

Aalto University
School of Science
Degree Programme of Computer Science and Engineering

Patrik Nisén

Implementation of a timeline analysis software for digital forensic investigations

Master's Thesis
Espoo, March 4, 2013

Supervisor: Professor Tuomas Aura
Instructor: Antti Nuopponen M.Sc. (Tech.)

Author:	Patrik Nisén	
Title:	Implementation of a timeline analysis software for digital forensic investigations	
Date:	March 4, 2013	Pages: 77
Professorship:	Data Communication Software	Code: T-110
Supervisor:	Professor Tuomas Aura	
Instructor:	Antti Nuopponen M.Sc. (Tech.)	
<p>Organizations today are trying to manage the many risks they perceive to be threatening the security of their valuable information assets, but often these risks realize into security incidents. Managing risks proactively is important, but equally important and challenging is to efficiently respond to the incidents that have already occurred, to minimize their impact on business processes.</p> <p>A part of managing security incidents is the technical analysis of any related computer systems, also known as digital forensic investigations. As a result of collecting evidence such as log files from these systems, the analysts end up with large amounts of data, which can form a timeline of events. These events describe different actions performed on the system in question. Analysing the timelines to find any events of interest is challenging due to the vast amount of data available on modern systems. The goal of this thesis is to create a software program to support the analysis of very large timelines as a part of digital forensic investigations.</p> <p>As a result, we have implemented a software with an efficient query interface, which supports iterative exploration of the data and more complex analytical queries. Furthermore, we use a timeline visualization to compactly represent different properties of the data, which enables analysts to detect potential anomalies in an efficient way. This software also serves as a platform for future work, to experiment with more automated analysis techniques.</p> <p>We evaluated the software in a case study, in which it proved to show a great level of flexibility and performance compared to more traditional ways of working.</p>		
Keywords:	digital forensics, incident management, information security, databases, visualization	
Language:	English	

Tekijä:	Patrik Nisén		
Työn nimi:	Aikajanojen analysointiohjelmiston toteutus tietoturvapoikkeamien tutkintaan		
Päiväys:	4. maaliskuuta 2013	Sivumäärä:	77
Professori:	Tietoliikenneohjelmistot	Koodi:	T-110
Valvoja:	Professori Tuomas Aura		
Ohjaaja:	Diplomi-insinööri Antti Nuopponen		
<p>Tärkeä osa nykypäivän organisaatioiden riskienhallintaa on tietopääoman turvaamiseen liittyvien riskien tunnistaminen. Näitä riskejä ei kuitenkaan usein oteta tarpeeksi vakavasti, sillä monesti ne myös realisoituvat tietoturvapoikkeamina. Kattava etukäteisvalmistautuminen on tärkeää, mutta poikkeamien vaikutusten minimoimisen kannalta oleellista on myös valmius tehokkaaseen poikkeamatilanteiden hallintaan.</p> <p>Osana tietoturvapoikkeamien hallintaa toteutetaan siihen liittyvien järjestelmien tekninen analyysi. Todistusaineiston, kuten erilaisten lokitiedostojen, keruun tuloksena tutkijat muodostavat aikajanan järjestelmässä suoritetuista toiminnoista. Koska modernien järjestelmien sisältämä tiedon määrä on poikkeuksetta suuri, on aikajanan analysointi mielenkiintoisten jälkien löytämiseksi erityisen haastavaa. Tämän diplomityön tavoitteena onkin luoda ohjelmisto tukemaan kooltaan erityisen suurten aikajanojen analysointia.</p> <p>Työn tuloksena luotiin ohjelmisto, joka tarjoaa tehokkaan kyselyrajapinnan, tukee tutkimukselle tyypillistä iteratiivista tiedon etsintää ja monimutkaisempia analyttisiä kyselyitä. Lisäksi ohjelmisto mahdollistaa monipuolisen aikajanan visualisoinnin, mikä helpottaa huomattavasti käytöspoikkeamien löytämistä. Tavoitteena oli myös tuottaa alusta, jota voidaan käyttää jatkossa uusien automaattisten analyysitekniikoiden kehittämisessä.</p> <p>Ohjelmiston toimivuus todennettiin tapaustutkimuksessa, joka osoitti ohjelmiston olevan erityisen joustava ja suorituskykyinen verrattuna aikaisempiin toimintatapoihin.</p>			
Asiasanat:	digitaalinen forensiikka, tietoturvapoikkeaminen hallinta, tietoturva, tietokannat, visualisointi		
Kieli:	Englanti		

Acknowledgements

Foremost, I would like to express my gratitude to my employer Nixu and namely Antti Nuopponen for making this thesis possible and for providing an interesting subject to study.

I also wish to thank my friends and colleagues who have listened to my ideas and given feedback.

The most sincere thanks to my family, without you none of this would have been possible. Finally, thank you Anna for your love and support.

Espoo, March 4, 2013

Patrik Nisén

Abbreviations and Acronyms

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
CD	Compact Disk
CERT	Computer Emergency Response Team
CPU	Central Processing Unit
CSIRT	Computer Security Incident Response Team
CSS	Cascading Style Sheets
CSV	Comma Separated Value
DBMS	Database Management System
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
I/O	Input/Output
IDS	Intrusion Detection System
IIS	Internet Information Services
IP	Internet Protocol
IT	Information Technology
ITIL	Information Technology Infrastructure Library
JSON	Javascript Object Notation
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
OS	Operating System
RAM	Random Access Memory
REST	Representational State Transfer
SHA	Secure Hashing Algorithm
SIEM	Security Information and Event Management
SQL	Structured Query Language
UI	User Interface
URL	Uniform Resource Locator

USB
WAH

Universal Serial Bus
Word Aligned Hybrid

Contents

Abbreviations and Acronyms	5
1 Introduction	9
2 Responding to computer security incidents	11
2.1 Incident management in general	11
2.1.1 Objectives	12
2.1.2 Prioritization	12
2.1.3 Escalation	13
2.2 Security incidents	13
2.2.1 Lessons from past incidents	14
2.2.2 Risks	15
2.3 Security incident management	15
2.4 Management process	17
2.5 Digital forensics	19
2.5.1 Forensics and digital investigation	19
2.5.2 Forensics process	20
2.5.2.1 Collection	20
2.5.2.2 Examination & Analysis	23
2.5.2.3 Reporting	25
3 Analyzing compromised systems in practice	26
3.1 Beginning for a forensic investigation	26
3.2 Methods for investigations	27
3.3 Efficient analysis	28
4 Analysis methods	29
4.1 Storing and querying event data	29
4.1.1 Databases	30
4.1.1.1 Row and column orientation	31
4.1.1.2 Full-text search engines	31

4.1.2	Indexing	32
4.1.2.1	Single level index	32
4.1.2.2	B-trees	32
4.1.2.3	Hash maps	33
4.1.2.4	Bitmaps	34
4.1.3	Progressive queries	35
4.2	Information visualization	36
4.2.1	Visualization	36
4.2.2	Visualization process	39
4.2.3	Timelines	40
4.2.4	Network flow data	41
4.2.5	Attack visualization	43
4.3	Anomaly detection	45
4.3.1	Denning's model	46
4.3.2	Quantitative and statistical methods	47
4.3.3	Data mining and machine learning	48
4.3.4	Application specific techniques	48
4.4	Existing solutions	49
4.5	Suitability of existing solutions	51
5	Design and implementation	52
5.1	Flexible query interface	52
5.1.1	Database	53
5.1.2	Query pipeline	54
5.1.2.1	Data reduction	54
5.1.2.2	Progressive queries and caching	55
5.1.2.3	Paging	56
5.1.2.4	Core commands	57
5.2	Timeline visualization	58
5.3	Application	59
5.3.1	Server	59
5.3.2	Client	59
5.3.3	Parser and importer	60
6	Case study	62
6.1	Background	62
6.2	Investigation	63
6.3	Comparison	68
6.4	Summary	69
7	Conclusions	70

Chapter 1

Introduction

Business processes today are increasingly dependent on information. The efficient management of information reduces the risk of interruptions in these processes and also the related costs. Information is therefore also an important asset, which organizations need to protect just as any other business critical asset, and which needs to be managed as a part of everyday risk management. However, most organizations are still not adequately protected, which is made especially noticeable from the increased reporting of information security incidents in the media. Even if they were adequately protected, security incidents would still occur. There has to be an efficient way to manage these incidents to minimize the damage and restore operations to the normal level of service. This is the objective of security incident management.

The technical analysis of the incidents, known as digital forensic investigations, is an important part of the security incident management process. It is performed to understand the full impact of an incident and even to collect evidence admissible in court. In that respect, it can be seen as a digital equivalent to more typical crime scene investigations. What makes digital forensic investigations challenging, though, is the vast amount of data available on modern computer systems. Collecting this data is relatively easy due to the many tools created for this very purpose. However, processing and analysing the data to find events or artifacts of interest can be a daunting task without efficient supporting tools.

As a result of collecting evidence, the analysts typically end up with a single, very large timeline of events. These events are collected from all the possible logs saved on a system, and describe in detail what has happened and what kind of actions have been performed on that system. The goal of this thesis is to implement an efficient software program, which assists the analysts during digital forensic investigations to analyse these timelines or

other individual log files. The software should help to find good starting points to guide the analysis and support iterative exploration of the data. In addition, we want to develop a platform for the experimentation of more automated analysis techniques.

Chapter 2

Responding to computer security incidents

The following sections will introduce the reader to management of computer security incidents and digital forensics. First, section 2.1 explains how incidents are managed within organizations in general and how they relate to operations of an organization. Next, section 2.2 expands the concept of an incident to security incidents and describes what kind of challenges they pose for today's organizations. Chapter 2.3 will then define different concepts around managing security incidents, and chapter 2.4 introduces an incident management process from a security point-of-view. After the general introduction, the term *incident* is used interchangeably with the term *security incident*.

As a result of understanding the process, we will notice the importance of efficient technical analysis of these incidents and take a closer look into the triage phase of the process in the form of digital forensics in section 2.5.

2.1 Incident management in general

As a part of everyday operations of an organization, it will regularly face events that cause interruptions or reductions in the quality of their services. These unexpected interruptions are also called *incidents*, and are normally first handled by the Service Desk function of an organization. Because of the general problem-solving nature incident management process spans through all of the functions of an organization. [49]

In the context of information technology (IT), Information Technology Infrastructure Library (ITIL), which provides practices for IT service management, defines incident in the following way: [49]

An unplanned interruption to an IT service or a reduction in the quality of an IT service. Failure of a configuration item that has not yet impacted service is also an incident. For example, failure of one disk from a mirror set.

The following sections first introduce the objectives of incident management in chapter 2.1.1, then explain the basic concepts of prioritization (section 2.1.2) and escalation (section 2.1.3).

2.1.1 Objectives

The objective of incident management is to restore the operations of an organization to the normal level of service with the smallest possible impact, which is either defined for the organization itself or committed to the customers. Efficient and controlled intervening to deviations on service level is crucial to not only constrain the direct costs related to resolving and incident, but also for maintaining the productivity of the whole organization. [49]

Naturally, when talking about possible risks that an organization might be facing, the foremost action is to first identify them. Only after knowing the risks, one is able to work pro-actively to prevent them from realizing into incidents. However, almost never it is possible to remove all of the risk, because some of the parameters are always out of the hands of an organization. That is why there must be a way to respond efficiently when all the preventive measures have failed. Furthermore, only being prepared helps to constrain the costs of a possible incident. [49]

Incident response in general terms is an organized and managed approach to addressing these unexpected events of emergency (incidents). Incident response team (IRT) is then a group of people which is trained and prepared to respond to these incidents. Team consists of pre-selected members with different capabilities to fill roles needed to handle the situation in question. For instance, there needs to exist an incident commander who takes responsibility of managing the situation, team and other resources.

2.1.2 Prioritization

Prioritization is an important part of the incident management process. As there will always be multiple incidents submitted to be resolved at the same time, being able to recognize the overall importance and priority of an incident helps to resolve the most threatening incidents much more efficiently. It is important to realize that the priority is actually a sum of two different factors: *impact* and *urgency*.

Impact of the incident is the overall disruption for the level of service in terms of the number of business process affected. Furthermore, an incident is considered to have major impact also when timespan of the disruption is remarkable, even when the number of affected users is low.

Urgency of the impact, on the other hand, is the acceptable delay for the affected functions in solving the incident.

This means that even though a certain incident might at first sight seem to threaten the whole organization, its urgency might be very low and allow more urgent incidents to be dealt with first. Incident management may approach the overall risk by reducing either impact or urgency of the incident. It has to be understood that impact or urgency may also change during the lifetime of an incident. [49]

2.1.3 Escalation

Another important part of managing incidents is the concept of escalation. To resolve an incident, it is not necessary to engage more resources than absolutely required, because constant interruptions of, for instance, certain experts will decrease their productivity substantially. That is why a structured approach will yield more efficient results, but on the other hand requires clear policies to be defined. When the incident cannot be resolved by the first line within a certain time span, it will be *escalated*, that is, more expertise will be involved to resolve the incident. *Functional escalation* will involve personnel with more specialist skills or authority. *Hierarchical escalation* means involving personnel with a higher level of authority in the organization. [49]

2.2 Security incidents

In the previous section we introduced the notion of incidents and important basic concepts related to management of incidents. This section will concentrate on opening the notion of security incidents and the challenges they pose for today's organizations.

Most of the business process today are fully dependent on supply of information, and the true value of information increases at the same time. Failing to protect one of the three basic factors of security (confidentiality, integrity and availability) leads to security incidents, where one of the factors is either threatened or already compromised.

However, information security should not be seen as a goal of its own, but rather linked to the operational efficiency where correct and complete

information is available where needed, and to the quality of service provided to the customers. [49]

The following chapters will discuss the history and lessons from past security incidents based on reported data (section 2.2.1), and the potential risks and reasons for preparing to security incidents (section 2.2.2)

2.2.1 Lessons from past incidents

In 1986 occurred the first well-documented international security incident on ARPANET, the predecessor of Internet. Only two years later, the first automated security incident called "the Morris worm" spread on the ARPANET and caused a first known denial-of-service on about 10 % of the computers connected to the ARPANET. This major event of the time triggered founding of the first computer emergency response team known as CERT Coordination Center, to help responding to computer security incidents. After that, we have witnessed an explosive growth in security incidents. [11] Already in 1995 the number of reported incidents to CERT was at a level of 2500. After that, the problem has exploded, and most of the small incidents go either unnoticed or are left unreported. [18]

What is notable here, is that the majority of the breaches reported to Verizon's Data Breach Investigations Report 2012 [62] are opportunistic and low difficulty [62], which has been the case since the very beginning [11]. In 85 % of these cases it took more than two weeks before the breach was noticed and in over half of the cases months. Furthermore, 97 % of the breaches were not detected by the organizations themselves, but rather by third-parties, mainly by the law enforcement. After the breach is detected, 42 % of the problems are fixed within days. This strongly implies that in most of the organizations incident prevention and detection are not adequately taken care of.

The report also notes that even though being a very typical driver behind preparing to incidents, the vast majority of the victims did not suffer any extensive financial losses in addition to the incident handling costs, or long term brand damage because of the breach. However, a few reported losses of hundreds of millions and some companies went out of business because of a breach. One has to remember though, that a lot of the costs may impact other organizations indirectly and more severely, as it is the case in payment card breaches where the card issuer might have to absorb the costs of the compromised credit cards. [62]

2.2.2 Risks

Unfortunately many organizations realize the importance of proper security management only after their first severe incident. Even if the possible risks have been considered, they are often times widely neglected. Proper incident management should be an integral part of organizations' normal risk management strategy. [48]

Some of the important risks are introduced below. [15]

Direct costs A clearly understandable risk, however not that obvious to fully recognize, is the direct costs of the incident. Handling and fully resolving an incident and all of its after-effects might result in major financial expenses. Even though, as noted in chapter 2.2.1, in most of the cases the expenses stay relatively low, they are still always present and hard to estimate.

Business impacts Perhaps even bigger threats are all the unclear indirect effects on the business itself. A breach may have an effect on daily operations of the business making it hard or impossible to operate. In addition, it can cause permanent damage on the reputation and brand of the organization or a product, and result in loss of confidence in the eyes of stock owners and clients.

Legal reasons Legal concerns may also require an organization to recognize security risks and prepare oneself to respond to incidents. For instance, entities handling credit cards, healthcare-related information or being part of a critical infrastructure might be required to obey certain regulations and even individuals may be held accountable for severe incidents. Furthermore, privacy concerns in general have become a major driving force behind preparing organizations to face this risk. European Union, for instance, has taken actions to strengthen privacy in its member countries by enforcing it in a specific regulation (on proposal stage). [19]

2.3 Security incident management

Typically, the term *incident response* has been used to describe the activities of a Computer Security Incident Response Team (CSIRT), which is an organization or a team taking care of performing the response within an organization. However, in [4] the definition is restructured into the following phrase:

a capability or team that provides services and support to a defined constituency for preventing, handling and responding to computer security incidents

The authors wanted to emphasize that the response "team" can exist in many forms, depending on the organization and operational purpose in question, and it should be seen more as a capability therein. Nevertheless, CSIRT is the entity formed to take care of the handling of security incidents within an organization.

However, this definition also resulted in reconsideration of the term incident response, as it was seen as too narrow to include all the possible services a CSIRT might provide. The authors of [4] consider *incident response* to be part of a more general *incident handling* service, among other preceding functions such as *detecting and reporting*, *triage* and *analysis*. That is, *incident response* is the last process of *incident handling*.

Even further, CSIRT may provide many other services outside of the scope of only reacting to incidents and handling them. Term covering everything around managing incidents is *incident management* and it includes also all the pro-active services such as vulnerability handling and awareness training. Division of these concepts is illustrated in figure 2.1.

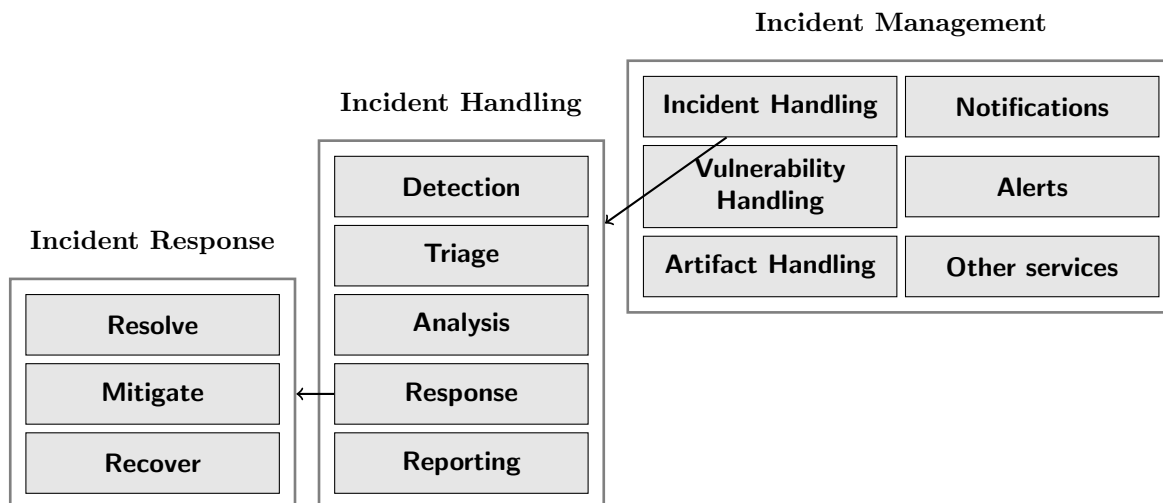


Figure 2.1: Incident management concepts

Building a well functioning capability is not a trivial task, since typically knowledge is needed from many different divisions, such as management, IT, Security, HR, legal and public relations. Thus, the capability must consist of knowledge from many different disciplines. Having such as a dedicated

multi-disciplinary capability might be hard to accomplish and that is the reason why it can take so many forms within an organization and only a strictly dedicated team might not be a reasonable form for it. [58]

It is even possible that the capability exists in a distributed form without the organization realizing it. This means that some processes and plans are in place, but not in a coordinated way. Depending on the level of implementation, the capability can be a comprehensive collection of policies and procedures, or more loose group responsible of handling incidents.

Furthermore, management is often under pressure to implement the security strategy of the organization as cost-efficiently as possible. This typically leads to centralization of the IT related tasks to avoid duplication and outsourcing of the tasks that cannot be executed efficiently in-house. Outsourcing, however, results in a situation where the managers are not anymore in direct control of the execution, but still responsible for the completion of the security strategy. [4]

2.4 Management process

During the years, many different authors have published their own view about efficient incident handling processes, thoroughly summarized in [4]. All of the authors agree on a high level, but differences can be seen in granularity. The process introduced in [4] and illustrated in Figure (2.2) is based on this summary and experience of its authors, and designed to support existing organizational mission, policies and procedures. Here, detailed descriptions of underlying subprocesses are omitted and instead introduced on a general level.

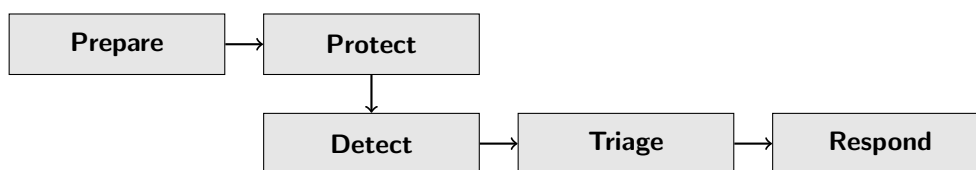


Figure 2.2: Incident management process

1. **Prepare:** In this phase, incident management or CSIRT capability is implemented in an organization and kept up-to-date. Plans are made and iteratively improved based on feedback from previous incidents. Information about found improvements is passed to the *protect* process. The base for effective response to incidents is built here and should

include planned processes on areas such as notification, communication, coordination, analysis of incidents and responding to incidents.

2. **Protect:** This phase implements changes in infrastructure to stop an on-going incident or to prevent incidents in future. Also, the infrastructure is pro-actively inspected and risks are evaluated. Any previously unknown security related information is transferred to the next *detect* process. Protection process ensures that the organization is adequately protected in advance and is aware of possible risks involved within its infrastructure.
3. **Detect:** Detection process actively monitors and analyzes events, forwards any suspicious events to the *triage* process and possibly reassigns events to other areas outside incident management processes. It is ensured that the organization is capable of detecting possible intrusions efficiently and communicating this information to responsible parties for further analysis.
4. **Triage:** Here, the overall impact of an event is investigated. Events arriving to this process are categorized, prioritized, correlated and forwarded to the responsible process depending on the context. This can mean *responding*, reassigning or closing unimportant events. Typically, this process is part of *responding* itself, but the authors argue that because it can be done in many different ways and by many different parties, even outsiders, it should be separated.
5. **Respond:** In this process, actions are taken to respond to an event. It is first analyzed to understand the full context, and then a response strategy is designed and executed. Actions concerning all the participating entities are coordinated and any actions required to resolve and recover from the incident are taken. Information to learn from is passed back to *prepare* process.

Especially important is to understand the links and communication between the processes. For instance, an important question when performing the analysis and responding is how and when should the system be contained to constrain the incident surface and preserve evidence. This is again only a matter of clearly defined and understood policies and procedures.

Detect and Triage phases of this process are the most technically challenging ones. As noted previously in the chapter 2.2.1, most of the security incidents typically remain unnoticed for a long period of time or even completely from the organization. And even though the issue is fixed relatively

shortly after, gaining a thorough understanding of what has really happened in the triage phase may be extremely challenging.

2.5 Digital forensics

In the previous section we introduced the reader to incidents, security incidents and their management. We identified that detection and triage or analysis phases of the management process are the most technically challenging and important for a successful handling of incidents.

This section will take describe how the actual analysis of incidents is performed on a general level, in the context of digital forensic investigation. It will first introduce forensics as a concept in section 2.5.1 and then look in more detail into a forensics process and an its many phases in section 2.5.2.

2.5.1 Forensics and digital investigation

Forensic science (or forensics) is typically defined as the application of a broad spectrum of sciences to answer questions of interest to a legal system. That is, the goal of forensics is to identify, collect, examine and analyse evidence that makes it possible to prove a certain chain of events. In addition, all this must happen while preserving the integrity of the evidence and information, and while maintaining a strict chain of custody for the evidence. Chain of custody means a chronological documentation of handling of the evidence.

This general definition applies as well to digital forensics, where the evidence is however, "information stored in binary form that may be introduced and relied on in court" [45]. Term data here refers to distinct pieces of information in a certain format.

Digital investigation is the process where hypothesis, which answer questions about digital events, are developed and then tested. Due to the complex nature of digital evidence and the technology underneath, analysts must realize the importance of fully understanding the results of the analysis, which means not only trusting the tools used, to be able to draw holding conclusions. [10]

Even though the definition of digital investigation lacks the aspect of proving the results in court, we will use the term interchangeably with digital forensics because the main difference is in management of the evidence.

In addition to using digital forensics in response to incidents, the same techniques can be used in plenty of other cases too, such as general troubleshooting, data recovery or regulatory compliance. In this case, however,

we limit our interest to firmly and efficiently proving a certain chain of events or finding other digital artifacts of interest.

2.5.2 Forensics process

A general forensics process is illustrated in figure 2.3. The phases of the process are collection, examination, analysis and reporting. A short description of each is given in here, and more in depth in the following sections.



Figure 2.3: Forensics process

1. **Collection:** This phase includes identifying, labeling, recording and acquiring relevant information from different sources, while preserving the integrity of the data.
2. **Examination:** In this phase, collected data is processed using automated and manual methods and particularly interesting data is extracted for further analysis.
3. **Analysis:** Next, information is derived and conclusions are drawn from the results of the examination. This phase finds answers to the research questions, which were the reason to start the investigation in the first place.
4. **Reporting:** Finally, the results of the analysis are presented. The report should describe how the data was processed, what tools and procedures were used and what are the following actions to take. In addition, recommendations for improving the forensics process may be given.

2.5.2.1 Collection

The first phase of the forensics process identifies relevant sources and collects data from them. Even identifying all the possible sources may be a daunting task considering the wide adoption of a variety of technologies in organizations. In addition to the obvious sources such as workstations, laptops, mobile phones and servers, one has to also consider the variety of transferable medias and other devices. External hard drives, USB memory sticks, memory cards and CDs may store vast amounts of data outside the control of

an organization. One problematic source, which has seen a quick adoption in many organizations, is all the different software and services hosted by other external service providers, from where the acquisition of data might be hard or even impossible. That is why one has to be able to think alternate sources that might provide the same information and circumvent possible legal and ownership considerations.

Today, what even small organizations have in their hands is an explosively growing amount of data that they do not know even exists. Having enormous amounts of data, however, results in equally big challenges when trying to analyse and extract information and evidence out of that data. Indeed, these facts are forcing analysts to begin using live-analysis data collection techniques, pinpointing important pieces of data remotely and retrieving only those absolutely necessary for further analysis. Full copies of all the data available might not be feasible or even possible anymore. [54]

If the organization has realized the need for preparing to the worst and taken proactive measures to respond to incidents they should have identified important systems and enhanced logging and audit capabilities in these systems. This data is typically extremely valuable for the forensics process.

Valuable sources for forensics data are:

Filesystem In addition to the files themselves, filesystems provide plenty of other information about the data it stores (metadata), which occasionally proves to be even more valuable than only the content of the files. Depending on the filesystem in question, this metadata includes information such as timestamps for file creation, access and modification. In addition, information about deleted files is usually preserved for a relatively long period of time. All this time information helps to build a detailed picture of actions performed on the filesystem, and on the system itself.

Network Because practically every computer is connected to some kind of network, data exchanged between different systems might reveal useful information about network-based attacks. Even though all the data within organizations usually cannot be saved due to the vast amount of data produced every day, routers, firewalls, Intrusion Detection Systems (IDS) and other systems produce lots of event data about what is happening inside an organization. In case more detailed data is needed for instance after a breach is suspected, a packet collection system or more specialized network forensics system can be set up, in case such a system is not already in place.

Application Most of the applications produce a log of their actions, which

is used for troubleshooting in error situations. These logs are often very useful for forensics as they reveal more in detail how a user was using the system. Not all applications are interesting nor produce a detailed log, but especially server applications such as web servers, keep a log that can be used to investigate how the application running behind it was used on a higher level.

Operating system In addition to the OS level logging, operating systems provide information about what is happening on a particular system at the very moment. Most of this data is volatile and can be lost due to any actions taken. Important volatile data are for example network connections, login session, contents of memory, running processes and open files. Furthermore, any changes in configuration might be a sign of a potential issue.

Acquiring the data should follow a three-step process: developing a plan, acquiring data and verifying integrity of the data. This process is illustrated in figure 2.4



Figure 2.4: Collection process

Making a plan beforehand helps to better understand the environment and prioritize the sources. As mentioned previously, one of the issues in data collection is handling the enormous amounts of data available. This is why identifying the right sources is important, instead of collecting everything available. The collection plan should take into account all the information and pointers available to detect the right sources. These pointers may include, for instance, any suspicious events or alerts from virus or intrusion detection systems.

Prioritization should take into consideration the following elements:

Value Estimating relative value of the data from each source helps to focus on most important sources.

Volatility Volatile data means all the data that is lost or modified due to actions performed on a running system or when the system is shut down. Because the data might be lost unexpectedly, it should be preserved earlier than non-volatile data. Also, data saved on non-volatile media might also have a volatile nature if it is modified often. [46] One has to

remember that actions that at first might not seem to be harmful might still modify the state of the system and lose valuable information. For instance, only accessing a file might destroy an important timestamp from the metadata of the file. So at first, one should collect information only by observing the running system and systems interacting with it, with no active actions taken.[60]

Effort Some sources might be a lot harder and costly to get access to, and thus taking a lot of time and effort. It might be more useful to focus on sources within an organization rather than trying to access, for instance, networks of third-parties.

Data acquisition is normally performed using automated forensics tools, which collect volatile data and takes copies of non-volatile data. Acquisition can be performed also over the network, but generally doing it locally is the preferred way because of the additional control and the certainty it brings.

However, one has to keep in mind that every action performed on the system in question will have an effect on its running state. That is why all the interaction has to be kept to minimum and it has to be verified that the actions of the tools used are understood correctly.

After the collection, integrity of the data needs to be verified to ensure that the data was transferred correctly and has not been tampered with. This is typically done by calculating a digest of the original and the copy, using algorithms such the SHA family [47].

2.5.2.2 Examination & Analysis

After the collection, what analysts typically have in their hands is a extensive collection of different types of data from many different sources. Next, the relevant pieces of information are extracted and conclusions are drawn. Having vast amounts of data makes finding the relevant pieces difficult and time-consuming, just as finding a needle from a haystack. This is true especially when the amount of initial, prior knowledge is limited or non-existent. Of the millions of events, only a couple might end up being relevant to the case. That is why the help and quality of supporting tools and techniques is particularly important.

Different types of data require their own specific domain knowledge and analysis techniques. Figure 2.5 introduced in [10] illustrates a layered model of the different types of analysis for digital evidence. This model first separates the analysis of storage devices from communication devices. The next level makes a difference between volatile, RAM memory, and persistent storage, such as hard drives. Volume analysis is a general term for the analysing

of "collections of storage locations", under which the real data is stored, most typically under file systems. This layer tells where the actual data is located, and where possibly hidden data could exist. Inside volumes are the actual file systems (or other structured layers). The last important layer is application and OS layer, which links files from file systems to applications. This layer also includes all the log data generated by applications or different monitoring systems. [10]

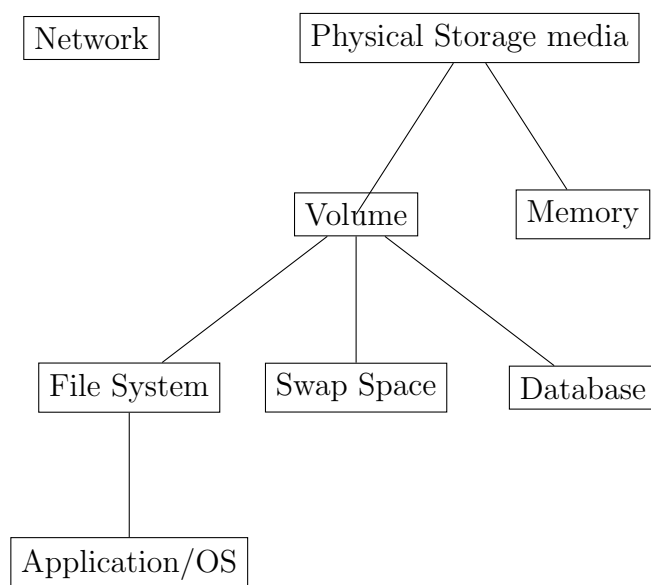


Figure 2.5: Analysis types

All these layers may provide a lot of information about the activities occurred on a system, but they all require their own tools and techniques to extract the relevant information. Despite the differences, even the tools typically output a vast amount of data that needs to be filtered and interpreted by the analyst. And this can be a daunting task to perform especially without any helpful prior knowledge. Nevertheless, many times the output of these tools resemble a log of any normal application.

As explained previously, the result of the analysis is typically to discover a chain of events that prove certain incident or event in detail. All this log data collected or extracted from the sources is chronological by nature and typically contains some sort of time information. This allows to combine that data into a single timeline, giving a deep view on the use of a system or many systems. The resulting timeline is then analyzed to prove or disprove certain defined hypothesis.

One point which characterizes the analysis phase is its ad-hoc nature.

In other phases, known steps are followed to first find the sources, prioritize and collect the data. Here, however, it is not known beforehand what kind of information or patterns are found, which makes entirely automated analysis difficult. Instead, tools and different methods are applied creatively to test hypothesis, and that is why the tools should also support this kind of iterative exploration of the data.

2.5.2.3 Reporting

After having gained a thorough understanding, conclusions are reported. This report contains the discovered issues, methods used to find them, and actions that needs to be taken to remedy these issues. Suggested improvements can be related to policies and procedures of the organization, or to the infrastructure itself. This important to prevent possible future incidents and to learn from the experience.

It might happen that the research questions were not fully answered, and more information is still needed to remedy that. In this case, the needed sources are reported and collection is continued.

Chapter 3

Analyzing compromised systems in practice

In the previous chapter we learned how organizations manage the risks related to security incidents. In addition, we described how these incidents are investigated on a very general level. This chapter will give a brief introduction to digital forensics investigations in practice. First, we describe how the investigations typically begin in section 3.1. Then we describe what methods are used in different cases in chapter 3.2. And finally, we describe the methods needed for efficient forensic investigations in chapter 3.3.

3.1 Beginning for a forensic investigation

As explained previously, in an ideal situation an organization has a management process for security incidents in place and the incidents should be discovered in the detect phase of the process (section 2.4). In practice this typically happens through an automated monitoring system. In case of a malware infection this system can be an anti-virus software and other types of attacks might be detected by firewalls or intrusion detection systems (IDS). These systems generate alerts containing information about the origin of the event and about the reason, which makes a certain event suspicious. The events are then taken further in the process and responded to.

As explained in section 2.2.1, however, this is not how these situations typically proceed. Most of the organizations are not able to detect or handle the incidents themselves. Rather, a third-party, such as law enforcement or an internet service provider, might detect the incident months later, when the worst might have already happened. Only at this point, when the organization realized the importance and impact of the incident, a full investigation

is started to minimize the damage and to understand what has happened and how.

It is also possible that the organization has been a target of a more directed attack, even by an insider. These cases can be very hard to detect because the actions might not be directly malicious by nature at all, and thus not detectable by any monitoring systems.

3.2 Methods for investigations

When an incident has been detected, the goal of the started investigations is to try to find answers, for instance, to the following questions. Has a breach or infection happened at all? How and when? What has a particular user done? Is there any evidence of certain actions executed? Depending on the question, different methods can be used to look for an answer.

From the different analysis types introduced in 2.5.2.2, in practice the most common ones are analysis of volatile memory (RAM), analysis of an individual application or operating system, and analysis of a full hard drive.

Volatile memory is analyzed typically in cases where a malware infection or a direct attack is suspected, and a full image of the running system is needed. Analysing artifacts from the volatile memory can reveal many interesting things that simply are not available on the disk or application logs. This is due to advanced hiding techniques of modern malware and hackers, there might be data available only on volatile memory and hidden from normal ways of accessing it through the operating system. For instance, we might find hidden processes, files, or network connections. The Volatility Framework [59] is an open source toolkit used for this purpose.

Similar individual artifacts can also be found from the hard drive. We might be interested to look for certain keywords from the files available on a system, find certain types of files or suspicious executables created by malware. Tools such as EnCase [24] and FTK [3] can be used to index the entire contents of a hard drive and then execute queries against this data.

More importantly, a forensic investigation is typically interested in proving a certain chain of events. For this, we must be able to build some kind of a timeline of all the events occurred in a system. As explained previously, most of the applications and operating system and file systems save a great amount of information on disk in different log files and formats. All this information can be combined to build a full timeline of events occurred on a system. Log2timeline [39] is able to combine logs from at least 35 different sources, including for instance, WWW-browser usage, Apache and IIS, Windows event log, anti-virus engines, firewall logs and generic linux logs,

among many others. The result is called a super timeline. This timeline is then analyzed in detail to understand what has happened on a system or what kind of actions or chain of actions a particular user has taken.

3.3 Efficient analysis

What is typically needed here, in addition to manual exploration of the raw events or artifacts, is support for more analytical queries and more compact ways of representing the timeline. We might want to know what moments of time differ from the normal use, based on a selected property of the events. For example, if the overall level of activity has changed, if a distinct amount of data was transferred out of the system or if suspicious files or other artifacts were brought into the system. Representing these results visually can benefit the analyst considerably, because of the capabilities of humans to process visual information efficiently, explained in detail in the following chapter. However, no such ready system currently exists for analysing these kinds of timelines in detail.

What typically complicates all these methods, is the vast amount of data available for analysis. Without an efficient tool to support this kind of exploration and more intelligent analytical queries, it is very time taking and difficult to find relevant information from the data. In addition, it is important to be able to efficiently find starting points for the investigation. This is where visualizations can be of a great help.

It is very possible, however, that no usable results are found at all within the time reserved for analysis. Analysing very large datasets is challenging, but often the biggest issue is the lack of relevant logging enabled in the configuration of the system.

Chapter 4

Analysis methods

The previous chapters gave background for the problem we are solving. We now understand incident management and digital forensics on a general level and the challenges present in that context. We also gave an introduction to investigations more in practice. We chose to look into solving the technical challenges of a digital forensic investigation, more specifically the challenge of handling big amounts of data either the systems under investigation have produced or the tools used to extract information produce.

The following sections will give the reader an overview into technical themes we identified to be of importance in designing an investigation supporting system. First, we study efficient storing and querying of data in section 4.1. Then, we look into compact representation of data in forms of visualizations, which help the analyst to find anomalies in the data, in section 4.2. Further, we will introduce automatic anomaly detection in section 4.3.

Finally, we will first introduce currently existing software solutions for forensic analysis in section 4.4 and then evaluate their suitability for our purposes in section 4.5.

4.1 Storing and querying event data

As explained in section 2.5.2, after the collection of many different types of data, the evidence is examined and analyzed. To be able to efficiently process the vast number of data collected, the underlying system must be designed for highest possible performance and to support specific needs. The raw data itself, for instance a log file with potentially millions of lines, may be hard to process in the original format. However, the lines are typically formatted in an unified way and thus have an internal structure that contains

more meaningful information. This is where database systems come into the picture. They provide means to store this structured data in a way that enables execution of complex queries against the data and information represented by the structure.

The following sections will introduce different types of data management systems and features related to the efficient analysis of event data. Section 4.1.1 will look into database systems in general, section 4.1.2 introduces different indexing techniques for efficient querying, and finally section 4.1.3 will discuss background for a novel technique of progressive queries to support efficient exploration of the data in an iterative investigation process.

4.1.1 Databases

By definition, *database* is a collection of related data, that have an implicit meaning. Further, the common meaning of database also includes the following properties: database represents some aspect of the real world, database is a logically coherent collection with an inherent meaning and database is designed, built and populated with data for a specific purpose. A database itself represents only the collection of data and is unusable on its own. *Database management system* (DBMS) defines a collection of programs or routines to enable defining, constructing and manipulating databases. Combining a database and software yields a full *database system*. The main idea behind database system is to provide a *data abstraction* and representation through a standard interface based on the defined *data model*. This means that a database system hides the details of storage access behind the system, thus making accessing it more feasible to access. [52]

A general purpose database is typically meant to allow concurrent use from many users at the same time through a unified interface. This means that the database system must implement some kind of transaction processing, which follows so called ACID properties [31]:

- **Atomicity:** It is required that either all or none of the parts of a transaction are executed.
- **Consistency:** The system must assure that after a transaction, the database satisfies all of the defined consistency requirements.
- **Isolation:** Two or more concurrent transactions must be isolated from one another in a way that the result would be the same when ran sequentially.
- **Durability:** After a successful transaction, the effects must not get lost in case of a system failure.

In our case, we want to store and query very large collections of events, that have an order (time) and a description of what happened (event). In addition, the event is typically carrying other pieces of data, defined by a data model, that describe the event in more detail.

There are many types of database systems meant to be used for different purposes. These systems can be categorized using many criteria, but here we are interested in high performance analytics, so we are going to focus on the difference between online transaction processing (OLTP) and online analytical processing (OLAP), and row versus column oriented databases.

4.1.1.1 Row and column orientation

Traditional databases store records in a row-oriented fashion. This means that pieces of data related to an individual record are stored sequentially allowing efficient processing of these full records. The design is well suited for the general-purpose databases, traditionally processing transactions on records (OLTP). However, for more analytical needs (OLAP) this is not optimal.

Along with the growth in need for efficient analytical data processing we have seen an emergence of new DBMSs focusing on delivering the highest performance on read-intensive analytical workloads. These new technologies like MonetDB [42] and InfiniDB [43] are promising and can provide even an order-of-magnitude gains on certain workloads. These *column-oriented* databases store the data of each column or field sequentially, which enables I/O efficiency and better CPU and cache performance. This difference in efficiency is true even when using techniques such as vertical partitioning, index-only plans or a specific collection of materialized views on row oriented databases. Other optimizations related to column-oriented design are late materialization of the records, block iteration and column compression techniques. [2]

4.1.1.2 Full-text search engines

Full text search engines are a special type of document databases that employ special indexing structures to allow fast searching against different fields in its documents. What makes them relevant here is their optimized string search performance and the ability use them also as a more typical document database.

The most known and widely used such database is Lucene [6], and its derivatives Solr [7] and elasticsearch [16]. However, as these systems are mostly search oriented and optimized to return only a few top matching results, they

lack many more analytical features present in traditional database systems. Although features such as Field Collapsing [8] are available in, for instance, Solr, they cannot match the performance of traditional databases and thus they are many times used to add certain search functionalities on top of traditional databases.

4.1.2 Indexing

Indexes are access structures used to speed up retrieval of records. When queries are executed to retrieve records based on values on certain column, that is, the primary organization cannot be used, a full scan over the records is needed. To avoid scanning through all the records, an index may be constructed to map values of a column to individual records, thus enabling efficient queries.[52]

Efficient retrieval of interesting data is crucial when performing an iterative investigation. The goal of this section is to understand what kind of techniques are available for efficient querying and retrieval of records from the database. Commonly used indexing techniques are introduced in the following sections.

4.1.2.1 Single level index

The most simple access structure is an ordered file with two fields. The first one, also by which the file is sorted, is the indexing field containing the value of the column in the indexed record. The second field acts as a pointer to the original record. The index file contains one entry for each record, thus allowing a quick traversal by the column values.

Because the index is sorted by the indexing field, search can be performed using binary search, which has a logarithmic execution time. This is already a great improvement over the linear search in the case of no indexing at all. [52]

4.1.2.2 B-trees

Single level index provides already a significant performance improvement. However, to reduce the number of different blocks accessed during a search, we can introduce additional levels to the index. These multilevel indexes reduce the search space even faster. They are typically implemented as a tree structure, where each node guides the search to the next subtree of the search tree. Generally, a node in a search tree contains a field value and the corresponding pointer to the record itself. In addition, node contains

pointers to the next level in the tree, for both smaller or larger values than stored in the node in question. The details then vary a lot, how the tree is structured and maintained. The most common version of a search tree used in databases is a B-tree and its variants B+-tree and B*-tree. [52]

B-tree is a constrained instance of a tree structure, which ensures that the tree stays balanced, meaning all the leaf nodes are at the same level. This holds for both insert and delete operations. The basic B-tree stores the indexed values on nodes of all the levels, along with a pointer to the record itself. B+-tree, which is the most used variant, takes another kind of approach. It stores all the data pointers only at the leaf nodes, creating a leaf node for every unique value of the index. In case of a not unique search field, an extra level of indirection is added from the leaf node, pointing to a block containing all the data pointers (row identifiers). In addition, leaf nodes form a linked list, which allows efficient ordered traversal on the search field. [52]

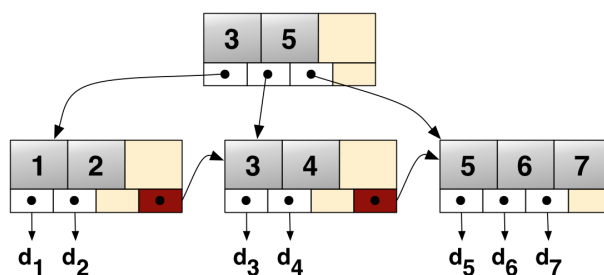


Figure 4.1: B+-tree [23]

4.1.2.3 Hash maps

Hash map is a data structure that maps keys with their corresponding values using a hash function. Hashing derives a compact representation of the key and is used internally, for instance as a table index, to efficiently find the corresponding value. Due to its great performance, hash map is widely used for cases where functionality of an associative array is needed.

Generally, hash map performs better than tree based structures as its average cost of each operation is independent of the number of elements stored in the structure. However, even though the average cost is very low, the optimal performance is achieved mainly in cases where the use pattern can be predicted. Furthermore, hash map lacks other features of trees as well, such as ordered traversal. [52]

4.1.2.4 Bitmaps

Typical indexing, as explained for instance in section 4.1.2.2, reference each row individually in the structure using a row identifier, which is a pointer to the storage system (disk). This means that for very large databases, as encountered for instance in data warehousing, indexes typically grow very large in size and become hard to maintain. Bitmaps have become an interesting solution for very large scale database indexing for read intensive use patterns. [50]

In this case, bitmap is basically a compact representation of the set of row identifiers. For a given set of rows, n th bit in the bitmap is set when the corresponding n th row has a certain property. An example of a bitmap index is illustrated in figure 4.2. This is especially efficient for columns with low cardinality, such as boolean columns, as the required number of bitmaps is low. In case of a column with unique values, one bitmap is needed for each value, thus the number of bitmaps would equal the number of rows and can become inefficient. However, proper use of compression and encoding techniques have been shown to overcome this inefficiency and yield better performance for columns of any cardinality. Indeed, using bitmaps or not is mainly a question of application. [50, 63] FastBit project [20] uses Word-Aligned Hybrid (WAH) code, analyzed in [64], for compression and a multi-level encoding [65]. The authors argue that these are the most efficient methods known in literature.

row ID	X	bitmap index			
		b_0 =0	b_1 =1	b_2 =2	b_3 =3
1	0	1	0	0	0
2	1	0	1	0	0
3	3	0	0	0	1
4	2	0	0	1	0
5	3	0	0	0	1
6	3	0	0	0	1
7	1	0	1	0	0
8	3	0	0	0	1

Figure 4.2: A sample bitmap index [64]

In addition to the space efficiency, handling bitmaps benefit of CPUs performant bit operations, and for instance boolean operations on bitmaps become extremely efficient. Furthermore, as needed in analytical use cases, bitmaps are well suited for grouping and calculating aggregates in ad hoc

queries. For OLTP applications normal B-tree indexes are a better fit because the data changes more, but the executed queries are typically known beforehand [50, 63].

4.1.3 Progressive queries

Data exploration is typically an iterative process. The queries are not necessarily known beforehand, and thus supporting data structures to answer the queries cannot be constructed in advance. Iterative querying also implies that the results of the next slightly modified query is many times a subset of the previous query. If the result set is populated entirely, it could be possible to use this set as a starting point for the following queries and reduce the number of records to be inspected radically.

Four general level techniques to support these kinds of queries are presented in [67].

1. **Repeated evaluation via query merging:** In this strategy, the previous results are not used in the following step queries. Rather, a new query is evaluated fully with only the differing conditions, and later the results are merged with all the previous step queries.
2. **Incremental evaluation via sequential scan:** This technique uses the results of a previous step query as an input for the following step query. For each subsequent step, a sequential scan has to be performed because the intermediate results are temporary and thus the indexes created for the original data cannot be employed.
3. **Incremental evaluation via dynamically created indexes:** Dynamic creation of indexes resolves the problem of the previous technique. This would make the query evaluation itself faster but decrease the overall performance, because of the dynamic index creation.
4. **Incremental evaluation via dynamically maintained indexes:** This technique transforms the indexes dynamically to suit the needs of the subsequent queries.

The last technique was considered to be the most promising one in the same paper [67]. The authors introduced an additional layer between the actual records and indexes, which they named as "router". This router adds a layer of indirection that can be dynamically updated to support progressive queries.

Typically this kind of functionality is constructed over the view functionality of the database systems. Another paper [66] investigates the use of dynamic materialized views to speed up the execution of progressive queries. The authors build a superior-relationship graph based on historical queries and deploy an algorithm to decide if the sub result set should be materialized as a view or not. Using views to speed up queries is well studied subject and summarized in the same paper.

Another approach [28] studies automatic recycling of intermediate results of a query plan to speed up any following queries with overlapping intermediates.

4.2 Information visualization

This section will introduce the reader to information visualization. The goal is it understand, what kind of techniques and information are needed to create supporting visualizations, that would help the analysts to process vast amounts of information more efficiently.

We will first briefly explain theory behind visualizing data in sections 4.2.1 and 4.2.2. Then, we will take a closer look into useful visualization techniques in the context of security visualizations in the following sections. Section 4.2.3 introduces visualizations of timelines, section 4.2.4 explains visualizations related to network flow data, and finally section 4.2.5 shows techniques to detect attacks using visualizations.

4.2.1 Visualization

Information visualization produces visual representations of abstract data, with a goal of reinforcing human cognition to enable the viewer to understand the structure and relationships in the data.

visual representations and interaction techniques take advantage of the human eye's broad bandwidth pathway into the mind to allow users to see, explore, and understand large amounts of information at once. Information visualization focused on the creation of approaches for conveying abstract information in intuitive ways. [29]

This means that because visualizations make use of the human mind itself, analysts do not need to learn any specific skills to interpret the data. They are able to quickly draw conclusions or generate hypothesis to later verify

with more formal analysis. Visualization is thus an important part of data analysis techniques, among with other tools such as statistics. This ability of humans was successfully exploited, for instance, to solve an previously unknown structure of a protein, which had troubled scientists for more than a decade. This breakthrough was achieved by playing a game called Foldit. [13, 22, 33]

In addition to more basic tools of representing data, such as charts (pie chart, bar chart, histogram, etc.), plots and graphs, the information visualization community has come up with plenty of other intelligent ways of visualizing different types of data. Examples of these include treemap, which can be used for hierarchical data, and heatmap, which illustrates intensity of a value in an matrix.

Information security is facing the same problems as for instance business corporations. Plenty of data is available or generated in data collection, but the problem is to make a good use of it. After a security incident, the analysts might be facing an amount of data that cannot be handled only through manual inspection, and looking only the raw data might be very hard to interpret any clear patterns in the data. Visualization is a useful way of getting a quick insight of the data to guide the investigation further.

Security visualization is still relatively young area of research and the techniques used are mainly based on the results of more traditional information visualization research. The following chapters will present techniques to support the analysis and to detect attacks, anomalies and structures. The information presented here is mainly based on the work done by Raffael Marty [51] and the SecViz-community.

Benefits of visualization include: [51]

Answer and pose questions It is possible to create a visualization for each of the question that needs to be answered during an investigation. Very often these visualizations then make the viewer to pose new questions, which makes the process iterative by nature.

Explore and discover Visualizations make it possible to explore very large datasets through a compact representation. Different representations are needed to highlight different properties and relationships of the underlying data.

Communicate information Due to the fact that visualizations encode a lot of information into a easy to understand format, they are a way to communicate information efficiently.

Increase efficiency Visualizations can speed up the investigation process dramatically compared to for instance raw log analysis, because the

pattern and relationship recognition capabilities of humans are in full use.

However, these described benefits are only valid when the visualization in question is created properly and by recognizing how human eye processes information. Visualizations done wrong can even misguide the analyst and result in completely incorrect conclusions. Human visual system has a certain set of rules that define how visual perception works.

Some elements in an image are detected immediately when the image is first viewed. These so called pre-attentive visual properties require no conscious attention to interpret. These basic visual properties can be grouped into four groups: form, color, position and motion. Each of the groups have additional sub-elements to consider, and they are illustrated in figure 4.3. Important information should be presented using these properties to make the viewer spot important parts immediately.










Group	Attribute	Illustration	Group	Attribute	Illustration
Form	Orientation		Position	2D location	
	Line length		Color	Hue	
	Line width			Intensity	
	Size				
	Shape				
	Enclosure				

Figure 4.3: Pre-attentive visual properties [51]

Keeping these basic features of human vision in mind, we are able to present a set of basic principles of graph design [51]. Following these principles results in better quality and understandable graphs.

Reduce nondata ink The less ink is used to draw the graph the better. All the ink spent should be used to present the data, everything else should be removed.

Distinct attributes Previously presented visual properties are used to highlight important parts of the graph. However, humans cannot remember too many features of a graph, so one should not try to encode too many properties into a single graph.

Gestalt principles Gestalt principles are a set of visual characteristics that help viewer to find a graph and detect important parts of it. Gestalt principles are proximity, closure, similarity, continuity, enclosure and connection.

Emphasize exceptions One should make exceptions stand out even more, because usually those are the things the graph tries to communicate in the first place.

Show comparisons Instead of presenting only the raw data, showing something to compare with allows the viewer to understand the differences better and find the interesting anomalies.

Annotate data Graph should also state what the data is about by using legends and axis labels. Sometimes even a text explaining certain values can be used.

Show causality When a graph shows an exception, the next questions typically are looking for a reason for it. Graph should try to give an explanation for the root cause behind the exception. Here, one graph might not be enough and a second one is needed to clarify the interpretation.

4.2.2 Visualization process

Previous section discussed the basic elements that make a good visualization. The path from a collection of data to highly informative visualization is, however, not just a easy single step, but requires a careful processing of the data through a complete visualization process.

Marty introduces a six-step information visualization process in [51].

1. **Describe the problem:** First, define the questions that need to be answered by the resulting visualization.
2. **Assess available data:** Find out what data is available and what additional information is needed.
3. **Process information:** In this phase, information is extracted from the source, filtered and processed into a format that can be visualized.
4. **Visual transformation:** Next, the extracted data is transformed into a graph, based on the selected visual properties.

5. **View transformation:** If the previous phase did not result in already easily interpretable result, it can be modified by zooming, aggregating the data or for instance modifying the selected properties to better represent the interesting parts. This phase is also iterative in nature, since the result is viewed in between and then further modified to better represent the wanted features.
6. **Interpret and decide:** Finally, the conclusions are drawn based on the result from previous phases. Often, however, the result may be hard to interpret on its own and needs other graphs or verifying to support the decision. At this point it is also useful to remind oneself if the result really answered to the original questions posed, or did it reveal some other features of the data that should be investigated with new question and perhaps with a better suiting graph.

4.2.3 Timelines

When performing a forensic analysis, the time factor is strongly present. Analysis tries to discover when and in which order certain events happened, thus a timeline is generated from the data obtained from different sources. Log entries typically carry a timestamp revealing when that entry was written, and often the tools used to extract data from the inspected system also are able to attach a time for the data.

Analyzing this kind of timeline, that is, a sorted set of log events, by only looking at the entries does not easily give a good image about the level of activity of the system at a given point of time. A visualization of the level of activity over a selected field and aggregation function can help to better understand the time relations between entries. Furthermore, many times anomalous activities can be seen to differ from the normal activity profile of the system, either as a increase or decrease, by looking at different metrics. These points of time are good pointers to focus the investigation to. Figure 4.4 shows an example of activity visualization, clearly indicating an increased level of activity.

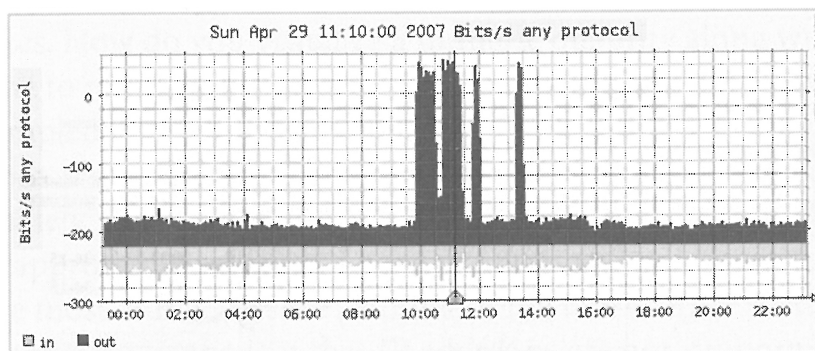


Figure 4.4: Visualization showing an increased level of activity [51].

Count is the basic aggregate function that can be applied for every set of log entries. Increased logging activity implies increased activity of the system and can reveal an ongoing attack. In addition, a specific log type may carry other information that is suitable for visualization. For instance, WWW server log typically includes fields such as bytes transferred and response time. Aggregating, for example, bytes transferred by summing or response time by averaging can reveal other types of anomalous activity that only the event count is not revealing.

Combining these kinds of activity measures with the raw data allows a better understanding of the system and relations between the activity level and certain events.

4.2.4 Network flow data

As previously described in section 2.5.2.2, network forensics can be seen as one of the main areas of digital forensics that needs its own analysis techniques. This is mainly because individual log entries in a packet capture have very little information in themselves about the overall events. Entries need to be aggregated to produce meaningful information, and even then the information might be hard to interpret. This is where meaningful visualization can be of great help.

Steps to analyze network flow data using visualizations are presented in [51] and shortly described below.

1. Gain an overview
2. Analyze overview graphs
3. Discover services of target machines

data using a treemap, which shows the distribution of protocols between the network machines. An example of a treemap is shown in figure 4.6.

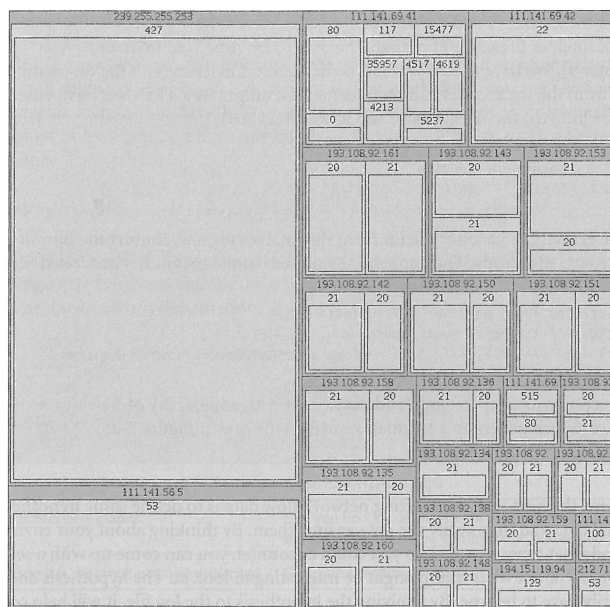


Figure 4.6: Treemap showing an overview of services used in a network [51]

After gaining an overview, hypothesis are formed and tested using custom generated visualization for each individual case.

4.2.5 Attack visualization

Expressiveness of parallel coordinates for visualizing different types of network attacks was presented in [12]. The authors argue that parallel coordinates is a relevant visualization for many attacks, because the attacks impose one-to-many relationship between some of the characterizing variables. By analysing different attacks types, they selected four variables that are enough to detect those attacks: source address, destination address, destination port and packet size. For instance, typical port scan includes one source, one destination, many ports of that destination and constant packet size. Figure 4.7 illustrates signatures of different attacks using parallel coordinates.

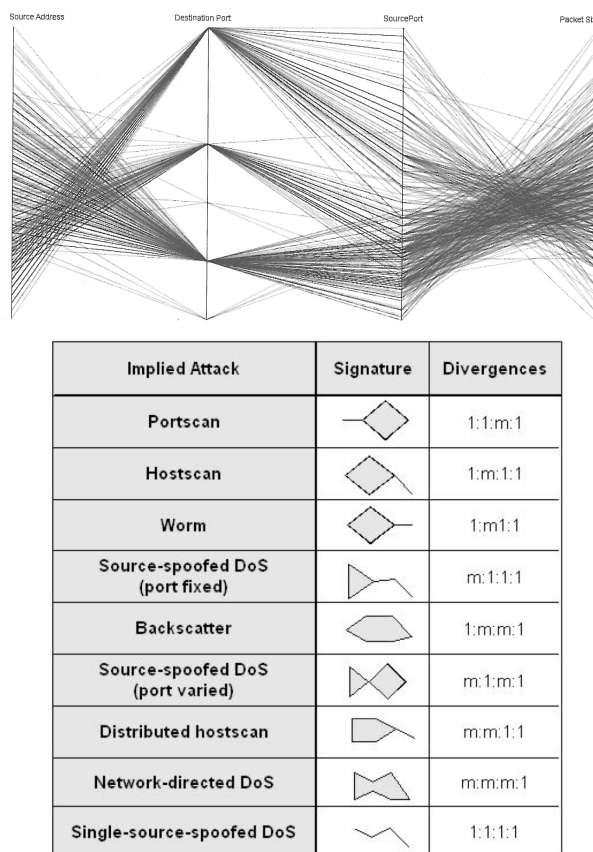


Figure 4.7: The upper image shows network traffic to a single host visualized with parallel coordinates [51]. The lower image shows different attack signatures seen from visualizations such as the upper. [12]

Another research investigated detecting distributed scans using query-driven visualization.[56] The authors reasoned that the current tools (typical shell utilities) are not sufficient considering the fast increase in data volumes. As a result, they generated multi-dimensional histograms from very large datasets (almost 300 GB) using the FastBit-library [20] to identify the scans and the attacking hosts visually. Even though they had a pre-defined assumptions to investigate in their research, we believe that these results may be usable in a more broad context too. Figure 4.8 shows some of their visualizations, clearly indicating malicious activity.

Network and port scans were also researched in [44], where the focus was limited to port activities in a few levels of detail. Figure 4.9 shows the different visualizations used. The level of detail between the views increases from left to right, from a summary timeline to individual port details. This implementation, however, is not as self-explanatory as other simpler visualizations

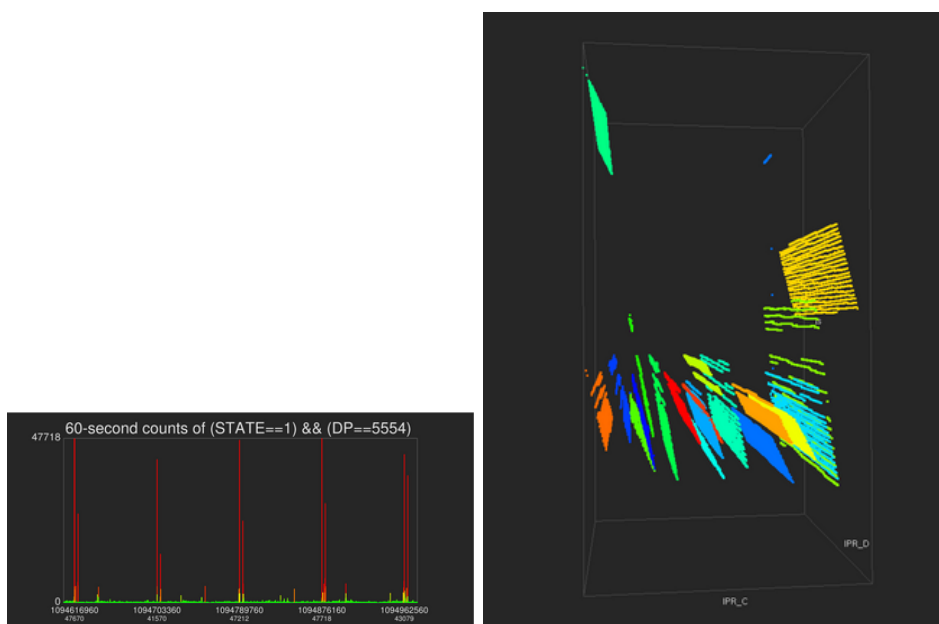


Figure 4.8: Distributed scans [56]

and thus requires more domain understanding from the analyst.

4.3 Anomaly detection

The previous section introduced visualization techniques to help an analyst to interpret data more efficiently. Visualizations present high volumes of data in a very compact form and allow the analyst to use visual capabilities of humans to detect anomalies and different relationships effortlessly. This way, however, the detection is still done manually and could be improved even further.

Automatic detection of intrusions have been widely researched in the context of firewalls, intrusion detection and intrusion prevention systems. Generally, the techniques can be divided into two categories: misuse detection and anomaly detection. [53]

Misuse detection is based on an extensive collection of fingerprints that represent activity that is considered harmful. This means that the system is only able to catch activities that are already identified and known before. It requires a very good knowledge of what is suspicious and what is not, and keeping the collection of fingerprints up-to-date can be a difficult task. To be able to detect misuse behaviour in data previously unknown or not completely understood, we need more elaborate techniques. [53]

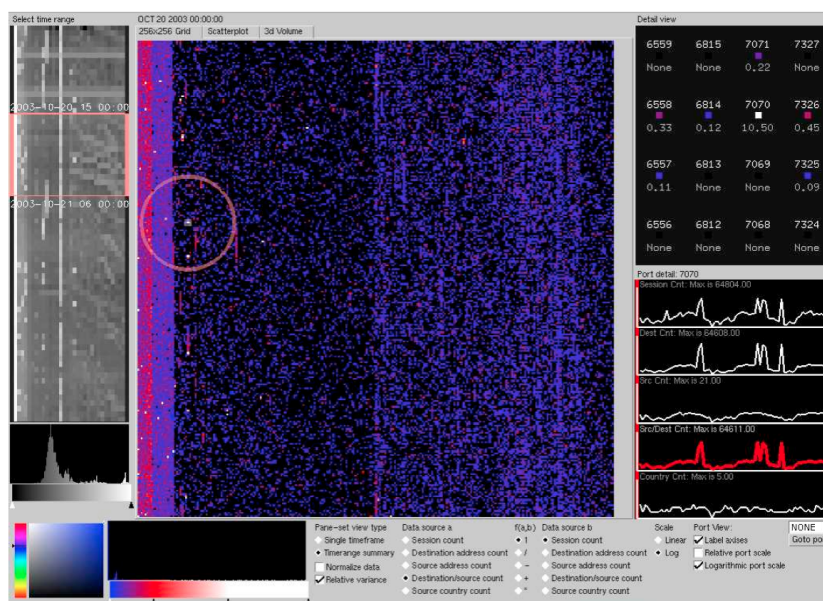


Figure 4.9: Visualizations of port activity [44]

Anomaly detection tries to overcome this issue by creating a profile of the activity that is considered normal. Profile here is a set of metrics that measure certain aspects of the behaviour. New data is then compared against this profile representing a typical activity pattern to detect deviations from it. In other words, anomaly detection assumes that the behaviour of users can be predicted and modelled. Flawless modelling is, however, very difficult. Often the most reliable approach is a hybrid one, combining many different techniques starting from misuse detection. [53]

The following sections will explain different methods for automatic detection of anomalies. Section 4.3.1 introduces a historically important model for intrusion detection. Then, section 4.3.2 explains basic quantitative and statistical methods. Section 4.3.3 gives an overview to modern techniques for extracting features from data and automatic classification. Finally, section 4.3.4 shows that the most reliable results are achieved using techniques exploiting application-specific features.

4.3.1 Denning's model

The first remarkable research in the field of intrusion detection happened when Denning published her intrusion detection model, now known as Denning's model. It is widely used as a base for intrusion detection systems, although its focus is more on detection of malicious activities on hosts rather

than network. It consists of four models that are suitable for different metrics of the system. [53]

- **Operational model:** Operational model gives certain metrics a threshold that is considered normal behaviour within a timespan. Any deviations from it are considered anomalous. This model is mainly suitable for different event counters.
- **Mean and standard deviation model:** This model assumes that mean and standard deviation are enough to describe the behaviour patterns. Observations outside the confidence level characterized by these parameters are then marked abnormal. According to Denning this method is relevant for event counters, timers and resource usage measures.
- **Multivariate model:** Multivariate model extends the previous model by adding correlations between different metrics.
- **Markov process model:** The last method is a stateful model that builds a matrix of possible states in the system, transitions between the states and their probabilities. This means that the model is not focusing on detection of single suspicious events but rather anomalous sequences of events. When the probability of a certain sequence is very low, it is considered abnormal. Use of this model is limited to event counters.

4.3.2 Quantitative and statistical methods

Commonly used and the simplest technique in anomaly detection is quantitative analysis of certain numeric features. For instance, the overall activity measured needs to stay within a specified threshold for the activity to be considered normal. This is called *threshold detection* and can be applied to any numeric, also calculated, features. This simple threshold is user defined and static, thus probably hard to define reliably. Applying a simple heuristic can make this more adaptive to different scenarios by utilizing more elaborate functions to define what is normal and what is not. For instance, rather than defining that a certain static number of login failures are allowed within a timespan, it might be more accurate to make that number vary as a function of the time or even based on statistical deviations. [53]

Statistical methods, described in [30], take the inspection of individual features one step further. They create a complete statistical profile from the individual metrics. In addition to comparing absolute values of these

individual measurements, they take into account all the metrics and their relationships, that is, the correlation between metrics. This means that the system creates an alert only if a metric has a value that is not expected considering the system's full state and historic values.

These techniques in their simple forms are included in the Denning's model introduced in section 4.3.1.

4.3.3 Data mining and machine learning

Quantitative and statistical methods described in the previous section can only be used with strictly measurable metrics, such as event counts. However, the events might be carrying much more features than these metrics can directly present. Typically expert knowledge about the inspected data provides the most accurate modeling, but in cases where it is not feasible, automatic feature extraction techniques can be used. Data mining is the discipline commonly used to perform this kind of classification, link analysis and sequence analysis over large datasets.

Many techniques for this purpose have been studied, including classification and regression trees [9], support vector machines and neural networks [57]. In addition, [38] concluded that mined frequent patterns can be used as reliable detection models for detecting anomalies. Frequent episodes for intrusion detection have been studied also studied in [25] and [34].

Machine learning, on the other hand, is a term used for prediction based on already known properties of the data. It shares many of techniques of data mining, and is partially overlapping discipline. Once a model is constructed using data mining on the existing data, new events can be classified using machine learning algorithms.

4.3.4 Application specific techniques

In the previous chapters we introduced techniques that are applicable for the general purpose anomaly detection. However, the problem typically with these techniques is accuracy, which means that they may result in a high number of false-positives. To achieve very low levels of false-positives, it is required to create techniques that take into account other characterizing features of the application data and not only the raw numbers. This was researched in [36] and continued in [35] where the authors introduced anomaly detection techniques for web application traffic and were able to achieve false-positive rate as low as 0.2 %.

Intrusion detection systems typically inspect web traffic utilizing misuse detection techniques, using a large collection of signatures describing different

attacks. The problem is that these collections quickly become obsolete as new attacks emerge at a remarkable speed. This is where anomaly detection can be a great help. The techniques presented in [35] take a close look into the web request and their parameters to detect unusual parameter content. These techniques are shortly described below.

1. **Attribute length:** This technique assumes that the parameters are mostly defined by the application and should remain relatively static in length. Thus, it is possible to approximate the distribution of these lengths and detect deviations from the typical behaviour.
2. **Character distribution:** This technique finds a "normal" structure for the query parameters by looking at the characters in query parameters. Typical parameters consist of a relatively restricted set of characters, and just as in normal languages their relative frequencies are not constant. This way, it is possible to detect in a flexible way when the parameters contain unusual content.
3. **Structural inference:** The previous method looked only on the character distribution without considering the more specific structure of the parameters. This can be taken into account by building a grammar that describes the legitimate structural behaviour of the parameters.
4. **Token finder:** Often the attributes are tokens or flags that can have a few different values. Discovering these value sets allows immediate detection of illegal values and a possible attack. However, if no such rule can be discovered, the values for the field in question have to be considered as completely user-defined.
5. **Attribute presence, absence and order:** Assumption that the user is not writing the queries by himself and they are rather generated by the application implies that a change in the presence or absence of an attribute is a sign of tampering of the request. Hence checking that parameters that normally are found together in the request are there and in the right order can effectively reveal an attack.

4.4 Existing solutions

The previous sections have introduced important themes related to the capabilities that an analysis supporting software product has to consider. These themes are taken into account in our implementation. This section will introduce currently available products for different use cases.

Log management is a big and growing area for software systems. Many organizations, partially due to compliance requirements, have understood the benefits of centralized management of logs and security events. These log management systems typically extend their capabilities from only storing the events to intelligent querying, aggregation and correlation. In addition to log management, so called security information and event management systems (SIEM) have much in common with these systems. Many companies have their own solution in their offering, but also many open source options are available. Some of these solutions include Arcsight [27], OSSIM [5], ELSA [17] and Splunk [55], which is known for its flexible querying and reporting system. However, none of these products have been designed for high-performance analysis of a fixed set of events, but rather to support a continuous flow of events and reporting without strict performance requirements.

Some tools exist for analysing individual log formats only. Probably one of the most powerful software, in addition to Splunk, supporting many different formats and allowing analytical querying using a SQL-like syntax is Microsoft Log Parser [41]. Log Parser integrates especially well to Microsoft environment, but has use also outside of that. However, it has a very strong focus on analytics, rather than analysis of individual log entries. On contrary, Mandiant Highlighter [40] focuses on manual analysis of any line-oriented data files with features such as searching, filtering and highlighting. However, it lacks any advanced analytics that the Log Parser has.

Furthermore, computer forensics is another special field for software. There are two main commercial software packages for performing forensic analysis for hard drives: FTK [3] and Encase [24]. Both offer extensive indexing and analysis of artifacts found from a hard drive, but neither of them support full analysis of a timeline built from the events and artifacts discovered.

Open source tool log2timeline [39] is probably the most used tool in computer forensics community for building these types of timelines. However, the output from this tool is simply another log file containing all the different events combined from the system's files. The analysis of this result is left for the analyst, and is often performed using traditional shell tools (less, grep, sed, awk), Microsoft Excel or in some limited cases Splunk, due to its strict licencing based on the amount of data indexed. Very recently a project trying to solve this challenge emerged. L2t Review [37] generates a database from the output of log2timeline and allows querying over this database using a designated GUI. However, it is strictly limited to be used with log2timeline.

4.5 Suitability of existing solutions

We presented and briefly evaluated many existing tools in the previous section. This section summarizes the differences to our requirements and clarifies the motivation for our implementation.

The only tool satisfying our functional requirements is Splunk. It has a very extensive query interface and can be extended to support almost any type of log entries. However, based on our initial testing, performance of a single system is inadequate for iterative exploration of big datasets. In addition, it is a commercial product with inflexible licencing for varying indexing needs. Furthermore, it is not easily extendable to support experimentation and adaption for varying use cases.

What we are trying to achieve is a combination of Microsoft Log Parser and Mandiant Highlighter with better support for interactive querying and inspection of individual events with supporting visualizations. However, also these are closed source products that do not allow extending the features to better fit our needs.

The first version of L2t Review was released only after we had investigated the current offering. As log2timeline is able, or extendable, to parse almost any format, and this tool is open source, it could have been a good candidate as a starting point for our customizations. However, considering our requirements for example for performing progressive querying, we believe that our choices of technologies introduced later are more suitable and scalable for this specific use.

Chapter 5

Design and implementation

This chapter explains the design and implementation of the software we created to support the analysis of large datasets during forensic investigations. We will first present the two most important contributions: the query interface in section 5.1 and timeline visualization in section 5.2. Then, the basic architecture of the whole application is explained in section 5.3

We decided to select Python as our language of choice, because it is widely used by scientific communities and corporate users. Furthermore, the wide and open community and availability of thousands of ready software packages and libraries allows to focus on solving the problem itself without having to reinvent everything from scratch.

5.1 Flexible query interface

In this section we will introduce the most important contribution of this thesis: implementation of a query language to support analysis of large datasets.

Instead of having complex and static forms used for building queries, a flexible "pipe based" language is used, widely used on unix shell environments. The language is based on commands that produce results, which can then be given as input for a subsequent command. That is, the commands are *piped* together to form a so called *pipeline*. Instead of piping only text or other unstructured data, as done in unix shells, the commands produce objects. This allows more flexible, efficient and lazy population of the end result. In addition, this language is extendable in a way, that lets the user easily define new commands on the fly and use these new commands as part of the pipeline.

For instance, the following query demonstrates the core features we are going to support. The result shows the amount of transferred data per month,

excluding certain images, only from specified IP address, and ordered by the amount of data.

```
filter .jpg .png .gif
  | where ip='127.0.0.1'
  | groupby year, month, sum(bytes) as bytes
  | orderby bytes
```

We will first introduce our database system of choice in the section 5.1.1, which we are then querying using our language. Then, we will introduce the query pipeline in detail in section 5.1.2, and the features implemented on top of the database system.

5.1.1 Database

Because the end result, the application, is basically an easy to use user interface on top of a very capable database system, the choice of a correct system was seen as crucial to be able to fulfill the goals for our implementation.. Based on the research done in chapter 4 we understood that our use case, querying over a static set of entries, allowed us to abandon most of the typically important properties of a general purpose database system. We also understood that column orientation and bitmap indexing technologies will help to deliver the best possible performance from a single system, that is, without employing a bigger cluster of computing power.

In addition, we discovered that certain requirements, such as exploration of data using progressive or iterative querying, are features that no currently available, fully SQL-based system is able to deliver easily. These reasons made us select Fastbit [20] as our database of choice.

Fastbit is a C++ based database library, one of the pioneers in taking the most out of bitmap indexing technologies. Its performance for more analytical workloads, as needed by us as well, is well evaluated in [1] and supports our choice. Being a library it also allows easier experimentation on a lower level to meet all the requirements, without sacrificing performance. The only limitation with very large datasets is that the data needed for a query needs to fit into the memory to be efficient. For most queries this is not an issue, since they can be answered solely with indexes, but for instance aggregate queries might bring more data into memory.

Because Fastbit is a library implemented in C++ rather than a typical database system, we need ways to program with it using Python. We decided to use Cython [14] to develop the bindings between Python and C++. Cython is a full language, a superset of the Python language, which allows to

interact with C or C++ code in an easy and efficient way. An extension written in Cython is first compiled into C code using many optimizations. The resulting code can then be compiled normally with any C or C++ compiler and used as a Python extension. Cython can also be used in our implementation to optimize other critical parts.

5.1.2 Query pipeline

Implementing a query pipeline itself is a fairly simple task considering the flexibility of the Python language. Most of the challenges we are facing only when trying to meet the efficiency and other specified requirements.

The pipeline is simply constructed by implementing all the commands as functions inside a specific module and dynamically locating the function by the string representation of its name. Then the commands are executed step by step, passing the results of a previous command as input for the next one. However, the biggest challenge here is to ensure efficient resource usage between the commands.

Solutions to this and other challenges on a lower level are presented in the following sections. However, for Python commands generator-objects are returned where feasible to allow lazy evaluation of commands and to reduce processing needs. We will first look into efficient data reduction in section 5.1.2.1. Then, the implementation of progressive queries is explained in section 5.1.2.2. Implementation of efficient paging is explained in section 5.1.2.3. Finally, the core commands are introduced in section 5.1.2.4.

5.1.2.1 Data reduction

One of the main use-cases for the software is filtering the set of events, excluding uninteresting pieces of information and on the other hand finding the relevant information. The user can filter or search events matching certain criteria using the query language either by matching a string against the full log lines or by querying more efficiently against the parsed fields stored in bitmap indexed columns.

However, the biggest challenge is to store the result of these filtering queries in an efficient way, because the following commands may want to use the results for further filtering or more analytical queries. This means that we cannot bring the data itself into memory until the very last moment and that there must be a way to efficiently represent the individual query results and use them as a starting point for subsequent queries.

Fastbit makes this possible because it can answer all the simple filtering queries by using its bitmap indexing technology only. The resulting set of

entries is represented by a data structure called *bitvector*, which is a vector of single bits, having a *n*th bit set for the *n*th record in the data partition matching the query in question. This allows us to pass a result set forward in the pipeline in the most compact form, without any of the data itself. Once the data is needed, it is brought into memory as Python's objects.

As a simple benchmark, let us assume that we have a dataset of 10 million records, and a query returns a subset of 8 million records. A typical way to represent this result set without bringing the data itself into memory, is a set of row identifiers, which is typically a vector of 64 bit integers. 8 million such ids would take approximately $8 * 10^6 * 8 * 8bits = 512 * 10^6bits$ of memory to represent, which is approximately 64 megabytes. Storing that much data for every intermediate result would incur a significant overhead. Representing the same result set as a bitvector takes only 10 million bits, which is approximately 1 megabyte in uncompressed form. After Fastbit's compression the memory requirements are typically in hundreds of kilobytes, depending on compressibility even only kilobytes or bytes, and thus already on a very low level.

5.1.2.2 Progressive queries and caching

The idea of progressive queries was introduced in section 4.1.3. Basically, as data exploration is an iterative process, we want to efficiently use the previously calculated results for the following, slightly modified queries.

We decided to implement progressive queries using the technique "repeated evaluation via query merging" introduced in section 4.1.3. This is currently possible because most of the queries can be executed very fast and the merging process is also extremely fast due to the use of bit operations only. Fastbit's programming interface currently does not support the use of bitvectors as a starting point for new queries, so the merging is done manually in the query pipeline, but implementing it is investigated as future work. However, if the query can be answered using the indexes only, having an initial row set does not yield in any better performance.

As explained previously, the most expensive operation for Fastbit is filtering by string matching, because it cannot be answered using bitmap indexing and a full sequential scan is required. Typically, filtering or data reduction is done incrementally, so that new filtering strings are added when new uninteresting entries are encountered. Fastbit seems to implement these multi-term queries by doing a full scan for each of the strings. This means that the query time increases linearly with the number of strings in the query. To resolve this and better support progressive queries, we implemented a separate bitvector cache for string filtering. The string terms used and the resulting

bitvector are saved in memory, and when a new query is executed, the cache is first searched for a subset of the string terms needed. If an intermediate result is found, only the differing strings are used in the query and the result is merged with the result of the subset query to form the final result. This results in a great performance improvement for iterative exploration of the data.

5.1.2.3 Paging

When analysing the results of a query executed, there is only a limited amount space on the screen to display the results. In addition, the analyst might be interested to inspect only a small number of results to see if the query was satisfying or not. This is why paging is an important feature for the end user, but also for performance of the whole system because less data can be brought into memory and less data needs to be processed later in the pipeline.

To follow the technique of avoiding construction of python objects until the very last moment and bringing only the required entries into memory, paging has to be an integral part of the query pipeline. This is because as bitvectors are passed through the pipeline, the data selecting command has to be aware of the requested offsets and counts, and the selection cannot be done later in the application efficiently. We chose to implement paging as efficiently as possible using bitvector masks that are constructed and applied just before selecting the data to memory.

Paging masks are constructed as follows. We first take the requested offset and create a bitvector mask having all the bits set starting from that offset. We then perform an AND bit operation with the bitvector resulting from the filtering operations. The resulting vector of the operation then represents all the entries starting from the given offset. We then iterate through the bitvector's set bits decrementing the given count at the same time. This way we are able to find the given number of bits and the offset of the first and the last matching entry. Now, we take these offsets and construct a mask having all bits set between the offsets. Performing an AND bit operation between this bitvector and the original bitvector, which was the result of filtering, gives as the final result, which contains only the requested number of entries starting from the given offset.

The same routine can also be performed to other direction, in case we need to find a certain number of entries starting from a given offset but counting backwards. This situation is represented by a negative count parameter in the paging system.

5.1.2.4 Core commands

The core commands supported by the query interface are introduced below. In addition to these commands, of course many others are needed to support an extensive investigation. However, these are the commands that are provided by the system, mostly executed efficiently on a lower level, and used as a base for new, more expressive commands. More detailed usage examples are introduced in the case study in section 6.

where This command takes a string of conditions as an argument. This string is passed directly to the Fastbit query processor and follows the syntax of WHERE-clause in SQL-language. The result of this command is a bitvector.

filter This command is a special case of the where command. It takes a set of strings as arguments and uses these strings as either excluding or including filters against the full text representation of entries. As explained previously, it also intelligently caches the results and is able to use previous subset queries to optimize performance. The following is a valid example of the arguments given to this command.

```
filter foo +"foo bar"
```

Plus sign ('+') represents an including term, and terms containing special characters like spaces can be quoted.

The result is a bitvector.

select This command is used to select data from the database. The argument follows the syntax of SELECT-clause of SQL-language. Input for this command is a bitvector, which is used together with the arguments to bring only relevant information into memory. The results are not directly instantiated as Python objects, but instead kept inside the Fastbit engine as a so called table-object. This command returns a Python wrapped table object that can be iterated through normally in Python. This ensures that no unused Python objects are created too early because they are instantiated lazily when needed.

groupby Groupby command acts very much in a same way as the select command. The only difference is that it allows the use of aggregate functions in the select clause. Here, the functionality differs from typical SQL-statements. Fastbit performs implicit grouping operation if any aggregate functions are defined in the select clause. For instance,

the statement `foo, bar, sum(baz)` first groups records by `foo` and `bar` columns. After that, within the resulting groups, the operation `sum(baz)` is performed and the result is represented by a single row per group. The result, like in the `select` command, is a table-object allowing lazy instantiation of Python objects.

orderby This command takes a table-object as input and sorts it according to the column names given as arguments. The result is also a table-object.

5.2 Timeline visualization

Because the log entries can be considered as an timeline with large amount of information, an important supporting feature for analysis is visualization. Timeline visualization allows to encode very large amounts of information into a compact representation, which can be analyzed visually.

We implemented a flexible timeline element according to the theory introduced in section 4.2 that allows the analyst to visualize different properties of the log entries easily to understand how they develop over a given period of time. The visualized data is calculated using the standard query interface introduced previously and thus allows the use of more complex aggregate functions to derive more information from the data. In addition, multiple timelines can be drawn at the same time, which allows correlation between the selected properties.

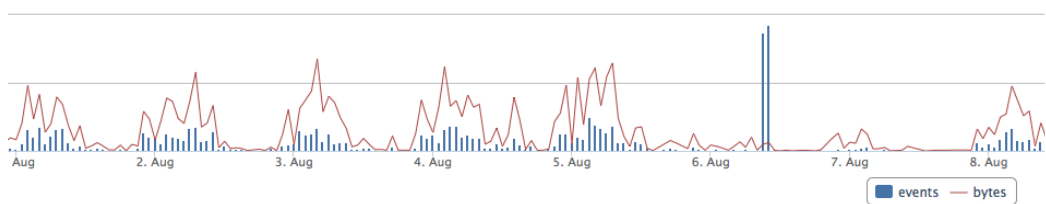


Figure 5.1: Two timeline visualizations showing event count (blue) and bytes transferred (red) and revealing suspicious activity (cropped)

For instance, figure 5.1 shows two timeline visualizations from a web server log file of 8 days of data. We can clearly identify the working days and identify suspicious activity during the weekend.

In addition, as the timeline visualizes different moments of time, the user can select a moment from the timeline and the events from that moment are shown and can be inspected more in detail.

5.3 Application

The application itself was divided into two pieces: a server and a client serving the user interface. The idea is to separate the core system from the user interface, and only provide a clean interface for the use of a client application. The details of this separation is explained in the following two sections. First, we will describe the implementation of the server in section 5.3.1, followed by description of the currently implemented client in section 5.3.2

5.3.1 Server

The main functionality of the server is to integrate the database and the query pipeline together and provide a simple interface for different clients to use. It takes the query in, executes it through the query pipeline and returns the result to the client, possibly extending the query pipeline if the result was nothing useful for the client. For instance, selecting default values from the database if only a bitvector was returned from the pipeline.

We decided to use the currently popular method of exposing application programming interfaces (API) through REST-like (representational state transfer) interface over HTTP-protocol. However, we expose only one resource, the query interface, which all of the different functionalities implemented in the client must use. The only addition to the interface, which can be done outside the query string, is paging. A query is executed by sending a POST-request to the resource `"/query/[count]/[offset]"`, where count and offset are parameters given to the paging system. The result is a text response formatted in JSON (javascript object notation).

The server is implemented using the Flask [21] microframework for web development, because of its great simplicity and our modest requirements for the interface.

The server is also used to transfer the default client code to a web browser, which is explained in detail in the following section.

5.3.2 Client

We selected web browser as our default client for one simple reason: platform independence. Web browsers are available for every operating system and their capabilities are now on a level that meets our requirements for a client. Full applications can be run completely inside a browser using Javascript, without any specific needs from the server side.

However, as browsers can be considered as application platforms, development using only raw Javascript, HTML and CSS can be relatively laborious. We decided to select a couple of libraries to speed up the development and improve experimentation and maintainability. The current development stack includes the following libraries: jQuery [32] (a general purpose Javascript library), Bootstrap [61] (user interface framework) and Highcharts [26] (Javascript charting and visualization library).

The user interface (UI) currently consists of three parts: timeline, query input field and data viewer. These elements are shown in figure 5.2



Figure 5.2: User interface of the client

With the timeline, the user is able to easily visualize different properties of the underlying data over time. Basically any query result having a timestamp column and another column with numeric values can be drawn into the timeline. In addition, the user can select any point of time from the timeline and the current query is pagged to show entries from that point of time.

The query input field is used to write queries, which are then executed on the server. The result of the query is shown in the data viewer. The client decides how the result is rendered and has different output modes for different situations. For instance, when the log entries are inspected, the lines rendered directly and highlighted using a syntax highlighter. If on the other hand different columns are selected by hand, they can be viewed as a table. Further, the results of more analytical queries can be even visualized in the client.

5.3.3 Parser and importer

Fastbit stores the data in so called *data partitions*, using a partition specific schema. To be able to import the data entries, the schema must be specified

first. This means that whenever we meet a new file format, we look for fields and their types in the entries and define the schema based on that information. Once that is done, reusing the information later is easy.

Fastbit includes a fast importer for CSV (comma separated values) file format, so we decided to use that instead of populating the partitions ourselves. This means that, for each different file format, we only need to first define the schema and write a parser that converts the file into CSV format. We use Python's standard regular expression library to parse and extract field information from the data entries and write them to a file. Support for new file formats can be easily added by implementing a regular expression for the format in question. After that, population of a new data partition out of an input file can be done by a simple script integrating the schema information and a parser.

Chapter 6

Case study

We have now implemented an application to support digital forensic investigations. This application basically acts as a query interface to a very efficient database system allowing efficient iterative investigation and timeline visualization. In this chapter we will evaluate the suitability of this application for real investigations.

We were provided with a dataset from a real security incident. The next chapters will look into that dataset with a goal of not completing the full assignment with our application, but rather evaluating its capabilities in finding first signs of possible breaches as fast as possible.

For the evaluation, no exceptionally powerful or dedicated server was used, but instead we wanted to see how a typical work computer would perform. The evaluation was executed with a normal MacBook Pro laptop having 6 GB of memory and a Intel Core 2 duo (2.66 GHz) processor. All the measurements reported are measured by the client (web browser), and therefore include also all the overhead of processing the data in Python, possible paging and the transfer from the server to the client. We decided to measure the whole request time because it is the time the user perceives.

6.1 Background

In this case the assignment was to analyse a hard drive of a server where an anti-virus system had detected malware. This system was not part of a business critical infrastructure but still inside an isolated network which no malware should be able to enter. The customer wanted to know how and when the system was infected. As background information, initially only the time when the customer detected the issue was known.

Because only a hard drive of the system was provided for investigations,

copies of the drives were created and log2timeline and Sleuthkit were used to generate a timeline of the information available on the drive. The timeline generated was about 1.2 GB in size and included approximately 3.3 million events. This data was then analysed manually to answer the questions posed by the client.

Parsing and importing the generated timeline to our tool took 15 minutes and generated approximately 20MB of additional data for the indexes.

6.2 Investigation

The following sections will describe how we used our software for analysis of the generated data. The goal is to show how our tool enables very flexible and iterative analysis of any log data. As a result of the investigation performed for this evaluation, we will have clear signs of infection, a few moments of time when these suspicious events occurred and a few potential attack vectors.

Signs of infection

We started the investigation by filtering out any events that we considered not to have meaningful input in this phase. This data reduction, however, is an iterative process and thus new rules are added when uninteresting and repeating events are encountered. After the data is in memory, filtering with a single string takes 2-4 seconds and is still very efficient for this purpose. Queries involving parsed fields are fast by design and the evaluation itself does not take almost any time but the request returns to the client in typically under 100 milliseconds.

As a result, we were left with the following initial query:

```
filter "[EXIF metadata]" "[SetupAPI Log]" | where year=2012
```

This query visualized in the timeline, using day as a grouping precision gave us the following picture (figure 6.1) and returned almost immediately because of the filter cache and fast aggregation capabilities of our database system. The whole query took about 100 milliseconds.

This visualization shows two clear spikes in the level of activity. On the right side of the image, however, we can see an increase in the activity over a longer period of time, which is about a month. In addition, the time when the client detected the infection is inside this period, so we will take a closer look into that first. This can be done by clicking the highest bar, which takes the query viewer to that point of time.

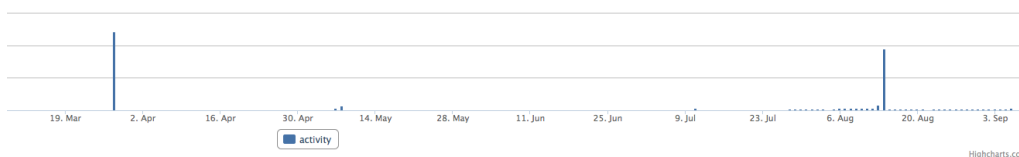


Figure 6.1: Activity of the system on the timeline (cropped)

We then started to browse the raw events and add new filtering terms. Retrieval of the next page takes only under 100 milliseconds. We were quickly faced with very suspicious events and could verify that the system was indeed infected. These events are shown in figure 6.2

```
[2012-08-14 03:05:47] [...b] "C:/WINDOWS/system32/11b32wapz.exe"
[2012-08-14 03:05:50] [m...] "C:/WINDOWS/system32/11b32wapz.exe"
[2012-08-14 03:05:58] [...b] "C:/Documents and Settings/LocalService/Local Settings/Temporary Internet Files/Content.IE5/RVN9726V/A12[1].exe"
[2012-08-14 03:05:59] [m...] "[Internet Explorer] (Client Last Accessed/Server Modified) URL:ftp://gg11.in.int04.info/A12.exe cache stored in: RVN9726V/A12[1].exe (file: C:/Document
ES/Index.dat)"
[2012-08-14 03:05:59] [m.c.] "C:/Documents and Settings/LocalService/Local Settings/Temporary Internet Files/Content.IE5/RVN9726V/A12[1].exe"
[2012-08-14 03:06:00] [...b] "C:/Documents and Settings/LocalService/Local Settings/Temporary Internet Files/Content.IE5/E94BUS5R/B12[1].exe"
[2012-08-14 03:06:04] [m...] "[Internet Explorer] (Client Last Accessed/Server Modified) URL:ftp://gg11.in.int04.info/B12.exe cache stored in: E94BUS5R/B12[1].exe (file: C:/Document
ES/Index.dat)"
[2012-08-14 03:06:04] [m.c.] "C:/Documents and Settings/LocalService/Local Settings/Temporary Internet Files/Content.IE5/E94BUS5R/B12[1].exe"
[2012-08-14 03:06:09] [m...] "[Internet Explorer] (Client Last Accessed/Server Modified) URL:ftp://gg11.in.int04.info/C001.exe cache stored in: SSUWLAN/C001[1].exe (file: C:/Docume
.IE5/Index.dat)"
[2012-08-14 03:06:09] [...b] "C:/WINDOWS/system32/ccffhhj.exe"
[2012-08-14 03:06:10] [...b] "C:/Documents and Settings/LocalService/Local Settings/Temporary Internet Files/Content.IE5/CD23STUJ/D001[1].exe"
[2012-08-14 03:06:11] [m...] "[Internet Explorer] (Client Last Accessed/Server Modified) URL:ftp://gg11.in.int04.info/D001.exe cache stored in: CD23STUJ/D001[1].exe (file: C:/Docume
.IE5/Index.dat)"
[2012-08-14 03:06:11] [m.c.] "C:/WINDOWS/system32/ccffhhj.exe"
```

Figure 6.2: Suspicious events

As we can see, suspiciously named executables are created under system32 and downloaded using Internet Explorer. We can easily query all the executables created under system32 during that specific year with the following query:

```
filter [...] +system32 +.exe "Event Log"
| where year=2012 and birth=1
```

As a result, we found 12 executables, shown in figure 6.3, created during the very same day and those files were later verified as malware using anti-virus software. This information could have been also found by running anti-virus software against the provided hard drive in the very beginning. Nevertheless, it was only a matter of minutes with our tool, and having this information and after additional browsing of the events, we are able to discover the time when the first malware activated. However, this still does not answer the questions how or when the system got infected.


```

[2012-08-14 03:05:47] [...b] "C:/WINDOWS/system32/lib32wapz.exe"
[2012-08-14 03:06:09] [...b] "C:/WINDOWS/system32/ccffhhj.exe"
[2012-08-14 03:14:52] [...b] "C:/WINDOWS/system32/rppnkn.exe"
[2012-08-14 05:53:41] [...b] "C:/WINDOWS/system32/hffcaa.exe"
[2012-08-14 07:40:56] [...b] "C:/WINDOWS/system32/MKKIIGG.exe"
[2012-08-14 07:41:01] [...b] "C:/WINDOWS/system32/zzwwuus.exe"
[2012-08-14 09:18:47] [...b] "C:/WINDOWS/system32/NvddLL.exe"
[2012-08-14 09:18:50] [...b] "C:/WINDOWS/system32/bbJJrra.exe"
[2012-08-14 10:37:45] [...b] "C:/WINDOWS/system32/JNNSXX.exe"
[2012-08-14 10:37:49] [...b] "C:/WINDOWS/system32/PUYYddi.exe"
[2012-08-14 12:41:26] [...b] "C:/WINDOWS/system32/IIDDzzu.exe"
[2012-08-14 12:41:29] [...b] "C:/WINDOWS/system32/PPLLGGC.exe"

```

Figure 6.3: Files created under system32 by the malware.

Internet browsing activity

After confirming the infection, we are faced with the challenge of finding how the system got infected. During the time of activation of the malware, we also saw events showing use of the web browser (Internet Explorer). This is suspicious because the system in question should not be used for anything else than running a specific server software. We then decided to see when exactly has the browser been used using the timeline. (figure 6.4).

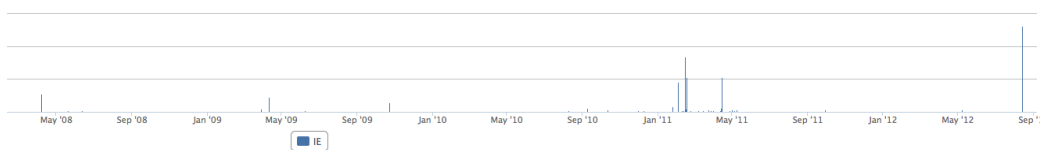


Figure 6.4: Internet Explorer activity (cropped)

From this timeline we can see some individual activities starting from the year 2006. On the right side of the image, however, there are two interesting time timeframes. During 2011 we can see a clearly raised level of activity, and then on the day of activation there is a clear spike. As the formatting of `log2timeline` for these events is quite verbose, we decided to extract all the URLs accessed and inspect them closer. This was done using the following query:

```

filter +"[Internet Explorer]"
  | where year>=2006
  | select_all default
  | field default
  | extract https?://\S+

```

The request took about 350 milliseconds to complete when the filter was already cached. We immediately see in figure 6.5 that the browser has visited,

among others, a lot of Chinese websites. This raises suspicions of the original attack vector being the web browser, although the URLs do not show any clear and extensive use of the browser. However, a few other URLs, which are shown in figure 6.6, caught our attention.

```
http://sclick.baidu.com/w.gif?q=0x0y0f6&fm=se&T=1344938542&y=DE7EF5DF&rsrv_sid=1288_1329_1267_1228_1360_1208_1362_1186_1282_1284_1319_1295_1331&cid=0&qid=d9c9b3b14eabce2&t=1344939284&y=E5D296CE&rsrv_sid=&cid=0&qid=8b79fba14f1da9f9&t=1344939482122&path=http://www.baidu.com/s?wd=脚臭的治疗方法&on
http://sclick.baidu.com/w.gif?q=0x000003000&fm=se&T=1344939554&y=872E9BE8&rsrv_sid=1289_1328_1268_1227_1186_1283_1179_1287_1319_1330&cid=0&qid=aff14ba613f3bb47&t=1344939558699&
http://sclick.baidu.com/w.gif?q=0x00002330000000&fm=se&T=1344931712&y=8FFB8E8&rsrv_sid=&cid=0&qid=e8f2e9614614dff&t=1344931717847&path=http://www.baidu.com/s?wd=诺基亚5233软件
http://sclick.baidu.com/w.gif?q=0x0000100000&fm=as&F=778317EA&F1=9D33F1E4&F2=4CAGDE68&F3=54E5243F&T=1344944565&title=移动电源什么牌子好呀_移动电源哪个牌子好啊_品牌推荐哇哦哟_新建
id=bia566f015427bde&t=1344944612060&path=http://www.baidu.com/s?wd=0x0000100000&
http://sclick.baidu.com/w.gif?q=0x00000000000&fm=se&T=1344935199&y=FE8BFDB8&rsrv_sid=1288_1329_1263_1227_1325_1231_1208_1352_1185_1280_1178_1287_1320_1294_1332&cid=0&qid=be10e0
```

Figure 6.5: Suspicious visits to Chinese web sites

```
http://js.users.51.la/3513006.js
http://www.bicycledebt.info/kernel/511a.htm?Ver=20120814
http://www.bicycledebt.info/kernel/511a.htm?Ver=20120814
```

Figure 6.6: Suspicious URLs

The parameter *Ver* in the request to bicycledebt.info is clearly the date when the malware activated. This indicates that the initial infection might have been occurred a lot before and the activation has been only deferred to a time later in the future. When we create a filtering query using that URL, we see that the first occurrence of this URL is indeed about a month earlier from the activation (20.07.2012). When browsing the events before that we cannot, however, find any interesting events in the near past.

IIS log entries

Trying to figure out other possible attack vectors, we remember seeing IIS (Microsoft Internet Information Services) log entries near the activation date. We can take a look a closer look at those using the following query:

```
filter +"[IIS Log File]"
```

And drawing a timeline out of these events gives us a visualization shown in figure 6.7.

Browsing through the first events shows us only local events of an internal system, but clicking one of the higher bars on the right side of the timeline gives us interesting events showing (in figure 6.8) a brute force attack against the FTP service running under IIS. Notice that log2timelines's IIS plugin has a bug, since the event line is so unclear.

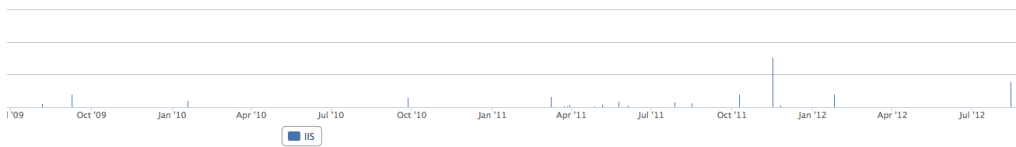


Figure 6.7: IIS server activity (cropped)

```
[2011-11-17 17:23:40] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]USER administrator using -
[2011-11-17 17:23:41] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]PASS - using - status code
[2011-11-17 17:23:42] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]USER administrator using -
[2011-11-17 17:23:43] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]PASS - using - status code
[2011-11-17 17:23:45] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]PASS - using - status code
[2011-11-17 17:23:45] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]USER administrator using -
[2011-11-17 17:23:47] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]USER administrator using -
[2011-11-17 17:23:48] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]PASS - using - status code
[2011-11-17 17:23:49] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]USER administrator using -
[2011-11-17 17:23:50] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]PASS - using - status code
[2011-11-17 17:23:50] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]USER administrator using -
[2011-11-17 17:23:52] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]PASS - using - status code
[2011-11-17 17:23:54] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]USER administrator using -
[2011-11-17 17:23:55] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]PASS - using - status code
[2011-11-17 17:23:56] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]PASS - using - status code
[2011-11-17 17:23:56] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]USER administrator using -
[2011-11-17 17:23:58] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]USER administrator using -
[2011-11-17 17:23:59] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]PASS - using - status code
[2011-11-17 17:24:00] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]USER administrator using -
[2011-11-17 17:24:02] [macb] "[IIS Log File] (Entry written) User: .152.4 .152.4 connect to ':' URI: [1]PASS - using - status code
```

Figure 6.8: Brute force attack against IIS

This output also shows a public IP address, which is suspicious, because the system was not supposed to be connected to public Internet. The following query gives all the IP addresses from IIS log entries and their corresponding physical locations:

```
filter +"[IIS Log File]"
  | select_all default
  | field default
  | extract_ip
  | uniq | geoup | csv
```

This query has a lot of previously unrepresented commands and details specific to the current implementation of the query pipeline, but the basic idea is simple: get all IIS entries, ignore paging, select only a single column from the database, extract unique IP addresses using a regular expression, add IP geolocation information and output in CSV format.

As we can tell from the figure 6.9, there are entries from almost all around the world, 141 unique IP addresses in total. This strongly suggests that the system has indeed been connected to public Internet and leaves IIS as one potential attack vector.



Figure 6.9: IP addresses and their geolocations from IIS log entries (cropped)

6.3 Comparison

Comparing the features of our implementation to the traditional way of working is difficult, because no other product offers the same kind of functionality in a single package. However, we will present some simple benchmarks of similar kind of queries executed using traditional shell tools, which is a common way of analysing this kind of log data. We will query against the CSV file, which we used for importing to our application.

Lets first start with the initial query of our investigations. We translated the query into the following grep pipeline:

```
cat l2tdata.csv | grep -vF "[EXIF metadata]"
                | grep -vF "[SetupApi Log]"
                | grep -F "[2012-"
                > intermediate.csv
```

Runtime of this query was about 1.5 minutes, which is considerably higher than our 2-4 seconds. One of the reasons for this could be the amount of data also written out. The size of *intermediate.csv* was approximately 135 megabytes.

We did not have tools to visualize these results, but we built the following query to calculate the same information needed for visualization:

```
cat intermediate.csv | cut -d "," -f 1-3 | uniq -c > timeline.res
```

This query took 3 seconds to complete, compared to our hundred milliseconds for the whole request.

Finally, we decided to compare the URL extraction query and came up with the following grep pipeline:

```
cat l2tdata.csv | grep -F "[Internet Explorer]"
  | awk -F "," '{ if ($1 >= 2006) print $0 }'
  | egrep -o "https?://\S+"
  > urls.txt
```

Evaluation of this took about 35 seconds, as our tool was able to deliver the same results in a few hundred milliseconds, or a few seconds if we include the time used for the filtering.

6.4 Summary

In this case, the timeline visualization immediately gave us a clear starting point to begin investigations, where we were able to detect plenty of malicious executables brought into the system. We followed the creation of these executables and were able to detect the time when the malware activated.

We then looked more in detail one possible attack vector, the web browser. As a result, we detected signs showing that the initial infection had possibly been deferred in the past, and verified these moments of time.

We then followed another lead, IIS server log entries, and discovered that the server had been publicly accessible and under a brute force attack. This gave us another potential attack vector and many moments of time to investigate further.

This is where we stopped the evaluation, because the following leads started to spread outside the scope of this the evaluation, and customer information would have to be revealed. However, this was already enough for us to be convinced that the flexibility of the tool is sufficient for performing these kinds of investigations. Furthermore, the query performance was more than sufficient for this particular use and allowed responsive and iterative exploration of the dataset, which was not the case using traditional methods. Query times of about a minute are not flexible enough for iterative investigations.

Chapter 7

Conclusions

In the very beginning of this thesis, in chapter 2, we began by introducing the concept of security incidents and their management in organizations. After the introduction, main focus was put on the analysis phase, which was identified as the technically most challenging phase. We then dived deeper into digital forensics, its challenges and the general process needed to execute such investigations. We came to the conclusion that currently the biggest challenge is not managing or collecting the evidence but rather the amount of data available for investigations. Analysis of growing amounts of data is becoming more and more difficult by using traditional methods. Chapter 3 gave more practical information of the investigations.

Having these challenges as a background, chapter 4 then presented techniques and methods to store and query data, compactly represent the data to human analysts and automatically detect of anomalies. In addition, we presented existing tools and evaluated their suitability. Efficient and flexible, but also more analytical querying, was seen as the key functionality for solving the challenges presented. However, as analysing only raw data can be a daunting task, using the visual capabilities of humans to detect different anomalies in visualizations was identified to play an important role in efficient data analysis. Taking this even further, possibility to detect the same anomalies automatically by algorithms could still speed the analysis process.

Chapter 5 then combined the challenges and techniques, and presented the design and implementation of our software, which supports efficient analysis of large amounts of data. It was built around three main ideas: efficient and flexible query pipeline, timeline visualization and inspection of raw query results. In this phase, automatic anomaly detection was left as future work. Furthermore, we implemented a novel idea of caching intermediate sets of rows as bitvectors from queries not employing indexes, and recycling these results also for new superset queries.

Finally, in chapter 6, we evaluated the suitability of our software for investigation of a real incident. We showed that performance of the software as a whole and flexibility of the query pipeline is well suitable for performing such investigations. In addition, the timeline visualization allows very efficient detection of possibly interesting points of time to investigate in detail. We showed query comparisons to traditional shell tools and concluded that our tool greatly outperforms these tools. Still, the dataset used for the evaluation was relatively small and future work will include more testing with considerably larger datasets to research the scalability of our approaches.

As a conclusion, we were able to meet the specified goals for this thesis, which resulted in a concrete working software and a platform for future work, especially considering more automated analysis techniques.

Bibliography

- [1] <http://luca.ntop.org/Fastbit.pdf>.
- [2] ABADI, D. J., MADDEN, S. R., AND HACHEM, N. Column-stores vs. row-stores: how different are they really? In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2008), SIGMOD '08, ACM, pp. 967–980.
- [3] ACCESSDATA. Ftk. <http://www.accessdata.com/products/digital-forensics/ftk>, 2012.
- [4] ALBERTS, C., DOROFEE, A., KILLCRECE, G., RUEFLE, R., AND ZAJICEK, M. Defining incident management processes for CIRTs: A work in progress. <http://www.cert.org/archive/pdf/04tr015.pdf>, 2004.
- [5] ALIENVault. Ossim. <http://communities.alienvault.com/>, 2012.
- [6] APACHE. Lucene. <http://lucene.apache.org/core>, 2012.
- [7] APACHE. Solr. <http://lucene.apache.org/solr>, 2012.
- [8] APACHE SOLR. Field collapsing. <http://lucene.apache.org/solr/FieldCollapsing>, 2012.
- [9] BREIMAN, L., FRIEDMAN, J., STONE, C. J., AND OLSHEN, R. A. *Classification and Regression Trees*, 1 ed. Chapman and Hall/CRC, Jan. 1984.
- [10] BRIAN CARRIER. *File System Forensic Analysis*. Pearson Education Inc, 2008.
- [11] CERT. Security of the internet. http://www.cert.org/encyc_article/tocencyc.html#History, 1997.
- [12] CHOI, H., AND LEE, H. PCAV: Internet Attack Visualization on Parallel Coordinates. 2005, pp. 454–466.

- [13] COOPER, S., KHATIB, F., TREUILLE, A., BARBERO, J., LEE, J., BEENEN, M., LEAVER-FAY, A., BAKER, D., POPOVIC, Z., AND PLAYERS, F. Predicting protein structures with a multiplayer online game. *Nature* 466, 7307 (August 2010), 756–760.
- [14] CYTHON. Cython: C-extensions for python. <http://cython.org>, 2012.
- [15] DAMIR RAJNOVIC. Computer incident response and product security, 2010.
- [16] ELASTICSEARCH. elasticsearch. <http://www.elasticsearch.org>, 2012.
- [17] ELSA. Enterprise log search and archive. <http://code.google.com/p/enterprise-log-search-and-archive/>, 2012.
- [18] ENISA. Cyber incident reporting in the eu. http://www.enisa.europa.eu/activities/Resilience-and-CIIP/Incidents-reporting/cyber-incident-reporting-in-the-eu/at_download/fullReport, 2012.
- [19] EUROPEAN COUNCIL. Proposal for a regulation of the european parliament and of the council on the protection of individuals with regard to the processing of personal data and on the free movement of such data. http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf, 2012.
- [20] FASTBIT. Fastbit. <https://sdm.lbl.gov/fastbit/>.
- [21] FLASK. Flask. <http://flask.pocoo.org>, 2012.
- [22] FOLDIT. Foldit. <http://fold.it/portal/>.
- [23] GRUNDRINZIP. B+-tree image. <http://en.wikipedia.org/wiki/File:Bplustree.png>, 2010.
- [24] GUIDANCE SOFTWARE. Encase. <http://www.guidancesoftware.com/encase-forensic.htm>, 2012.
- [25] HÄTÖNEN, K. *Data mining for telecommunications network log analysis*. Department of Computer Science, series of publications A, report A-2009-1. University of Helsinki, 2009.
- [26] HIGHCHARTS. Highcharts. <http://www.highcharts.com>, 2012.
- [27] HP. Arcsight. <http://www.hpenterprisesecurity.com/products/hp-arcsight-security-intelligence>, 2012.

- [28] IVANOVA, M., KERSTEN, M. L., NES, N. J., AND GONCALVES, R. A. An Architecture For Recycling Intermediates In A Column-Store. *ACM Transactions on Database Systems* 35, 4 (December 2010), 1 – 41.
- [29] JAMES J. THOMAS AND KRISTIN A. COOK. *Illuminating the path: The R&D agenda for visual analytics*, 2005.
- [30] JAVITZ, H., AND VALDES, A. The SRI IDES statistical anomaly detector. In *Research in Security and Privacy, 1991. Proceedings., 1991 IEEE Computer Society Symposium on* (may 1991), pp. 316 –326.
- [31] JEFFREY D. ULLMAN AND JENNIFER WIDOM. *A First Course in Database Systems*. Prentice-Hall International, Inc., 1997.
- [32] JQUERY. jquery. <http://jquery.com>, 2012.
- [33] KHATIB, F., DiMAIO, F., COOPER, S., KAZMIERCZYK, M., GILSKI, M., KRZYWDA, S., ZABRANSKA, H., PICOVA, I., THOMPSON, J., POPOVIĆ, Z., JASKOLSKI, M., AND BAKER, D. Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature on Structural and Molecular Biology* 18, 10 (10 2011), 1175–1177.
- [34] KOKORINA, L. *The use of frequent episodes in intrusion detection*. 2009.
- [35] KRUEGEL, C., AND VIGNA, G. Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM conference on Computer and communications security* (New York, NY, USA, 2003), CCS '03, ACM, pp. 251–261.
- [36] KRÜGEL, C., TOTH, T., AND KIRDA, E. Service specific anomaly detection for network intrusion detection. In *Proceedings of the 2002 ACM symposium on Applied computing* (New York, NY, USA, 2002), SAC '02, ACM, pp. 201–208.
- [37] L2T REVIEW. L2t review. http://code.google.com/p/l2t-tools/wiki/l2t_Review, 2012.
- [38] LEE, W., AND STOLFO, S. J. A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inf. Syst. Secur.* 3, 4 (Nov. 2000), 227–261.
- [39] LOG2TIMELINE. log2timeline. <http://log2timeline.net>, 2012.
- [40] MANDIANT. Highlighter. <http://www.mandiant.com/resources/download/highlighter>, 2012.

- [41] MICROSOFT. Log parser. www.microsoft.com/en-us/download/details.aspx?id=24659, 2012.
- [42] MONETDB. Monetdb. <http://www.monetdb.org>, 2012.
- [43] MONETDB. Monetdb. <http://www.infinidb.org>, 2012.
- [44] MUELDER, C., MA, K.-L., AND BARTOLETTI, T. Interactive visualization for network and port scan detection. In *Proceedings of the 8th international conference on Recent Advances in Intrusion Detection* (Berlin, Heidelberg, 2006), RAID'05, Springer-Verlag, pp. 265–283.
- [45] NATIONAL INSTITUTE OF JUSTICE. Digital evidence and forensics. <http://nij.gov/topics/forensics/evidence/digital/welcome.htm>, 2010.
- [46] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Guide to integrating forensic techniques into incident response. <http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf>, 2006.
- [47] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Cryptographic hash project. <http://csrc.nist.gov/groups/ST/hash/index.html>, 2012.
- [48] NEIRA JONES. Incident management and risk management go hand in hand. <http://www.infosecisland.com/blogview/20013-Incident-Response-and-Risk-Management-Go-Hand-in-Hand.html>, 2012.
- [49] OFFICE OF GOVERNMENT COMMERCE. *Introduction to ITIL*. TSO, 2005.
- [50] O'NEIL, P., AND QUASS, D. Improved query performance with variant indexes. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 1997), SIGMOD '97, ACM, pp. 38–49.
- [51] RAFFAEL MARTY. *Applied Security Visualization*. Addison-Wesley, 2010.
- [52] RAMEZ A. ELMASRI AND SHAMKANT B. NAVATHE. *Fundamentals of Database Systems, second edition*. The Benjamin/Cummings Publishing Company, Inc., 1994.

- [53] REBECCA GURLEY BACE. *Intrusion Detection*. Macmillan Technical Publishing, 2000.
- [54] RICHARD BEJTLICH. Incident response and forensics: The basics. <http://www.csoonline.com/article/205960/incident-detection-response-and-forensics-the-basics>, 2008.
- [55] SPLUNK INC. Splunk. <http://www.splunk.com>, 2012.
- [56] STOCKINGER, K., BETHEL, E. W., CAMPBELL, S., DART, E., AND WU, K. Detecting distributed scans using high-performance query-driven visualization. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing* (New York, NY, USA, 2006), SC '06, ACM.
- [57] SUNG, A., AND MUKKAMALA, S. Identifying important features for intrusion detection using support vector machines and neural networks. In *Applications and the Internet, 2003. Proceedings. 2003 Symposium on* (jan. 2003), pp. 209 – 216.
- [58] THE SANS INSTITUTE. Incident handling step-by-step and computer crime investigation, 2007.
- [59] THE VOLATILITY FRAMEWORK. The volatility framework. <https://www.volatilesystems.com/default/volatility>, 2012.
- [60] TIMOTHY E. WRIGHT. A method for forensic previews. <http://www.symantec.com/connect/articles/method-forensic-previews>, 2010.
- [61] TWITTER. Bootstrap. <http://twitter.github.com/bootstrap>, 2012.
- [62] VERIZON. Data breach investigations report. , 2012.
- [63] VIVEK SHARMA. Bitmap index vs. b-tree index: Which and when? <http://www.oracle.com/technetwork/articles/sharma-indexes-093638.html>, 2005.
- [64] WU, K., OTOO, E. J., AND SHOSHANI, A. Optimizing bitmap indices with efficient compression. *ACM Trans. Database Syst.* 31, 1 (Mar. 2006), 1–38.
- [65] WU, K., SHOSHANI, A., AND STOCKINGER, K. Analyses of multi-level and multi-component compressed bitmap indexes. *ACM Trans. Database Syst.* 35, 1 (Feb. 2008), 2:1–2:52.

- [66] ZHU, C., ZHU, Q., AND ZUZARTE, C. Efficient processing of monotonic linear progressive queries via dynamic materialized views. In *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research* (Riverton, NJ, USA, 2010), CASCON '10, IBM Corp., pp. 224–237.
- [67] ZHU, Q., MEDJAHED, B., SHARMA, A., AND HUANG, H. The collective index: A technique for efficient processing of progressive queries. *Comput. J.* 51, 6 (2008), 662–676.