily.

2.3.3 Incident/Crime Scene Protocols

If the decision in the previous step is to conduct a full investigation of the incident, all potential evidence at the crime scene has to be secured and documented using accepted protocols and procedures. The goal is to "freeze" the evidence in place and provide a "ground truth for all activities to follow" [3]. Actual analysis is not done at this point.

2.3.4 Identification or Seizure

After securing the crime scene, all items that may contain potential evidence for a suspected incident have to be seized and a *chain of custody* for all evidence must be started.

There are numerous legal implications and restrictions to securing a crime scene and seizing material from a crime scene. Excellent resources regarding these two steps during an investigation are U.S. Department of Justice [8] and UK Association of Chief Police Officers [7], which describe the recovery of digital evidence from law enforcement's point of view. This knowledge can also be valuable to a digital investigator and helping him to work together with the other parties involved in an investigation.

2.3.5 Preservation

After all potential evidence has been seized and is available to the digital investigator, duplicate copies of all data sources are made to ensure integrity of the original evidence. Actual analysis is only done on the copies, while the original evidence is securely stored and cataloged. This step is in fact the first step that is part of the actual digital forensic investigation, and it makes use of trusted and reliable tools to perform the duplication of evidence.

2.3.6 Recovery

The first step of a computer forensic analysis consists of recovering any data that may have been deleted, hidden, encrypted or is otherwise unavailable to the forensic investigator in its current state. The goal is to recover anything possible — "throwing out a large net" — so that the maximum of data is available for further analysis, hoping that it will contain valuable evidence.

2.3.7 Harvesting

During the Harvesting phase, metadata, i.e. data about data, are collected and used to organize the large amount of data that the last step produced. Often, files are grouped according to their filetype, or by their temporal relationship with respect to file timestamps. By slowly starting to make sense of all the available data, the reconstruction of events and the development of different hypotheses about what might have happened starts. The structured data that is the output of this phase represents the first step towards extracting the evidence out of all the data that was collected in the beginning.

2.3.8 Reduction

Since the volume of data under investigation may be very large, it is crucial for a forensic examiner to eliminate any unnecessary data in order to focus on the more important pieces of information. Still, this does not mean that evidence is reviewed based on content, but rather that data is eliminated based on filetype or by using hash databases of known good files. The desired result is "the smallest set of digital information that has the highest potential for containing data of probative value" [3].

2.3.9 Organization and Search

Organization of the data extracted in the Reduction step helps to make the actual analysis easier and can also simplify referencing to specific data during the following phases. A searchable index is often used to allow efficient retrieval of data and thus speed up the analysis.

2.3.10 Analysis

In the Analysis step, all the data that has been prepared in the steps before will be analyzed in detail, this time also incorporating the actual content, e.g. the content of text files. Finding out how and why an offense occured is investigated by establishing links between different pieces of evidence, ultimately trying to identify the offender and offer comprehensive proof that supports the conclusion. With standard scientific methods such as showing correlation between certain events, experimenting with the data and rigorously validating the results, an investigator can present digital evidence that can be relied upon even under high standards such as that of a court of law.

2.3.11 Reporting

The final report should accurately document each step of the investigation, back up any conclusions drawn during the examination with evidence and report on the methods used to obtain the evidence. Whenever possible, accepted and known protocols and methods applied during the analysis should be referenced to increase the credibility of the investigation and its results.

2.3.12 Persuasion and Testimony

Sometimes it may be necessary for an examiner to testify in court or to answer to decision makers regarding the results of his report. Trying to explain the details of an investigation to a mostly non-technical audience can be difficult and therefore special techniques exist that allow an expert to do so in an understandable fashion.

2.4 Comparison between IR and CF

The Incident Response process clearly focusses on management issues and integration of the investigative process into a business environment. For this reason, it has to take into account things such as a possible hit to an organization's reputation and weighing off the monetary cost of an investigation against the potential damage the incident may cause. When proposing a common model for Incident Response and Computer Forensics in Sect. 3, the management issues will be addressed in a similar way as in the Incident Response model described here. The different emphasis is also reflected in the way a response strategy is developed in the IR model, compared to the Assessment of Worth step in the CF model. While in the case of IR, there is a wide variety of possible response strategies, in the case of the Investigative Process the question at this point will simply be if it is necessary to assign additional resources to the case to investigate further, or if the investigation can be stopped altogether.

Another striking discrepancy between the IR and CF model is, that all the steps starting from Preservation up to the Analysis step in the CF process model correspond to just one step in the IR process model, the Investigation phase. This is because during Incident Response, the actual investigation can take on various forms or might also be left out altogether in some situations. A CF investigator on the other hand will always stress the importance of scientific and forensic methods during an investigation in order to obtain evidence that is admissible, i.e. that it can be presented in a court of law. For that reason the analysis steps are clearly seperated and structured in a subtle way to allow a systematic approach to an investigation. It is mandatory to use the scientific principle when drawing conclusions from the evidence, which means that not only has to be reconstructed what has happened, but also it has to be shown why other explanations can be ruled out; this practice is also known as *falsification*. The common model for IR and CF will incorporate these forensic principles and procedures into the investigation step that was described rather loosely in the IR model.

3 Common Process Model for Incident Response and Computer Forensics

We now present a Common Process Model for IR and CF and explain the different steps of this model. More details on the function of these steps, the methods and specific

techniques that are used today can be found in Schwittay [6].

3.1 Overview

The Common Process Model for Incident and Computer Forensics is a proposal for a new process model to investigate computer security incidents, and its aim is to combine the two concepts of Incident Response and Computer Forensics to improve the overall process of investigation. In fact the Common Model somewhat resembles a Computer Forensic investigation which is embedded into an Incident Response procedure.

The Common Model consists of three main phases: Pre-Analysis, Analysis and Post-Analysis (see also Fig. 3).

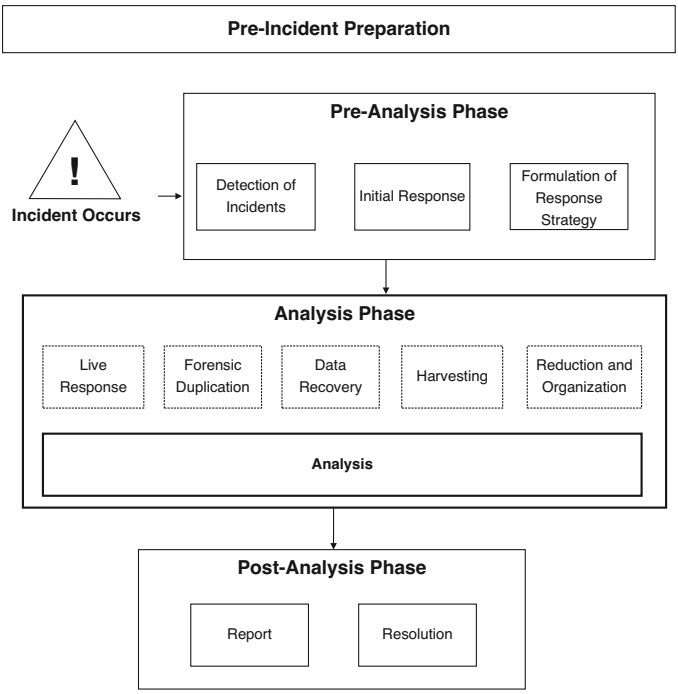

Figure 3: Common Process Model for Incident Reponse and Computer Forensics

Pre-Analysis Phase The *Pre-Analysis Phase* contains all steps and activities that are performed before the actual analysis starts; in this case, analysis means that compromised hosts or data are reviewed in detail with the intention to reconstruct the reason for the computer security incident in question. Using this notion, a quick survey of the affected host during the *Initial Response* step does not qualify as actual analysis. The three steps of the Pre-Analysis Phase correspond to the respective steps

in the Incident Response process model introduced earlier, although the last step, *Formulation of Response Strategy*, differs slightly.

Analysis Phase Actual analysis takes place in the *Analysis Phase*, which uses part of the Investigative Process Model's steps presented earlier. The sequence of steps is based on the Computer Forensics process model, except for the step of *Live Response*, which did not appear explicitly in either of the two previous models. In short, Live Response means collecting information about an incident on hosts that are still running, i.e. "live" — as opposed to a "dead" analysis of devices after the host containing it has been powered down.

Post-Analysis Phase Following the Analysis Phase, the *Post-Analysis Phase* is first of all concerned with documentation of the whole activities during the investigation in a written report; both classic models include this step.

Having introduced the Common Model for Incident Response and Computer Forensics, the specific steps of the Common Model will now be described in more detail, pointing out their significance within the whole process. For more details, see Schwittay [6].

3.2 Pre-Analysis Phase

Not surprisingly, the Pre-Analysis Phase (see Fig. 4) is comprised of all steps that are performed before the actual analysis of an incident starts. This includes Pre-incident Preparation, which is different from the rest of the steps because it is an ongoing phase, containing preparation procedures for a possible incident. Incident Detection is concerned with incident detection mechanism and procedures, and Initial Response deals with gathering initial information that allow to choose an appropriate response strategy in the Formulation of Response Strategy step.

3.2.1 Pre-incident Preparation

The goal of Pre-incident Preparation is to enable an organization to handle a computer security incident in a well-defined manner that allows quick and effective resolution.

Usual measures taken during Pre-incident Preparation can be divided into two main categories: those that prepare the personnel that is responsible for responding to incidents, in particular the CSIRT, and those that concern preparation of the organization or environment that an incident may occur in.

Another aspect concerning the preparation of an organization for possible computer security incidents is the definition of proper policies. This includes both a general statement of an organization's response stance, called Response Posture, as well as policies and rules regarding the monitoring of end users or network traffic. It has to be assured, that a proper investigation can take place without breaching privacy rights or violating public law. So-called Acceptable Use Policies define what is considered an acceptable or unacceptable

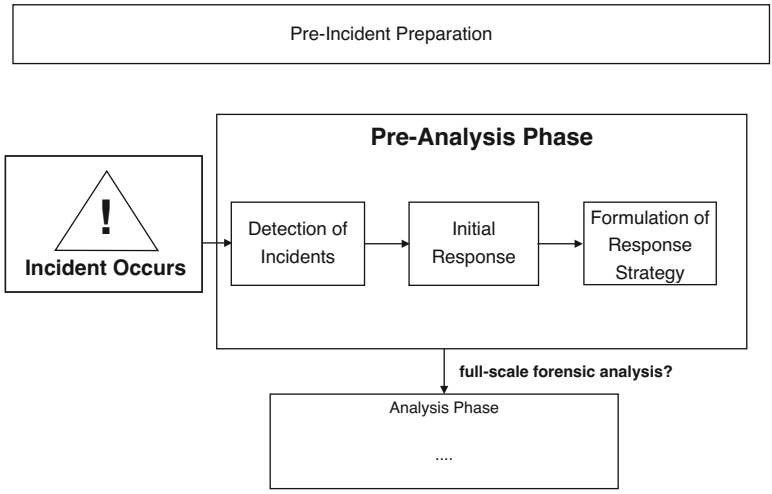

Figure 4: Pre-Analysis Phase of the Common Model

use of computer resources, and thus which behaviour can be considered a computer security incident.

3.2.2 Incident Detection

Incident Detection is about establishing proper incident detection guidelines, allowing for quick detection of computer security incidents. Proper notification and reporting procedures have to be in place, making it possible to transfer control over the remaining investigation to the CSIRT as soon as possible.

Incident Detection normally occurs whenever a person or security mechanism suspects an unauthorized or unlawful action involving a computer system or network. Actually suspicion of an incident can come from a lot of different sources, e.g. end users, security personnel, Intrusion Detection Systems, or system administrators. This is why there have to be clear guidelines for everyone that may detect a possible incident, whom to contact and how to react in such a situation, so that no potential evidence is destroyed. If an incident is reported, some initial information concerning the detection should be recorded, including date and time of detection, who is involved in the incident (by detection or otherwise), what kind of incident is suspected, and/or which hosts/networks are involved. Ideally, appropriate forms for this checklist should be provided, to allow a standardized way of documenting the incident detection. Once completed, the CSIRT should be alerted and informed about the suspected incident, so that they can take control of any further investigation.

3.2.3 Initial Response

During the Initial Response step, the goal is to have the CSIRT confirm a suspected incident or discard the suspicion by collecting and reviewing information related to the incident. If the incident is confirmed, its type and scope should be determined, in order to be able to develop a suitable response strategy in the following step of the Pre-Analsis Phase. Initial Response also includes initializing well-documented containment measures to limit the potential damage of an ongoing incident.

Often network monitoring will be initiated to confirm an ongoing incident and possibly collect additional data. In rare cases, the Initial Response may also include procedures normally taken during *Live Response*; in this case all precautions that apply for Live Response also have to be respected in this earlier part of the process. If necessary, possible harm for an organization can be avoided by containing the incident, e.g. by removing compromised hosts from the network, initializing packet filtering at routers or firewalls or by keeping suspected persons away from the "crime scene". As always all activities have to be documented accurately to maintain well-organized incident handling.

Once an incident has been verified, the information collected is used to estimate the incident's impact on users, systems and business operations. This assessment will then be used to formulate an adequate response strategy.

3.2.4 Formulation of Response Strategy

The goal of the Formulation of Response Strategy step is to determine the most appropriate strategy to handle the incident in question. In particular, the response team will have to decide whether a full-scale forensic analysis of the incident is warranted.

When trying to find an optimal response strategy to deal with a computer security incident, a lot of factors will influence the process of reaching a decision; this is also sometimes called "considering the totality of the circumstances" [5]. Some of the more obvious properties of a specific incident, which have an influence on the adopted response strategy are the criticality of the compromised hosts/data, potential perpetrators, apparent skill level of the attacker, downtime caused by the incident, or monetary loss. These are factors that are directly related to the incident and will influence how many resources are deployed to investigate the case and what kind of approach will be used to tackle the problem. Apart from these factors, there are also a number of more subtle ones that nevertheless can have a large impact on the choice of a response strategy, as for example political considerations (e.g. what happens if the incident becomes public), legal constraints (e.g. liability for not reporting certain incident to law enforcement), business objectives, acceptable use policy, monitoring policy, previous enforcement of policies, etc.

In the Common Model, the Formulation of Response Strategy step is also the time when it has to be decided, whether a full-scale forensic analysis should be conducted. We will describe later in this paper, what criteria is relevant to such a decision.

3.3 Analysis Phase

During the Analysis Phase (Fig. 5), the actual analysis of the compromised computer systems takes place, as outlined in a response strategy that was developed in the previous phase. Starting with Live Response, data concerning the incident are first gathered while the computer systems in question are still running. The rest of the Analysis Phase deals with "dead" analysis of systems, i.e. when they have been powered down.

If the previous phase resulted in opting for a full-scale forensic analysis, then all of the steps of the Analysis Phase have to be performed without taking any shortcuts. If such a thorough investigation is not wanted, some of them may be skipped or shortened; this will be mentioned specifically for each step later on. Notice that some response strategies may not even demand any analysis at all and a response team will go straight to the resolution phases.

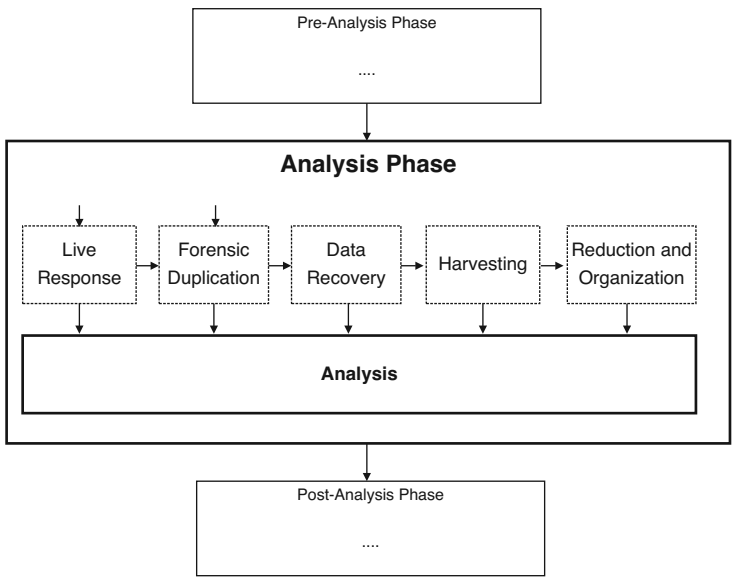

Figure 5: Analysis Phase of the Common Model

3.3.1 Live Response

Live Response is concerned with collecting data from "live" computer systems, that means the systems under analysis are still powered on and running, in contrast to "dead" analysis on systems that have been powered down. The goal is to collect *volatile* data, i.e. data that can not be recovered from a forensic duplication alone after having cut off the power supply. In addition to this it has been common to collect a number of non-volatile information too for convenience. All these data collection activities should modify the running system as little as possible so that the original evidence is not altered more than absolutely

necessary.

Certain pieces of data on a running computer are only present in main memory, that means that they will be erased once the computer is switched off. There is no way to reconstruct these information later, including for example the system date and time, list of open network connections etc. In addition to collecting these volatile data, it has become common to collect non-volatile information from running systems, usually out of convenience. Non-volatile data could also be recovered during a standard dead analysis, but collecting it from a live host may have certain advantages. For example, if a rootkit is installed on an evidence system, the output of Live Response tools may be manipulated to hide certain information. So checking for rootkits should can quickly validate or invalidate the corresponding hypothesis.

The combination of volatile and non-volatile information collected in Live Response can often contain valuable clues for an investigator to estimate the extent of an incident and take measures to contain it.

3.3.2 Forensic Duplication

In the Forensic Duplication step, exact copies of all storage media that are involved in the incident are generated, while the original evidence has to stay unaltered. A chain of custody for all media is started and the original media are stored in a safe place along with the duplicates.

3.3.3 Data Recovery

Working on the output of the Forensic Duplication step, mostly disk image files, Data Recovery is concerned with the recovery of data that in the current state of the image is not available for analysis. This includes recovery of deleted, damaged, hidden, or otherwise inaccessible data on a file system image, as well as uncovering data that is hidden otherwise, e.g. in unallocated space. For more details, see Carrier [2], which is also a major reference for forensic file system analysis.

3.3.4 Harvesting

During Harvesting, the investigator begins to gather *metadata* (data about data, like file names, file types, file sizes etc.) about the preserved material which is the output of the Data Recovery step. This allows to structure the largely unorganized material based on certain criteria, typically file timestamps, permissions and other file attributes, and the file type. This structure will help during the rest of the analysis, because often sets of data with certain properties "seem or are known to be related to the major facts of the case or incident known to this point in the investigation" [3].

When investigating files, the file type information can then be used to group files of a certain type or class, e.g. when investigating a case of suspected distribution of child pornography one would focus on image and video files, whereas in case a computer intrusion

with a suspected rootkit installation the investigator would rather focus on executables or driver files.

3.3.5 Reduction and Organization

During Reduction and Organization all data that can be identified as irrelevant to the case is eliminated, and the remaining data is organized to alleviate access to the data later. The goal is to reduce the large set of structured data that was the output of the previous step to "the smallest set of digital information that has the highest potential for containing data or probative value" [3], and then to organize the data to allow efficient search, identification and reference to relevant data in the Analysis step and the whole Post-Analysis Phase.

3.3.6 Analysis

After having recovered, harvested, reduced and organized the data related to an incident in the previous steps of the Analysis Phase, the actual reasoned analysis start in the Analysis step. An investigator develops a detailed reconstruction of the events that comprise the incident and tries to answer the questions of what happened, when and how did it happen, and who is responsible. During an investigation of suspected criminal activity, the perpetrator has to be identified, along with determining the means, motivation and opportunity. By reviewing the actual contents of the data, different pieces of evidence are correlated to establish links between them. By carefully documenting the activities and validating the results, this results in a thorough reconstruction of the incident based on objectivity and scientific principles.

To guarantee an objective analysis and interpretation of the results, the scientific method has to be applied, that means an investigator should test multiple theories regarding an incident and try to disprove them, instead of trying to verify them. By eliminating theories that conflict with intermediate results, a remaining theory has a high possibility to represent a correct reconstruction of the events.

Another very important property of analysis results is that they are *repeatable*, meaning that any observer could make the same observations as the investigator using the same methods. To allow a smooth and accurate Post-Analysis Phase, every action taken, technique applied or tools used should be documented precisely and immediately, along with its results and impact on the considered theories.

3.4 Post-Analysis Phase

The Post-Analysis Phase (see Fig. 6) starts when all activities regarding the collection and analysis of digital evidence have ended, and the objectives set by the response strategy for the Analysis Phase have been fulfilled.

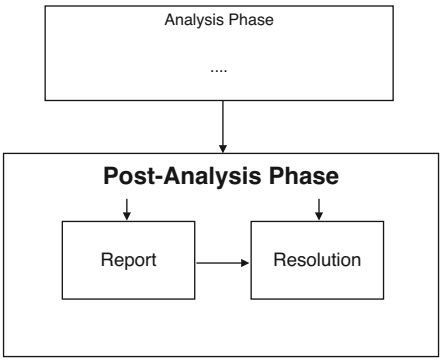

Figure 6: Post-Analysis Phase of the Common Model

3.4.1 Report

The Report step deals with writing a precise report that describes the details of an incident, is understandable to non-technical readers or executives and meets the legal standard for admissibility in court. It contains all of the documentation material that (hopefully) was produced during the Pre-Analysis and Analysis Phase respectively, and assembles the different pieces into a comprehensive overview of the whole case, while pointing out the most important analysis results and implications for resolution of the incident.

3.4.2 Resolution

The goal of Resolution is to contain the problem that caused the incident, solve it and take appropriate measures to keep it from occurring again in the future. In case of an ongoing incident, these measures may also be taken prior to conclusion of the analysis, but if possible they should only be put in place after all potential evidence has been secured for further analysis, because the evidence will generally be altered by the implemented security measures. To verify that the chosen security mechanism have been integrated successfully and that they are effective, the implementation should be supervised and the effectiveness of the measures should be validated in the future.

3.5 Discussion

The Common Model for Incident Response and Computer Forensics offers a way to conduct proper Incident Response while applying principles known from Computer Forensics during the actual analysis phase. In this way it addresses the shortcomings of both seperate models, effectively integrating a forensic analysis into an Incident Response framework. This model makes use of the management procedures developed in Incident Response process models [5], which mainly concern the Pre-Analysis and Post-Analysis phases. Typically these measure like Pre-Incident Preparation and Incident Detection are not in-

cluded in traditional investigative models, since those tend to approach the problem from a law enforcement's point of view.

4 Unifying IR and CF

After a computer security incident has been detected and some initial information concerning the incident has been collected, a suitable response strategy has to be chosen; this step has been called *Formulation of Response Strategy* above. In the case of the Incident Response process model, this simply means that after assessing the potential damage, criticality of the affected hosts and data and taking into account an organization's response posture, a strategy is formulated which seems to be the optimal way to resolve the incident. It has to be kept in mind, that a viable response strategy might in some cases be to just reinstall a compromised hosts's operating system, apply all patches and then resume normal operations again. Also, such a response strategy does not have to respect forensic principles, it may be perfectly adequate to disregard them altogether, depending on the particular incident.

In the Common Model, Formulation of Response Strategy includes an additional decision regarding the further investigation. In this step, an investigator or team of investigator has to decide, whether a *full-scale forensic analysis* is necessary.

It is clear that this type of investigation in general requires a lot of resources, because due diligence has to be invested for the analysis, and it may even cause downtime of its own, because hard drives have to be imaged, systems might have to be isolated or taken to the lab, and so on. This is why the decision to go for the full-scale forensic analysis should not be made lightly, but it should be made if the details of the incident in question call for it. The Common Model provides a comprehensive way to make the decision whether a full-scale forensic analysis is warranted or whether other response strategies may be more appropriate.

4.1 Determining factors

When deliberating the particular question of whether to conduct a full-scale forensic investigation, the Common Model focusses on two main factors, which will also be called "soft" factors, because they can only be estimated and depend largely on the properties of each particular incident:

1. attacker threat level
2. potential damage

The *attacker threat level* denotes the estimated threat of an attacker, which for this matter is any person who is directly responsible for the incident in question, whether it be by actively attacking a computer system or for example by violation of a security policy.

Such a threat measure can be derived from a more detailed *attacker model*, which is a model of the attacker with his skills and intentions, thus allowing to assess the threat he poses for the organization under attack. In the Common Model, a high attacker threat level favors a full-scale forensic analysis, because a highly skilled attacker is likely to cover his tracks or even plant false clues. It may also be desirable to completely reconstruct the incident if a skilled attacker is involved, in order to find out what exactly the attacker did and how he did it; only a thorough forensic analysis can fulfill this requirement.

The *potential damage* of the incident under investigation is the second key factor to consider when considering the type of analysis to go for. Damage in this case means financial loss as well as loss of reputation or credibility, and since the actual damage of in incident is generally not known exactly in such an early state of the response process, it can only be approximated. Generally, a high potential damage will favor the decision for a full-scale forensic investigation, because in these cases, criminal prosecution or filing for civil complaint is a likely outcome. Hence, reliable and admissible evidence may be needed to fully resolve the incident.

With these two factors in mind, an abstract equation can be used to illustrate the decision:

$$AttackerThreatLevel \times PotentialDamage > Threshold \tag{1}$$

If this equation evaluates to true, a full-scale forensic analysis should be conducted. That means that the combination of a skilled attacker and high potential damage will influence the response strategy in a way that favors a forensic analysis, while in the case of an unskilled attacker or low potential damage, such an analysis is not justifiable.

4.2 Constraints

Apart from the above soft factors, there are also two "hard" factors which are mostly independent from each other and from the soft factors, and which can call for a full-scale forensic investigation: an organization's Response Posture and legal constraints.

A *Response Posture* describes an organization's general stance towards responding to incidents. It contains general guidelines as well as prefered response procedures for a specific type of incident, e.g. what has to be done if confidential customer data has been compromised, or the organization's web site has been defaced. It may also include recommendations regarding administrative actions should an employee violate a corporate policy, e.g. how to deal with theft of trade secrets or illicit internet use.

In some cases, a full-scale forensic analysis may be demanded explicitly by the Response Posture, for example because the organization pursues a "zero-tolerance" policy towards computer security incidents and will always try to get to the bottom of each case. In other cases a forensic investigation might be necessary because of the likely outcome of the investigation, in particular whenever the resolution could include administrative or legal action. In these cases admissible evidence is required, regardless of the attacker threat level or the potential damage of then incident, therefore it is a hard factor that can indicate the need for a full-scale forensic analysis.

Legal constraints can be another hard factor that will demand a full-scale forensic analysis of the incident. In some cases, an organization will choose to involve law enforcement and make a report against a suspected attacker. It may even be mandatory to report a suspected crime, e.g. the failure to notify the authorities about suspected possesion of child pornography may make an organization liable [5]. There have also been cases where the failure to objectively analyze an incident has led to negligence charges against the investigators [3]. Because of these possible complications, a well documented and objective approach to analysis of such incidents is necessary; that means a full-scale forensic analysis has to be conducted.

4.3 Discussion

Equation 1 is related to the well-known *Risk Equation* which is often used to define risk [1]:

$$Risk = Threat \times Vulnerability \times Cost \tag{2}$$

The Threat value corresponds approximately to the Attacker Threat Level and the Cost value corresponds to Potential Damage. Vulnerability normally denotes the likelyhood that a particular attacks succeeds, which in the case of Incident Response has in fact already happened; therefore this factor could be set equal to 1, and the two equations would draw even nearer to each other. In this sense, the value that determines whether a full-scale forensic analysis is necessary equals Risk under the condition that a vulnerability has already been exploited.

Acknowledgments

We wish to thank Maximillian Dornseif for helpful discussions.

References

[1] The Risk Equation. `http://www.icharter.org/articles/risk_equation.html`.

[2] Brian Carrier. *File System Forensic Analysis*. Addison-Wesley, 2005.

[3] Eoghan Casey. *Digital Evidence and Computer Crime - 2nd Edition*. Academic Press, 2004.

[4] Tim Grance, Karen Kent, and Brian Kim. *Computer Security Incident Handling Guide*. National Institute of Standards and Technology, 2004.

[5] Kevin Mandia, Chris Prosise, and Matt Pepe. *Incident Response & Computer Forensics - 2nd Edition*. McGraw-Hill, 2003.

[6] Bastian Schwittay. Towards automating analysis in computer forensics. Master's thesis, RWTH Aachen University, Department of Computer Science, 2006.

[7] UK Association of Chief Police Officers. *Good Practice Guide for Computer based Eletronic Evidence*. National Hi-Tech Crime Unit, 2003.

[8] U.S. Department of Justice. *Electronic Crime Scene Investigation: A Guide for First Responders*. National Institute of Justice, 2001.

IT Incident Management and Structured Documentation

Sandra Frings

Fraunhofer Institut fuer Arbeitswirtschaft und Organisation (IAO)
Nobelstrasse 12
70569 Stuttgart, Germany
sandra.frings@iao.fraunhofer.de

Abstract: What is worse than having to deal with an IT incident? Dealing with a similar one a second time. Don't you wish you had some information to fall back on? This information is not being created by itself. Someone, or a system, has to collect, edit and store it in a way that other people can find and reuse it. Since written documentation of an investigation is in most cases an underrated, time consuming task, for which hardly any standardised procedures exist, the emphasis of this PhD thesis lies within the definition of a method for the well structured documentation of IT incident management - a holistic, process oriented approach of documenting IT security related incidents.

1 Introduction and Problem Area

The crime statistics of the German Federal Criminal Police Office BKA (www.bka.de) [BKA06] states that there was a decline of 4.9% of reported security related incidents in 2006 (59149) compared to 2005 (62186). This is mainly due to reduced incidents using PIN supported debit cards which where illegally acquired (-4885 / -15.2%) but this type of incident still covers almost 50% of all reported incidents (27347). Counterfeiting and forgery of probative data has increased on the other hand by 143.1% (1448). But no matter what the numbers say, those are merely the reported incidents.

The dark figures are much higher. One reason for this may be not wanting to go public with problems related to security issues. The other may be that IT security related incidents may be underestimated. Even if they are judged correctly, many organisations are not really prepared for emergencies and have hardly any standard procedures in place. They do not allocate adequate budget for preventing, detecting and handling IT incidents since awareness is missing.

Sadly this is still the case even though on the one hand preventive, detective and reactive IT security issues are being covered already by numerous books and publications (e.g. [Ca02], [Ge06], [Ka00], [Mi04], [Va02]). And on the other hand, organisations, like the German Federal Office for Information Security (BSI, www.bsi.org), try to push industrial companies and also citizens to start caring about security issues and understanding them. For example, the German IT Baseline Protection Manual (www.bsi.org) [BSI05] contains standard security safeguards and implementation

advices for organisations, and furthermore the BSI makes special IT security related information available for citizens (http://www.bsi-fuer-buerger.de).

An "IT incident" could be just a consequence of an undeliberate user action or of a deliberate action of a criminal. In this thesis, "IT incident management" starts with the preparation activities in case of an incident, covers the whole investigation of and recovery from the incident, and ends when the incident is filed. No matter what caused the incident, its investigation would probably reveal unawareness and/or lacking qualification of users, untreated security flaws in a system, or intentional wrongdoings.

The complexity of an investigation can range from taking a quick look at a log file or questioning a user, up to conducting a detailed forensic analysis of the event possibly supported or even initiated by law enforcement.

If you watch a criminal investigation on TV the detective writes down every single little item in his flip-over note book to be able to reconstruct the crime - to maintain the chain of evidence - and to find the motive and criminal(s) in the end. The same reasoning is used in investigations of IT incidents. The necessity for good documentation is being mentioned in all available literature covering IT incident management and IT forensics but no overall - covering the whole process of IT incident management - detailed, contiguous and applicable guidelines and procedures for the documentation are provided.

Documentation is part of the quality assurance process and supports in keeping track of activities performed, tracing problems and errors, reducing the time spent on a recurring problem, analysing a security lack and working on prevention methods, as well as collecting relevant evidence for following up a crime.

In general, written documentation of an investigation includes information about the different investigation steps, performed by which persons, at what time, using which kind of methods, finding what kind of evidence, and where.

Since documentation is often seen as useless effort, it is avoided or done improperly wherever possible. Sadly enough, the knowledge is missing, that documentation has high potential in improving the processes of an organisation.

2 State of the Art - Best Practices and Standards

In the area of IT security management, standards, best practices, and consultancy are available, but nonetheless organisations do not invest enough money in it. Aside from the already mentioned German IT Baseline Protection Manual, the following support (among many other publications) is available:

- standards like the ISO 17799:2005 - Code of practice for information security management [ISO02],
- or the ISO/IEC 27001:2005 - Information technology - Security techniques - Information security management systems - Requirements [ISO05],
- technical reports like the ISO/IEC TR 18044:2004 - Information Technology - Security Techniques - Information Security Incident Management [ISO04],
- procedures like the ACPO Good Practice Guide for Computer Based Electronic Evidence [ACPO03],
- best practices like IT Infrastructure Library Security Management [ITIL99], and
- judicial guidelines like the Convention on Cyber Crime [CCC01].

All these approaches for IT security management, and more importantly guidelines for IT forensics (seizing, collecting, analysing electronic evidence), e.g. [NTI02], [IOCE02], [USSS06], emphasise the fact that written documentation is a necessary and crucial part of an incident investigating procedure.

To be able to follow a process oriented approach for IT security management in an organisation and to be successful in going by it, the required processes need to be identified, defined, up and running, and to be cared for. If this is not the case then the processes need to be put in place, looked at, improved, integrated and/or attuned.

Literature shows that IT security management can be defined through different processes. As an example, the needed processes to be established within IT security management according to ISO/IEC 17799 [ISO02] are as follows:

- security policy,
- organising information security,
- asset management,
- human resources security,
- physical and environmental security,
- communications and operations management,
- access control,
- information systems acquisition, development and maintenance,
- information security incident management,
- business continuity management, and
- compliance.

It actually does not really matter which standards or guidelines to follow as long as experts set up and monitor the processes and follow the standard or guidelines according to the organisation's requirements.

Looking at guidelines for the documentation of IT security processes, the German Baseline Protection Manual [BSI05] (Safeguard S2.201) states that the documentation should as a minimum extend to the following:

- Information security policy,
- schedules of IT assets (including connectivity plans etc.),
- IT security concept(s),
- plans for implementation of IT security measures,
- procedures for the proper and secure use of IT facilities,
- documentation of reviews (checklists, interview notes etc.),
- minutes of meetings and decisions made by the IT security management team,
- management reports on IT security,
- IT security training plans, and
- reports on security-relevant incidents.

In [BSI05] (Safeguard 6.64), the documentation guideline for an incident goes as far as:

"All actions performed while dealing with a security problem should be documented in as much detail as possible so as to

- retain the details of what happened,
- make it possible to retrace the problems which occurred,
- be able to rectify any problems/faults which could result from hasty implementation of countermeasures,
- be able to resolve problems already known more quickly should they occur again,
- be able to eliminate the security weaknesses and draw up preventive measures, and
- collect evidence if a prosecution is to be brought.
  Such documentation includes not only a description of the actions carried out including the times at which they were taken, but also the log files of the affected IT systems."

This guideline is a start but definitely not sufficient for an organisation to tap the full potential of documentation. As a consequence each organisation trying to produce documentation of IT security issues details the advice given in [BSI05] and adds to them according to their expert's knowledge. This is a lot of work and not many organisations go through this effort. Especially small companies lack the knowledge and budget for company specific IT security procedures.

The conclusion derived from this situation is that there is a need for well-defined and standardised documentation of IT security processes, especially for IT incident management.

3 Documentation Requirements

Written documentation has diverse target audiences. For example system administrators see a different purpose in it than an IT manager of an organisation, or a judge in court.

For written documentation to be of any value to the reader, it has to fulfil several general requirements. It has to be at least

- clear and comprehensible (suited for the target reader),
- meaningful and reasonable (according to the specified purpose),
- complete, and
- structured (e.g. according to the sequence of the described steps).

Those simplistic requirements do not really sound impossible to be met unless of course no one demands written documentation in the organisation or the awareness for the advantages does not exist.

What happens if the documentation of an incident investigation is one or more of the following: unclear or incomprehensible, meaningless or unreasonable, incomplete or unstructured? This may lead to the following situation: the reader will not use the documentation and tell the author of the documentation that it is useless. The next time documentation is about to be created the author will probably do less of a job; except he himself has improved the procedure for creating the documentation, he was given support to improve it, or he has received an assignment to improve it. What happens if there is no documentation at all and the consequences of an incident need to be followed up?

On the other side, what happens if documentation is written, structured and kept updated? There will be detailed information on incidents and all the data gathered; the incident handling will be traceable; the loss of evidence and evidential information can be prevented; time and money can be saved due to general process improvement; the organisation is prepared - there is a process in place (incident readiness), also for protection against external accusations; comprehensible documentation is available for technicians, management, law enforcement, lawyers, judges; know-how is documented - not only in the head of the experts; there is support for quality control - verifiability according to compliance requirements, and a common terminology.

All these advantages should be objectives of an organisation and can be reached if documentation is a part of the every day work of participating roles within the IT incident management process. To make sure, that documentation is being written wherever appropriate or planned, the process of documentation needs to be defined. One way to do this is integrating documentation activities into a process model, in this case into a process model for IT incident management, in which activities and roles are defined for specific phases.

4 Reference Process Model

For this PhD thesis, IT incident management is divided into three major phases: preparation, investigation, post processing [Fr05]. Proving the chain of evidence, especially if the incident is planned to be taken to court, is an important objective. This can only be achieved if the essential steps have been taken during the preparation phase. During the post processing phase all data gathered within an investigation will be used to learn from the incident (better prevention and readiness) and to take further actions where appropriate (internal disciplinary actions, court, etc.).

Since available literature does not deliver detailed, contiguous guidelines on how to document the entire IT incident management process[1] and is therefore in most cases an underrated, time consuming task, the focus of this thesis lies exactly therein: defining a method to easily generate structured documentation of the entire IT incident management process. The structure is derived from the holistic, process oriented approach of managing IT security related incidents.

The process model this thesis builds upon is made up of three different phases. Within those, a structured set of actions and decisions is defined including the flow of actions and decisions accordingly which have to be considered or executed in the case of an IT incident. It also incorporates further information (e.g. roles and their necessary skills, checklists, references to documents and tools, and legal advice) to support the actions and/or decisions in each step. The user can, for example, consult a checklist, prior to reporting an IT incident, which covers the technical incident information law enforcement will need from the persons involved. This aims at optimising communication between the parties.

The following items summarise the major steps of the reference process model:

- Trigger: An incident happens and is discovered.
- Action: The incident is analysed.
- Action: Necessary actions are taken with the priority of stopping the incident, preventing the incident to impacting others which means keeping operations running and preventing major negative consequences for the organisation.
- Decision: Will the incident be analysed any further?
- Action (in case of further analysis): Detailed investigation of the incident, including IT forensics.
- Decision: Since more information is now available of the incident, the organisation may have to decide to involve experts, what kind of standard of proof would be required, which evidence to collect, whether to involve law enforcement and/or lawyers, and whether to go to court.
- Action: Conduct those actions according to previously taken decisions, including IT forensics.

[1] In some literature like [Ge06], [ISO04], and [ITIL99], forms for evidence documentation and log book examples are given.

– Action: Collection lessons learnt (post-processing) and get better prepared for the next incident.

After conducting an analysis of an organisation's relevant processes, policies, handbooks, etc. and therein an adequate risk analysis, improvement potentials will be derived. From this the reference model can be adapted according to the organisation's needs (see Figure 1).

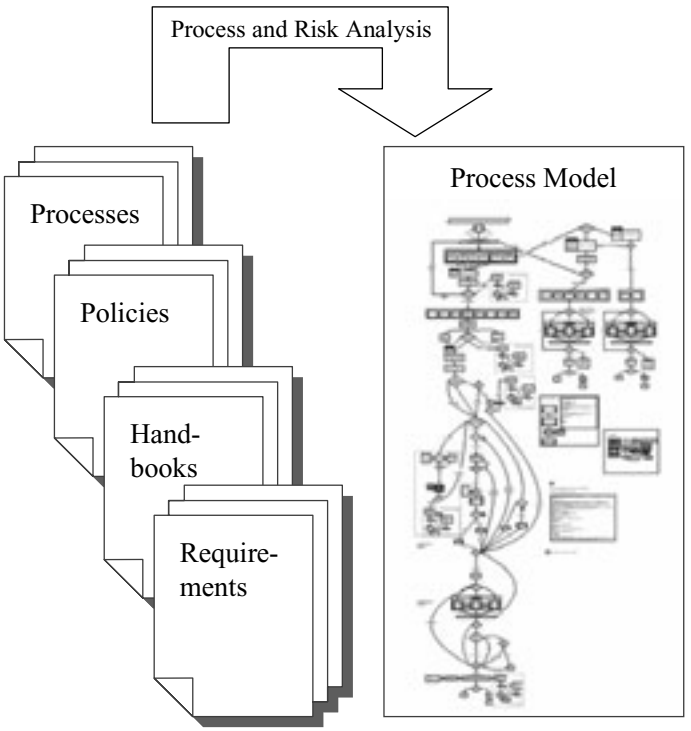

Figure 1: Defining an organisation specific process model

To be able to handle a complex process model and make it applicable, a software system is needed. Hence, a prototype was developed within a research project ([EU03], [Fr03]) which guides the user (e.g. the investigator) step by step through a predefined and company specific process model, gives additional information where appropriate, refers to relevant documents, and proposes e.g. that special roles are to be involved in the action, etc.

5 Method for Structured Documentation

One way to better handle an upcoming incident is improving the necessary steps to be taken ahead of time, and one way to learn from a previous case is to document each step of the way. Unfortunately there are many ways to document an incident, especially if different people are involved and it is not clear to them, what is relevant or irrelevant. This is usually the case if no or unclear guidelines are available. Furthermore, documentation is mostly seen as unnecessary effort. This is a consequence of lacking awareness that documentation does have its advantages and potentials.

Using a computer based system to support the documentation of IT incident management may be one possibility out of this predicament.

The technical aim within this thesis (Figure 2) is to add the documentation component to this software prototype mentioned above. This means, when going through the "advisory tool" steps, step by step, the software stores, on the one hand, the information given to the user (e.g. supporting information, how a decision should be dealt with), and, on the other hand, lets the user add investigation relevant information within each step. E.g. the software prototype will start with asking for the investigator's name, role in investigation, case name, other involved persons, short description, document a reason why a certain decision was taken, etc. The user can either select from a list, or add his individual comments. All this information is stored in an XML[2] document.

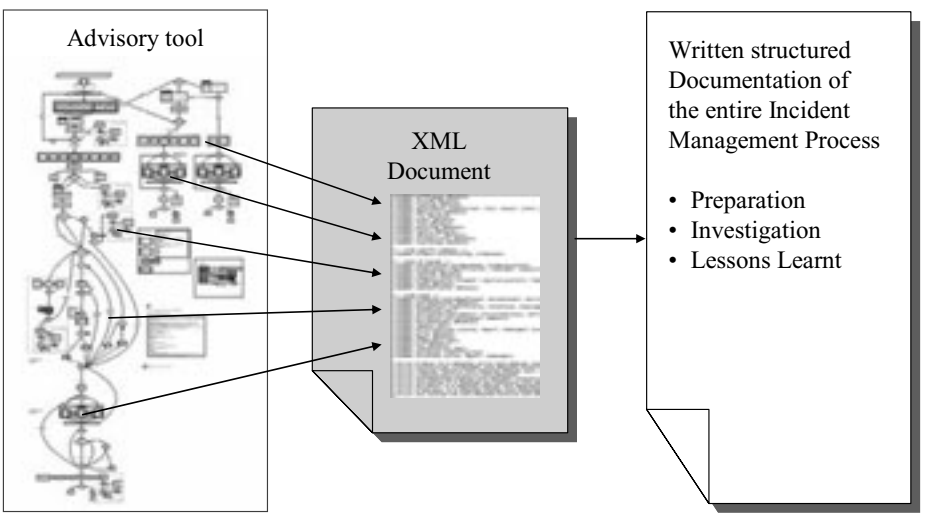

Figure 2: Overall Objective of the Thesis

[2] XML: Extensible Markup Language. Its primary purpose is to facilitate the sharing of data across different information systems (http://en.wikipedia.org/wiki/Extensible_Markup_Language).

To make the documentation flexible and applicable for different purposes, the software user (e.g. investigator) will have the option to add specific organisational, technical or legal aspects in each step of the software. Depending on e.g. preference, experience, role of the reader, the documentation structure can then semi-automatically be generated accordingly whereby the output will be made up of chapters and sections (the main structure) filled with the content needed (automatically) and added by the user (manually) including references to external documents like policies, handbook, contact lists, and guidelines.

For forensic analysis or prevention and detection tasks, several software tools already exist on the market where some are able to automatically generate reports, according to the steps taking within the tools by the user. These reports could then be added manually to the produced incident documentation described above. The documentation then results in a whole package of reports, used information, investigation steps, investigation meta date, etc.

6 Conclusion

The documentation of an incident investigation is of vital importance to an organisation. Having a structured and clear way of creating documentation within IT incident management - based on a defined and followed process - can lead to the following benefits:

- having detailed information on incidents,
- traceability of incident handling,
- prevention of loss of evidence and evidential information,
- time and money savings due to general process improvement,
- being prepared - having a process in place (incident readiness),
- protection against external accusations,
- comprehensible documentation for technicians, management, law enforcement, lawyers, judges,
- know-how is documented - not only in the head of the experts,
- support for quality control - verifiability according to compliance requirements,
- common terminology,
- support for investigations which have been internally or externally triggered, etc.

The general advantage of creating written documentation like described above is that this way is applicable to any defined and followed process and in any organisation where documentation is needed.

7 Bibliography

[ACPO03] Good Practice Guide for Computer Based Electronic Evidence, 2003. www.acpo.police.uk

[BKA06] Crime statistics of the German Federal Criminal Police Office BKA for 2006. http://www.bka.de/pks/pks2006/index.html

[BSI05] German IT Baseline Protection Manual, 2005. http://www.bsi.de/

[Ca00]] Casey, E.: Digital Evidence and Computer Crime. Academic Press, 2000.

[CCC01] Convention on Cyber Crime, 2001. http://conventions.coe.int/Treaty/EN/Treaties/Html/185.htm

[EU03] EU funded project: CTOSE - Cyber Tools On-Line Search for Evidence, 2001-2003.

[Fr03] Frings, S.: Prozessorientiertes Referenzmodel für IT-Incident Management. In: Netz- und Computersicherheit (Hrsg. Jan von Knop, Hans Frank), p. 115-122. Conference „Netz- und Computer-Sicherheit“, Düsseldorf, Germany, 07.10.2003

[Fr05] Frings, S.: IT-Forensik: Hintergründe und Ermittlungen zur Computerkriminalität.In: "Frauenarbeit und Informatik" der Gesellschaft für Informatik, # 30, ISSN 0944-0925, 2005.

[Ge06] Geschonneck, A.: Computer Forensik – Systemeinbrüche erkennen, ermitteln, aufklären. Heidelberg: dpunkt.verlag, 2006.

[IOCE02] Guidelines for the Best Practice in the Forensic Examination of Digital Technology, 2002. http://www.ioce.org

[ISO02] ISO 17799:2005 - Code of practice for information security management, BS 7799-2:2002 - Specification for Information Security Management. http://www.bsi-global.com, http://www.iso.org

[ISO04] ISO/IEC TR 18044:2004 - Information Technology. Security Techniques. Information Security Incident Management. http://www.iso.org

[ISO05] ISO/IEC 27001:2005 - Information technology -- Security techniques -- Information security management systems -- Requirements. http://www.iso.org

[ITIL99] Cazemier, J.; Overbeek, P. L.; Peters, L.: IT Infrastructure Library Security Management. London: OGC, 1999.

[Ka00] Kossakowski, K.-P.: Incident Response Capabilities. Libri Books on Demand, 2000.

[Mi04] Miora, M.: Incident Management. http://www.softwaresecuritysolutions.com/PDF/Incident%20Mgmt.pdf, 2004.

[NTI02] New Technologies Inc. Computer Evidence Processing Steps, 1999. Computer Incident Response Guidelines, 2002. http://www.forensics-intl.com

[USSS06] United States Secret Service: Best Practices for Seizing Electronic Evidence, 2006. http://www.ustreas.gov/usss/electronic_evidence.shtml

[Va02] Vacca, J.: Computer Forensics. Charles River Media, Inc., 2002.

Proposal Of A System For Computer-Based Case And Evidence Management

Prof. Dr. Fritjof Haft, Pascal Hassenpflug, Hans Lecker

Normfall GmbH
Bismarckstraße 22
80803 München, Germany
imf2007@normfall.de

Abstract: Evidence secured from computers created the need for computer-based evidence management. But there are other good reasons: Presently a growing number of major cases with a large amount of electronic and paper evidence creates a massive overload for the attorneys and judges involved. Hence a growing demand for better organisation and new tools in this area is evident. Leading auditing companies are offering forensic and dispute services, fraud services, forensic integrity services etc to their clients. But there is a big gap between their results and the everyday needs of criminal lawyers. The authors, one of which is a long-time university professor and practitioner of German criminal law and legal informatics, claim that there is significant potential to increase quality and lower cost through a fully computer-based case and evidence management process for large and complex legal cases. They define the process and discuss how Electronic Document Discovery (EDD) and Document Management Systems (DMS), which are commonly used by forensic services as well as innovative Case Management (CM) systems, can be combined to support it. Their company Normfall GmbH is one of the first to develop CM tools in Germany, which have already been used successfully in major cases of white-collar crime.

1 Overview

IT Forensics is the term applied to the area of securing evidence of IT security incidents. In a broader definition, of securing evidence from computer systems, which includes computer-based management of the evidence gathered. The next step is as natural as it is necessary – to include the management of any kind of evidence by computer systems.

Whereas standards for electronic document exchange with juridical authorities are finally being established, cases and paper evidence are still often managed in a non-electronic fashion. The U.S.-American market for legal software is more developed than the European, but not all products can be easily adapted to suit European needs.

This paper deals with the design of a computer system to support document-intensive cases such as in commercial law.

We will first present a motivation from the point of view of a legal practitioner experienced with the current situation in Germany. We will then lay out how work on a large case can be done in an efficient, cost-effective way under the aspect of process organization. To support this process, several kinds of software tools can be used. We will present the main categories of existing tools, then formulate requirements for an overall solution and finally discuss ways of combining existing tools to fulfill our requirements.

2 Motivation

On February 25th 2005 the German Parliament („Deutscher Bundestag“) enacted the public statute concerning the use of electronic communication in the judiciary („Gesetz über die Verwendung elektronischer Kommunikationsformen in der Justiz Justizkommunikationsgesetz – JKomG“, BGBl I 2006, S. 837 ff). It regulates the electronic relations and the use of electronic files in all areas of the German legal system. But no thought has been given to the question what substantial improvements of legal procedure are possible when using IT technology. Although there is worldwide research in legal informatics, German courts and lawyers don’t go much beyond production of electronic documents, the exchange of emails and the use of the Internet.[1]

This defect produces damages especially in complex criminal procedures, which occur more and more in many fields of criminal law (i.e. penal law concerning business, fiscal fraud, deceit in the medical field, corruption in companies etc.) Judicial evidence often consists of many thousands of files. The principle of oral presentation and the lack of a written protocol in all cases brought the county courts („Landgerichte“) to overestimate the testimonial evidence, even though this is the least valuable evidence of all. The factual truth that is hidden in the files often cannot be found simply because nobody – neither the judges nor the attorneys – is able to deal with the overabundance of files.

Electronic documentation offers some progress like OCR and full text search. But this gives little help if the searched descriptors are not very specific. Furthermore, it takes too much time. If somebody appears as witness in court and she or he is asked questions, the questioner must be able to retrieve relevant documents within seconds. No known document retrieval system was designed to enable judges and lawyers to do this. As a result presumption often were (and still are) the bases of the court decisions although certainty was (and is) in reach.

[1] Legal informatics pertains to the application of informatics within the context of the legal environment. It encompasses two conceptual areas: IT in law and law of IT. While the latter (issues such as privacy law, copyright law etc.) is well developed in Germany, the former (issues such as information retrieval, artificial intelligence and even such practice issues as applications which help lawyers in their day-to-day operations) is less developed.

The German law system, like many other continental European law systems, follows the tradition of the Roman law. So a short look at the history of law might be helpful for understanding the way a modern tool for computer-based case and evidence management has to be designed.

Originally, Roman law was, like modern Anglo-American law, case law. The capabilities of Roman jurists were demonstrated by their capacity to find elegant solutions for specific cases, many of which have lasted up to the present. These solutions were recorded in the so-called responsa. In the responsa the case and its solution were formulated and presented as briefly and lucidly as possible.

As the centuries passed, thousands and thousands of such responsa came into existence. This created two problems. How could one find existing precedents when dealing with new cases? And, even more importantly, how should one organize any meaningful law education in the face of such an abundance of legal material?

The answer to this was found by the Roman jurist Gaius (around 160 A.D). He made use of the insights formulated in the philosophy of science by the Greeks, whereby terms have both content and scope; a reciprocal relationship exists between the two. A lot of content means little scope and vice versa. The Greeks used this to create hierarchically ordered and structured pyramids of terms which were used to try and explore the essence of things. The Romans were more practically minded – and used this concept to create something which could be applied in practice. Gaius made use of the insights of the Greeks to develop a system of hierarchical, structured legal terms and definitions. The legal systems of continental Europe have developed out of this tradition. Hierarchies of juridical terms have been developed for almost two thousand years. In the 19th century a so-called *Begriffsjurisprudenz* was created in Germany, whereby every individual case could be subsumed under a codified legal term; leading German jurists were of the opinion that the construction of legal terms was the sole task of the study of law.

The interesting thing is that the structural systems are so complex that – although they are formulated using specific terms - they can no longer be adequately represented by language alone. We can only practice solving a case by training the use of the system on the basis of legal cases. Just as with typological terms, certain forms of knowledge exist which are located in language but which cannot be comprehensively described in language. Unlike typological terms, however, there is no everyday experience to help us out. Most people will know whether something is good or bad on the basis of their daily lives. Only the trained jurist will “know”, if this is in accordance with the rules of law without him being able to grasp this knowledge fully and put it in words. This knowledge is embedded in the system and constitutes a specifically juridical variation of the sense of justice.

This finding is particularly interesting from the point of view of behavioral theory and has as yet not been explored in any depth. We consider this to be an interesting field for further research and we are convinced that it should be possible to use computers to create tools for investigation, which will be able to help us cope with the challenges mentioned.

This background, which is familiar to every German lawyer, induced the creation of our software tool – the Normfall Manager – that has been used successfully in several criminal cases.

3 A Process Organization To Support Large Cases

In our vision of how large cases will be managed in the future, the key element is the organizational step after the evidence has been secured: The evidence must be evaluated and organized. This can be done by a team other than the people actually working on the case. It can even be outsourced. In the past, highly paid professionals often spent their expensive hours trying to extract the information important to them from a large amount of paper evidence. This is still a reality for many, but some have already managed the step to use DMS systems (in conjunction with EDD systems and scanners) to enable full-text search of documentary evidence. However, this creates the illusion that computers can take the task of interpreting the evidence off our hands. But whereas full-text search is a good tool, the evidence must still be viewed and evaluated by legal professionals in its entirety. Computer systems must support this evaluation process and provide a means of making the result accessible to everyone who is working on the case.

The outline of this organizational process is as follows (Figure 1):

3.1 Capture and Discovery

Capture and discovery refers to the phase where evidence is discovered and copied from electronic storage devices and paper evidence is made available digitally through scanning. This process needs to be integrated with the archive layer so that each copy of the evidence can be traced back to the original piece of evidence.

At this stage, technical expertise is needed rather than legal expertise. Electronic Document Discovery (EDD) systems should be used to gather documents that may be relevant as evidence. All printable document formats can be used as input to the next stages, but content in binary or other form that is not easily printable or viewable outside the system it was authored on (e.g. a database) must be analyzed by EDD experts, and their findings submitted to the next stages.

3.2 Document Evaluation

Lawyers work together with a team of qualified assistants, whom they instruct to evaluate the documents.

3.3 Case Management

The findings will be entered into the case management system, in which the lawyers that work on the case organize all the information pertaining to it.

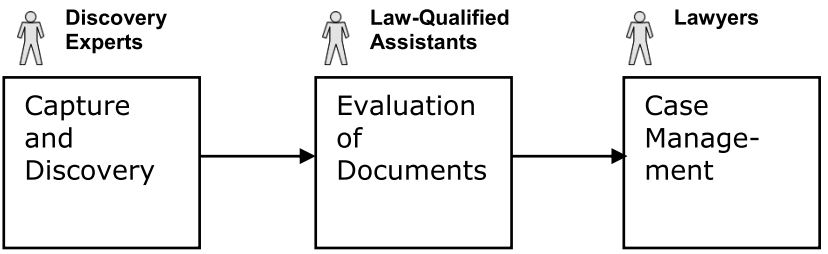

Figure 1: Three stages of preparing documentary evidence

4 Existing software systems

We will now present different categories of software systems that are used in case and evidence management.

4.1 Electronic Document Discovery systems (EDD)

Electronic Document Discovery (EDD) refers to the process of retrieving documents from electronic data storage devices as is necessary for using the information as evidence in a trial.

4.2 Document Management Systems (DMS)

Document Management Systems (DMS) manage storage and retrieval of documents. They can have very different characteristics, depending on what they are used for. The main characteristics of a DMS used in evidence management would be:

- Read-only archive that controls access to the documents
- High stability and good data backup capabilities
- High performance, file-format independent full-text search engine
- Document origin tags that point to the original piece of evidence.

4.3 Case Management / Litigation Software

Case management software, also known as litigation software, lets lawyers organize the information about a legal case. As with DMS, the particular features may vary depending on the intended use. In this document, where we assume large and complex cases, the aim is to build an informational structure of the case, that helps users understand the case, communicate about the case and access (evidence) documents using the case structure as a key.

In contrast to the United States, German attorneys and prosecutors rarely use case management tools – yet. We think that this has mainly to do with the pace at which German lawyers embrace technological innovation. The need for such tools seems obvious to us, especially in large and complex cases. Normfall GmbH, the authors' company, is one of the first to offer this kind of tools in Germany.

The main idea behind case management is that all information pertaining to a case is organized in a systematic manner, so that it can quickly be retrieved, and the people working on a case can easily get an overview of it. For cases with large amounts of documentary evidence, it is especially important to organize this evidence around the case issues. We propose a tree structure of the entire information in a case, e.g. of the legal issues, where all correspondence that is found in the evidence and corresponds to a given issue is attached to the structure node representing that issue. We have also created a tool for this which is currently marketed as the Normfall Manager.

The following are typical information categories for case management, which could be represented as nodes in a tree structure, or as records in a table:

Facts: Claimed facts of both sides, confirmed or unconfirmed, organized e.g. by

Issues: Organizes the legal battleground – what are the legal or other (e.g. moral) issues in the case.

Cast of Characters: Which natural or juridical persons are involved, and what are their roles and relations to each other?

Evidence: Lists all the evidence belonging to the case.

Investigation Questions: What information has yet to be investigated?

Law Research: What is the law relevant to this case?

4.4 Trial Management (Process-Enhanced Case Management)

Although not yet implemented, this is another area for possible computer support in the future.

The so-called *Relationstechnik* („Relation“) is a juridical method which is used in German civil cases for the assessment, arrangement and appraisal of complex contradictory facts. It is used by judges as well as by lawyers. Every German jurist is training this method during his education at universities, courts and law firms. For the judges, *Relation* is the quickest and cheapest way to find a correct judgement. For the lawyer it is an proven method as well.

The *Relation* consists of six possible steps: Sufficiency of motion, logical coherence of action, relevance of answer, contestation and plea, evidence and judgement.

These steps or *stations* can be reproduced in software, so that each case's current station can be captured. The claims of each of the opposing sides can be tracked with their status, e.g. which side needs to prove the claimed fact, whether it is disputed or not, and what both sides have said about it.

The trial management tool could be combined with features supporting electronic document exchange with the courts as have been specified by the German government.

5 Functional Requirements

We will now outline the functional requirements for the combined case and evidence management system. These are only the requirements that are relevant to the combination of systems (EDD, DMS, case management) whose properties are already well-known.

5.1 Capture and discovery

Documents created by Electronic Document Discovery and paper scanners must be tagged to identify their original source. The tag will be maintained throughout the entire system.

5.2 Original Documents

If digitized evidence documents will be changed or converted during the case management process, their originals must be maintained on an archive system.

5.3 Universal viewing capabilities

After the capture and discovery stage, all documents in the system must be easily viewable, independent of their digital format (at least for all digital formats that are designed to be printable).

5.4 Full Text Search

Mainly during document evaluation phase, users must be able to perform a full text search of all documents.

5.5 Commenting

The system needs to support commenting of the documents by multiple users. Ideally, the comments should be visible in the document viewer, but not be stored within the document files themselves.

5.6 Structuring and case management

To prepare for the case, lawyers need to organize the information around the case issues.

5.7 Multi-user access

On a complex case, lawyers from different firms may work on a case, each contributing their expertise in a special area. The system must therefore support multiple users.

5.8 Remote access

Given that the users may come from different firms, they need remote access to work with the system from their office.

5.9 Offline availability

A lawyer should be able to bring all case information to court on a laptop computer.

5.10 Ad-hoc installation

We assume ad-hoc collaboration of lawyers with different kinds of expertise. This implies the need for ad-hoc installation of the client software, taking into account the various network and software security policies that may be in effect at each lawyer's office.

6 Composing the system

Existing software systems can and should be used, leveraging time-tested technology. The challenge is to combine them to actually support a distributed team of law professionals working on a large and complex legal case.

6.1 Document Sources

At first, we need to look at the sources we want to use for our evidence management – these are mainly electronic data storage devices such as hard disks, and paper evidence. The paper evidence is captured using a scanner. Typically, it is enriched by *Optical Character Recognition (OCR)* to enable full-text search. Probably the most popular format to combine picture and OCR data in one file is the *Portable Document Format (PDF).*

In contrast to purely electronic documents, scanned documents always contain a picture layer which puts certain restraint on the way the document can be handled in a scenario of remote access, as we will see later.

From the start, the DMS needs to keep account of the origin of a document. Users must be able to trace it back to the original piece of evidence.

6.2 Original Live Systems

The software used to retrieve data off electronic storage devices is often referred to as Electronic Document Discovery (EDD) software, as mentioned before. The main challenge with data retrieved by the EDD is the file formats – they can be anything. Now one approach is to try and rebuild the live system a document base was used on, say on a virtual machine. Whereas this may in some instances be justified, for example to use document retrieval functions unique to this system, it puts great constraints on our possibilities to organize the documents in a case management software (we would have to run the case management software on the system that we built).

6.3 Universal Viewing

Our preference is to have the information “dead”, on our own system, but still be able to view it. There are two ways to achieve this: By using a universal document viewer, or by using a universal document converter, that converts all documents to a common format such as PDF. As we will see, the conversion scenario proves to be advantageous when dealing with offline access.

Either way, we still want to be able to access a document in its original file format if necessary. Therefore, we need a DMS system that simply stores all of the files in their original format. Note that there are special considerations for viewing files in their original form.

6.4 Protection From Malicious Code

Many documents such as Word documents may contain malicious program code that must be restrained from executing on the machine of the evidence management system’s user. Even document formats that do not support executable code may contain code that is brought to execution on a particular viewer using buffer overflow techniques.

The most practical ways of achieving protection are:

- Accessing the documents on an isolated execution engine (stand-alone computer or virtual machine).

- Converting all document types to a single document viewer format such as PDF. The converter must not be vulnerable to malicious code within the documents or must be isolated.

- Making sure that only the universal viewer can access the documents, and that the viewer is not vulnerable to malicious code within the documents.

6.5 From DMS To Case Management

Our next step is to go from a simple DMS to a case management system. The documents must be viewed before they can be attached to the case structure in the case management system. This might be a good point to do the conversion as well: A universal document viewer might come with conversion capabilities. So after a document has been viewed, it is converted and transferred to the case management system. If PDF is used, the document origin tag may be translated into embedded metadata. The information stored with the document in the case management system must also enable us to find the copy of the document in the DMS efficiently if needs be.

An alternative is to do batch conversion (the question here is whether user assistance might be necessary in the conversion process), and to store the converted documents in the DMS. This might be a good solution if the DMS also supports project-based offline caches (see below).

6.6 DMS / Case Management Integration

The case management system can either refer to the documents by their ID in the DMS, or keep its own copy, such as in the scenario where we generate PDF versions of all the documents we want to use in the case structure. As the case management system does not need advanced DMS functions (such as file format independent full-text search), it may store the documents in a project folder in the file system, giving us an easy implementation of an offline cache: simply copy the folder.

To sum it up, the options are:

- **Viewable documents stored in DMS:** The case management system only stores document IDs from the DMS.

- **Viewable documents stored in case manager:** The case management system keeps its own copy of each document in a standard format.

6.7 Remote Access

We need to ask ourselves: At which stage is remote access needed, for what tasks, and what are the requirements? Preferable, we do not want to transfer original documents because of the security considerations stated above. That leaves us with two options:

- To deliver documents in a “harmless” viewing format (e.g. PDF).
- Remote Desktop access to a safe environment such as a virtual machine.

The weak point in all remote desktop scenarios is the use of scanned documents because they consist of image data. We created a test setup where we accessed documents over a VPN tunnel. With an underlying Internet link of 2 Mbit/s, which should be a realistic value for many remote workplaces, we could scroll through a document with considerable flickering - still workable, but not comfortable. Once an area of the document was outside of the screen, it had to be reloaded completely the next time it was scrolled into view. The bandwidth requirement on the server is particularly high, because new image data is requested any time a client scrolls.

If the actual document is to be transferred to the client, the performance for large documents can only be acceptable if partial transfer is supported. E.g., when using PDF over the HTTP protocol, the Adobe Reader supports a feature known as *byteserving,* where only individual pages are transmitted and not the whole document. There is an important option in the Adobe Reader software which is set to load all pages in the background when byteserving is used. If this option is turned off, loading individual pages over a 640 kBit/s link is a very smooth experience and puts much less stress on the server. It is no problem to use this feature from a Web application to request an individual page of a document.

We believe that a Web application is the best document viewing solution, but we have to weigh it against architectural issues (on the evaluation stage, we may not have the documents in a Web-friendly format), and implementation issues (it may not be practical to develop a high-end, powerful case management tool for the Web).

6.8 Remote Client Installation

Deployment of the remote client is an important issue. That is because, as mentioned above, heterogeneous groups of lawyers may work on a project in an ad-hoc manner. They don’t all use the same infrastructure. The deployment solution should ideally work without the need for an administrator, particularly for updates. Whereas the Java applet framework was once designed for this purpose, thin client solutions such as remote desktop clients and Web browsers are far more popular with system designers today. Thin clients eliminate the need for updates completely because the actual application logic runs on the server.

We need to consider network access in our deployment scenarios as well. Most naturally, with sensitive data, the connection would have to be encrypted. Whereas it is generally no problem to establish an encrypted HTTP over SSL connection for a normal user at any organization, remote desktop protocols or VPN tunnels may not be routed.

The design of the case manager plays another role here. If it is a complex application, it may be difficult to port to the Web.

6.9 Offline Clients

We still need an offline client. There are two offline client scenarios, depending on where the viewable documents are stored:

- **DMS:** The case management system only stores document IDs from the DMS. The DMS has integrated offline capabilities, so the whole document base for a project can be exported to a machine and used with an offline client of the DMS.

- **Case manager:** The case management system keeps its own copy of each document in a standard format. It can then implement the offline functionality, for example by simply copying the project folder to the offline machine.

We see the first scenario as the more elegant one because the boundaries of competence of each system are better defined. We would only prefer the second scenario if the documents managed by the case management software would somehow differ from those in the DMS, such as when the content is enhanced by comments of the users working on the case, which brings us to our next consideration:

6.10 Comments and Collaboration

An ideal commenting system supports comments that refer to certain passages in the document, and the system is able to show the document with a marker at that passage. The comment itself should be shown outside of the document viewing area, and other users should be able to add their own comments to the primary comment.

The system should also be able to show documents without the comment markers, and comments would best be implemented as data structures outside of the actual document file, so the file is kept in its original form.

Commenting might be a feature of the DMS as well as the case manager.

If the DMS provides an commenting system, the case management system should be able to “call” the comment by an ID.

In a simple solution, PDF annotations would be used as comments. They can be linked to by the case management system directly through an ID.

7 Proposals of an evidence management system

Building on the previous discussion, we present the following proposals (see also: Figure Figure 2). Note that PDF can be substituted by an equivalent format.

Alternative 1: PDF on the DMS side

- Documents are converted to a universal format (PDF) at an early stage.
- PDF copies and offline cache are both managed by the DMS.
- DMS comes with remote viewing client and offline cache client.
- Case management system links to comment IDs in the DMS. Case management system has its own offline and remote access solution.

Alternative 2: PDF on the case manager side

- Documents are converted to a universal format (PDF) at an early stage.
- PDF copies and offline cache are both managed by the case management system.
- Case management system comes with remote viewing client as a Web application, and an offline cache client.

Alternative 3: Original documents / universal viewer

- Documents are left in their original state.
- Documents and case management software can only be accessed from within a virtual machine.
- With online access, virtual machines running on the server may be an alternative.
- With offline access, virtual machine including document offline cache must be installed on the offline computer.
- Can be combined with document management through a dedicated DMS or through a case manager tool (in the table below: “Access by DMS / CM”).

		Alternative 1: PDF / Access by DMS	Alternative 2: PDF / Access by CM	Alternative 3: Original Format
	Access binary data	No	No	Yes
	Viewer	PDF	PDF	Universal Viewer
	Comment Display	PDF Viewer or DMS or Case Manager	PDF Viewer or Case Manager	Universal Viewer or DMS or Case Manager
	Comment Management	DMS or Case Manager	Case Manager	DMS or Case Manager
	Viewer Security	Doc Conversion	Doc Conversion	Virtual Machine
	Special Capabilities	DMS: Manage PDF version	Case Manager or PDF export: Manage origin tag	Alt. 1 or 2
	Overall document availability rating	**good**	**medium**	**optimal**
Remote	Protocol	Web	Web	Terminal (RDP)
	Speed (Text)	good	good	good
	Speed (Image)	good	good	medium
	Remote delivery server (HTTP)	DMS	Case Manager	Alt. 1 or 2
	Client Installation	Alt. 2 + possibly DMS client (not if case manager generates access URLs)	Possibly case manager client (not if Web application)	RDP Client
	Overall remote access rating	**good**	**good**	**medium**
	Case Manager Offline Capability	Case data	Case data + documents	Case data + documents
Offline	DMS Offline Capability necessary	Yes	No	Alt. 1 or 2
	Installation	Case Manager + DMS	Case Manager	Virtualization Software
	Document Availability	All	Case Manager Documents	Alt. 1 or 2
	Offline Export	Case Manager + DMS	Case Manager	Alt. 1 or 2 to dedicated file system
	Offline Changes	Comment + case manager must support it	Case manager must support it	Alt. 1 or 2 + file system must support it
	Overall offline access rating	**good**	**optimal**	**medium**

Table 1: Overview of the three alternative solutions proposed

Remote access: If the case management tool integrates document delivery and case management into one Web application, we see the *alternative 2* architecture as superior. In all other cases, *alternative 1* seems the better integration option.

Note that we do not present a total overall rating for the three alternatives. The choice may depend on the tools available. We see alternative 1 as having the best overall architecture, alternative 2 as the easiest implementation and alternative 3 as the best availability if combined with features from alternative 1 (i.e. the documents are delivered by the DMS).

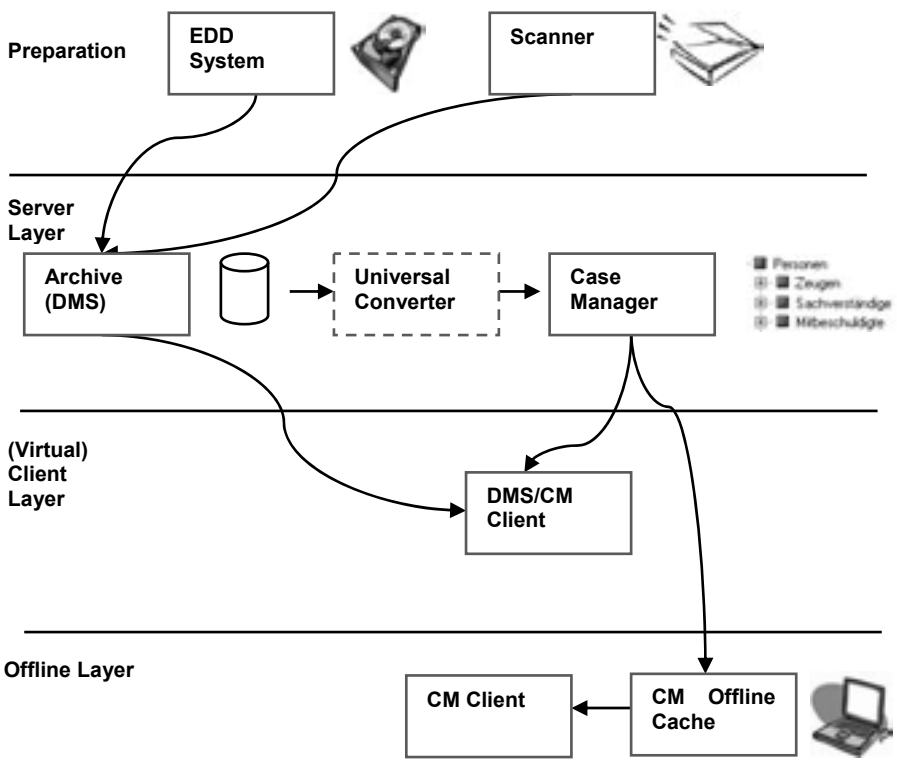

Figure 2: Proposal of an evidence management system, including EDD, DMS and Case Manager (alternative 2)

Of course, the splitting of the architecture into DMS and case management tool is somewhat arbitrary. The idea behind it is that most DMS will not have sufficient case management functions and most case management tools will lack DMS features. Both could be combined in one tool, but to us, this doesn't really reflect the current market situation.

8 Conclusion

In this paper, we have proposed a process organization for managing large cases, emphasizing a step where evidence is evaluated by law-qualified professionals.

We have discussed how a system for computer-based case and evidence management supporting that process might look like. We have presented three alternative solutions for the architecture of such a system. Ad-hoc collaboration and remote access by independent lawyers were especially considered.

A solution involving terminal services and machine virtualization was considered best for document authenticity and availability. Two solutions involving conversion of documents to a universal file format (PDF) were considered best for usability.

We believe that the ideas presented in this paper may help to advance the discussion about electronic support for evidence and case management in German law practice, an area in which innovation, although essential, has stagnated for some time now. We are committed to continue work in this area, in theory as well as in practice through our company's Normfall Manager product line of case management software.

Information-Sharing System for Vulnerability Information Dissemination in Large-Scale Organization

Tohru Sato and Jumpei Watase

NTT Information Sharing Platform Laboratories
3-9-11, Midori-cho, Musashino-shi, Tokyo 180-8585, Japan
sato.tohru@lab.ntt.co.jp
watase.jumpei@lab.ntt.co.jp

Abstract: An organization has to deploy the capability to respond to vulnerabilities in products for which the organization is responsible. For a proper vulnerability response practice, disseminating vulnerability information to appropriate departments in an organization is critical. Some technologies for vulnerability information exchange have been presented. However, vulnerability information dissemination in a large-scale organization still has critical problems, mainly in management and maintenance of contact point information and confidentiality. To address these issues, we developed an information sharing system that utilizes a real-world social network in an organization and centralized information storage by means of a web application server. Our developed system makes information dissemination and organizational risk management more efficient and appropriate.

1 Introduction

To maintain the security of computer networks, dealing with vulnerabilities related to information technology products such as servers, routers, application software, and terminal devices, for example, is critical.

For proper vulnerability handling in an organization, deploying a vulnerability-information-sharing mechanism in an organization is necessary. The mechanism should include the capability to send vulnerability information in a timely manner to the appropriate product owners who are in charge of products/services.

The goal of our work is to integrate a system that comprehensively manages vulnerabilities. Our developed system is named “Security Information eXchange Infrastructure (SIXI)”. SIXI enables an organization to distribute vulnerability information in a timely and secure manner to product owners in an organization who are responsible for their products, systems, or services. Furthermore, SIXI also provides a vulnerability response status tracking function. This function collects information about the status of the ability to respond to vulnerabilities and shows security managers such as the Chief Security Officer (CSO) how many vulnerabilities have been fixed and how many vulnerabilities still exist in the organization.

In this paper, we discuss some problems in the vulnerability handling process in a large-scale organization. We also present a method and a conceptual diagram of our developed system. Target users of this system are security managers, system developers, and system operators in an organization.

1.1 Types of vulnerability information

In consideration of the confidentiality level of vulnerability information, we classify vulnerabilities into the following two types.

(Type A) Public

The word “Public” is defined by the Vendor Special Interest Group (Vender SIG) of the Forum of Incident Response and Security Teams (FIRST) as “entities outside of a vendor” [1]. They also defined “vendor” as “private or public corporations, partnerships or other for-profit business entities”.

This type of vulnerability information has been disclosed to the public. For instance, some security information websites or mailing lists disclose vulnerability information of this type. This type of information does not require a high-confidentiality level because the information has already become available to everyone.

However, if program codes that exploit certain vulnerabilities are published, the vulnerability information should be shared among developers and operators who are in charge of affected systems as soon as possible to facilitate early responses and minimize the potential damage.

(Type B) Undisclosed

This type of vulnerability information is secretly shared among one or more specific entities, such as vendors and coordinators, before being disclosed to the public. Vendors who supply affected products are required to respond, which includes developing and releasing security updates. They have to maintain the confidentiality of the undisclosed vulnerability information during their response because the inadvertent public disclosure of such information puts all vendors’ customers and product users at risk of a potential exploitation.

In this context, we discuss the system requirements that deal with the two above-mentioned types of vulnerability.

2 Related Work

The efficient exchange of vulnerability information using the standardized description format is one of the major problems in responding to vulnerabilities.

The Extended Incident Handling Working Group (INCH WG) [2] in the Internet Engineering Task Force (IETF) [3] defined the essential vulnerability information descriptions as the following [4].

v1) The affected product

v2) The nature of the problem

v3) The possible impact if the vulnerability and/or exploit were, accidentally or maliciously, triggered

v4) Available means of remediation

v5) Disclosure restrictions

Some description formats for efficient exchange of vulnerability information using the above description have been developed among security professionals.

Another major format is Mitre's Common Vulnerabilities and Exposures (CVE) [5]. CVE is the de facto standard for storage of vulnerability information. Mitre also introduced another format named the Open Vulnerability Assessment Language (OVAL) [6]. OVAL is the common language for security experts to discuss and agree upon technical details about how to check for the presence of vulnerabilities on a computer system.

The Organization for the Advancement of Structured Information Standards（OASIS） [7] developed another format named the Application Vulnerability Description Language (AVDL) [8]. AVDL was a candidate of a security interoperability standard to create a uniform manner of describing application security vulnerabilities using XML, but this effort has been deactivated at present.

The Open Security Project (Opensec) [9] provides another format named the Advisory and Notification Markup Language (ANML) [10]. ANML is an XML-based specification for describing advisories and other types of notifications. ANML intends to resolve the inconsistent use of terminology by software vendors in their advisories and enable applications to read these advisories easily.

In addition, a lot of applications to operate these formats have also been proposed.

While various vulnerability description formats and applications have been proposed by many security professionals, some critical issues remain in vulnerability handling in an organization. In the next section, we discuss these issues.

3 Vulnerability Handling Issues

In this section, we discuss the following four issues. These issues should be resolved to handle vulnerabilities properly, in a timely manner, and securely in an organization.

Issue 1. Notification to everyone who needs vulnerability information

To distribute vulnerability information among all developers and administrators who are responsible for the affected product in the organization, it is necessary to manage the list of the product developed/operated in the organization and the person in charge of these products. When vulnerability information is reported, the capability to search for appropriate people quickly is significant.

However, management and maintenance of contact point information including the components of products and their product owners is difficult work in a large-scale organization. One of the reasons is that a large organization owns a lot of products and there are many employees.

Another reason is that the components of products and their owners are frequently changed by the startup of a new development project, changes in product specifications, or reassignment of their owner/section. Even if the information has been collected successfully at some point in time, the information dynamically changes and becomes obsolete along with company activities.

Therefore, effective management methods that can maintain components of products and their owners are required.

Issue 2. Accurate understanding of product composition

Accurate understanding of product composition is also difficult work because products consist of a complex combination of operating system, middleware, application software, library, and protocol for example. Products are also supplied in various forms such as commercial, or OEM. In addition, in the case of outsourcing product development or failure to change owners of products, owners cannot understand the entire composition of their products.

An effective method of extracting product component information is required.

Issue 3. Observation of vulnerability information distribution and response status

To observe that vulnerability information is appropriately shared with the right person, a method to monitor the information distribution status in an organization is required.

Furthermore, in the case of dealing with undisclosed vulnerability information, distribution control is also needed to control confidentiality of the information. If the information were exposed to outsiders such as a hacker community, the risk of a potential exploitation would increase.

Similarly, existing vulnerabilities of products in an organization and the status of responding to those vulnerabilities should be monitored. However, the conventional distribution method by means of E-mail makes response status tracking difficult.

A monitoring method that enables a security management section to track the response status effectively is required.

Issue 4. Centralized control of vulnerability information

Distributing undisclosed vulnerability information by E-mail has another problem. The E-mails are stored in many unspecified computers by repeated forwarding. In this case, the risk of leaking undisclosed vulnerability information becomes higher.

Centralized control of vulnerability information in an organization is also required to reduce the risk of information leakage.

3.1 Conventional Practice of Vulnerability Information Dissemination

We analyze some vulnerability handling cases that we experienced and examine our conceptual design.

In one case, we dealt with undisclosed vulnerability information. In maintaining the confidentiality of undisclosed vulnerability information, we were careful to distribute only to product owners who were thought to be responsible for affected products.

At the beginning, we selected people who were eligible to receive the information according to the following criteria.

(A) Product owners in charge of affected products.

(B) People who are in charge of managing the point of contact of development sections and operations sections.

(C) People who have wide human networks in related fields.

(D) People who have extensive knowledge of related technologies.

However, we could not distribute vulnerability information to all people who needed the information in the first stage. Then, we asked the above selected people to introduce us to other people who should be notified. The vulnerability information was distributed in our organization more widely and properly due to the introductions or information forwarding by the selected people.

In this case, to confirm that the introduced people are truly eligible to receive the information, we checked who received the information. Furthermore, we also checked their response status. This work took a huge amount of time and human resources.

We assume that efficient organizational vulnerability handling requires a system that has the following capabilities. These are 1) to identify "what types of vulnerability information each product owner needs", 2) to forward vulnerability information to individual users securely, and 3) to track distribution and response status.

The next section discusses our approaches to solve these problems.

4 SIXI System

4.1 Overview

The objective of our developed system is to provide effective functions for organizational vulnerability handling. The main features of SIXI are an efficient management of contact point information and dissemination of vulnerability information based on a social network in an organization.

SIXI consists of three subsystems. The first subsystem is the Contact Point Management (CPM) subsystem. The function of this subsystem is to collect contact information of individual product owners such as developers/administrators. In addition, CPM also provides a function to search their contact information.

The second subsystem is the Vulnerability Information Management (VIM) subsystem. One function of this subsystem is to collect vulnerability information and information about the distribution and status of a response by product owners. Other functions are to summarize that the distribution and response status information, and to provide reports on various aspects of vulnerability handling.

The third is the Contact Point Information Update (CPIU) subsystem. This subsystem provides a mechanism that autonomously updates the Contact Point Information according to the vulnerability information distribution process. This mechanism enables more efficient maintenance of the Contact Point Information.

The major components in SIXI are depicted in Figure 4.1. In the following sections of this paper, the details and features of the two subsystems are discussed.

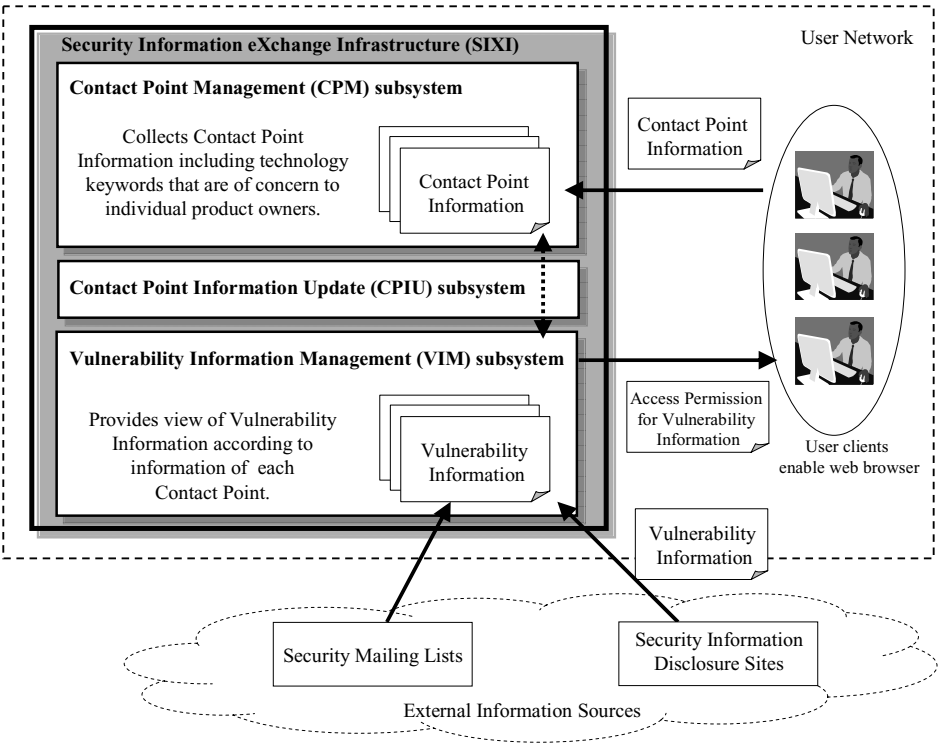

Figure 4.1: Structure of system

4.2 CPM Subsystem

The purpose of the CPM subsystem is to provide a capability to search for product owners quickly and accurately. This capability is a solution to the above-mentioned problems of section 3.

The subsystem deals with two types of information. One is Contact Point Information. The other is the Keyword Template. The components in the CPM are depicted in Figure 4.2. The definitions of the two types of information are as follows.

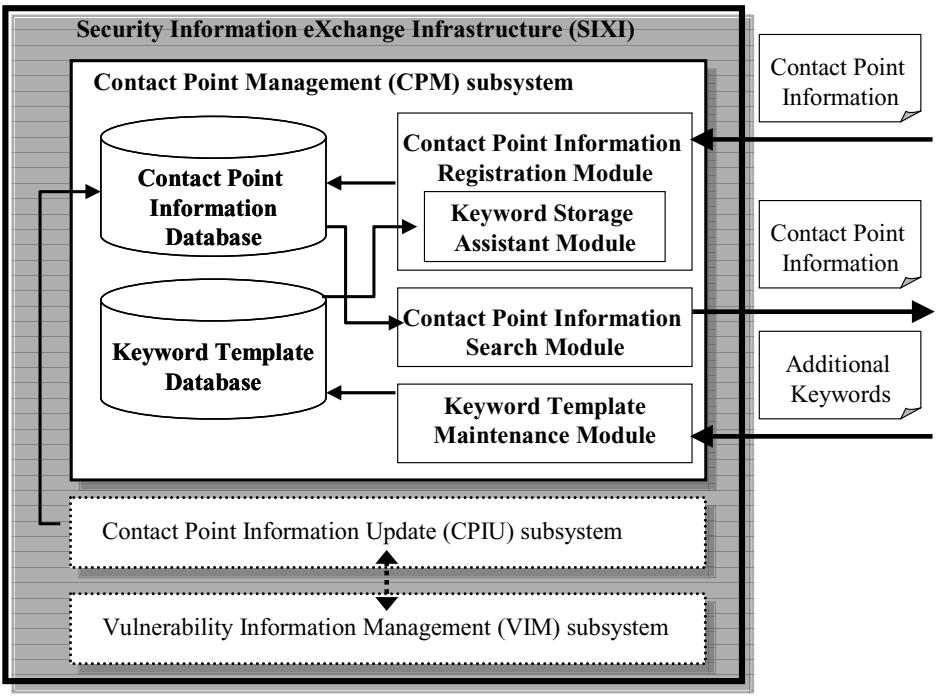

Figure 4.2: Internal Structure of Contact Point Management Subsystem

4.2.1 Contact Point Information

Contact Point Information is a user's profile to be used to search and determine the product owners who should receive certain vulnerability information.

The following components are described:

c1) Product owner's name

c2) Product owner's contact information

c3) Product name that he/she is responsible for

c4) Product composition described with technical keywords

c5) Other technical keywords that the owner has an interest in or to which he/she pays attention

Technical keywords describe product composition such as operating system, middleware, application software, libraries, and protocols.

Contact Point Information will be maintained by product owners themselves. For example, if an owner is responsible for a certain product, the owner will store his/her product name and composition with his/her name and contact. The owner also stores other technical keywords in which he/she is interested.

4.2.2 Keyword Template

The Keyword Template is a set of technical keywords to offer choices of technical keywords that can be registered to product owners. In other words, keywords included in Contact Point Information about a certain product owner are a subset of the Keyword Template.

The purpose of the Keyword Template is to store technical keywords easily.

4.2.3 Keyword Storage Assistant

The Keyword Storage Assistant is a function that extracts product composition information as a list of keywords. The function extracts technical keywords from various resources by comparing them with the Keyword Template.

One of the simple ways to extract keywords is to compare the Keyword Template with specifications of a product document. If some software agents or devices that scan hardware to investigate composition of a certain product were available, keywords can also be extracted by comparing a scan result with the Keyword Templates.

4.2.4 Keyword Template Maintenance

A keyword that should be listed in the Keyword Template will change as a new protocol, middleware, and product appear. Therefore, the Keyword Template requires continuous maintenance.

We provide two methods to update the Keyword Template for maintenance. One method is a vulnerability information collector like a security web site crawler. The method observes vulnerability information including new technology keywords in some security information resources such as CVE. The crawler can observe new technology keywords and add them to the Keyword Template because security information resources provide vulnerability information described in a specific format.

Another method is manual editing of the Keyword Template by any product owner who noticed that he/she needs to register more keywords. To prevent an improper keyword from being added in this way, an approval process performed by a security manager is introduced.

4.3 Vulnerability Information Management Subsystem

The VIM subsystem provides services to handle vulnerability in the centralized management manner. This is for centralized management of vulnerability information distribution and the status of responding to vulnerabilities. In addition, information is prevented from proliferating toward many unspecified PCs. The components in the VIM are depicted in Figure 4.3.

The subsystem has the following features.

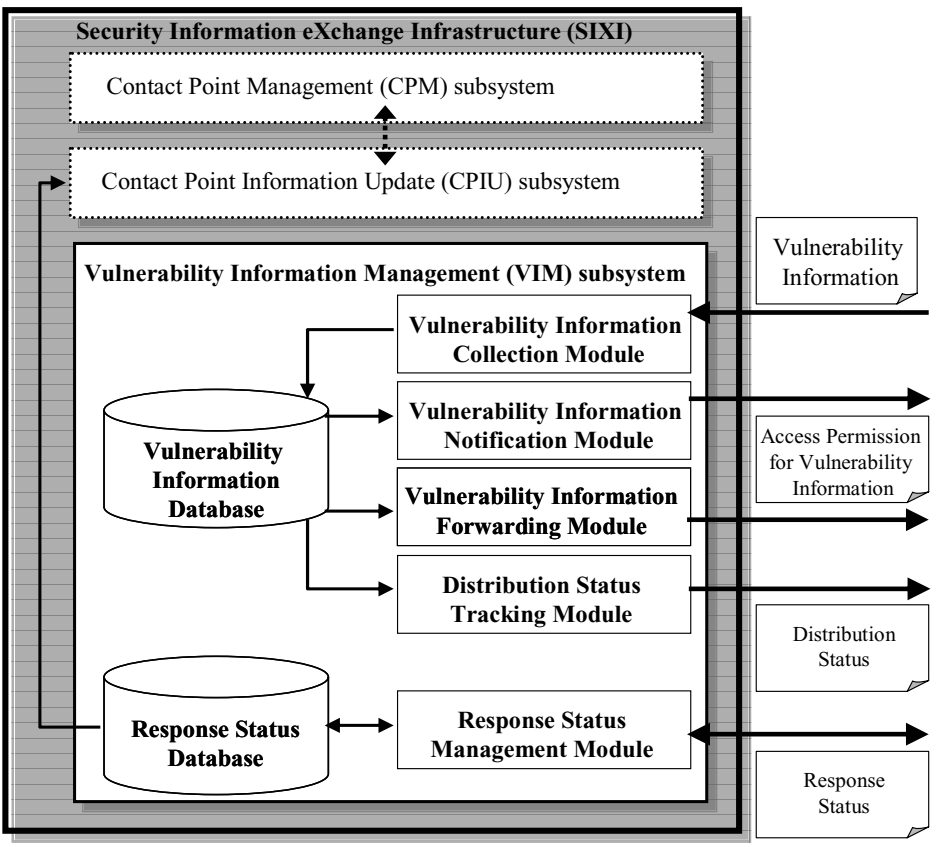

Figure 4.3: Internal Structure of Vulnerability Information Management Subsystem

4.3.1 Vulnerability Information Collection

Public vulnerability information can be collected by the Vulnerability Information Collection Module from external sources such as a security information web site and/or mailing lists. The module periodically observes new vulnerability information.

If new vulnerability information is published by a specific format such as CVE, the module will automatically parse the information and store it in an internal vulnerability information database.

Other vulnerability information without a specific format is stored by SIXI operators manually.

4.3.2 Vulnerability Information Notification

According to matching "description of affected products included in a vulnerability information (v1)" and "product composition keywords included in Contact Point information (c4, c5)", vulnerability information is only sent to the person in charge of affected products.

SIXI provides a view using a web application interface instead of E-mail of the vulnerability information sent to the person in charge of affected products. Product owners can view the vulnerability information using generic web browsers without saving the information to their hosts.

4.3.3 Information Distribution Status Tracking

Vulnerability information distribution status can be confirmed by extracting an access history of the vulnerability information database. The status would be disclosed to all SIXI users. Knowing which information has/has not been read by each product owner is possible.

4.3.4 Vulnerability Information Forwarding

The information forwarding function that enables any person to forward certain information to other product owner(s) is introduced. A person who has noticed that another person who should be notified has not been informed could transfer the information to that person. In this manner, we expect the information would be delivered to the appropriate people more comprehensively.

However, the forwarding of undisclosed vulnerability information should be restricted to people in charge of affected products. To hide undisclosed information from the public, we introduce a label, which represents confidentiality as metadata of vulnerability information. We also implement a process to audit and approve the forwarding of undisclosed vulnerability information.

When the forwarding of undisclosed vulnerability information with high confidentiality is required in the SIXI system, the forwarding will be executed after approval by the security manager in the organization.

4.3.5 Response Status Management

SIXI has a Response Status Database, Response Status Input Module, and Response Status Tracking Module. The Response Input Interface accepts status information about responses to certain vulnerability information from each product owner. The status information is stored in the Response Status Database.

The Response Status Tracking Module shows how many vulnerabilities remain in an individual product and summarizes how much risk is still in the organization in performing organizational risk management.

4.4 CPIU Subsystem

The CPIU subsystem is responsible for only one function. The subsystem helps the interaction between CPM and VIM. This subsystem updates Contact Point Information of a product owner regarding vulnerability information forwarding action as a trigger of updating. The object to be updated is the keyword set included in Contact Point Information about each product owner.

When a product owner receives the vulnerability information, he can confirm the distribution status of the information in his organization. If he noticed the information was not reported to colleagues of his acquaintances who should know that information, he can forward or transfer that to them. People who are notified by the forwarding can input their evaluation of the usefulness of the information to SIXI. Then, SIXI interprets the input and autonomously adds the related keywords to the product owners' Contact Point Information.

By means of the above mechanism, SIXI can maintain Contact Point Information effectively. The function to edit Contact Point Information manually is also provided.

5 Discussion

Correlation between contact point information and response action/status by the Contact Point Information Update Subsystem enables an organization to manage and maintain information that indicates who needs what kind of information in the organization. As a result, security managers can easily disseminate the critical vulnerability information to appropriate parts of the organization more comprehensively. This can resolve the issue 1 in chapter 3.

The introduction of Keyword Template and Keyword Registration Assistant capability confers an advantage in understanding products composition easily. This is a solution to issue 2.

The centralized management of vulnerability information and response status by means of the Vulnerability Information Management subsystem enables security manager to observe the progress of the response and the total amount of the risk remaining in the organization. This could be a solution of issues 3 and 4.

6 Conclusion and Future Work

In this paper, we have discussed some issues of organizational vulnerability handling. To handle vulnerability information appropriately, managing Contact Point information efficiently and reducing the risk of leaking undisclosed vulnerability information is significant.

We presented a vulnerability information management system named Security SIXI as an approach to resolving these problems.

By using the social network in an organization, SIXI effectively maintains Contact Point Information. Furthermore, SIXI makes vulnerability information notifications more secure by centralized control of vulnerability information. The centralization of individual response status information enables the security risk of the organization to become more visible.

However, to make organizational vulnerability handling better and more appropriate, a mature organizational culture and climate for vulnerability handling is necessary besides vulnerability management system like SIXI.

In the future, we will execute a field test of this system for evaluating the effectiveness of the approach. We will also examine the effect of various kinds of peer-recommendation using social networks such as “a person needs to know about this particular vulnerability” and “a person manages (owns) this type of system”. Furthermore, we have a plan to evaluate the interoperability between SIXI and other current vulnerability information management technologies.

Bibliography

[1] FIRST Internet Infrastructure Vendors Special Interest Group, “Guidelines for Vendor - Coordinators relationship”, October 2006, http://www.first.org/vendor-sig/vendor-coordinators-guidelines-public-v1.0.pdf.

[2] IETF INCH WG (Extended Incident Handling Working Group), May 2006, http://www.ietf.org/html.charters/OLD/inch-charter.html

[3] The Internet Engineering Task Force (IETF), http://www.ietf.org/.

[4] IETF INCH WG, “Vulnerability and Exploit Description and Exchange Format (VEDEF)”, June 2004, http://www3.ietf.org/proceedings/04aug/slides/iinch-3.pdf.

[5] The MITRE Cooperation, “How we build the CVE list”, May 2004, http://www.cve.mitre.org/about/list.html.

[6] The MITRE Cooperation, "The Open Vulnerability Assessment Language", October 2004, http://oval.mitre.org,

[7] Organization for the Advancement of Structured Information Standards（OASIS）, http://www.oasis-open.org/home/index.php
[8] OSAIS, “OASIS Application Vulnerability Description Language (AVDL) TC”, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=avdl.
[9] The Open Security Project (Opensec), http://www.opensec.org/index.html.
[10] Opensec, “Advisory and Notification Markup Language (ANML)”, http://www.opensec.org/anml/.

IMF 2007

Wednesday, September 12th, 2007

A Case Study on Constructing a Security Event Management System

Vijay K. Gurbani, Debra L. Cook, Lawrence E. Menten, and Thomas B. Reddington
Security Technology Research Group,
Bell Laboratories, Alcatel-Lucent
{vkg,dcook,lmenten,treddington}@alcatel-lucent.com

Abstract: We define *Security Event Management* (SEM) as the ability to analyze the information arriving as discrete events from various network services in order to determine whether the network, or a portion of the network, is in the process of being compromised *and* to undertake evasive action to mitigate the attack. In practice, a SEM system can be viewed as the collection of tools, technologies and policies related to presenting a security-specific view of the network at all times We describe our work in constructing a security event management system using a mix of open source and internally developed software. Our results in constructing such a system and lessons learned during the process are presented in this paper. We also outline an agenda for future research in this area.

1 Introduction

For the purpose of this paper, we define Security Event Management (SEM) is the ability of a network to analyze and interpret discrete events in order to enable better security assessment *and* undertake appropriate remedial action (this characterization is more expansive than the traditional held definition of a SEM as a detection only step; mitigation is provided by other means.) SEM capability is necessary in today's networks because network security is notoriously difficult to manage due to the following four reasons: First, network security has been largely delegated to individual hosts and applications executing on them; this leads to a plethora of points to monitor when a security breach occurs. Second, the hosts and applications executing on them typically have limited communications with other security products. As a result, there is substantial difficulty in detecting large scale network-based attacks. Third, most network security is reactive, not pro-active, and as a result is done after an attack has occurred and is well underway. Finally and most importantly, network security tools today do not lend themselves to adequate situational awareness by failing to provide an integrated network view regarding the state of the network.

Figure 1 depicts a bi-directed cyclical graph that can be used as a model for securing the network (we consider the graph bi-directed because for every edge in the graph, the reverse edge also belongs to the graph.) The graph consists of three vertices: Prevention, Detection and Response; the edges are labeled according to the functions they provide between the vertices.

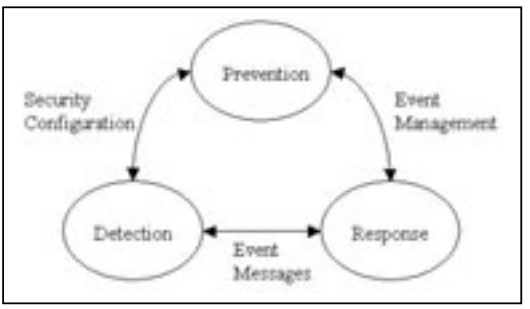

Figure 1: The Security Triad

The graph forms a feed-forward system (see Figure 2) where the output from a state is used to drive the behavior of the next state. For example, if a host detects that it is undergoing a port scan (current state: Detection), it will transition to the next state (Response) by issuing an event message. In this state, the host uses event management request („Block source IP address X“) to transition to the next state (Prevention), where the event management request is subsequently converted to an event management directive enforceable locally or at a router or firewall to prevent the host from being scanned further.

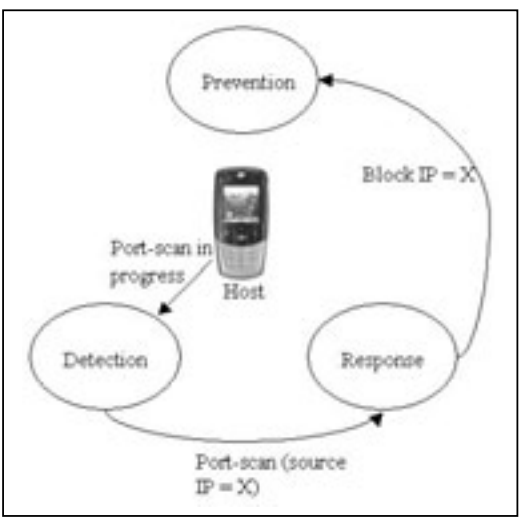

Figure 2: Feed-Forward System

Figures 1 and 2 essentially describe a SEM system. SEM is the „glue“ that allows the security triad to work together in an integrated fashion. A typical SEM architecture (see Figure 3) consists of three layers – a user interface, an event correlation engine, and event collectors. The user interface provides a graphical view of the situational awareness to the network the operator, and depending on the specifics of the SEM system, may be used to specify policies (authentication, access control, etc.) on the individual hosts and applications in the network.

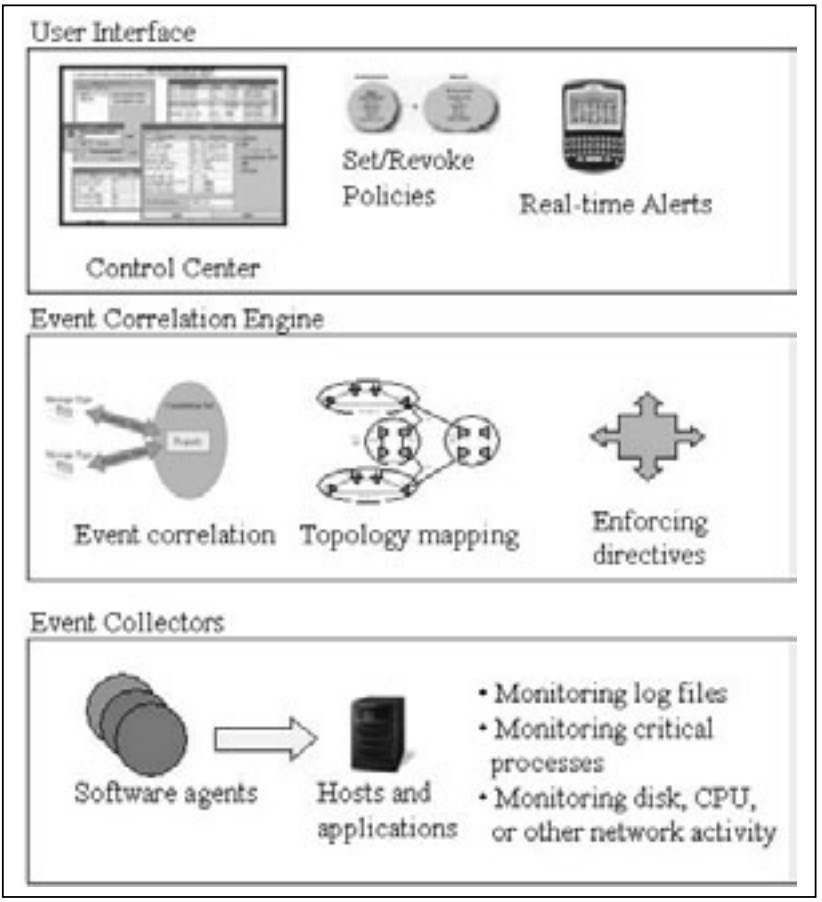

Figure 3: A Prototypical SEM Architecture.

The second layer, the event correlation engine, corresponds to the „Response“ state in Figure 1: it is concerned with real-time response that quarantines affected parts of the network. A robust correlation engine will require a meta-language to specify and capture events; „data fusion“ capability to describe the overall behavior of the attack based on discrete events arriving into the engine; semi-automatic response to thwart

attacks before they infect the entire network; and the ability to securely execute commands on other network elements.

The third layer, event collector, corresponds to the „Detection“ state of Figure 1. Event collectors are software agents co-located on hosts or with applications. When an agent detects behavior that is contrary to the normal operation of the host or application, it informs the event correlation engine of this occurrence. The event collectors agents have to be tuned to a particular application, for instance, an agent for a web server may monitor the log file and intimate the correlation engine of abnormal behavior (such as rapid requests for accessing resources coming from the same IP address, an attempt to access a resource known to be vulnerable, etc.)

The remainder of this paper describes our work at creating a SEM system using open source and internally developed software. Section 2 reviews the existing literature on SEM systems and Section 3 contains the system details of our SEM system. This is followed by a series of „what-if“ scenarios that tests the system by launching attacks on the network. Section 5 details observations and lessons learned from building a SEM system; Section 6 outlines an agenda for future research in this area.

2 Related Work

The concept of and the need for a SEM system has been recognized in the industry; there are a number of white papers on SEM systems and SEM implementations from commercial companies. However, for our continuing research outlined in Section 6, we require a SEM framework that we could control on a level that would exceed the control offered by a commercial implementation. For instance, the ability to swap the event correlation engine, or the ability to extend the SEM system through new software agents that are closely coupled to leverage the correlation engine or the topology mapping engine would be immensely beneficial. Primarily to gain such fine grained control of the system, we do not consider commercial implementations further in this paper.

Liu et al. [11] describe a SEM framework constructed using case-based reasoning. Their framework consists of four modules that bear similarity to the components we used in constructing our SEM system. The „Data Collection“ module uses agents to collect raw data by analyzing event log files and security log files . The „Data Standardization and Aggregation Module“ cannonicalizes the data from multiple sources into a format that the next module, the „Event Correlation and Scenario Analysis Module“ can operate upon. Finally, their „Quantization and Output Module“ uses data-mining techniques to compute the threat level of an event. Ertoz et al. [8] describe MINDS – Minnesota Intrusion Detection System, a project that also uses a suite of data mining techniques to automatically detect attacks against computer networks and systems. The traffic from the University of Minnesota is monitored using snort, an open source network intrusion detection system. The data is filtered and fed into the MINDS system, where the known-attack detection module detects attacks that correspond to known signatures. The remaining data is fed into an anomaly detection module, which assigns a score that reflects how anomalous the data is compared to

normal network traffic. Highly anomalous attacks are further categorized to create new signatures and models for emerging attacks; thus forming a feedback loop.

There is tremendous activity in the techniques associated with a SEM system, such as intrusion detection and data mining techniques to determine root cause analysis. Duan et al. [7] and Sekar et al. [15] suggest techniques to enhance intrusion detection systems to minimize the false alarm rate. Ning et al. [13] present a method for constructing attack scenarios through alert correlation and Julisch [10] observes that a few dozen persistent root causes accounts for over 90% of the alarms that an intrusion detection system triggers. They propose a novel alarm-clustering mechanism to identify the root causes of an alarm. Devitt et al. [6] uses the topological proximity approach that exploits topological information embedded in alarm data to filter out alarms that are not plausible from the point of view of the network topology.

In a SEM system, a standard protocol and an associated set of application programming interfaces (APIs) between the event correlation engine and the specific network component reporting the event is required. Debar et al. [5] have defined the Intrusion Detection Message Exchange Format (IDEMF) protocol that describes data formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to the management systems that may need to interact with them. The Intrusion Detection Exchange Protocol (IDXP), is an application-level protocol for exchanging data between intrusion detection entities [9]. IDXP supports mutual-authentication, integrity, and confidentiality over a connection-oriented protocol. The protocol provides for the exchange of IDMEF messages, unstructured text, and binary data. However, it is unclear how widespread the use of IDMEF and IDXP is; the protocols are considered experimental in nature.

More recently, Mitre has developed the Common Event Expression (CEE) which addresses the problem of vendors and products employ varying logging practices such as using inconsistent formats and terminology when describing events [12]. This alleviates the significant burden to analysts and products in normalizing the vast quantities of heterogeneous log records, allowing for aggregation, correlation, and further processing. At this time, there aren't any published papers on CEE to evaluate it in depth that the authors are aware of.

The work we present in this paper is orthogonal to intrusion detection, data mining and event reporting techniques. Indeed, the module in our SEM framework that correlates the event can be augmented (or even replaced) to take advantage of the results obtained by other researchers working in this area. In the same vein, even though the protocol used in our SEM to report events of interest to the event correlation engine is not standardized, it could in the future be replaced with a widely deployed one should such a protocol become available.

3 System Design and Architecture

Security threats have an exploitation cycle that permits a SEM system to act early enough to be effective; time is of essence during an exploitation cycle: if the SEM system is slow to react, the security breach would have already occurred. Our goal in constructing the SEM system was simple: we wanted to detect the discrete events as they build up to a security attack and acting consistent with the security policy, undertake mitigation actions before the attack could have an impact on the network resources.

System Components

To construct a SEM system that met this goal, we needed a strong event correlation engine. There are several examples of scalable event-based middleware: CEA [1], Sienna [2], JEDI [4] and TOPSS [3]. However, the common drawback with all of these frameworks is their lack of strong correlation capabilities. Correlating discrete events from multiple sources of information into one view is an absolute requirement and a cornerstone of an SEM system. From this correlation arises the ability to track the security event and automatically respond through rules that are triggered as certain conditions are met. A triggered rule results in an automatic response to contain the attack and alert the security operator so that human-decision making can be subsequently engaged.

In the end, we constructed our SEM system using a in-house developed high-performance fault manager as the core of the system to achieve distributed platform with good alarm thresholding and suppression features, a language for creating rules to correlate events of different types and from different sources and with network topology information from an integral topology database. The platform also provided flexible interfaces for collecting event information from network devices and good facilities for the customization of event collection and the reformatting of events into alarms. Another advantage of the in-house software was that it was integrated into a browser, complete with color-coding of alarms and the ability to send email or page an operator. With this piece of software, we essentially had Layers 1 and 2 (User interface and event correlation engine, respectively) from Figure 3 in place.

In order to collect the events, we used a variety of in-house and open source tools. For HTTP and SSH common log file analysis, we used an in-house log file analyzer. This analyzer attached itself to the log files to be monitored and if it detected an anomaly in the normal operation of the HTTP or SSH daemon, it would send a number of denunciations to the correlation engine. For HTTP, the analyzer software maintained a list of over 2000 known web server vulnerabilities catalogued by Common Vulnerabilities and Exposure dictionary (http://cve.mitre.org). If the software detected that a cracker was attempting to access a resource listed in the CVE database, it would immediately issue a denunciation to the correlation engine.

In addition to the static list of vulnerabilities, the analyzer is also tuned to dynamically perform statistical analysis on the requests to find attack patterns. For HTTP requests,

the statistical analysis observes certain characteristics, such as inter-arrival time, errors generated, and links accessed. Some other measures like bandwidth utilized can also be factored in to compute a weight for the IP address corresponding to the connection at the server. Depending on the value of the weight, the connection could get dropped, experience differential service, or allowed. Some weighting factors are exponential, for example, if a connection is using more than 75% of the bandwidth, it will get dropped. For SSH requests, the analyzer software looks for failed attempts from the same source IP address; if it detects such a pattern, it will issue a denunciation for that IP address.

We used snort for an intrusion detection system (IDS). Snort was configured to write messages to the system's */var/log/messages*. A Perl script constantly monitored the file and forwarded the messages of interest to the correlation engine as a security event. We also wrote a file signature verification software that was used to track changes to files and the addition or removal of files from a user's directory. Finally, we used a firewall to protect access to the core hosts in our network (see Section 4 for a network diagram of our laboratory setup) and as an additional security event detector in our network. Note that although we used a firewall developed internally and sold commercially, any other vendors firewall could be used. The firewall that we used provided session establishment rate limiting, traffic rate-limiting, and good detection and alerting features for invalid IP address and IP header content, and detection and alerting for TCP state violations. We used these features to provide an additional source of security events for the correlation engine. Collectively, the log analyzer software, the snort Perl logger and the firewall logger act as data collectors with reference to Figure 3.

Architecture

Figure 4 contains the SEM framework corresponding to the discrete parts we describe above. There are three datasources where external events are fed into the system. The events enter into the system in a canonical format, from where the data collector distributes the events to a system logger and a data distributor. The data distributor can be programmed to distribute these events to operation support interfaces or to other external destinations. The data distributor also sends the data to a correlation engine, which accepts the events and processes them according to rate, time, group or value thresholds to create specific alarms or to send a rule to another managed element (firewall, for example) for limiting (or allowing) access to an IP address or a range of IP addresses.

4 Using the SEM Framework

We constructed a laboratory network depicted in Figure 5 to test our SEM system. The network consists of four machines behind a firewall. Machine A is the primary attack machine that the cracker will try to break into. Machine B runs snort in promiscuous mode, thus it has access to all the traffic occurring on the subnet. Machine C is the management console for the firewall; it accepts directives from the SEM system

(Machine D) to disallow access for a set of IP addresses into the network. The cracker itself is assumed to be coming in from an external network.

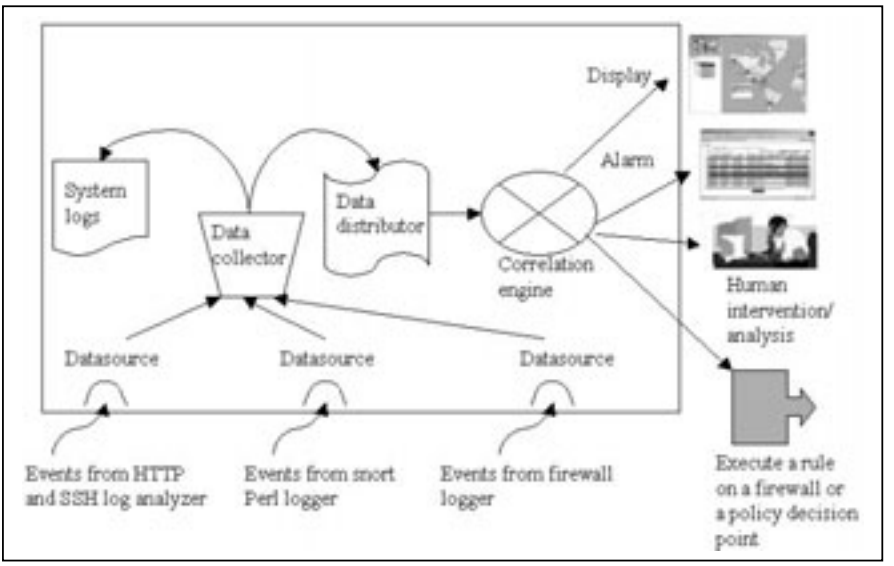

Figure 4: SEM Framework.

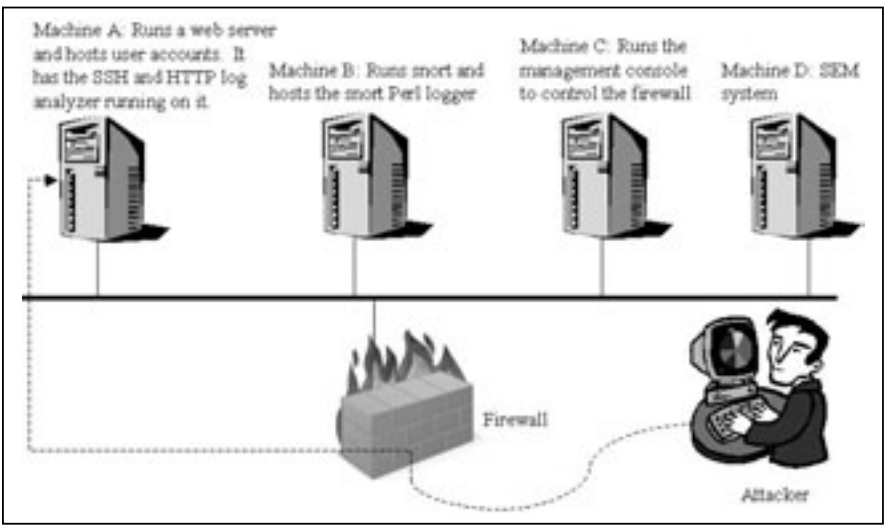

Figure 5: Laboratory Setup.

Typical security attacks follow a pattern: reconnaissance to find vulnerabilities, followed by a break-in, followed by expanding access to more system and network resources. This cycle is depicted in Figure 6(a). Mirroring this pattern, the attack proceeds in four stages. Note that in a deployed system, the attack can be stopped on the first occurrence of its detection. However, since we were interested in allowing the attack to proceed in a controlled environment, we at times permitted the attack to continue before having the firewall issue a block on the attacker's IP address.

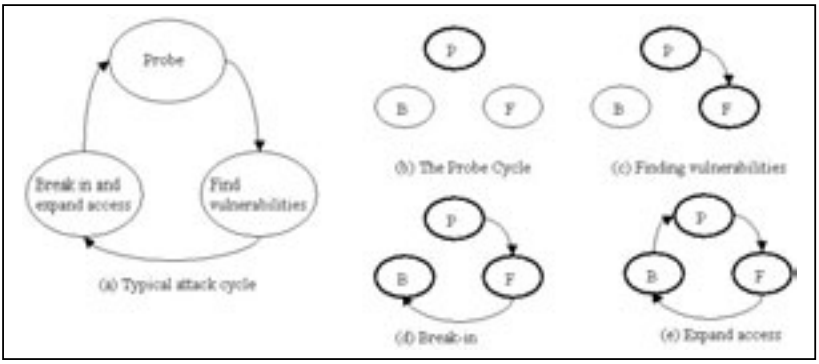

Figure 6: Attack cycle states.

Stage 1: The first stage is very preliminary; in the attack cycle of Figure 6, the attacker is in the probe cycle state, Figure 6(b). Here, the cracker attempts to reconnaissance the subnet using nmap (nmap, or „Network Mapper" is an open source tool for network exploration or security auditing; see http://insecure.org/nmap). When the attacker does this, many events are generated simultaneously: snort generates events about network scanning and sends them to the SEM system. In addition, the firewall also generates rate-limit violation alarms (i.e., incoming packet rate or session-establishment rate has exceeded a set threshold). All these events are sent to the SEM system, which has to correlate them and associate them with the same attack. Upon correlation, the SEM system issues a „Possible ICMP Probe" advisory. At this stage, we allow the attack to proceed into Stage 2.

Stage 2: Based on the output of the nmap probe, the attacker has now zeroed in on Machine A, the machine that hosts a web server . Using an open source web scanner called nikto (http://www.cirt.net/code/nikto.shtml), the attacker now tries to exploit a known vulnerability in the web server (in the attack cycle of Figure 6, the attacker is in the finding vulnerabilities state, Figure 6(c).) Nikto performs comprehensive tests against web servers for multiple items, including over 3300 potentially dangerous files/CGIs, versions on over 625 servers, and version specific problems on over 230 servers.
Since it is not designed as an overly stealthy tool, it's fingerprint is fairly obvious in log files. The HTTP log analyzer running on Machine A detects activity consistent with accessing well-known vulnerabilities; it sends a security event to the SEM system.

The SEM system correlates this event with those of Stage 1 and issues a denunciation to Machine C, which runs the management console to control the firewall. The denunciation results in the firewall blocking packets with the source address of the attacker. The attack has now stopped.

Stage 3: Undeterred, the cracker uses a new IP address to continue the attack. Having failed to exploit any vulnerabilities in the web server, the attacker now turns his attentions to the host itself. Using nmap, he performs a TCP and UDP port scan on Machine A. The snort instance on Machine B as well as the firewall itself generate security events; the snort instance generates a port scan alert event and the firewall generates a quality of service event. As in Stage 1, we allow this particular attack to continue and do not issue a block for the attacker's address.

Stage 4: Using the new IP address, the attacker performs a port scan of higher numbered ports. He finds out that a high-numbered TCP port (10022) is open, i.e., in listen mode. In many systems, users learning network programming or system administrators often keep servers running on a port. Many times, these servers are benign and don't do much damage if compromised; but at other times, the servers may inadvertently release information that will enable an attacker in mounting an effective attack. This was the case with the server on port 10022. When a connection is made to the server, it prints a listing of the home directory on the host. The attacker now has the login names of the users on that host. Using the login name, it is trivial to run a password cracker to successfully guess passwords of one or more user accounts. In the attack cycle of Figure 6, the attacker is now in the break-in state, Figure 6(d).

Armed with possible passwords, the attacker attempts to secure shell (SSH) into the system. Note that the inbound ssh connection is not detected by snort since is very well a normal occurrence in any network. (we could have configured the SSH log analyzer to issue a denunciation upon observing three failed login attempt for any user name to thwart such an attack; but in our scenario we did not do so and allowed the attack to proceed.) The attacker now has successfully logged into the system by guessing a password to an account by the name of „demo".

Since the reconnaissance and break-in have occurred, the attacker moves to the third state: attacking more systems (Figure 6(e).) To do this, the attacker uses the *wget* utility to download a malicious program from an external website at his disposal (since *wget* uses HTTP and most firewalls are configured to allow outbound HTTP connections to go through, this would not be tagged as an anomalous event.) The attacker retrieves the malicious program and tries to innocuously save it in the local directory. He then executes the program, which makes an outbound connection. As soon as an outbound connection was made to an arbitrary server, snort detects this and sends an unknown-outbound-connection event to the SEM system; the source port on machine A used for this connection is sent to the SEM system as well. The SEM system decides that this is an anomalous event and sends a directive to Machine A that causes it to perform a self-check (the directive includes the source port from which the connection emanates.) Using the *lsof* tool, a program on Machine A first figures out the owner of the source port; it subsequently runs the file integrity check on the home directory belonging to the

owner of that source port (account „demo“.) The integrity check uncovers the malicious program that the attacker had saved before and the SEM system is notified of this event.

At this point, human intervention is a must; the attacker has successfully breached the defenses and commandeered Machine A. The SEM system isolates Machine A by having the firewall drop all packets going to it or originating from it and notifies a human operator to intervene.

This scenario has demonstrated the need for a central SEM platform that can co-ordinate a response to a possible attack. Without such a platform, a network operator would have a coarse view of the events occurring in the network and may not be able to piece together the events that could prove to be a precursor to a larger attack.

5 Observations and Lessons Learned

There were several observations from constructing a SEM system that drive the lessons learned from this exercise.

Network fault management complements but is different from SEM

A fault management system and a SEM system are very similar in some of the key functions that they perform. They both receive, interpret, threshold, and correlate events from alarms from a wide range of devices. In fact, our SEM system was built upon an in-house high-performance fault manager containing an expressive correlation engine. Thus, it is interesting to ask whether a fault management system be transformed successfully in a SEM system. As a result of our work, we have reached a conclusion that while these systems perform similar operations on network events, security event management has some important needs that are not met by a network fault manager.

The primary function of the fault manager is to determine the underlying root cause of multiple received events, facilitate a remedy of the problem, and suppress the display of those events that represent symptoms of the root cause. A SEM system performs a similar function. An attacker may trigger the receipt of events of many different types from many sources and it is the function of the SEM system to determine the nature and source of the attack (the „root cause“) from these events, facilitate the mitigation of the attack, and suppress the reporting of those events that are symptoms or evidence of the attack.

However, while in fault management the root-cause fault might be intermittent and may even evolve into a larger problem with time, the root cause is generally static and the appropriate solution to address the fault remains valid once the root cause is found. Importantly, the root cause for the fault is normally a single component in a fixed geographical location. Furthermore, unless the root cause is a fire, flood, or other threat to the larger infrastructure, the damage caused by the event rarely escalates significantly with time. By contrast, a network attack generally proceeds in phases, often with

different attacking/probing sources and attacking/probing destinations as the attack evolves. The type and volume of events associated with the different phases of an evolving attack will change dramatically as the attack proceeds. The interpretation of the security events ("root-cause)" may proceed from "topology mapping," to "vulnerability probe," to "host vulnerability exploit," to "new bot detected," as the attack progresses. Consequently, there is an opportunity to detect the precursors to a full-fledged attack and an opportunity to mitigate the attack that has no common parallel in the fault management domain. If appropriate mitigation is not performed quickly, the damage can escalate dramatically with time.

Network attacks are dynamic in nature and the symptoms can be difficult to detect and threshold

There are great many possible categories of root causes for network problems, but a relatively small and predictable set of symptoms for identifying the root cause of a network fault. Consequently, a relatively static configuration for thresholding, event correlation and event suppression can diagnose these root causes effectively. The tools and interfaces for configuring a network fault manager reflect the relatively stable nature of the volume and type of the events expected. The threshold settings are relatively stable and generally require adjustment only as the network infrastructure or topology is modified. The correlation rules change infrequently and depend upon topology information in very simple ways. By comparison, network attacks and the precursor probes to these attacks come in fewer varieties, but the events that announce them can vary widely in type and in volume for different instances of an attack, and different phases of a single attack. A network probe can be brute-force and easily detectable or it can be carefully crafted to be difficult or impossible to detect. Consequently, it is impossible to create a set of event thresholds and correlation rules to reliably detect any but the hastiest and naive network scans and vulnerability probes.

Automatic or facilitated action is effective and valuable in addressing network attacks and is a valuable feature in a SEM system. A relatively small number of easily implemented solutions can address a wide range of network attacks. In contrast, because of the wide range of possible problems and the wide variety of network elements that can be the source of a root cause, it is not often feasible to mitigate or resolve network faults by automatic action, and automatic action is rare in all but the most specialized fault management devices.

Because the sources and varieties of network fault events are so open-ended and the collection of possible root-causes is essentially unbounded, a fault manager must provide the means for very flexible event interpretation and must be equally open-ended in the ability to define new kinds of alarm. To accommodate open-ended events and extensible alarm definitions, the mechanisms for event interpretation, thresholding, and correlation must be completely flexible and general. These requirements impose two important consequences for the internal architecture of these systems.

First, it is often necessary to create multiple alarms from a single event in order to apply multiple thresholding rules against different "views" of the event. Because the alarm

generation is so open-ended and thresholding and correlation mechanisms are necessarily very general, alarms are generated from events, and correlation alarms are generated from other alarms as arrays of name-value pairs in text form. As a result, as the alarm moves through the system, the arrays are repeatedly searched for field name matches, and the value fields are repeatedly parsed, and checked for correct syntax as they are communicated through the fault management system. Although this approach is very flexible and open-ended, it does not scale well as the volume of incoming events and the rate of thresholding and correlation operations increases.

Second, because the thresholding and correlation features of a fault manager are necessarily very general they are not always a good match to the capacity requirements of a SEM system. For example, the algorithms and data structures used within a fault management system for relating multiple failures to the failure of a shared network resource will probably not scale well when applied to the similar problem of detecting attacks originating from many sources or to many targets.

Event thresholding, correlation and mitigation must be distributed and pushed down to the lowest possible level

Because the volume of events received during an attack is generally much greater and less predictable than would normally be observed in a fault manager, without some care in the design of the detection architecture, the flood of events received during an attack can easily overwhelm the SEM system. Fortunately though, because security events are correlated into a smaller number of root causes, events can be correlated and summarized in a distributed manner forwarding a much smaller volume of event reporting to the central SEM system.

In a security event detection architecture, it is critically important to push event thresholding, correlation, and suppression down to the lowest possible level. For example, many firewall devices, including the one we used in our investigations provide features to set thresholds for traffic rate at the level of an individual session, a firewall rule, a particular network interface, or an entire rule set. In the firewall device we used, alerts can be generated for these events down to the level of rules but (for very good reasons) not to the level of individual sessions. In addition, the generation of logging records for session begin and termination can be enabled at the rule level. We found that if we were to naively enable these event reporting features and deliver the events directly to the SEM system, during a DoS attack or even during a benign traffic spike, the volume of information generated quickly overwhelmed the ability of the SEM system to digest it. Consequently, the logging features of the firewall device were typically employed very selectively and it is often not until the firewall reports that a resource threshold has been reached that the network operators are even aware that an attack has been underway. In a modern high-capacity firewall it can take hours to reach this condition.

Pushing thresholding and correlation features for DoS attacks down to the level of the firewall logging system would be a good, and necessary first step, but to address the problem effectively, we found that it would be necessary to implement new thresholding and correlation features within the firewall device itself. Furthermore, sampling of the event traffic and adaptive tracking of the top sources of firewall events would be necessary if we were to be able to perform this function effectively.

Security event records must be designed for SEM system consumption

When we applied firewall devices as security event detection points and forwarded the session events to the SEM system we discovered that the content and formatting of the event records was not well suited to interpretation by the system. Important session data was present in both the session start record and the session end record but there was no reliable identifier with which to associate the two records.

Furthermore, in the event that a session start record was discarded in the firewall by the automatic throttling mechanisms, important data would be lost. In our opinion, the best solution would be to provide the operator with the option of having the firewall device deliver all of the key fields in the session termination, and any intermediate event records. In addition, we decided that it was important to have every event record that was associated with a session contain an identifier to reliably associate those events. A similar example that we encountered was the way in which rules were identified in the session records with which they were associated. In the firewall device we used, the rule was named in the session record using the numerical order of the rule within the ruleset. The insertion or deletion of a rule would have the consequence of changing the identifier for the rule making it difficult to interpret the preceding session events for temporal ordering. In our opinion, having a reliable identifier to associate session records with the rules that enabled them is a key requirement for this class of devices.

Remediation of security events

Network operators whom we talked to during the early phases of constructing the SEM system recognize the value in having the security event management system propose and facilitate action to remediate security events. However, they have legitimate concerns about having such a system automatically generate and activate firewall policies and they have similar concerns about having the security event manager automatically block or terminate user sessions. Firewall policies often contain over 1000 hand-crafted rules and are modified infrequently and with great care. Automatic insertion and deletion of rules from these rule sets may be outside of the comfort level for many network operators. We have also observed reluctance to having the security management system automatically terminate sessions from traffic sources that have been judged to be sources of attack or egregious policy violation. Consequently, in addition to session termination and host blocking, we employed rate limiting on session establishment, threshold limits on session data rates, and redirection of sessions from suspect hosts to limit the damage that a suspect source can do. To address concerns about automatic policy generation, we employed modification of the host lists associated with source address matching on

certain rules to accomplish policy modification in a very constrained and well understood way. The redirection of sessions is accomplished using the policy-based routing features of the firewall. In our experimental network, we can redirect all or a subset of the sessions (e.g. HTTP/HTTPS) from the suspect sources through an intrusion detection system or into a sandbox server.

The firewall device that we have used in our work supports a device-independent zone-based security policy. With this design, a single collection of firewall policies are shared across an entire network by many firewall instances. This simplifies the remediation of attacks in a large network environment by enabling a single remediation action to be applied across the network avoiding the necessity to pinpoint the entry point of the attacker.

In summary, the lessons that we have learned from our work are:

1. Automatic action or operator-approved mechanized action for problem mitigation is a more feasible, valuable and desirable feature in a SEM system than it is in a fault manager.

2. Topology information is much more easily applied in a fault management system than it is in a SEM system.

3. Built-in correlation rules and analysis features for the detection and classification of security events is valuable in a SEM system but has no parallel in a fault manager.

4. A fault manager lacks the features necessary to efficiently correlate the events of a large scale DDoS attack.

5. Pushing down event correlation and suppression into detection devices (IDS), network element event logging systems, and network elements is crucial to effectively countering DoS and DDoS attacks in a large network.

6. Network elements that act as detection points may require modification of their event reporting mechanisms in order to work well as an element of a SEM system deployment.

7. The mechanisms provided for remediation of an attack must be designed with care to address the reluctance of the network operator to the introduction into the network of automatic policy control.

6 Agenda for Future Research in SEM Systems

The manner by which security events get to a SEM system today is largely ad-hoc; there isn't a formal language by which a SEM system can indicate interest in a subset of security events from the edge devices, nor is there a formal language for describing the events from the devices to the SEM system. While SEM systems may have a limited ability to reconfigure a device (allow or drop packets with certain IP addresses), they have no ability to query it pro-actively. Building SEM systems today amounts to performing integration work for the discrete pieces to communicate with each other. Continued research in SEM systems is essential to ensure that this is not the case in the future. In this section, we outline a research plan for SEM systems directed at solving the associated problems we have observed as a result of our work.

Better Network Reconnaissance Techniques

Much of the focus of today's system is on the detection of denial of service attacks. While still important, interest is rapidly growing in attacker activities that can be characterized as low-level scanning for network reconnaissance. Such activities are performed by skilled hackers or botnets as a means of understanding the vulnerabilities of a target network prior to exploitation. These and other activities are largely undetected today and collection of correlation techniques will have to be created to account for them.

Developing Resilient Protocols

Today, a SEM system is largely a one-directional system in that information is fed into the SEM. There is little in the way of two-way communications to, for instance, query or reconfigure detectors, or take action to reconfigure network devices to mitigate an attack. To do so, resilient protocols need to be developed for communication flow from the devices to the SEM system and from the SEM system to individual devices.

Designing a resilient protocols for communication is an open research problem in SEM systems. Ironically, it is precisely when a network is under attack that it may be least able to devote bandwidth resources for informing a SEM system. A protocol designed for communication from the discrete devices that are reporting their status to a SEM system thus needs to be resilient in the face of an adverse network. It should ensure that the information that is sent to the SEM is idempotent, self-contained and requiring minimal overhead in form of retransmissions and acknowledgements. For example, if five copies of a packet are sent by a device to a SEM system to increase the probability that at least one arrives at the SEM system, then even if the packet loss is 20% (and drop rates in today's network are typically under 5%) the odds are extremely high that at least one copy of the packet will get through. With a 20% packet loss and a five copies transmitted, assuming that the packet transmissions are spread out so that all losses are independent the odds are still greater than 99.6% that at least one copy gets through.

In the same vein as above, a resilient and universal protocol is required for the SEM system to impose policies and control the discrete devices in a network. Unlike the protocol discussed in the above paragraph, this protocol flows from the SEM system to the devices, thus it needs support in the individual devices themselves. Today, the commercial SEM system vendors depend on the device manufacturer to drive this communication. Thus, if the device can only be controlled using a proprietary protocol over TCP, then the SEM system would open up a TCP connection to the device and send it appropriate directives to cause it to change its behavior. As noted previously, a network under attack may not be quick to establish a TCP connection and to transmit directives over this reliable connection. Instead, what may be required here is the same mix of idempotency and multiple packet transmission as we discuss above. Furthermore, the protocol may well need to be standardized if it is to be implemented in a wide variety of edge devices.

Policy Languages and Rule-based Systems

Today's solutions are largely concerned with the event management of security devices such as firewalls, intrusion detection systems or intrusion prevention systems. The reality is, however, that every network device is a collector of information that may lead to a better understanding of what kinds of attacks are progressing. This information, if collected today, is usually done by enabling syslogs (system logs) for recording purposes. However, there isn't a formal mechanism to specify what information should be collected and how. We believe that a formal, universally implemented and verifiable mechanism is essential to for identifying security attacks in early stages.

Another area of research is the need for a security policy language for SEM systems. A robust policy language should be expressive enough to describe events of interest and to perform causal analysis. Related to this is the exploring the possible use of rule-based languages in a correlation engine: information that arrives at a correlation engine is typically in the form of facts („File X is accessed by IP address Y," „File X has been modified," „Web server is being scanned for vulnerability analysis from IP address Y," etc.) Once these facts are input into the system, it would seem rule-based logic langauge (like Prolog or the CLIPS expert system, see http://www.ghg.net/clips/CLIPS.html) should be able to run through its predicates to derive necessary alarms and denunciations for the remediation of attacks.

Device Modeling

Device modeling is yet another important aspect for further study in SEM systems. To achieve automatic and adaptive control of security monitoring and control points, it will be necessary for the SEM system to have knowledge of the characteristics of each of the controlled devices and of the location of each device in the network. Ideally, the SEM system should be able to query the capabilities and the physical location in the network of each of its detection and control points to be able to discover the best resources at each point in the network for monitoring network events and mitigating attacks. In order to do this, the security devices employed by the SEM system must be described to it in a

common model and be available to the system in a knowledge base that can be queried to adapt the monitoring and attack mitigation to changes in device inventory and network topology. The design of a device model that can encompass the full range of detection and control capabilities of these devices and the creation of a knowledge-driven system that can adaptively apply these resources, collecting feedback on the effectiveness of these actions to automatically improve behavior is a rich area for further exploration.

Correlation Rules and Network Topology

Correlation rules and the desired actions to mitigate an attack are inseparable from network topology. Specifics about the network topology, IP addresses, domain names, routing policy, network architecture, and network services are embedded in both the rules and actions that are encoded in a SEM system. Any change to network topology, therefore, affects and probably breaks the ruleset. Further research is required in this area to study how network topology impacts correlation rules, and to perhaps even allow for an automatic adjustment to the ruleset when the topology changes.

Integration with Operations, Administration, Maintenance and Provisioning

Most of the time, the output of an SEM system will be a rule that is enforced in the traffic admission component to mitigate an attack. But in some cases, an attack may simply consist of one packet that ends up crippling key servers such as a Session Initiation Protocol server (SIP is an Internet telephony signaling and services protocol [14]). For example, a documented attack in SIP consists of sending a request to a forking proxy that does not perform loop detection. As a result, the proxy is forced to maintain 2^{69} transactions in memory; clearly an untenable task [16]. If a SEM system observes the same SIP request traversing the network multiple times, it can restart the SIP proxy so that the proxy does not send out the looped request all over again. To this extent, is may be worth investigating into the feasibility of coupling of a SEM system with the OAM&P framework on a host.

Developing Human Computer Interaction for security (HCISec)

While one goal of a SEM system is to minimize human involvement, human input is still required. This includes entering network configuration information into various components, including the policy manager, detectors, correlators and simulators, defining policies and responding to alerts in cases where responses have not been automated. The APIs for modeling the network within the components and defining policies must not be overly complex or abstruse in order to limit inaccuracies and errors in human entered data. The manner in which alerts and supporting information is presented to humans must be carefully considered. One common problem today when presenting alerts to users is the quantity of information, both in the number of alerts and supporting information. The ease of using the system and the inclusion of safeguards against human error must also be considered. Approaches have typically followed that taken in network management where there is generally one root cause alarm and many

secondary and tertiary alarms. Processing has evolved to suppressing secondary and tertiary alarms, and displaying the root alarm to the user. While this approach has been suitable for network management because alarm hierarchy is fairly well understood at both the network and component levels, this is not appropriate, in general, for SEM systems where security events do not adhere to a known hierarchy and do not necessarily follow known sequences of events.

Bibliography

[1] Bacon, J.; Moody, K.; Bates, J.; Chaoying, M.; McNeil, A.; Seidel, O.; Spiteri, M.: Generic Support for Distributed Applications, IEEE Computer, 33(3), 2000, pp. 68-76.

[2] Carzaniga, A,; Rosenblum, D.; Wolf, A.: Design and Evaluation of a Wide Area Event Notification Service, ACM Transactions on Computer Systems, 19(3), 2001, pp. 332-383.

[3] Cugola, G.; Jacobsen, H-A.: Using Publish/Subscribe Middleware for Mobile Systems, ACM Mobile Computing and Communications Review, 6(4), 2002, pp. 25-33.

[4] Cugola, G.; Di Nitto, E.; Fugetta, A.: The JEDI Event-based Infrastructure and its Application to the Development of the OPSS WFMS, IEEE Transaction on Software Engineering, 27(9), 2001, pp. 827-850.

[5] Debar, H.; Curry, D.; Feinstein, B.: The Intrusion Detection Message Exchange Protocol (IDMEF), IETF RFC 4765, March 2007. Available online at http://tools.ietf.org/rfc4765..

[6] Devitt, A.; Duffin, J.; Moloney, R.: Topographical Proximity for Mining Network Alarm Data, Proc. of ACM SIGCOMM Workshop on Mining Network Data, Philadelphia, USA, 2005; pp. 179-184.

[7] Duan, Q.; Hu, C.; Wei, H-C.: Enhancing Network Intrusion Detection Systems with Interval Methods, Proc. of ACM Symposium on Applied Computing (SAC), New Mexico, USA, 2005; pp 1444-1448.

[8] Ertoz, L.; Eilerston, E.; Lazarevic, A.; Tan, P-N.; Kumar, V.; Srivastava, J.; Dokas, P.: MINDS – Minnesota Intrusion Detection System, Next Generation Data Mining, MIT Press, 2004.

[9] Feinsten, B.; Matthews, G.: The Intrusion Detection Exchange Protocol (IDXP), IETF RFC 4767, March 2007. Available online at http://tools.ietf.org/rfc4767.

[10] Julisch, K.: Clustering Intrusion Detection Alarms to Support Root Cause Analysis, ACM Transactions on Information and System Security, 6(4), 2003, pp. 443-471.

[11] Liu, L.; Li, Z.; Xu, L.; Chen, H.: A Security Event Management Framework Using Wavelet and Data-Mining Technique, Proc. of IEEE International Conference on Communications, Circuits and Systems, China, 2006; pp. 1566-1569.

[12] McQuaid, R.; Security Information Management for Enclave Networks (SIMEN), Mitre Corporation, unpublished. Available online at http://www.mitre.org/news/events/tech07/2606.pdf, 2007.

[13] Ning, P.; Cui, Y.; Reeves, D.; Xu, D.: Techniques and Tools for Analyzing Intrusion Alerts, ACM Transactions on Information and System Security, 7(2), 2004, pp. 274-318.

[14] Rosenberg, J.; Schulzrinne, H.; Camarillo, G.; Johnston, A.; Peterson, J.; Sparks, R.; Handley, M.; Schooler, E.: SIP Session Initiation Protocol, IETF RFC 3261, June 2002. Available online at http://tools.ietf.org/html/rfc3261.

[15] Sekar, R.; Gupta, A.; Frullo, J.; Shanbhag, T.; Tiwari, A.; Yang, H.; Zhou, S.: Specification-based Anomaly Detection: A New Approach for Detecting Network Intrusions, Proc. of ACM Computer and Communications Security (CCS), Washington, DC, 2002, pp. 265-274.

[16] Sparks, R.; Lawrence, S.; Hawrylyshen, A.: Addressing an Amplification Vulnerability in Session Initiation Protocol (SIP) Proxies, Internet-Draft, work in progress, March 2007. Available online at http://tools.ietf.org/html/draft-ietf-sip-fork-loop-fix-05.

Taxonomy of Anti-Computer Forensics Threats

Joseph C. Sremack, Alexandre V. Antonov

LECG, LLC
Washington, DC USA
jsremack@lecg.com
aantonov@lecg.com

Abstract: Threats to computer forensics are increasingly becoming more prevalent. Attacks against underlying forensic methodologies have come to the forefront during the past five years, in which attacks have become more sophisticated and difficult to prove in a court of law. These threats can negatively affect an investigation, if not completely stymie it. No complete taxonomy of threats to computer forensics posed by anti-forensic techniques currently exists. This paper attempts to construct a comprehensive taxonomy of anti-forensic threats by investigation and threat type.

1 Introduction

Recent developments in several critical areas have drawn attention to the threats facing computer forensics. Advanced anti-forensics techniques are being developed in practice [Gr02] and [PR05], as well as theoretically [Se06]. These anti-forensic techniques can threaten a computer forensic investigation in its entirety. In addition, changing requirements in US and EU case laws are impacting the existing processes and foundations of computer forensics and allow for a new breed of anti-forensics in which case law can be used against the forensic investigator.

Computer forensics, as a whole, is a relatively young field. Compared to traditional forensics, computer forensics is especially young. Traditional forensics has been establishing its body of knowledge and best practices for more than one hundred and thirty years. Compare this to computer forensics, where the practice has largely been performed for only the past two decades.

While traditional forensics is still rapidly developing, little work or examinations are being done with respect to threats. Threats have largely been identified and classified in the areas of security and traditional forensics. Many models and taxonomies have been established to not only identify, but to classify and rank those threats accordingly. Some efforts have been made [Bu06] and [Ha06], but computer forensics has not established these threats in a complete and organized manner. This deficiency can lead to problems in which an investigation can be thwarted by any number of undocumented threats.

The threats facing computer forensics are multi-faceted. Legal, technical, and physical factors must all be considered with respect to the threats. The combination of these factors differs from the mostly technical threats accounted for by security threat models. In addition, threats in computer forensics differ based on the jurisdiction(s) in which an investigation takes place. Differing case law and other jurisprudence factors means that a threat facing a solely US-based investigation may or may not apply to an international investigation.

In addition to the threats differing, the types of investigations can differ. Three major types of investigations are performed: internal, criminal, and civil. Internal investigations are performed outside of a court of law and are typically performed within a single organization to respond to some event. Criminal investigations are performed when prosecuting one or parties for a criminal offense. Civil investigations are performed within the realm of a court of law, but they are conducted in order to settle a civil dispute rather than for prosecution. Threats, for each of these investigation types, differ according to the requirements for proof and what constitutes proper evidence.

In this paper, a taxonomy of threats to computer forensics is developed. To achieve this, threats to computer forensics are defined based on the multitude of factors and types of investigation. Since the threats are specific to each combination, a general taxonomy is developed. Practical considerations for applying this taxonomy are then shown. Section 2 discusses the overall requirements for computer forensic investigations. Section 3 briefly outlines the three types of computer forensic investigations and how they differ. Section 4 discusses the current state of anti-forensics, and how practitioners and theoreticians are developing techniques and approaches to defeating forensic techniques. Section 5 is the taxonomy of anti-forensic threats. Section 6 provides a case study that highlights several of the threats. Section 7 provides concluding remarks and future work on this topic.

2 Computer Forensic Investigation Requirements

Most computer forensic investigations follow a general structure. The first phase of every investigation is preparation. This phase establishes the overall plan for acquisition and analysis to ensure that all data are completely and correctly acquired. The next phase is collection, in which all data are acquired. Analysis is then performed against the collected data. Finally, the presentation phase is performed in order to present the analysis in a clear manner to the adjudicating body (court of law or otherwise). This section outlines the general requirements for each phase and the overall investigation.

2.1 Preparation Phase Requirements

The preparation phase includes all steps necessary to ensure that a complete and correct investigation is performed. This entails accounting for all data that should be acquired, the analysis that is expected to be performed, and how the findings should be presented. This stage is critical and often begins based on nature of the event. For example, a hacked database server would lead an investigator to capture all evidence related to accessing the database server.

There are five requirements for the preparation phase. The first requirement is that the full scope of the investigation be understood. Understanding the scope of the investigation consists of learning the timeline of events leading up to, occurring during, and occurring after the incident or event of interest. This information can arise from system documentation, legal documentation, and/or verbal discussions. The second requirement is the interview process in which all pertinent and accurate information is received relevant stakeholders and witnesses. The number of witnesses and stakeholders varies based on the investigation, where it could be a single person or consist of tens to hundreds of people. From the information gained through interviews and background material, the next requirement is to determine all data points that need to be acquired and how they are to be acquired. Next, the means for analysis must be determined. The analysis may be a simple keyword search across a single hard drive, or it may be complex steganalysis. While the analysis plan may change once the data has been acquired, setting up an analysis plan *a priori* is helpful in organizing and streamlining the analysis. Finally, the venue for presenting the analysis findings must be known. The analysis may need to be presented in court, or it may only be shared internal to a company. Each venue has its own requirements for rigor and presentation, and the analysis should be aligned with those requirements.

2.2 Collection Phase Requirements

The collection phase involves acquiring all data and verifying that the data were properly and fully acquired. There are four main collection phase requirements. The first is that all relevant data be acquired. The second is that the acquired data be verified through some means, such as matching hash values. The third is that the entire process be fully documented. Finally, chain of custody must be maintained and demonstrated through documentation.

The analysis phase is where all data are analyzed based on the investigation plan. The analysis phase consists of six major requirements. The analysis, most importantly, must be performed completely and accurately. The second requirement corresponds to the first; evidence should be cross-verified with other evidence, where appropriate. For example, router log entries should be cross-verified with file server logs. Industry best practices should be employed. Fourth, court-tested tools and techniques should be employed over novel or otherwise untested tools and techniques. The entire process should be documented, and finally, chain of custody must be maintained.

2.4 Presentation Phase Requirements

The presentation phase involves culling the relevant data and assembling in a logical manner to present to the adjudicating body. The presentation phase requires that all relevant information be presented clearly and that the analysis conform to the rules of admissibility of the adjudicating body.

2.5 Overall Investigation Requirements

All phases of an investigation must be performed so that the following requirements are met:

- Follows rules of admissibility
- Findings are convincing and based on court-tested industry best practices
- Full process is documented
- Performed in a reasonable amount of time

3. Types of Computer Forensic Investigations

Three main types of computer forensic investigations exist: internal, criminal, and civil. These types all have different purposes and levels of required rigor. Internal investigations are performed outside of a court of law and are typically performed within a single organization to respond to some event. Criminal investigations are performed when prosecuting one or parties within a court of law. Civil investigations are performed within the realm of a court of law, but they are conducted in order to settle a dispute rather than for prosecution. All three types try to answer the following questions:

1. Who (e.g., who were the sources of the event?)
2. What (e.g., what exactly was the event?)
3. Where (e.g., on what systems and at what locations did the event occur?)
4. When (e.g., what is the timeline of events?)
5. How (e.g., what conditions allowed the event to occur?)

Internal investigations are typically performed in order to resolve an event outside of a court of law. The rigor required for collecting and analyzing evidence for an internal investigation is less, because the requirements for evidence is based solely on the interested party and no other third parties. The first goal is to determine the cause(s) and source(s) of the event. The next goal depends on the event. Containment and remediation may be all that is required of the investigation if the event is small and does not require disclosure. In other instances, determining the exact source of the event is the prime objective. Companies operating in the State of California - online or otherwise - must disclose all hacker incidents that may affect customer information [Ca02]. In cases such as this, which affect customer records or may later require legal intervention, greater care must be given. The full scope of the event must be determined and reported accordingly.

Criminal investigations are presented to a court of law and require that the evidence prove, beyond a shadow of a doubt, the source(s) of the event. A criminal investigation's findings cannot leave any doubts as to who committed the crime and how. These cases are handled by law enforcement professionals.

Civil investigations are also presented to a court of law for adjudication. There are two main differences between civil and criminal investigations. First, criminal cases involve a party(ies) having broken a societal law, whereas civil cases allow citizens to protect their individual rights in court. This difference is telling with respect to the length of the investigation and the types of evidence involved. Second, civil cases only require a preponderance of evidence, instead of the criminal cases' requirement of being proven beyond a shadow of a doubt.

4. Anti-Forensics

Anti-forensics is the practice of thwarting a proper forensic investigation. Any activity that intentionally aims to deceive or impede the forensic analysis is classified as anti-forensics. There are two classes of threats posed by anti-forensics: threats to digital evidence and threats to the legal process.

Anti-forensics is typically considered from a digital evidence perspective. The four main types of threats to digital evidence are data preservation, data counterfeiting, data hiding, and data destruction. There are two main subclasses of threats to digital evidence: physical and technical. From a technical perspective, data hiding and data destruction techniques have existed for some time in the hacker community, e.g. log deletion. In 2002, the first paper was published on intentionally deleting data to avoid detection by a well-known forensics software [1]. Data preservation is the process of ensuring that no forensic evidence is created, be they newly created evidence or the alteration of existing evidence. These techniques have been published recently in which software are loaded into memory for execution and are subsequently wiped from memory upon completion [2]. Data counterfeiting is the process of creating false and/or misleading data. These techniques have been known for many years and include techniques as simple as creating false log entries.

The same four threats to digital evidence exist from a physical perspective. First, media preservation can be achieved through write blocking devices and other means of ensuring that the media are not altered. Media destruction can be performed through the use of chemical, magnetic, and mechanical means. Mechanical threats, such as the use of a hammer or a knife, the use of a magnet or by a chemical solution can all destroy the media. Media replacement is done by replacing the affected system(s) with replicas. The other physical threat is displacement, in which the devices or media are stolen or otherwise moved.

Legal anti-forensics techniques exist as well. Legal doubt can be intentionally created in order to avoid prosecution. If it is not clear who performed an event or how it occurred, prosecution is difficult, especially in criminal cases. Crossing jurisdictions increases the difficulty to bring a matter to court, as well as increasing the difficulty of acquiring all evidence. Additionally, privacy laws, theoretical doubts (e.g., theoretical breakthroughs against MD5), and creating new legal precedents pose threats to investigations.

5. Taxonomy of Threats

Two major categories of threats to computer forensics by the effect they have on the forensic environment exist: threats to digital evidence and threats to the legal process.

5.1 Threats to Digital Evidence

The class of threats to digital evidence includes potential actions that can negatively affect the goals of the investigatory process by compromising digital evidence. There are the following subclasses of technical threats:

- Evidence Preservation: prevention of creation data that later may be used as evidence.
- Evidence Destruction: destruction of data that later may be used as evidence.
- Evidence Hiding: taking special steps to prevent investigators from accessing data.
- Evidence Counterfeiting: creation of misleading digital evidence.

All four subclasses can be divided into technical and physical threats by the way these threats affect the evidence. Technical threats are projected through software, while physical threats are projected through processes outside of the computer logic.

Table 1 provides examples of such threats.

Class	Subclass	Example
Evidence Preservation	Technical	Prevention from writing to hard drive.
Evidence Preservation	Physical	Installation of data gathering equipment that does not communicate with host network, such as a silent sniffer.
Evidence Destruction	Technical	Deletion of log file entries.
Evidence Destruction	Physical	Chemical, magnetic, mechanical destruction of media containing evidence.
Evidence Hiding	Technical	Use of encryption or steganography.
Evidence Hiding	Physical	Use of smart cards or hardware cryptographic modules.
Evidence Counterfeiting	Technical	Creation of misleading log file entries.
Evidence Counterfeiting	Physical	Physical replacement of system hard drive with a ghost image of the original hard drive with non-incriminating digital evidence.

Table 1: Examples of threats by class and subclass.

5.2 Threats to Legal Process

The class of threats to the legal process includes potential actions that can negatively affect the goals of the investigatory process by providing legal obstacles to collection, analysis, or presentation of digital evidence. The following are subclasses of threats to legal process:

- Sufficient Doubt: rules, regulations, and steps taken that ease the process of creation of sufficient doubt regarding the collected digital evidence.
- Privacy: rules, regulations, and steps taken that ease the process of denying the prevention of collection or presentation of digital data due to privacy concerns, which creates problems for the authentication of evidence,
- Cross-Jurisdictional Nature: rules, regulations, steps taken that prevent a forensic investigation started in a given jurisdiction to successfully obtain evidence from another jurisdiction.
- Significant Changes in Scientific Foundation: the threat is posed by scientific research that changes the environment that the proceedings are relied on. Such environmental changes may make the process of collection, preservation, and analysis of digital evidence unacceptable or foundationless.

The following provides examples of threats to the legal process:

Sufficient Doubt: Perform a crime from publicly-accessible or virus-infected computer; use of repudiation techniques such as communication through public forums or "mixed networks;" the creation of legal precedents.

Privacy: European Union laws prohibit transfer of personal data of EU citizens to outside of the EU.

Cross-Jurisdictional Nature: Performing crime from a jurisdiction with no extradition and no working relationship with the law enforcement of the target jurisdiction.

Significant Changes in Scientific Foundation: Recent proofs of weakness in SHA-1 and MD5 algorithms [Sc05].

6. Case Study

While several of the threats are widely known in practice, some are rarely seen in the field. This section presents an example of a case in which some of the less common forensic threats were encountered. The case involved a US-based company who was facing internal intellectual property theft, where evidence destruction and hiding, as well as cross-jurisdictional issues were occurring during the internal investigation.

A US-based medical device company ("A") who had a principal founder leave the company to start a competing company ("B") outside of the US in a country whose extradition and data export laws are known to be difficult for investigations. Because "B" was located in a country that is known to be uncooperative, evidence could not be gathered directly by "A." This legal impediment meant that "A" had to internally handle the intellectual property theft and stop future theft from occurring. This is an example of a legal process threat involving cross-jurisdictional issues.

The initial scope of the investigation was to determine employees who were stealing intellectual property and providing them to "B." The suspected employees consisted of executives, R&D scientists, and IT staff. The first step was to install sniffer devices for email and instant messages. "A" had three locations, one in the US, one in the EU, and one in Latin America. Sniffers were installed at all three locations, and current server email files were acquired.

Initial analysis of the email confirmed the belief that some of the original suspected employees were involved in suspicious emails, though those emails were not themselves smoking guns or sufficient evidence for employment termination. Analysis began on the instant message traffic via keyword searching. Instantly it was noted that data from one of the locations was missing. "A's" Latin American site's IT staff had stolen the device (as well as other equipment) before leaving the company, which prevented traffic analysis via this form of physical evidence destruction. The instant message traffic was surprisingly devoid of any relevant evidence or clues.

Further email analysis was performed – this time focusing on two pieces: the IT staff personnel who may have stolen the sniffer device and any correspondence relating to tipping off other employees of the internal investigation. Keyword searches, such as "investigation" and "instant message," yielded some results that allowed "A" to hone in on a specific set of employees, whereby those employees' hard drives were acquired and analyzed. Although the investigation did yield results, evidence was lost and additional intellectual property was most likely lost due to anti-forensic techniques and issues.

7. Conclusion

Anti-forensics poses a large challenge to each type of forensic investigation. As advancements in anti-forensics are made, the computer forensic body of knowledge and best practices must adapt. In order to do so, an overall taxonomy of threats is required. These threats must then be accounted for and properly classified. This paper provides the overall taxonomy and means for classifying future threats.

There exist two major categories of threats: threats to digital evidence and threats to the legal process. Threats to digital evidence are based on four classes: preservation, destruction, hiding, and counterfeiting. The subclasses for these threats are physical and technical. In addition, threats to the legal process exist. These threats are cross-jurisdictional, creating sufficient doubt, significant changes in scientific foundation, and threats based on privacy.

7.1 Future Work

Future work on this topic includes several steps. First, the development of a full taxonomy of threats, both anti-forensics and unintentional should be created to account for all possible threats. This paper focused on the intentional threats by an adverse party. Outside threats exist in which inherit weaknesses in the process, such as MD5 and chain of custody can prove to be problematic.

The second work is the creation of controls to account for and mitigate the threats posed to computer forensics. As in security, controls can be created for computer forensics. The threats posed to computer forensics are not insurmountable, and as such, they should be controlled through the methodical creation of better software and processes.

The third extension of this work is research into a third subclass of threats, namely "social" anti-forensic threats. Social threats include many of the same threats and themes as social threats to computer security. Collusion, for one, can have a negative impact on a forensic investigation if one person warns others involved in an incident to cover their digital tracks. Incorporating social activities and other non-digital evidence into computer forensic research can provide valuable insights for practitioners.

Bibliography

[BM06] Burmester, M.; Mulholand, J.: The Advent of Trusted Computing: Implications for Digital Forensics. 21st ACM Symposium on Applied Computing, Computer Forensics Track. Dijon, France. April 2006.

[Ca02] California Senate: California Database Breach Act (SB 1386). 2002. Online athttp://info.sen.ca.gov/pub/01-02/bill/sen/sb_13511400/sb_1386_bill_20020926_chaptered.htm

[Gr02] the grugq: Defeating forensic analysis on unix. Phrack 59, July 2002. Online at http://www.phrack.org/archives/59/p59-0x06.txt

[Ha06] Harris, R.: Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem. The Proceedings of the 6th Annual Digital Forensic Research Workshop. 2006; pp 44-49.

[PR05] Pluf; Ripe.: Advanced Antiforensics: SELF. Phrack 63, July 2005. Online at http://www.phrack.org/archives/63/p630x0b_Advanced_Antiforensics_and_SELF.txt

[Sc05] Schneier, B.: SHA-1 Broken. Februrary 15, 2005. Online at http://www.schneier.com/blog/archives/2005/02/sha1_broken.html

[Se06] Selinger, P.: MD5 Collision Demo. 2006. Online at http://www.mscs.dal.ca/~selinger/md5collision

Testing Forensic Hash Tools on Sparse Files

Harish Daiya
IIT Kharagpur, India
hady.iit@gmail.com

Maximillian Dornseif
Hudora GmbH, Germany
md@hudora.de

Felix C. Freiling
Laboratory of Dependable Distributed Systems
University of Mannheim, Germany
freiling@informatik.uni-mannheim.de

Abstract: Forensic hash tools are usually used to prove and protect the integrity of digital evidence: When a file is intercepted by law enforcement, a cryprographic fingerprint is taken by using a forensic hash tool. If later in a court of law the identical fingerprint can be computed from the presented evidence, the evidence is taken to be original. In this paper we demonstrate that most of the freely available forensic hash tools fail to support this conclusion at the file system level for *sparse files*, a particular class of files in Unix systems that contain holes. We describe an experimental setup by which existing and future hash tools can be easily tested for this border case. In conclusion, we argue that further efforts are necessary to test and validate common forensic hash tools so that the significance of their results can be better judged.

1 Introduction

Computer forensics deals with analysis and investigation of computer systems using scientific methods to test whether or not the system was used for unauthorized activities. One of the first steps in forensic investigations is the preservation of digital evidence. If this evidence comes in the form of a file system on a magnetic disk, *forensic imaging* is one of the first steps performed. Forensic imaging is the process in which exact copies of file systems are made. Forensic analysis can then performed on the copy of the data and not on the original to preserve the integrity of the evidence. To be valid in a court of law, examiners must argue that the copy is identical to the original evidence. This is done today by using *forensic hash tools*, i.e. tools based on cryptographic hash functions. Roughly speaking, such tools compute a unique and unforgeable fingerprint of a file system. File systems with identical content should have the same fingerprint while different file systems should have a distinct fingerprint.

Since evidence collected during a forensic investigation can be used in a court of law, it is mandatory to have high confidence in the correctness of such forensic hash tools. One way to increase this confidence is to test the forensic tools under stress and using border cases as inputs. Such border cases should contain incomplete inputs, malformed file structures, corrupted directory structures etc. Only if the tools behave predictably and

in their intended manner in all these cases should they be acceptable in court.

1.1 Previous Work on Forensic Tool Testing

The *Computer Forensics Tool Testing Project* (CFTT) [10] is a joint project of the United States' National Institute of Justice, National Institute of Standards and Technology, and other agencies, such as the Technical Support Working Group. The objective of the CFTT project test the accuracy of the computer forensic tools and increase the confidence in them so that they can be used by the forensic community and legal organizations during digital investigations. The results of the tests done as a part of the project also provide useful information to the tool makers to improve the tools.

The activities of forensic investigations under CFTT is divided into catogories, such as hard disk write protection, disk imaging, string searching, etc. They then develop a test methodology for each category and select a tool in each category. The test results are posted on NIJ's website. Disk imaging tools like dd, SafeBack, EnCase have been tested. Write block tools like RCMP HDL and PDBLOCK have also been tested. Currently the project is working on developing a test methodology for testing deleted file recovery tools.

Apart from the CFTT project, the *Digital Forensics Tool Testing Project* (DFTT) [3] has developed several small test cases for testing various forensic tools. The test images for all the tested tools are available on the website. Some such test cases include the keyword search test for Ext3, Fat and NTFS file systems. This test focuses on searching an ASCII string in a file. The goal is to identify which tool can find different types of strings in a file. For example, a string can cross between end of a file into the slack space of the file. Some tools can identify this others might not.

Forensic hash tools have neither been tested by DFTT nor has been any such test been posted on CFTT.

1.2 Contributions

In this paper we report on experiences in investigating forensic hash tools on the border case of a very simple and common file structure: *sparse files* which are also known as *files with holes*. Briefly spoken, sparse files result from using the `lseek` system call in Unix operating systems. Using hash tools in the context of sparse files can result in surprising results, as we now explain.

The experimental setup was as follows. On a given file system we created two files: The first file `withhole` was a sparse file that contained a hole, i.e., the beginning and the end of the file contained a fixed bitstring while the transition from beginning to end was performed using the `lseek` system call. The second file `withzero` contained the same bitstrings at beginning and end while the intermediate part of the file was filled with zeros. Obviously, these two files have different representations on the raw hard disc while many

operating systems will treat both files similarly at the system call interface, so it is not immediately clear how forensic hash tools should treat them. We tested a variety of forensic hash tools (`md5sum`, `md5deep`, `sha1sum`, `sha1deep`, `tigerdeep`, `sha256deep`, `whirlpooldeep`) on these files for a variety of file system types (`ext3`, `reiserfs`, `vfat`, `jfs`, `minix`, `ext2`, `msdos`). In summary all of the hash tools computed the same hash for both files. However, the file size of the two files (as reported by invoking the Unix command `du`) was different for all file systems except `vfat`, `minix` and `msdos`.

At first sight the above results may lead to the conclusion that all of the common forensic hash tools have problems when confronted with sparse files on most file systems. However, at second sight the problem is more complex: As pointed out by Carrier [2, p. 10f], data analysis can be performed at different levels of abstraction. For example, data can be analyzed directly on the *physical storage medium* (the raw hard disk sectors), within a *volume* (a collection of sectors accessible to an application), within a *file system* (a collection of data structures that allow an application to create, read, and write files). Forensic imaging, which is where hash tools are commonly used, is performed at the physical storage medium level or the volume level. Sparse files are a feature supported at the file system level and so the results of our study do *not* directly affect standard practices of digital forensic investigations. However, we feel that investigators should be aware of the pitfalls which arise from using the *same tools* on *different analysis layers*.

The set of sample file system images is publicly available for download [4].

1.3 Paper Outline

The paper is structured as follows. We first recall the principles of cryptographic hash functions and forensic hash tools in Section 2. We then explain the concept of sparse files in Section 3. Section 4 reports on the experiments we performed using forensic hash tools on different file systems. Finally, Section 5 summarizes our results and concludes the paper.

2 Hash Functions

A cryptographic hash function is a mathematical function which satisfies certain properties to make it suitable for use as primitive in various information security applications, such as authentication and message integrity. In this section we recall these special properties, explain their importance in forensic investigations and enumerate the most relevant hash algorithms used in practice.

2.1 Properties of Hash Functions

A cryptographic hash function H maps an arbitrary size input string x to a fixed size output string y. The output y is usually called the *message digest* or *fingerprint* of x.

The security properties which H must satisfy are as follows:

- H must be a *one way function*, i.e., for a given hash value y it is computationally infeasible to find a message $y' \neq y$ such that $H(x) = H(y)$.
- H must be *collision free*, i.e., it is computationally infeasible to find two different messages x and x' such that $H(x) = H(x')$.

2.2 Usefulness of Hash Functions

Since the input of a hash function can be of arbitrary length and the output is of fixed length, in principle there must exist distinct files that have the same fingerprint. However, because of the security properties of the hash function it is very hard to find such two files. Therefore it can be safely assumed that distinct files result in different fingerprints.

One of the most important purposes of hash functions is to check the integrity of files and file systems. For this, a hashing tool that implements the hash function initially computes the fingerprint of a file or file system. This fingerprint is recorded in a place where integrity can be assured (e.g. the investigator's paper notebook or a printed report). In case the integrity of the original files is questioned, an investigator or analyst can run the same hash tool on the files. In case the fingerprint is equal to the one initially recorded, the evidence is original with very high probability. The alternative to using hash functions is to store a complete copy of the original files in a protected place. Especially for large data sources this is very inconvenient.

2.3 Hash Functions and Tools

We now give an overview over the most common hash functions in use today.

MD5 The MD5 (Message Digest 5) family of hash functions was developed by Rivest [5]. The output size of MD5 is 128 bits. Many tools implement the MD5 algorithm, e.g., `md5sum`, `md5deep`. In contrast to `md5sum`, the tool `md5deep` can create MD5 hashes of whole directory trees and is extensively used for forensic puroposes

SHA The SHA (Secure Hash Algorithm) was designed by the National Security Agency (NSA) and published as US government standard [6, 7]. There are several different instantiations of SHA, e.g., SHA-0, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512. The tools used were `sha1sum`, `sha1deep` and `sha256deep`.

TIGER The Tiger family of hash functions was designed by Anderson and Beham in 1995 [8]. This family can produce hashes of lenghts 128 bits, 160 bits and 192 bits. The tool used to produce the message digest is called `tigerdeep`.

WHIRLPOOL The Whirlpool family was developed by Rijmen and Barreto [9]. The hash is 512 bits in length. We used the tool `whirlpooldeep` to produce the message digest.

3 Sparse Files

We now recall the concept of sparse files. A file is sparse if it has unallocated blocks i.e holes. The reported size of such files is always larger than the actual disk space consumed by them. The Unix disk untility `du` (disk usage) by default gives the real disk space consumed by a sparse file (i.e. the space used by disk blocks), while `du` with the option `--apparant-size` gives the size of a file after reading it through the system call interface. The holes are always read as zeros. All major Linux filesystems like ext3/2, JFS, reiserfs, etc. supports sparse files. But some file systems like ISO 9600 CD-ROM filesystem does not. Later in this paper we will see a program that can be used to create a sparse file.

Overall we can say that it is easy to create files with holes, but it is difficult to distinguish them at the system call interface by reading and comparing then. If read, the holes will be treated as zeros. Using the Unix utility du it is always possible to find whether a file is sparse or not. In this sense, sparse files are a hybrid concept: meant to be transparent at the system call interface, but distinguishable using other means.

4 Testing Forensic Hash Tools on Sparse Files

We now describe our experiments. First we explain the experimental setup, i.e., how we created different file systems and how we created test files (sparse and non-sparse). Then we enumerate the test results for different hash tools and different file systems.

4.1 Experimental Setup

All the analysis and tests are done on separate file system images of different types. To set up these images, we used the Unix `dd` command to create a disk partition within a file. Then we formatted the (partition) file to establish a file system of a particular type. Using the loopback device option of the `mount` command it is then possible to mount the file system like a regular disk partition into the directory tree and to create a number of test files.

A common sequence of the necessary commands looks like this:

- Create a 10 MB image containing zeros:

```
# dd if=/dev/zero of=image bs=1M count=10
```

- Format it as for e.g. the `ext3` file system:

```
# mkfs.ext3 image
```

- Mount the image using the loopback device:

```
# mount -o loop image /mnt
```

We created file system images for the following file system types: `ext3`, `reiserfs`, `vfat`, `jfs`, `minix`, `ext2`, `msdos`. On these file systems we created two different but similar files:

- A sparse file named `withhole` containing the following data:
  - The first 10 bytes of the file contain the string `beg_string`.
  - Then a fixed number n of bytes is skipped using the `seek` system call. In our experiments we used $n = 100000$.
  - The final 10 bytes of the file contain the string `end_string`.
- A non-sparse file named `withzero` containing the following data:
  - The first 10 bytes contain the string `beg_string`.
  - Then n null characters are written to the file (ASCII value 0).
  - The final 10 bytes of the file contain the string `end_string`.

We used the following program in Figure 1 to create both files.

Counting the number of bytes and skipped byte positions, both files have exactly the same length. Our aim now is to see the behaviour of hash tools while dealing these two files.

4.2 Testing of Tools

All the mentioned hash tools (`md5sum`, `md5deep`, `sha1sum`, `sha1deep`, `tigerdeep`, `sha256deep`, `whirlpooldeep`) were tested by comparing the hash or message digest produced by them for the two created files. The message digest of the two files was exactly the same for all the tools. We also tested whether the two files were distinguishable at the

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<string.h>
#include<sys/stat.h>
#include<fcntl.h>

/* set up variables */
#define beg_str   "beg_string"
#define end_str   "end_string"
#define withhole  "withhole"
#define withzero  "withzero"
#define n         100000        /* size of hole */

int main()
{
  int offset=n, fd1, fd2, i;
  char *buff1 = beg_str;
  char buff2[offset];
  char *buff3 = end_str;

  fd1=creat(withhole, 0);
  fd2=creat(withzero, 0);

  /* create common header */
  write(fd1,buff1,10);
  write(fd2,buff1,10);

  for(i=0;i<offset;i++)
    buff2[i]=0;
  lseek(fd1,offset,SEEK_CUR);
  write(fd2,buff2,offset);

  /* create common trailer */
  write(fd1,buff3,10);
  write(fd2,buff3,10);
  return(0);
}
```

Figure 1: Program to create files used in experiment.

level of the file system. In most cases this was the case, i.e., the disk usage of the two files was different.

To test the disk usage of the two files, we used the Unix command `du` which shows the number of bytes allocated for the file at the file system level. This can give different results for sparse and non-sparse files as shown below. There is a special option `--apparent-size` of `du` that shows the apparent size of a file, i.e., a possibly larger value due to holes, internal fragmentation, indirect blocks etc. For example, a 3KB file with holes in it with `du --apparant size` will show the size as 98KB which is same if these holes were read as zeros. As explained before in many file systems there is no special provision for reading a hole. Whenever it is read it is interpreted as zero.

4.3 Test Cases

We now enumerate the investigated file systems. The hash values for the different hash tools as well as the MD5 hash of the complete file systems are given in the summary below.

Ext3 This test image is a raw partition image of an Ext3 file system. The size of the image is 16MB. For this file system it was possible to distinguish the two files from their file size.

Reiserfs This test image is again a raw partition image of a Reiser file system. For this file system it was possible to distinguish the two files from their file size.

vfat The test image is raw partition image of a FAT file system. The size of the image is 11MB. In contrast to the first two file systems, it was *not* possible to distinguish the file sizes using `du`.

JFS The test image is raw partition image of a JFS file system. The size of image is 17MB. For this file system it was possible to distinguish the two files from their file size.

Minix The test image is raw partition image of a Minix file system. The size of image is 5.1MB. For this file system it was possible to distinguish the two files from their file size.

Ext2 This test image is raw partition image of an Ext2 file system. The size of the image is 11MB. For this file system it was possible to distinguish the two files from their file size.

Msdos The test image is a raw partition image of a MS-Dos FAT file system. The size of the image is 5.1MB. Like the vfat file system, here it was *not* possible to distinguish the two files using `du`.

filesystem	img size	du		du –app size		msg digest
		withhole	withzero	withhole	withzero	identical
ext3	16MB	3KB	99KB	98KB	98KB	yes
reiserfs	76MB	8KB	100KB	98KB	98KB	yes
vfat	11MB	98KB	98KB	98KB	98KB	yes
jfs	17MB	8KB	100KB	98KB	98KB	yes
minix	5.1MB	99KB	99KB	98KB	98KB	yes
ext2	11MB	3KB	100KB	98KB	98KB	yes
msdos	5.1MB	98KB	98KB	98KB	98KB	yes

Table 1: Results of applying the tested tools on the two files.

hashtool	hash
md5sum	458b5ebc8c1bf7cacc4684e67eabb409
md5deep	458b5ebc8c1bf7cacc4684e67eabb409
sha1sum	6beb9d5bfe512fdf34aa33f0c258a72caeb64995
sha1deep	6beb9d5bfe512fdf34aa33f0c258a72caeb64995
tigerdeep	21aca239cefd99d2f441eb3dd4576898961758292515897l
sha256deep	7bce318f4ce2833a02decbcde475c0b1164d1557567e55cc004f883276811d21
whirlpooldeep	d9ae3e0342558c1f1fc0eb5d8fcfcd97a4c932ddd869cba141d77367594660fb fefb292a895d97174757bbd85a1befe5d1d8e018b0f5d3bb0ceae05ded02f2f9

Table 2: Hash values for both files using different hash tools.

4.4 Summary and Discussion

Tables 1 and 2 show a summary of the results of the experiment done. It should be noted that the disk usage in vfat, minix and msdos file systems is the same. This is because these file systems do not seem to support sparse files. Overall the apparant sizes of the two files for all the file systems is the same and so is the message digest for all the hash tools. Table 3 show the reference values computed by all hash tools.

4.5 Attack Scenario

Can the observed behavior of hash tools be exploited by an attacker? One scenario we can envision is this context is an insider who wants to steal a large confidential file by copying it to a relatively small removable storage medium. The attacker prepares the file with holes and is able to copy it to the storage device and steal it. When accused in court of stealing the file, the attacker can claim that it is impossible to copy the file to the storage medium because of its size. Since only the hashes have been recorded, it is questionable whether the attacker is telling the truth or the evidence has been tampered with.

A second attack scenario occurs in the context of denial-of-service attacks. An attacker

File System	md5sum Hash
Ext3	e4eb6cc8f026f676ee7b1249251905bf
Reiserfs	07c7e4afbc788799b83f398614e09d49
vfat	4e32ebb9784a01c176a7f371e3a680c4
JFS	c7ac4f517683b4ee5dde89a54346cc1d
Minix	76a8da6e046ee1690a26212c13f5d100
Ext2	ed3f984761861d24fbaeab1c6b476d40
Msdos	56324b6d97230650e29a09d5c4dd6501

Table 3: File system hashes for different file systems.

may prepare a large file and place it on a disk device to consume sufficient disk space to cause service disruption. Later, when accused of denial-of-service, the attacker may claim of having placed a sparse file which consumed less disk space. Since only the hashes have been recorded, it is questionable whether the attacker is telling the truth or evidence has been tampered with.

5 Conclusion

In this work we investigated how forensic hash tools perform on sparse files (i.e., files with holes) in comparison to non-sparse files. We therefore analyzed the tools for this border case on different file systems. All tools gave the same hash for the two created files even if the disk usage of the two files was different. This is mainly because the hole in the sparse file when read by the tools is interpreted as a sequence of zero bytes. The results of the tests are as follows:

- All the tools resulted in same message digests in all the file systems.
- Disk usage was different in Ext2, Ext3, Reiser and Jfs file systems.
- Disk usage was exactly same in vfat, minix and msdos file system.
- The apparent sizes of the files however was same for all the file systems.

This points to a weak point in the usage of hash tools today if they are used on different analysis layers within the same investigation. If used on the file system layer, two files which have different physical representations on the disk had the same message digest. These differences become apparent if *the same* hash tools are used on the volume layer. The caveat here is that even on the file system layer (i.e., using `du`), the two files with identical hashes can be distinguished (at least in many file systems). This opens an applicability gap for hash tools and a need for a convincing argument in case the integrity of evidence is challenged in court.

Overall we feel that current tools should at least warn the user about such an inconsistency if they are used at the file system level. This is possible since the file sizes can be observed

using the du command. Overall we argue that all common hash tools need to be modified for dealing correctly with sparse files.

Acknowledgments

We wish to thank the anonymous reviewers for helpful feedback. In particular we acknowledge the comments by reviewer #3 who pointed out the difference of using hash tools at different levels of abstraction (i.e., on volumes and on file systems).

References

[1] Dan Farmer and Wietse Venema: Forensic Discovery, Addison Wesley Professional, 2005.

[2] Brian Carrier: File System Forensic Analysis, Addison-Wesley Professional, 2005.

[3] Brian Carrier: Digital Forensics Tool Testing Images, http://www.dftt.sourceforge.net, last visited: Dec 2006.

[4] Harish Daiya: Sparse file testing images. Available online at pi1.informatik.uni-mannheim.de/filepool/projects/hash-tool-testing/images.zip

[5] Ronald Rivest: The MD5 Message-Digest Algorithm, RFC 1321, April 1992.

[6] D. Eastlake, T. Hansen: US Secure Hash Algorithms (SHA and HMAC-SHA), RFC 4364, July 2006.

[7] D. Eastlake: US Secure Hash Algorithm 1 (SHA1), RFC 3174, September 2001.

[8] Ross Anderson and Eli Biham: Tiger - A Fast New Hash Function, Proceedings of Fast Software Encryption 3, Cambridge, 1996.

[9] P.S.L.M. Barreto, V. Rijmen: The Whirlpool Hashing Function, First Open NESSIE Workshop, Leuven, 13-14 November 2000.

[10] National Institute of Standards and Technology: The Computer Forensics Tool Testing (CFTT) Project Homepage, http://www.cftt.nist.gov, last visited: December 2006.

Towards Reliable Rootkit Detection in Live Response

Felix C. Freiling
Laboratory for Dependable Distributed Systems
University of Mannheim, Germany
freiling@informatik.uni-mannheim.de

Bastian Schwittay
Symantec (Deutschland) GmbH, Germany
Bastian_Schwittay@symantec.com

Abstract: Within digital forensics investigations, the term *Live Response* refers to all activities that collect evidence on live systems. Though Live Response in general alters the state of the suspect system, it is becoming increasingly popular because it can recover valuable information that is lost in normal investigations that power down a suspect computer and perform analysis on its hard disk image. Current best practices for Live Response however fail to take into account the possibility of false information being gathered due to the presence of rootkits on the system. In this paper we propose to establish rootkit detection as a standard part of Live Response. We argue that the credibility of the recovered information can be substantially increased by regular empirical experiments using known rootkits and rootkit detectors. We present the results of such an experiment in this paper showing that a redundant combination of three tools can discover all rootkits which were publicly available as of June 2006.

1 Introduction

1.1 Motivating Live Response

After detecting a computer misuse, the classical approach during a digital investigation is to pull the plug of all computer systems involved and analyse an image of the hard drives in the respective systems. This kind of analysis is commonly called "dead" analysis. In recent years, this principle has been reconsidered, and methods of data collection on systems that are still running have become popular. In contrast to dead analysis, this is called "live" analysis, and all activities concerning data collection on live systems are summarized under the term *Live Response*.

The reason why Live Response has become an important part of digital forensic investigations is that when powering down a computer system, a lot of volatile information is lost, because the RAM is cleared. This information will not be available for further analysis, although it may contain valuable clues regarding the incident, and there is sometimes no other way to obtain it other than collecting it on the running system. The volatile information that can be collected during Live Response includes the system date and time, a list of current network connections and open TCP/UDP ports, a list of running processes, the

names of the loaded kernel modules, and a list of open files. Especially the information regarding network connections and ports, and running processes can be crucial information that allows to figure out what caused an incident or how a system was compromised.

In addition to the collection of volatile data, it has also become common to collect certain non-volatile data too, i.e. data that could also be recovered in dead analysis. Sometimes this is done out of convenience, e.g. because certain logfiles can easily be recovered from a running system with special commands, but in a dead analysis the investigator would have to parse the logfile format with large effort. In some cases file formats are even proprietary so no publicly available parsers do exist, yet it is very easy to recover the files on a running system with the right tools.

Another reason for collecting non-volatile data is that this information will already be available in a very early stage of an analysis and can improve containment of the incident. With the amount of data that can already be recovered during Live Response, performing a forensic duplication can become redundant, because enough information to explain and resolve the incident is already there. On the other hand, if Live Response yields unexplainable results or raises further suspicions, the decision to perform a forensic duplication and thorough dead analysis is strengthened.

1.2 Live Response Considerations

A severe problem with Live Response techniques is that they will generally alter the running system's state, which contradicts the general paradigm of computer forensics to never modify potential evidence. Issuing commands on an evidence system will change that system's RAM contents, create processes, and even the hard drive may be modified, e.g. by unnoticed creation of cache files by the operating system. However, if the changes that the use of a Live Response tools will make to the target system are well understood and documented, its use can be admitted even if it alters evidence slightly. This tradeoff is justified by the fact, that the information gained by conducting Live Response is often so valuable, that a small and controlled modification of the evidence is negligible.

It is well-known, that information offered by a compromised computer system cannot be trusted. Especially rootkit software can alter status information in arbitrary ways. Before performing Live Response, it is therefore important to aquire confidence that no rootkits are installed on the computer system in question. However, standard operational procedures for Live Response (see for example Jones et al. [14]) fail to acknowledge this fact.

One could argue that although a rootkit can hide certain information on a running system, a dead analysis will always be able to recover all information that the rootkit hid, and it will also reveal the presence of the rootkit. First of all, any volatile data that is manipulated is not reconstructable during a dead analysis, so the otherwise valuable information about running processes, open ports etc. is lost, unless special techniques to correct the rootkit's manipulations are used. Secondly, it is much more convenient to collect evidence about rootkits on a live system than during dead analysis since the latter often necessitates searching thousands of files for malicious traces. Thirdly, rootkits exist that leave no trace

on the file system at all, running completely in main memory; these rootkits will be undetectable on a file system image, and if their presence is not noticed during Live Response, there is a severe danger that an investigation will be using false data collected during Live Response without ever knowing.

Therefore, if a rootkit is detected, *any* information collected during Live Response must be checked against other evidence, e.g., evidence collected during dead analysis. The existence of a rootkit can additionally be the cause for further actions. For example, rootkits can initiate electronic booby traps for logical bombs, in case of which it may be advisable to freeze the system immediately to prevent any loss of valuable evidence.

1.3 Contributions

In this paper, we aim to improve the *credibility* of Live Response procedures for the Win32 family of operating systems. We contribute to the methodology of computer forensics by exhibiting a method to detect rootkits on a compromised system with high confidence. The method is based on the concept of *diversity*, i.e. redundantly using different rootkit detectors on a set of publicly available rootkits. We performed experiments with 11 publicly available rootkits and 12 available rootkit detection tools as of June 2006. By experimenting with the rootkits and detection tools, it is shown that using a combination of 3 different detection tools, 100% of all tested rootkit installations can be detected. Due to the diversity in detection techniques it can be expected that a similarly high detection rate can be achieved for unknown rootkits, allthough it is hard to precisely quantify this of course. We therefore argue that rootkit detection should be a standard part of Live Response and should follow an empirically evaluated strategy like the one presented in this paper to raise confidence in the detected evidence.

1.4 Paper Outline

This paper is structured as follows: Sect. 2 gives some background on Windows rootkits and techniques for their detection. The experimental setup is explained in Sect. 3 while the results of the experiments are reported on in Sect. 4. We conclude in Sect. 5. For lack of space we omit a discussion of standard practices for Live Response and refer the reader to Schwittay [17] for more information.

2 Windows Rootkits

We now take a detailed look at rootkits for Windows operating systems and their basic working principles and techniques. There are no novel insights in this section, but knowledge of rootkit internals is necessary to understand the different ways to detect rootkits despite their stealth capabilities. These differences are vital to the diversity of detection

techniques used in our methodology later in this paper.

A rootkit can be defined as "a set of programs and code that allows a permanent and undetectable presence on a computer" [13]. Typically rootkits are used to hide certain files, processes and other information on a running system, making them one of the most dangerous classes of malware, because they can exist on a compromised system unnoticed for years. Additional functionality like a backdoor or keyboard sniffers can often be added, turning rootkits into the ultimate attacker's tool. By examining current rootkits and detection tools and putting them to the test, in Sect. 4 it will be shown how rootkits can nevertheless be detected, resulting in a methodology to safely detect publicly available rootkits.

2.1 Background: Windows Kernel and Rootkits

To understand how rootkits work it is first of all important to understand the basics of the Microsoft Windows kernel architecture and the way the operating system works beyond the normal graphical user interface. The techniques used by rootkits to subvert the operating system are fundamentally different from the way normal software operates. Often they manipulate internal operating system tables that are used to control the execution flow of other software on the system, causing the rootkit code to be executed instead of the original code. In other cases rootkits will filter or forge the output of system calls to hide themselves or other objects and yet another class of rootkits modifies memory contents directly to fool the operating system.

Because of the complexity of the underlying technology, it will hardly be possible to explain all details of the operating system mechanism completely. Therefore this introduction will only focus on a few central concepts that allow presentation of the most common rootkit techniques, namely *Hooking* and *Direct Kernel Object Manipulation*. For detailed description of Windows operating system internals, refer to Solomon and Russinovich [18] or Hoglund and Butler [13]; the latter can also be considered to be a reference work on rootkits.

2.1.1 Windows Internals

The first important concept to understand is how *access control* is realized on the x86 processor family, which is the processor type used in most personal computers today. In this context, access control means how the hardware controls which kind of instructions may be used by a particular process, which areas of memory or files may be modified, and how hardware components may be accessed and written to. Without access control at the hardware level, any process would theoretically be allowed to access and modify every file, alter the memory contents belonging to another process and issue every CPU instruction, no matter whether that could disturb other processes or even crash the whole system.

The x86 processor type has provided the capability for access control for a long time, in the

form of four privilege levels called *rings*, where *Ring 0* is the most privileged and *Ring 3* is the least privileged (see Fig. 1). Since Windows NT, Microsoft operating systems use the two rings Ring 0 and Ring 3 for access control. All user-mode programs, i.e. most normal applications, run in Ring 3 and have only limited access to memory, and cannot access hardware directly; this will be prevented by the CPU itself. Ring 0 contains all software that runs with full system priviliges, for example the Hardware Abstraction Layer (HAL), device drivers, IO and memory management – essentially all components of the operating system kernel. Code running in Ring 0 can access the whole memory space and is able to execute privileged instructions, e.g. allowing access to hardware components like graphic cards or the keyboard. Programs running in Ring 3 are often called *userland programs* and those running at Ring 0 are called *kernel programs*.

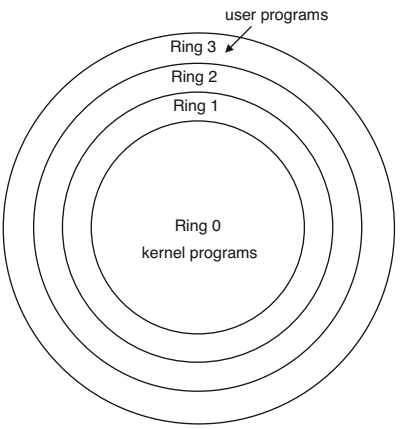

Figure 1: Ring architecture of the Intel x86 processor family.

Since even userland applications may have to access hardware from time to time – e.g. when using a system administration programm to install new device drivers for a graphics card – special mechanism exist that allow a userland program to cross the barrier to Ring 0 and use the otherwise unavailable functions in a controlled way. Some rootkits exploit this capability to plant their own device driver (a `*.sys` file) containing their malicious code in Ring 0, allowing them to run at full privilege level; these rootkits are also called *kernel rootkits*. It is important to notice that any Ring 3 detection tool will have a large disadvantage against a kernel rootkit, because the two are basically not competing with each other on equal terms. Because a Ring 0 rootkit can control the environment in which other software runs, it can achieve stealth and avoid detection in a very effective way.

The System Service Dispatch Table. Whenever a userland program wants to use a function that is only available to Ring 0 programs, it issues a so-called *system call*. A system call interrupts execution of the user program and transfers execution control to the kernel, which will then process the requested service as indicated by the system call number stored in the CPU register `EAX`, possibly using a set of input parameters received from the userland application. After the system call has been processed, execution of the userland

program is resumed, which can now use the system call's results for further operation.

On Windows, there is a large number of these service functions provided by the kernel, and each function is identified by a unique system call number. Whenever a system call is made, a special kernel routine called `KiSystemService` is executed in Ring 0, which reads the system call number from the `EAX` register and looks up the matching function in a table, the *System Service Dispatch Table (SSDT)*. The SSDT contains the memory addresses of all functions that correspond system calls, making it one of the central points of execution flow control in the kernel. Thus the SSDT naturally represents an opportunity for malicious modifications by a kernel rootkit; the technique for SSDT manipulation will be described in more detail later.

2.1.2 Hooking Techniques

Altering the execution flow of programs is one of the most prominent techniques of rootkits to subvert the Windows operating system. By having code that belongs to the rootkit execute instead of the original, a rootkit can manipulate the operating system and the programs that are run on it to achieve stealth, i.e. operate completely unnoticed and hidden from the user. The various techniques of rerouting the execution of programs are commonly referred to as *Hooking*. It should be noted that often legitimate software uses Hooking techniques too, so the presence of a hook on a system does not automatically mean that a rootkit is installed.

IAT Hooking. The easiest form of hooks manipulate the way in which individual applications are executed, in particular how they import library functions, that are typically provided on Windows operating systems in the form of DLL files. Whenever an application is run, a list of required library files included in the application executable file is used to load the matching library files into the application's memory space completely. Apart from the list of required DLL files, the executable also contains a structure that stores the specific functions inside the respective library files that the application needs. These functions' locations in memory are determined and a special table called *Import Address Table (IAT)* is loaded with the library functions' names and the corresponding address of the function in the application's memory space.

A rootkit that has access to an application's address space can manipulate the IAT in order to make its own code execute instead of the original library function. It will do this by overwriting the address of the IAT functions with the address of the rootkit function's location in memory space, allowing the rootkit to manipulate the parameters or output of the original function. There are a number of userland rootkits that will use IAT hooking to manipulate the IAT of every application running on a system, which enables them to hide their presence globally across the whole operating system. Fig. 2 shows how IAT Hooking alters the execution path of a program.

Inline Function Hooking. A slightly more elegant hooking technique does not overwrite the IAT address of the library function to be hooked, but instead modifies the func-

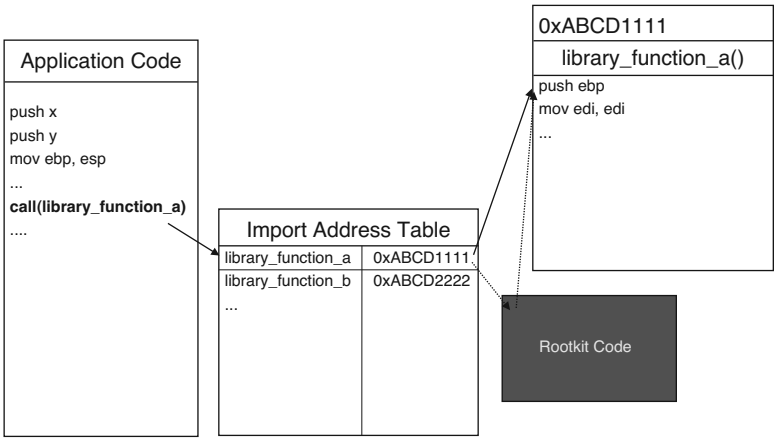

Figure 2: IAT Hooking – The solid line represents the normal execution path, the dotted line shows the hooked path.

tion's code directly in memory. Such an *inline function hook* will usually patch the first few instructions of the hooked function with a special jump instruction to the rootkit code, which will then be executed before the original code. Usually the rootkit code will then call the original function, because when execution of the original function has completed, control will return to the rootkit code, allowing to modify the results of the library function. Inline hooks have several advantages over standard IAT hooks, mainly because the technique avoids problems when the function is not called using the IAT, because no matter how a process calls the respective function, the inline hook is always effective.

SSDT Hooking. Whereas IAT and Inline Function Hooking all take place in the userland realm, there are also techniques to hook execution in the kernel. As previously explained, one of the central tables that control execution of kernel functions is the System Service Dispatch Table. Whenever a system call is performed, the kernel function `KiSystemService` takes control over the execution and looks up the memory address of the kernel function corresponding to the system call number in the SSDT.

A rootkit can subvert this mechanism by exchanging the original function's address with the address of the rootkit's hook function. The hook function can then imitate the original function but choose to filter out certain information, e.g. a function that reports a list of open ports could be replaced by a similar function that does the same but does not display a certain range of ports, which is used by a built-in backdoor of the rootkit. Overwriting the addresses in the SSDT is generally a lot harder than IAT Hooking, but the details have intentionally been left out in this context to be able to just explain the rough concepts. A detailed description of the method can be found in Hoglund and Butler [13]. Fig. 3 is a simplified representation of the technique.

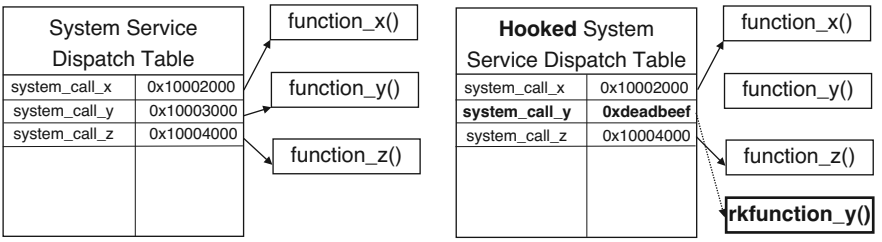

Figure 3: SSDT Hooking – left: before hooking, right: after hooking.

2.1.3 Direct Kernel Object Manipulation

Hooking techniques are a solid technique of subverting an operating system, but their main drawback is, that a hook can usually be detected and there are several tools for hook detection, some of which will also be introduced in later sections. An advanced technique for subversion, that e.g. allows to hide processes, directly manipulates the very data structures in kernel memory that keep track of the state of the operating system, therefore it is called *Direct Kernel Object Manipulation (DKOM)*.

Windows keeps a number of undocumented data structures in kernel memory which for example contain a list of running processes, of threads scheduled for execution etc. Part of DKOM techniques is therefore trying to understand the structure of these kernel objects to be able to manipulate them without making the operating system unstable or even crashing it completely. The technique then consists of careful modifications to the in-memory data structures to hide certain objects from the user. For example, processes are recorded in a doubly-linked list, so that if the offset of the forward and backward pointers to the next or previous process in the list is known, it is possible to exclude a process from the list with simple pointer reorganization. Because the scheduling of execution of processes does not depend on a process being present in that list, this technique hides the process successfully (e.g. from the Task Manager), but the process is still executed unnoticed (see figure 4 for a simplified model of the process hiding technique).

DKOM has certain benefits over Hooking, especially because it is very hard to detect, since directly modifying the raw main memory contents with a Ring 0 rootkit cannot be controlled by any built-in security mechanism in Windows. The disadvantage of DKOM is its extreme instability, since there is no documentation about the actual structure and usage of the kernel objects by the operating system, and often even minor operating system upgrades change the way they look like and how the operating system uses the kernel objects. Nevertheless DKOM has become a concept that has to be considered one of the most sophisticated and dangerous rootkit technologies that exists.

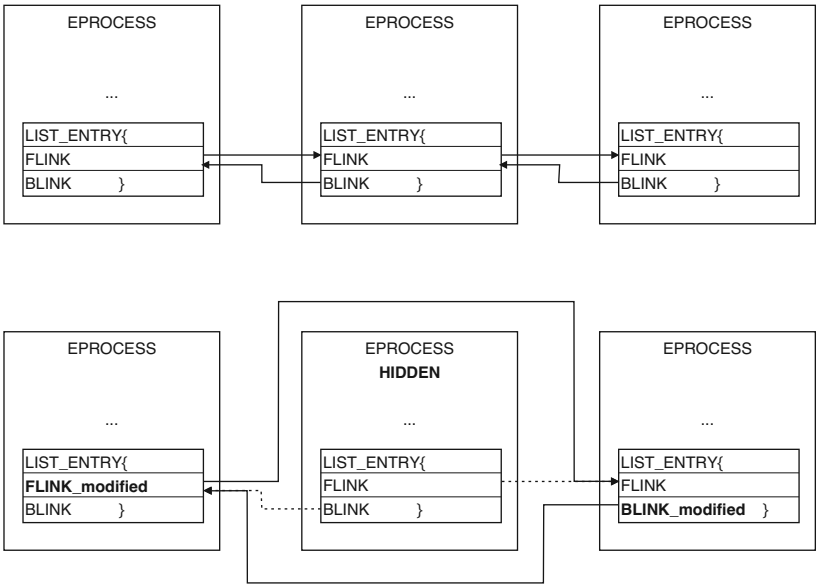

Figure 4: Process Hiding with DKOM [13]– The upper figure shows a part of the original linked list of processes, the lower figure shows how the middle process is hidden by pointer manipulation.

2.2 Overview of Rootkits and Detectors

2.2.1 Rootkits used

For the experiments on rootkit detection, a number of publicly available rootkits were used, all of which can be obtained via the rootkit.com website [8]. This includes userland rootkits as well as kernel rootkits, and rootkits which use Hooking techniques as well as some that use DKOM to hide their presence. The userland rootkits included in the experiments were AFX Rookit, Hacker Defender (also called HxDef), He4Hook, NtIllusion and Vanquish. All of these rootkits offer about the same capabilities of hiding processes and files, sometimes drivers or services; the details will be described in the next section.

Kernel rootkits included in the experiments were FU, FUTo, phide and HideProcessHook-MDL (HPHM), where the latter is more of a proof-of-concept device driver that allows to hide processes by using SSDT Hooking. It was chosen nevertheless because no other rootkit could be included that uses this technique. FU, FUTo and phide all use DKOM techniques to hide processes, FU and FUTo even offer functions to elevate a process's privileges and also alter other kernel objects than the linked list of processes.

There were also some related pieces of software, namely cfsd and Klog, which were included in the trials although they are not rootkits, because they use very similar techniques. Cfsd is a file system driver, which can be used to misrepresent the contents of a file system, for example allowing to hide certain files. Klog is a device driver that acts as a keyboard sniffer, i.e. it logs everything typed into the keyboard into a file without being noticed.

2.2.2 Detection tools used

After selecting the rootkits for the experiments an assortment of rootkit detection tools was assembled to be able to test the various technologies used by rootkits against the different kind of detection tools. Among them were two tools that explicitly look for hooks, VICE [8] and Respendence Software Rootkit Hook Analyzer (RKHA) [10]. VICE detects both userland and kernel hooks, and can discover normal IAT hooks, inline function hooks, SSDT hooks and also more advanced hooks, while Rootkit Hook Analyzer only detects kernel hooks.

Another class of detection tools directly looks for hidden objects on the running operating system, the ones used in this paper are F-Secure Blacklight [1], RKDetector [9], Rootkit Revealer [11], and UnhackMe [12]. They are generally run once and the output is a list of all hidden objects found. Blacklight can detect hidden processes and files, RKDetector and UnhackMe detect hidden processes and will sometimes also alert because of suspicious services and corresponding registry keys. Rootkit Revealer can detect hidden files and registry keys, including those belonging to a system service, making it possible to also detect hidden services indirectly.

It is known that Blacklight and Rootkit Revealer use a *cross-view detection* approach, which identifies hidden objects by comparing the output of high-level API reporting functions with results gained from parsing the appropriate structures on a very low level. For example, to find hidden files, a cross-view detection tool would first use the Windows API to request information about the contents of the file system. It would then use low-level methods of parsing the file system itself, and compare the output with the API output, uncovering any files that were hidden at API level. A cross-view approach only works if the subversion of the rootkit does not manipulate the operating system at the same or even lower level than the detection tool uses for its baseline data collection. Apart from this limitation, cross-view detection is one of the most effective ways to find rootkits, because it is "generic" in the sense that is does not scan for specific rootkit signatures or for signs of a data hiding technique like hooking.

The tools IceSword [5] and DarkSpy [2] are both freeware tools that offer the possibility to look for rootkits in an interactive fashion. For example both offer a function to view a list of currently running processes, like the Windows Task Manager, or an Explorer-like application to browse the file system or the Windows registry. The difference to the ordinary reporting and browsing tools is that they use very advanced techniques to gather the data about the respective objects, often exploiting the way in which rootkits typically hide processes or files. Although IceSword and DarkSpy may sometimes highlight hidden processes using similar techniques as the cross-view detection tools, generally they have to be used in an interactive fashion by an investigator who systematically looks for signs of a rootkit.

To complete the set of detection tools there are also a number of command line tools that cannot be classified like the tools above, these are modgreper, flister and System Virginity Verifier [6]. Flister is a tool that can be used to detect hidden files, exploiting some common bugs in rootkits when using file hiding techniques. Modgreper scans kernel memory for hidden modules, which can sometimes reveal the presence of a rootkit that uses a hid-

den module in the kernel to do its work. The System Virginity Verifier (SVV) written by Rutkowska is a unique tool, which is related to the concept of *Explicit Compromise Detection* [16], which means that the integrity of critical operating system elements is checked to detect a possible compromise. SVV compares the code sections of kernel modules in memory, i.e. drivers and DLLs, with their representation on the file system, where the files are stored. If discrepancies are detected between the stored file and its image in memory, SVV evaluates the type of the change and outputs an infection level that denotes the severity of the modification, and whether it is likely to be a malicious modification.

3 Experimental Setup

3.1 Test Platforms and Compatibility Issues

To get an idea of how rootkits work on different version of the Windows operating system, all experiments were executed on four different operating systems, namely on Windows 2000 Service Pack 4 with all official update as of June 1, 2006, Windows XP without any updates, Windows XP Service Pack 2 with all updates and Windows 2003 Server Service Pack 1 with all updates. All systems were standard installations, with a minimum of required device drivers installed to be able to run properly. After the setup, all systems were kept offline and all software was copied from a CD to the test platform, to avoid accidental compromise by a real attack.

There were significant compatibility problems with the rootkits, in particular, the only rootkits that could be installed on Windows 2003 were Hacker Defender, Klog and Vanquish. Other notable problems occured with cfsd and He4Hook, none of which could be successfully installed on any of the test platforms; it is not clear whether this was a known issue or caused by incorrect use of the supplied files, because these rootkits come with no documentation at all. On top of that, NtIllusion did not work properly on any operating system but Windows XP Service Pack 2, HideProcessHookMdl did not work on Windows 2000 and the AFX Rootkit could not be executed on Windows XP Service Pack 2.

There were also minor compatibility issues with some of the detection tools, although they appeared to be far more robust than the rootkits themselves. In particular, VICE does not execute properly on Windows XP Service Pack 2; the problem seems to be a patch to the .NET framework, which is used by VICE. In addition to these incompatibilities, SVV does not work properly on Windows XP Service Pack 2 with all patches installed as of June 1, 2006. Although it does execute normally and outputs a result, it also outputs an error message and it can be clearly seen from the resuts that it did not operate at full effectiveness. The last incompatibility was with DarkSpy and Windows 2000 SP4: when starting DarkSpy, the operating system would simply crash immediately with a Blue Screen, making it impossible to test DarkSpy on this platform.

3.2 Configuration of the rootkits

To start the experiments the rootkits were, if necessary, configured using the included documentation to use all the capabilities for stealth that they provide. Specifically, this means:

- If the rootkit offers a file hiding capability, a sample folder and files that should be hidden were created.
- If the rootkit offers a process hiding capability, an appropriate process that should be hidden was started.
- If the rootkit offers hiding of network ports, an open port was created using a listening `netcat` session.
- If the rootkit offers a registry key hiding feature, sample registry keys and values to hide were created.
- If the rootkit allows hiding of services and drivers, sample objects of the respective type to be hidden were configured.

Some rootkits hide files or processes by using a *magic string*, i.e. if a file or process name contains a specific string, it will be hidden; AFX, NtIllusion, Vanquish, and HideProcessHookMdl use this technique. Hacker Defender has a real configuration file which allows definition of different magic strings for different classes of objects to be hidden, and it allows to define network port ranges to hide. FU, FUTo and phide can hide processes by referring the process's identification number, or PID. The keyboard sniffer Klog did not have to be configured. Tab. 1 gives an overview over the hiding capabilities of the different rootkits.

	Files	Processes	Registry Keys	Ports	Services	Drivers
AFX Rootkit	X	X	X	X	X	–
FU	–	X	–	–	–	X
FUTo	–	X	–	–	–	X
Hacker Defender	X	X	X	X	X	X
Klog	–	–	–	–	–	–
NtIllusion	X	X	X	X	–	–
Vanquish	X	X	X	–	X	–
phide	–	X	–	–	–	–
HideProcessHookMdl	–	X	–	–	–	–

Table 1: Hiding capabilities of rootkits: an X denotes that the rootkit can hide objects of the respective type

3.3 Rootkit installation

After configuration the rootkits were installed using the included loader programs, which is mostly a simple EXE file that will load the rootkit into memory and start it, which results in immediate hiding of the desired objects. In this case, *hidden* means that files are no longer visible in the Windows Explorer, processes are invisible in the Task Manager, registry keys are invisible in the Registry Editor, and open network ports are not detectable with `netstat`. Drivers and services were checked using the System Information tool included in Windows.

Generally, the rootkits succeeded in hiding the respective objects they were designed to hide very well, all rootkits that have the capability the hide processes, files, registry keys or network ports worked exactly as expected, hiding the target elements from the applications mentioned above. The only exception is AFX Rootkit, which did not hide network ports as it was supposed to do. Vanquish and Hacker Defender use a special service to operate, and they both succeeded in hiding "their own" service. Driver hiding, which is offered by FU, FUTo and Hacker Defender, did not work at all; all loaded rootkit drivers were still visible with the System Information program.

3.4 Application of Detection Tools

After having installed a single rootkit and having documented its stealth properties, the rootkit detection tools were run using a small batch script to see if they would detect the hidden objects and the rootkits themselves respectively. The output was then recorded and it was checked whether all hidden objects that the detection tool can detect have actually been detected; of course it makes no sense to blame a tool for not detecting a hidden process, if it is not designed to do so. Having documented the results for the detection tool, it was then uninstalled if necessary, and any drivers it used were unloaded. After a system reboot, the state prior to testing the previous detection tool was reestablished, if necessary by restarting the rootkit and reconfiguring the sample processes and network ports. Then the next detection tool was tested in the exact same way as described above.

4 Experimental Results

4.1 Differences between operating system versions

As a first result, there were absolutely no significant differences in the detection performance between the different operating system versions used during the experiments. In particular, there was no instance, in which a detection tool detected any rootkit modification successfully on one operating system, but failed on another one. The only differences were the compatibility issues of some rootkits and detection tools that were also described in detail in the Setup section. For this reason the results will be mainly be evaluated for

the Windows XP SP2 system, which was compatible to most rootkits and detection tools and can thus be regarded as the reference operating system concerning the results of the rootkit detection experiments. For the AFX Rootkit and VICE, which were incompatible with Windows XP SP 2, the results on Windows XP without any Service Packs were used.

4.2 Detailed Results

It turned out that rootkit detection was not a "black-and-white" issue, but that in some cases detection tools would only partially detect the rootkits modifications. In this context, complete detection would mean, that the detector detected all changes that it is designed to detect. Tab. 2 gives an overview over detection tools and the kind of objects they were designed to detect.

	Files	Processes	R'stry Keys	Ports	Services	Drivers	Hooks
Darkspy	X	X	–	X	–	X	–
Flister	X	–	–	–	–	–	–
F-Sec. Blacklight	X	X	–	–	–	–	–
IceSword	X	X	X	X	X	X	X
modgreper	–	–	–	X	X	–	–
RKDetector	–	X	–	–	X	X	X
RKHA	–	–	–	–	–	–	X
RK Revealer	X	–	X	–	–	–	–
SVV	–	–	–	–	–	X	X
UnhackMe	–	X	X	–	X	–	–
VICE	–	–	–	–	–	–	X

Table 2: Rootkit Detectors: an X denotes that the detector can detect hidden objects of the respective type

A scale with three values was used to rate the amount of success that was achieved. A success variable s was used to account for the effectiveness of the detection tool as follows:

- $s = 0$: The detection tool did not detect anything.
- $s = 1$: The detection tool detected some rootkit modification or hidden objects, but not everything.
- $s = 2$: The detection tool detected all possible rootkit modifications and hidden objects.

The results can be seen in Tab. 3. A "–" means that the tool in question was not applicable, because it was not designed to detect the objects the particular rootkit hid. It is obvious, that the Klog keyboard sniffer does not really fit into this selection, because it does not

actually hide anything. This issue will be discussed after the analysis of the experimental results.

	Rootkits								
Detectors	AFX	FU	FUTo	HxDef	Klog	NtIll.	Vanq.	phide	HPHM
Darkspy	2	2	2	2	–	2	2	2	2
Flister	2	–	–	0	–	2	2	–	–
Blacklight	2	2	0	2	–	2	2	2	2
IceSword	2	2	2	2	–	2	2	2	2
modgreper	–	–	–	1	–	–	–	–	–
RKDetector	2	2	0	0	–	2	0	2	2
RKHA	–	–	–	–	–	–	–	–	2
RKRevealer	2	–	–	2	–	0	2	–	–
SVV	2	–	–	2	–	1	2	–	1
UnhackMe	2	0	0	2	–	1	2	0	2
VICE	2	–	–	2	–	–	2	–	2

Table 3: Experimental Results for s

4.3 Tests with Live Response tools

The results of the Live Response tools `netstat`, `fport` [3], `pslist` [7], `psservice` [7], `find`, and `regdmp`, included in the experiments can be summarized quickly: *none* of them reported any of the objects that were hidden by the rootkits. The only way that these tools could be used to indicated a potential rootkit installation, was when a hidden process had a non-hidden port open; in this case, the tools would detect the open port, but not the process, a discrepancy that could alert an investigator if he carefully analyzed the output of the tools.

4.4 Discussion

The experimental results show that there were several excellent tools that were able to detect all rootkit modifications, but there were also some that did not perform well. The two interactive tools DarkSpy and IceSword performed excellent, both reaching a mean score of 2, the maximum possible. F-Secure's cross-view detection tool Blacklight also detected all modifications except processes hidden by FUTo, which is an expected result, because FUTo is a modified version of the FU rootkit, specifically designed to be able to evade detection by Blacklight [4] .VICE also perfectly detected the hooks installed by the hooking rootkits, although it has previously been mentioned that identifying malicious hooks can sometimes be diffcult, because a lot of software, especially Antivirus software,

uses benign hooking.

Other tools that performed quite well were SVV with an average score of 1.6 and Rootkit Revealer (1.5), along with the file lister flister (average score of 1.5). The tools Rootkit Detector and UnhackMe did not perform satisfactorily, with scores of 1.25 and 1.125 respectively; in addition to the lower average results, these tools did not detect a number of rootkits at all, making their use questionable and the results not very reliable. The very specialized tools modgreper and Rootkit Hook Analyzer (RKHA) could not be evaluated properly based on the experiments that were conducted, because they were each only applicable in one case, which is certainly not representative of the tools' reliability. During the experiments it also became obvious that the keyboard sniffer Klog would not be detected, simply because it does not attempt to hide anything; this result was therefore expected.

The fact that the Live Response tools tested failed completely in trying to report the hidden objects does not surprise either: rootkits are specifically designed to subvert these tools' mechanisms, which generally means that they subvert at least the high level-API reporting functions used by the Live Response tools. This results shows that rootkit detection has indeed become very important in forensic analysis because the standard tools used in Live Response procedures do not work properly. Adressing this need for rootkit detection, a methodology for rootkit detection has been developed, which will be described in the next section.

5 Rootkit Detection Methodology

Based on the results of the experiments, a methodology for rootkit detection has been developed, which consists of using a combination of different detection tools to add redundancy to the detection procedures. The proposed methodology uses three tools: F-Secure Blacklight, IceSword and SVV. This combination was chosen for several reasons; first of all, all three tools achieved good average results in the experiments, reliably detecting rootkit modifications.

Secondly, these three tools represent three different approaches to rootkit detection. Blacklight uses a cross-view based detection approach, it offers a simple user interface, which allows to scan for hidden objects with a single mouse click. After checking the system, Blacklight outputs a list of hidden objects found. Usage of IceSword differs in the way that it offers information about certain operating system elements in an interactive way; it offers a function that resembles the Task Manager to review running processes, a browser for the Windows registry and the file system, which resembles the Windows Explorer, and it also includes function to view loaded device drivers, services, the SSDT and even more. IceSword does not offer a convenient scanning option such as Blacklight, but if an investigator knows what to look out for, it is a very flexible and powerful tool. SVV was chosen to be included in the methodology because it represents yet another technique to detect rootkits, by checking important operating system elements for their integrity. This technique requires the most technical knowledge about the concepts and techniques used by rootkits, but for an experienced investigator it is one of the most powerful and advanced

tools to analyze a rootkit infection.

This comparison of the three tools has also shown that the methodology is applicable for forensic examiners with different levels of understanding; Blacklight does not require any special background to be used, IceSword becomes more effective in the hand of an experienced investigator, and SVV offers detailed information which will be most helpful to a very skilled analyst. By correlating the output of all three tools, the reliability of rootkit detection is increased, because even if a rootkit manages to evade one of the tools, it can still be detected by another one that uses different methods for detection. None of the rootkits tested can evade the combination of Blacklight, IceSword and SVV, and only an extremely sophisticated rootkit could hope to avoid detection by all of the three tools together. Although it is difficult to quantify the success rate of detection when using this methodology for an arbitrary rootkit infection, it is reasonable to assume that when facing a rootkit that uses techniques similar to the ones tested in this paper, the rate of detection is close to 100%. Asssuming that the majority of rootkits used today fall into this category – which appears reasonable – the likelihood to detect an actual rootkit detection is equally high.

The results of these experiments were based on software which was available as of June 2006. Experiments like the ones performed in this paper should be repeated in regular intervals so that tools can be combined such that the detection rate can be kept high in practice.

6 Conclusions and Future Work

All experiments were planned and the test platform carefully prepared to allow objective and accurate testing of the tools, and to gain a representative view of their performance on the Windows operating system platform, multiple different operating systems were included in the tests. A selection of detection tools that use different methods for detection was chosen, among which were cross-view based detection tools and tools that use Explicit Compromise Detection (ECD) to detect rootkits.

It could be shown that there exist a number of very good rootkit detection tools, but also that some are not reliable enough to be used in a forensic investigation. By choosing a combination of three detection tools which achieved good results in the experiments and which each represent a different approach to rootkit detection, redundancy has been added to the process of rootkit detection, resulting in a reliable and accurate methodology. Using F-Secure Blacklight, IceSword and System Virginity Verifier, all publicly available rootkits can be detected safely, and depending on the skill of the investigator, this toolkit, especially IceSword and Blacklight, can also be used to analyze the rootkits in more detail.

There is one thing to keep in mind when using IceSword and several other rootkit detection tools: a special device driver, for IceSword it is called `IsPubDrv.sys`, is often copied to the systems hard drive to allow the detector to function properly. This of course contradicts the principle of not creating files on an evidence system, because this is a severe modification of the evidence. The principle problem here is, that in order to be able to detect a

rootkit's subversion, the detection software has to be able to deal with the problem on the same level, i.e. using a Ring 0 kernel component. Device drivers are the standard way to gain access to Ring 0, and that is why the creation of a single file can be tolerated in this context; otherwise the rootkit would have an implicit advantage, because it can manipulate the operating system at a lower level than the detection tool can reach.

As an avenue of future research, we wish to repeat the experiments performed in this paper in regular intervals. For example, in the time which elapsed since our study new rootkits based on virtualization have been widely discussed (see for example Rutkowska [15]) and these should be included as soon as possible.

Acknowledgments

We wish to thank Maximillian Dornseif for helpful discussions.

References

[1] F-Secure – Blacklight. `https://europe.f-secure.com/blacklight/`.

[2] DarkSpy 1.02. `http://d.hatena.ne.jp/tessy/20060417`.

[3] Foundstone, Inc. `http://www.foundstone.com/`.

[4] FUTo. `http://uninformed.org/?v=3&a=7&t=sumry`.

[5] IceSword 1.18. `http://soft.patching.net/list.asp?id=25`.

[6] Invisiblethings.org. `http://invisiblethings.org`.

[7] Sysinternals – PsTools Suite. `http://www.sysinternals.com/Utilities/PsTools.html`.

[8] rootkit.com. `http://www.rootkit.com/`.

[9] Rootkit Detector. `http://www.rootkitdetector.com/`.

[10] Resplendence Software – Rootkit Hook Analyzer. `http://www.resplendence.com/hookanalyzer`.

[11] Sysinternals – Rootkit Revealer. `http://www.sysinternals.com/Utilities/RootkitRevealer.html`.

[12] Greatis Software – UnhackMe. `http://www.greatis.com/unhackme/`.

[13] Greg Hoglund and James Butler. *Rootkits - Subverting The Windows Kernel*. Addison-Wesley, 2005.

[14] Keith J. Jones, Richard Bejtlich, and Curtis W. Rose. *Real Digital Forensics*. Addison-Wesley, 2005.

[15] Joanna Rutkowska. Subverting vista kernel for fun and profit. In *Black Hat*, Las Vegas, 2006.

[16] Joanna Rutkowska. Rootkit Hunting vs. Compromise Detection. `http://www.invisiblethings.org/papers/rutkowska_bhfederal2006.ppt`, 2005.

[17] Bastian Schwittay. Towards automating analysis in computer forensics. Master's thesis, RWTH Aachen University, Department of Computer Science, 2006.

[18] David Solomon and Mark E. Russinovich. *WIndows Internals*. Microsoft Press, 2005.

IMF 2007 Workshops

Thursday, September 13th, 2007

"Computer Forensics: High-tech tools for a high-tech problem"

Steven Wood
Alste.Technologies.GmbH

"Operationally Critical Threat, Asset, and Vulnerability Evaluation (Octave)"

Christian Paulsen
DFN-CERT Services GmbH

"Memory Analysis on the Microsoft Windows Platform"

Andreas Schuster
Deutsche Telekom AG

"Workshop on X.805: Security architecture for systems providing end-to-end communications"

Suhasini Sabnis
Alcatel-Lucent

"Virtualisation of forensic images"

Ralf Moll
LKA Baden-Württemberg

Holger Morgenstern
IT-Service / Sachverständigenbüro Morgenstern

GI-Edition Lecture Notes in Informatics

P-1 Gregor Engels, Andreas Oberweis, Albert Zündorf (Hrsg.): Modellierung 2001.

P-2 Mikhail Godlevsky, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications, ISTA'2001.

P-3 Ana M. Moreno, Reind P. van de Riet (Hrsg.): Applications of Natural Language to Information Systems, NLDB'2001.

P-4 H. Wörn, J. Mühling, C. Vahl, H.-P. Meinzer (Hrsg.): Rechner- und sensorgestützte Chirurgie; Workshop des SFB 414.

P-5 Andy Schürr (Hg.): OMER – Object-Oriented Modeling of Embedded Real-Time Systems.

P-6 Hans-Jürgen Appelrath, Rolf Beyer, Uwe Marquardt, Heinrich C. Mayr, Claudia Steinberger (Hrsg.): Unternehmen Hochschule, UH'2001.

P-7 Andy Evans, Robert France, Ana Moreira, Bernhard Rumpe (Hrsg.): Practical UML-Based Rigorous Development Methods – Countering or Integrating the extremists, pUML'2001.

P-8 Reinhard Keil-Slawik, Johannes Magenheim (Hrsg.): Informatikunterricht und Medienbildung, INFOS'2001.

P-9 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Innovative Anwendungen in Kommunikationsnetzen, 15. DFN Arbeitstagung.

P-10 Mirjam Minor, Steffen Staab (Hrsg.): 1st German Workshop on Experience Management: Sharing Experiences about the Sharing Experience.

P-11 Michael Weber, Frank Kargl (Hrsg.): Mobile Ad-Hoc Netzwerke, WMAN 2002.

P-12 Martin Glinz, Günther Müller-Luschnat (Hrsg.): Modellierung 2002.

P-13 Jan von Knop, Peter Schirmbacher and Viljan Mahni_ (Hrsg.): The Changing Universities – The Role of Technology.

P-14 Robert Tolksdorf, Rainer Eckstein (Hrsg.): XML-Technologien für das Semantic Web – XSW 2002.

P-15 Hans-Bernd Bludau, Andreas Koop (Hrsg.): Mobile Computing in Medicine.

P-16 J. Felix Hampe, Gerhard Schwabe (Hrsg.): Mobile and Collaborative Busi-ness 2002.

P-17 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Zukunft der Netze –Die Verletzbarkeit meistern, 16. DFN Arbeitstagung.

P-18 Elmar J. Sinz, Markus Plaha (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2002.

P-19 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3.Okt. 2002 in Dortmund.

P-20 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3.Okt. 2002 in Dortmund (Ergänzungsband).

P-21 Jörg Desel, Mathias Weske (Hrsg.): Promise 2002: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen.

P-22 Sigrid Schubert, Johannes Magenheim, Peter Hubwieser, Torsten Brinda (Hrsg.): Forschungsbeiträge zur "Didaktik der Informatik" – Theorie, Praxis, Evaluation.

P-23 Thorsten Spitta, Jens Borchers, Harry M. Sneed (Hrsg.): Software Management 2002 – Fortschritt durch Beständigkeit

P-24 Rainer Eckstein, Robert Tolksdorf (Hrsg.): XMIDX 2003 – XML-Technologien für Middleware – Middleware für XML-Anwendungen

P-25 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Commerce – Anwendungen und Perspektiven – 3. Workshop Mobile Commerce, Universität Augsburg, 04.02.2003

P-26 Gerhard Weikum, Harald Schöning, Erhard Rahm (Hrsg.): BTW 2003: Datenbanksysteme für Business, Technologie und Web

P-27 Michael Kroll, Hans-Gerd Lipinski, Kay Melzer (Hrsg.): Mobiles Computing in der Medizin

P-28 Ulrich Reimer, Andreas Abecker, Steffen Staab, Gerd Stumme (Hrsg.): WM 2003: Professionelles Wissensmanagement – Erfahrungen und Visionen

P-29 Antje Düsterhöft, Bernhard Thalheim (Eds.): NLDB'2003: Natural Language Processing and Information Systems

P-30 Mikhail Godlevsky, Stephen Liddle, Heinrich C. Mayr (Eds.): Information Systems Technology and its Applications

P-31 Arslan Brömme, Christoph Busch (Eds.): BIOSIG 2003: Biometric and Electronic Signatures

P-32 Peter Hubwieser (Hrsg.): Informatische Fachkonzepte im Unterricht – INFOS 2003

P-33 Andreas Geyer-Schulz, Alfred Taudes (Hrsg.): Informationswirtschaft: Ein Sektor mit Zukunft

P-34 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 1)

P-35 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 2)

P-36 Rüdiger Grimm, Hubert B. Keller, Kai Rannenberg (Hrsg.): Informatik 2003 – Mit Sicherheit Informatik

P-37 Arndt Bode, Jörg Desel, Sabine Rathmayer, Martin Wessner (Hrsg.): DeLFI 2003: e-Learning Fachtagung Informatik

P-38 E.J. Sinz, M. Plaha, P. Neckel (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2003

P-39 Jens Nedon, Sandra Frings, Oliver Göbel (Hrsg.): IT-Incident Management & IT-Forensics – IMF 2003

P-40 Michael Rebstock (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2004

P-41 Uwe Brinkschulte, Jürgen Becker, Dietmar Fey, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle, Thomas Runkler (Edts.): ARCS 2004 – Organic and Pervasive Computing

P-42 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Economy – Transaktionen und Prozesse, Anwendungen und Dienste

P-43 Birgitta König-Ries, Michael Klein, Philipp Obreiter (Hrsg.): Persistance, Scalability, Transactions – Database Mechanisms for Mobile Applications

P-44 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): Security, E-Learning. E-Services

P-45 Bernhard Rumpe, Wofgang Hesse (Hrsg.): Modellierung 2004

P-46 Ulrich Flegel, Michael Meier (Hrsg.): Detection of Intrusions of Malware & Vulnerability Assessment

P-47 Alexander Prosser, Robert Krimmer (Hrsg.): Electronic Voting in Europe – Technology, Law, Politics and Society

P-48 Anatoly Doroshenko, Terry Halpin, Stephen W. Liddle, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications

P-49 G. Schiefer, P. Wagner, M. Morgenstern, U. Rickert (Hrsg.): Integration und Datensicherheit – Anforderungen, Konflikte und Perspektiven

P-50 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 1) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-51 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 2) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-52 Gregor Engels, Silke Seehusen (Hrsg.): DELFI 2004 – Tagungsband der 2. e-Learning Fachtagung Informatik

P-53 Robert Giegerich, Jens Stoye (Hrsg.): German Conference on Bioinformatics – GCB 2004

P-54 Jens Borchers, Ralf Kneuper (Hrsg.): Softwaremanagement 2004 – Outsourcing und Integration

P-55 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): E-Science und Grid Adhoc-Netze Medienintegration

P-56 Fernand Feltz, Andreas Oberweis, Benoit Otjacques (Hrsg.): EMISA 2004 – Informationssysteme im E-Business und E-Government

P-57 Klaus Turowski (Hrsg.): Architekturen, Komponenten, Anwendungen

P-58 Sami Beydeda, Volker Gruhn, Johannes Mayer, Ralf Reussner, Franz Schweiggert (Hrsg.): Testing of Component-Based Systems and Software Quality

P-59 J. Felix Hampe, Franz Lehner, Key Pousttchi, Kai Ranneberg, Klaus Turowski (Hrsg.): Mobile Business – Processes, Platforms, Payments

P-60 Steffen Friedrich (Hrsg.): Unterrichtskonzepte für inforrmatische Bildung

P-61 Paul Müller, Reinhard Gotzhein, Jens B. Schmitt (Hrsg.): Kommunikation in verteilten Systemen

P-62 Federrath, Hannes (Hrsg.): „Sicherheit 2005“ – Sicherheit – Schutz und Zuverlässigkeit

P-63 Roland Kaschek, Heinrich C. Mayr, Stephen Liddle (Hrsg.): Information Systems – Technology and ist Applications

P-64 Peter Liggesmeyer, Klaus Pohl, Michael Goedicke (Hrsg.): Software Engineering 2005

P-65 Gottfried Vossen, Frank Leymann, Peter Lockemann, Wolffried Stucky (Hrsg.): Datenbanksysteme in Business, Technologie und Web

P-66 Jörg M. Haake, Ulrike Lucke, Djamshid Tavangarian (Hrsg.): DeLFI 2005: 3. deutsche e-Learning Fachtagung Informatik

P-67 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 1)

P-68 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 2)

P-69 Robert Hirschfeld, Ryszard Kowalcyk, Andreas Polze, Matthias Weske (Hrsg.): NODe 2005, GSEM 2005

P-70 Klaus Turowski, Johannes-Maria Zaha (Hrsg.): Component-oriented Enterprise Application (COAE 2005)

P-71 Andrew Torda, Stefan Kurz, Matthias Rarey (Hrsg.): German Conference on Bioinformatics 2005

P-72 Klaus P. Jantke, Klaus-Peter Fähnrich, Wolfgang S. Wittig (Hrsg.): Marktplatz Internet: Von e-Learning bis e-Payment

P-73 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): “Heute schon das Morgen sehen“

P-74 Christopher Wolf, Stefan Lucks, Po-Wah Yau (Hrsg.): WEWoRC 2005 – Western European Workshop on Research in Cryptology

P-75 Jörg Desel, Ulrich Frank (Hrsg.): Enterprise Modelling and Information Systems Architecture

P-76 Thomas Kirste, Birgitta König-Riess, Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Informationssysteme – Potentiale, Hindernisse, Einsatz

P-77 Jana Dittmann (Hrsg.): SICHERHEIT 2006

P-78 K.-O. Wenkel, P. Wagner, M. Morgenstern, K. Luzi, P. Eisermann (Hrsg.): Land- und Ernährungswirtschaft im Wandel

P-79 Bettina Biel, Matthias Book, Volker Gruhn (Hrsg.): Softwareengineering 2006

P-80 Mareike Schoop, Christian Huemer, Michael Rebstock, Martin Bichler (Hrsg.): Service-Oriented Electronic Commerce

P-81 Wolfgang Karl, Jürgen Becker, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle (Hrsg.): ARCS´06

P-82 Heinrich C. Mayr, Ruth Breu (Hrsg.): Modellierung 2006

P-83 Daniel Huson, Oliver Kohlbacher, Andrei Lupas, Kay Nieselt and Andreas Zell (eds.): German Conference on Bioinformatics

P-84 Dimitris Karagiannis, Heinrich C. Mayr, (Hrsg.): Information Systems Technology and its Applications

P-85 Witold Abramowicz, Heinrich C. Mayr, (Hrsg.): Business Information Systems

P-86 Robert Krimmer (Ed.): Electronic Voting 2006

P-87 Max Mühlhäuser, Guido Rößling, Ralf Steinmetz (Hrsg.): DELFI 2006: 4. e-Learning Fachtagung Informatik

P-88 Robert Hirschfeld, Andreas Polze, Ryszard Kowalczyk (Hrsg.): NODe 2006, GSEM 2006

P-90 Joachim Schelp, Robert Winter, Ulrich Frank, Bodo Rieger, Klaus Turowski (Hrsg.): Integration, Informationslogistik und Architektur

P-91 Henrik Stormer, Andreas Meier, Michael Schumacher (Eds.): European Conference on eHealth 2006

P-92 Fernand Feltz, Benoît Otjacques, Andreas Oberweis, Nicolas Poussing (Eds.): AIM 2006

P-93 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 1

P-94 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 2

P-95 Matthias Weske, Markus Nüttgens (Eds.): EMISA 2005: Methoden, Konzepte und Technologien für die Entwicklung von dienstbasierten Informationssystemen

P-96 Saartje Brockmans, Jürgen Jung, York Sure (Eds.): Meta-Modelling and Ontologies

P-97 Oliver Göbel, Dirk Schadt, Sandra Frings, Hardo Hase, Detlef Günther, Jens Nedon (Eds.): IT-Incident Mangament & IT-Forensics – IMF 2006

P-98 Hans Brandt-Pook, Werner Simonsmeier und Thorsten Spitta (Hrsg.): Beratung in der Softwareentwicklung – Modelle, Methoden, Best Practices

P-99 Andreas Schwill, Carsten Schulte, Marco Thomas (Hrsg.): Didaktik der Informatik

P-100 Peter Forbrig, Günter Siegel, Markus Schneider (Hrsg.): HDI 2006: Hochschuldidaktik der Informatik

P-101 Stefan Böttinger, Ludwig Theuvsen, Susanne Rank, Marlies Morgenstern (Hrsg.): Agrarinformatik im Spannungsfeld zwischen Regionalisierung und globalen Wertschöpfungsketten

P-102 Otto Spaniol (Eds.): Mobile Services and Personalized Environments

P-103 Alfons Kemper, Harald Schöning, Thomas Rose, Matthias Jarke, Thomas Seidl, Christoph Quix, Christoph Brochhaus (Hrsg.): Datenbanksysteme in Business, Technologie und Web (BTW 2007)

P-104 Birgitta König-Ries, Franz Lehner, Rainer Malaka, Can Türker (Hrsg.) MMS 2007: Mobilität und mobile Informationssysteme

P-105 Wolf-Gideon Bleek, Jörg Raasch, Heinz Züllighoven (Hrsg.) Software Engineering 2007

P-106 Wolf-Gideon Bleek, Henning Schwentner, Heinz Züllighoven (Hrsg.) Software Engineering 2007 – Beiträge zu den Workshops

P-107 Heinrich C. Mayr, Dimitris Karagiannis (eds.) Information Systems Technology and its Applications

P-108 Arslan Brömme, Christoph Busch, Detlef Hühnlein (eds.) BIOSIG 2007: Biometrics and Electronic Signatures

P-109 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 1

P-110 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 2

P-111 Christian Eibl, Johannes Magenheim, Sigrid Schubert, Martin Wessner (Hrsg.) DeLFI 2007: 5. e-Learning Fachtagung Informatik

P-112 Sigrid Schubert (Hrsg.) Didaktik der Informatik in Theorie und Praxis

P-113 Sören Auer, Christian Bizer, Claudia Müller, Anna V. Zhdanova (Eds.) The Social Semantic Web 2007 Proceedings of the 1st Conference on Social Semantic Web (CSSW)

P-114 Sandra Frings, Oliver Göbel, Detlef Günther, Hardo G. Hase, Jens Nedon, Dirk Schadt, Arslan Brömme (Eds.) IMF2007 IT-incident management & IT-forensics Proceedings of the 3rd International Conference on IT-Incident Management & IT-Forensics

P-115 Claudia Falter, Alexander Schliep, Joachim Selbig, Martin Vingron and Dirk Walther (Eds.) German conference on bioinformatics GCB 2007

P-116 Witold Abramowicz, Leszek Maciszek (Eds.) Business Process and Services Computing 1st International Working Conference on Business Process and Services Computing BPSC 2007

P-117 Ryszard Kowalczyk (Ed.) Grid service engineering and manegement The 4th International Conference on Grid Service Engineering and Management GSEM 2007

The titles can be purchased at:

Köllen Druck + Verlag GmbH
Ernst-Robert-Curtius-Str. 14 · D-53117 Bonn
Fax: +49 (0)228/9898222
E-Mail: druckverlag@koellen.de