Prioritizing Computer Forensics Using Triage Techniques

Author:
Matthijs Gielen

Supervisor:
Dr. Damiano Bolzoni

July 2014

UNIVERSITY OF TWENTE

Abstract

Faculty of Electrical Engineering, Mathematics and Computer Science

Prioritizing Computer Forensics Using Triage Techniques

by Matthijs Gielen

There is a lot of information contained on a single computer and a company can contain a lot of computer and other devices. If there is a breach somewhere in this organization how will a forensic analyst find the source and extend of the breach? Investigating all of the computer is not doable, there are simply too much computers and information. One of the solutions to this problem is the use of forensic triage. This research combines a couple of forensic triage methods and uses these techniques to classify computers into either malicious or clean. This method was tested on two datasets, a generated set and a set containing computers from real companies. The first dataset was reduced by 50% where the remaining computers were all infected. The second dataset was reduced by 79%, the result included all of the malicious computers. Thus this method can be used successful to reduce the workload of forensic analysts.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem statement

DigiNotar was a Trusted Third Party that maintained several Certificate Authorities (CA). DigiNotar issued certificates for, amongst others, various Dutch governmental applications. In June 2011 the company was breached by an outside attacker who managed to issue his own (rogue) certificates. These certificates were then abused in a man-in-the-middle attack on Google users. One of the rogue certificates was discovered by an user who used the Google Chrome browser. This browser has an extra safety precaution: Chrome checks if a certificate for the *.google.com domain was issued by the right CA. In the case of the rogue DigiNotar certificate the browser determined that it could not be trusted. After the breach went public DigiNotar asked Fox-It to investigate the incident. It was already known that some part of DigiNotar was breached and at least one of the CA's was abused but it was not yet known where the breach of DigiNotar originated from. Because some of the CA's maintained by DigiNotar issued certificates for the Dutch government, it was important to know for the Dutch government if all of the CA's DigiNotar maintained were breached: it could leave the Dutch governments applications at risk. However DigiNotar had a lot of computers and the manpower available to identify the computers was limited[66].

The main focus of the research is analyzing if it is possible to automatically determine which computer is most likely to be compromised. This is an important step for computer forensics analysts to make sure the manpower and time can be directed onto the identified systems and less time is wasted on investigating computers that may not even be compromised. The type of detection is anomaly based and a couple of classification models will be used. For a classification model both a clean and an infected baseline

must be established from a couple of features of a system. These features should be representative of the whole system and must give a good indication whether a system is infected. The classification model will compare the features from the baseline to the same features from a system that might have been breached. Each model will predict whether certain features are that of a malicious or a clean system. The output of these models are used to determine whether a computer is likely to be infected or more likely to be clean. The machines that are more likely to be infected are to be investigated by a forensic analyst. The time that can possibly be saved by using this method will be investigated. The smaller the remaining set is the more time of the analysts will be saved. However the set should include the actual infected machines otherwise not a lot of time will be saved. Thus it is important to know the false negative rate (FN) and to some extend the false positive (FP) rate. The false positive rate can be used to see if more time could possibly be saved. Thus a lower false positive rate means less computers have to be investigated and will result in a lower amount of time required to analyze the entire set. The false negative rate will show whether all infected computers are found using this method.

The problem statement is thus:
How can we automatically and quickly determine which computer systems within a set are most likely to be compromised?

Chapter 2

Related work

2.1 Computer forensics

2.1.1 What is computer forensics?

In the 1980s the term computer forensics was used to describe the examination of computers for uncovering digital evidence[55]. These computers were most of the time stand alone computers without network capability. Over time computers became more and more networked and the term computer forensics also included networks and servers within the definition[59]. Around the year 1984 the FBI started to develop programs to uncover evidence from computer systems [60].

Nowadays the definition can stretch from uncovering evidence within computers to network analysis and mobile devices like mobile phones and tablets. Because of the wide spread use of computers and embedded devices more and more digital evidence can be found. For example if the police finds a phone at a crime scene it can be checked for evidence of who the owner is. Phone records from cell towers can show (regionally) where a person, or at least his phone, is located during a certain time this can in turn be used to confirm or debunk someone's alibi or give an indication if someone can be seen as a suspect.

One of the goals of computer forensics is to provide evidence of crimes that is valid in a court of law. Therefore it is important that the investigator documents the process of uncovering evidence and preferable follows a certain protocol.[60, 41]

According to Yussuf et al. [60] the common phases of evidence collection are as follows:

1. Pre-Process: these are the tasks that needs to be performed before the actual investigation
2. Acquisition and Preservation: the tasks that are related to the acquisition, preservation, transportation and storing of data.
3. Analysis: analysis is the centre of the process of investigating the data. This contains the tasks that are related to obtaining evidence.
4. Presentation: these are the tasks that need to be done in order to present the evidence to the authority.
5. Post-Process: the tasks that are related to properly closing the investigation and returning the evidence to the owner.

They also suggest that it is not only possible to transition from a state to the next but it should also be possible to move back through the states.
There are some other terms used to describe computer forensics, these include digital forensics, cyber forensics, network and computer forensics and others. In this report the term computer forensics will be used.
There can be another use of computer forensics namely that it could be employed by a company to detect the scope of the damage done and to make sure the system will return to the normal mode of operation. In this situation documenting and keeping the evidence intact is not as important as fixing the current state of the system. When computer forensics is employed in such a way it is not as important not to make any traces yourself.

2.1.2 Challenges

Most of the challenges relate to the vast increase of storage available on a single computer. Nowadays hard drives with over 4 TB of storage are available whereas the speed at which the tools can process the data did not scale with the storage: it can take days for a forensic tool to process all of the data available[47]. Then there is another problem of scale: some of the tools that could give results against small data sets (n <100) will not give valid results when run against large data sets (n >10,000)[18]. This corresponds to the problem described in the problem statement: there is too much data available to have it all analyzed manually. One of the ways to solve or at least reduce the problem of an overflow of data is to use forensic triage.

2.1.3 Forensic Triage

Within medical care triage is commonly used to prioritize patients for care. When applied to computer forensics it becomes: selecting what evidence or which systems should be prioritized for the forensic analyst[48]. Triage is usually not included within the forensic models, in the models where it is included it is a stage before the gathering and analysis of evidence[8].
There are some practical examples of forensic triage: Kim et al[32] looked at a couple of indicators of malware and hacking attempts: Timeline analysis of system files, DNS Analysis, Phishing and Pharming Inspection, Correlation Analysis between Network, the Connection and Processes and ARP Analysis and whether these features can be used to see if a system is infected. This research can be seen as an example of how the problem can be addressed: They used some rules and signatures to determine whether the user station was within the expected values. However not everything they use is viable: changes to the host.txt file to redirect to another site is not done as frequently anymore. There are ways to achieve the same goal which are less detectable than to change the host.txt, however checking the host.txt for changes is an easy thing to do and still may be worth checking. The timeline analysis of system files will generate a lot of false positives: during the normal operation of the operating systems, anomalies (wrong combinations of timestamps) are generated. Thus this method will also find these false positives. Some of the other features could be looked at whether they prove viable.
Berte et al.[6] describe a way to execute postmortem forensic triage with regards to the computer system of a suspect. The focus of the research is to triage computers based on the likelihood that they were used in illegal activity. They suggest that someone who does triage on the computer looks at a couple of indicators within in a computer including the installed software, browser history and system event logs. While this research focuses on the triage of systems used by attackers the same method, with perhaps different features, could be used to determine whether a system is a victim of an attacker.
Marturana et al.[40] does a similar study into this. They use the same model as Berte et al. [6] and they use similar features like the installed programs, specifically file sharing programs, browser history but also the number of specific (.pdf, .iso, .divx) files on the system. They use a couple of machine learning algorithms and 10 folds cross-validation to see how well it performed. The classifiers managed to get up to 99% accuracy in determining whether a system was used to commit copyright infringement.
There are some tools with forensic triage in mind.
bulk_extractor [19], is a tool that uses bulk data analysis to find useful information on a hard drive. Bulk data analysis is different from file-base approaches that it does not use the file metadata to indicate what and where a file exists. The advantage of this approach is that bulk_extractor can find information that is not contained within a file

and it can handle information that is partly overwritten. The disadvantage is that it will take a long time to process an entire disc (1 to 8h depending on size). This is too long for the problem at hand.
Spektor forensics triage tools uses a remote connection to get assistance from additional analysts, however this is not exactly triage because it just uses more manpower to solve the problem. This tool cannot be used in the context of automatically determine which system is most likely to be infected. Another tool called ADFs Triage products just provides a list of important files based upon how much these files are accessed. This approach can just give some standard files that are accessed a lot and not provide any useful information.
There is another approach to triage by a tool called CBR, this tool uses a list of known cases and tries to map the current case to any of the known cases to get the significant data.[28]

These tools look and prioritize information on a single host. The problem of finding an infected computer within a large set of computer may require a different approach: there is too much information available on each system and scanning the entire system will take too much time. Focus must lie on having fast but representative techniques and features of a system that can indicate whether the system is interesting for manual inspection.

A last tool that could be used for this purpose is called Redline from Mandiant [68]. Redline has the following features:

- Timeline, a feature that gives a list of events sorted by time.
- Malware Risk Index (MRI) Score, gives an indication of how likely it is that a process is involved in a potential compromise.
- Indicators of Compromise (IOCs), this can be used to determine the indicators to identify the malicious activity.
- Whitelists, this can be used to whitelist files known to be valid.

The timeline feature will give a lot of entries, even though the Timeline feature can be filtered by process and other values. It may not be a good candidate for the problem at hand because it is unknown if the host is infected. The Malware Risk Index on the other hand may be a good indication per system whether there is something malicious on it: a high score can indicate a malicious process. The indicators of compromise feature is not as useful because first the system that is infected has to be determined. Redline is made to be deployed on a live system, instead from memory files. This can be a advantage

in the situation where the system is in use, in a system that is powered off it can be a disadvantage.
See Table 2.1 for a comparison of triage and anti-forensic techniques.

2.1.4 Work flow

This next section is about the work flow at an incident. What should the forensic analyst gather and which information should be prioritized. Section 2.1.4.1 is about the NIST guidelines. Section 2.1.4.2 lists two ways to approach the gathering of data.

2.1.4.1 NIST Guidelines

There are some guidelines for computer forensics with regards to the problem statement. The NIST guide to integrating forensic techniques into incident response [29] states that there are three steps to acquiring data.

1. Develop a plan to acquire the data
2. Acquire the data
3. Verify the integrity of the data

The first step is part of the forensic triage. Here the information is prioritized in which order the information should be acquired. There are some factors that should be included in the prioritization:

- Likely value of data source
- Volatility of data source
- Amount of Effort Required to obtain data source.

Applied to the situation of a lot of computers with only a few infections the analyst should prioritize data sources that have a high likely value, a lot of volatility and the least amount of effort required. To just load in all of the computers and take them to a forensic lab would not be the best way because it would require a lot of effort and some volatile information is lost in the process. However within the guidelines there are no other suggestions of how to tackle this problem other than to prioritize the data sources with regard to these aspects.

2.1.4.2 Data acquisition

Gómez [21] gives two mayor categories of collecting data.

1. Seize everything using staff with limited forensic training
2. Selective acquisition by forensic experts

The first strategy has a couple of advantages and disadvantages:

+ Limited forensic training
− Staff can damage digital evidence
− Every item has to be examined

The second strategy has the following advantages and disadvantages:

+ Less items to examine in the lab
− Important evidence can be overlooked

Both strategies have situations in which they are best, however none of these strategies reduce the actual workload for the forensic experts. The rest of the report of [21] gives a strategy on how to reduce the backlog of cases. This is less relevant because the problem is about the acquisition of data and not the actual analysis within the lab.

2.1.5 Forensic Analysis

The next section is about techniques and places in a computer that can be used to search for evidence of an intrusion of a system. Some of these techniques are already used in forensic triage, others may be good candidates for determining whether a computer is infected. The following paragraph is about the general forensic tools.

2.1.5.1 General forensic tools

There are some general forensic tools: FTK[63], Encase[71] and The Sleuth Kit[9]. FTK and Encase are commercial tools and The Sleuth Kit is an open source variant. These tools are all, to some extent, able to do the following[38]:

- Integrity checking (hashing)
- Finding & recovering deleted files
- Finding encrypted files
- Identify file extension mismatches
- Search through the evidence
- Finding specific information (cookies, URLs)

Some of the functions of the tools have can be interesting for solving the problem: finding encrypted files and file extension mismatches, encrypted files can indicate that someone wants to hide information and extension mismatches can also indicate strange behaviour. These tools however mostly need to process an entire hard drive which can take a long time. Encase has the ability to do a remote analysis of a (live) system. The drawbacks of Encase and FTK is that they are quite expensive. Most of these tools require an image of the memory and hard drive to analyze. This is usually not a good way to do forensic triage, because creating images of the memory and hard drive can take a long time and a lot of memory. Usually the triage has to be done before the data is loaded into these general tools.

2.1.5.2 Memory analysis

One of the executions of memory analysis is the use of a cold boot attack. This is a technique in which an attacker or forensic analyst lowers the temperature of the memory and uses a device to read out the current memory. This technique works because the state of the physical memory is not immediately lost when the power is turned off. The memory is cooled because it increases the duration at which the data in the memory is still readable. Other ways of obtaining the contents of physical memory are: dumping the contents of the memory to disc and live memory analysis. This requires access and/or credentials to the system.
Davidoff[12] shows that someone can find plain text passwords in the memory. She did this by dumping the memory of a machine. Searching this memory for the known passwords and determines whether this place is constant or whether there are signatures or indicators of this password in the memory. Finding passwords itself is not as interesting, however the same technique can be used to find other information like encryption keys. Halderman et al.[22] shows the decay of memory and manages to reconstruct encryption keys from partially decayed memory. This technique can be useful to see whether a process in memory has an encryption key. This encryption key can has some benign effect namely encrypting communication between hosts. But a process that has an encryption key but does not have any connections can be of value for a forensic analyst.

Not only passwords and encryption keys can be retrieved from memory, process structures and other file systems that are contained within memory can be retrieved.
Volatility[73] is a tool that can analyze the memory of a system. It requires a memory dump of the system and can look through structures and lists them for the user. Volatility can be an useful tool to look in a system for a rootkit or other malware that tries to hide. Currently the disadvantage is that it requires that the memory of the computer being investigated is acquired which can take some time and space. There may be ways to circumvent this limitation.
Redline[68] also has some capabilities of analyzing live memory. It can look for suspicious behavior in running processes, for example a process which uses unsigned drivers.
Freiling et al.[16] gives an overview of the available rootkit detector tools. The research shows that a combination of three different tools (Blacklight, IceSword, System Virginity Verifier (SVV)) will give the most optimal result. These three tools work all in a slightly different way: Blacklight looks for hidden objects in the running operating system by cross viewing it: the tool compares the responses of the high level APIs to the data gained from a lower level. If these two sets of data do not match there is an anomaly. IceSword is a tool that gives the ability to look for rootkits in a more interactive way. Finally SVV checks the integrity of critical operating system files, the tool does this by comparing the file in memory to the equivalent file on the hard drive. If an anomaly (difference) is detected, a rootkit is considered to be on the system. Of these three tools suggested only IceSword seems to be less effective in our problem: It requires interaction from a forensic analyst to function optimally. The other two tools may provide an addition to detect malicious activity on a computer system.
Arnold gives a more recent comparative analysis in [4], one of the things he states is that SVV, blacklight and Icesword are no longer updated. Thus these tools are probably not good candidates for rootkit detection. In [4] he investigates four rootkits: Rustock, TDL3, Black Energy and Zeus and he tests how good each of the rootkit detectors performs. A couple of other approaches are tested as well: the detection of hidden ports by using nmap and netstat, checking the system performance and the registry keys. By comparing the output of netstat and the output of a nmap scan he is able to determine whether there is a hidden port on the system. This can be a good approach for detecting suspicious behaviour, however it is not within the scope of our research. The others rely upon first building a baseline on the clean system and then infecting it with a rootkit and checking the same information again. It could be tested whether this approach can work with a predetermined baseline, however the normal user behaviour can vary a lot and the system performance and registry keys can in turn vary a lot in between systems.
Malware Bytes Anti-Malware and Combofix have the highest ranking in the overall score of rootkit detection. These programs can be used to detect rootkits on systems and may provide valuable information of whether the computer is infected or not.

Blacksheep[7] is a tool that uses the homogeneous property of computer systems within a company. They assume that the different computers within a company share a lot of the same properties. The hardware and software within a company are probably similar or identical. They use this property to look for hooks within this group of computer systems by comparing the memory dumps of these systems. The comparing of the systems is done by clustering memory images together on the bases of similarity. The most similar memory are placed together and some clusters will form. The biggest cluster of memory images is then assumed to be the clean dumps, the smaller clustered could be infected. They assume that only a small factor of machines is really infected and the infected machines are outliers to the 'crowd' of computers. Using this technique they manage to identify the memory images that contain the hooks.
Liang et al.[35] use a method called fine-grained impact analysis to identify hooking mechanisms. The technique works by analyzing all of the changes in the control flow of the system while running an unknown malicious executable. When a change in the system changes the control flow into the malicious code of the executable then a hooking mechanism is assumed. This approach is suitable for identifying hooking mechanisms and not so much for identifying whether a system is hooked by a malicious executable: at the time of execution it is unknown if there is a malicious executable present.
The use of memory analysis is a valuable tool, an analyst can detect malware hiding in memory or can find other signs of malware and rootkits. There are some disadvantages as well: if the malware is not currently active few traces can exist in memory and there can be a lot more traces at other places like the hard drives. Thus it is important to use memory analysis in combination with other techniques.

2.1.5.3 Executable analysis

The use of packers and encryption can cause an executable to have a higher entropy than 'normal' executables. These kind of techniques are almost exclusively used by malware and other programs that wants to obfuscate the real purpose of the program. About 80% of malware uses some kind of packer and about 50% of new malware is known malware repacked with another packer. One can measure the average or maximum entropy of all the executables on the system to determine if an anomalous program is present. One of the advantages of the use of file entropy is that a large percentage of malware can be identified, one of the disadvantages is that malware executables can look like a non executable file and can be missed if an analyst only checks the entropy of executables[45, 2, 37].
Lyda et al.[37] used the max and average entropy of PE files to determine whether they were packed or encrypted. Perdisci et al. [45] extended this approach by using

Number of sections (standard, non-standard etc.), Entropy of headers, sections and entire PE file to detect packed executables. As detection algorithms they use Naive Bayes, J48, Bagged-J48, iBk, MLP and an entropy threshold. They are able to detect a high percentage of executables which are packed: respectively 98.42%, 99.57%, 99.59%, 99.43%, 99.42% and 96.57%. The Entropy Threshold algorithms is just a fixed value which is compared to the entropy values of the executable file. Santos et al [51] used header and section characteristics, entropy values and the header block to detect the use of packers. Combined with the CollectiveIBK, CollectiveForest, CollectiveWoods and RandomWoods machine learning algorithms they are able to detect up to 99.8% of executables that use a packer. One of the disadvantages of these approaches is that it will also detect the use of packers by benign executables. So some false positives can be expected and may be of influence whether this approach will work in the context of detecting infected computers.
Another approach is used by Khan et al[31]: They use easy to extract features from PE files to determine whether a file is malicious or benign. These features include things like version number, number of imports and number of sections. They used 42 features and gotten a false positive rate of 6.7% and a detection rate of 78.1%. Raman [46] improved on this research: he used combination of 7 features which are a subset of the 42 features used by Khan et al. The research shows that while less features are used the detection and false positive rate are improved: 5.68% false positive rate with a detection rate of 98.56% . This research can be a valuable addition to packer detection: some malicious executables are not packed (around 20% of them) and this approach may be able to detect these executables.

2.1.5.4 Log Files

Log files can contain a wealth of information, the log files can contain file changes, file access and what kind of system functions were called. However these log files can become huge in size: a single log file can contain over a million entries. This can hinder a computer forensics analyst to find the relevant data within the log.[52, 3]
Abad et al.[1] argues that the combination the contents of multiple log files improves the detection rate. This because the intrusion of malware or an attacker will not only show up in one log but possible in multiple logs. Through the use of a data mining algorithm (RIPPER) and this correlation of logs they show that the correlation improves the detection of Intrusion Detection Systems (IDS). This approach is not as useful for our problem statement because the correlation is used for input into IDS instead of a detection of infected computers. It may be worth investigating whether the IDS can be

used in a forensic setting by using a baseline that is made beforehand.
Stallard et al.[52] use the java based tool JESS. As data input they also use a combination of logs, they use the log in times and modification times to check whether some file has been changed without the owner being logged in (this is a logical contradiction). This approach has as an advantage that nothing has to be done beforehand; the normal generated log files can be used. The disadvantage is that when the attacker does not access or change any file (for example a keylogger) this approach will not find any suspicious activity.
Al-Hammadi et al.[24] use log correlation as well. They employ log correlation to detect Botnets, however their implementation requires a pre loaded .dll file to log all system calls from all threads. As a result their implementation can not be used in a forensics setting.
Similar to the approach of Stallard et al.[52], CatDetect[39] use a combination of log files, timestamps and some other sources of information to build a model of the system that can be queried for inconsistencies, some of the examples they give is that user A first created a file, then logged in. This is not the logical way of file creation, the creator should first have been logged in. Thus this event is shown as an anomaly. There are some limitations to their approach, the user session start and end can not (yet) be detected automatically, this means that the investigator should have some information about the system. It could be investigated whether this approach works to detect malicious activity on a live system.
Splunk[67] is a data mining tool. It can be used for a range of things. One of the more interesting features is that it can mine information log files and extract anomalies from these logs. The drawback is that it requires micromanagement because it creates a overview of events and actions done but an user should look whether an event is relevant or not.

There are a lot of other log file forensic tools, most of them are to be used in a forensic context. One can use them to analyze a specific event or action done by the user.

2.1.5.5 File system

The file system of an OS can give information of what files can be important and which files should be looked at. The temp map of most computer OSes are used by programs to temporarily store information that the program needs. Someone can use this directory to hide information or to store executables because users will normally not look at this directory. There are many other ways to use the OS file system to look for anomalies.
Dirim[50] is a tool that analyzes the file system and uses this information to look for suspicious files within this file system. The tool can look for suspicious file extensions,

analyze files to see whether they are encrypted, look for suspicious paths, deliberate hash collisions, clusters of deletion and atypical drive averages. This tool uses Sleuthkit for prepossessing. The disadvantages of this tool is the long training phase and when the attacker does not leave any traces on the file system or when they manage to hide their traces by mimicking normal user behaviour this tool will not detect them. The advantages are short time to compare new drive and the ability to detect suspicious files on a hard drive.

2.1.5.6 Differential Analysis

A quite different approach in computer forensics is the use of differential analysis. This technique is checking what the differences are between two different (digital) objects. For example this is the process of comparing hard drives (or images of hard drives) from before and after a breach to determine which files were changed by the attacker. This process can be very useful for the impact of certain malware to a system[20]. Differential analysis is not only limited to hard drives but it can also be used to compare network traffic to determine if someone has send the traffic, determine if someone has the same file as the original or determine if some program has been altered by an attacker. One of the non forensic analysis application is the use of these systems in cloud sharing programs[20], for example Dropbox uses such a feature to determine which file an user has changed. The drawback of this technique is that you would need something to compare it with. Every computer can deviate a lot from a normal installation due to the different programs someone can install or different uses of their computer system. One of the ways to mitigate this problem is to use an image of the system the business uses on normal computers. The difference between this image and the real computer can be calculated. This however does not mean that the most anomalous system is the most malicious: the user of the system can require different (anomalous) applications for his or her normal work. Another way is to use the same technique as used by the tool Blacksheep[7]: to correlate the results between the computer systems instead of to calculate the difference between each system and the image. One could expect that each computer deviate somewhat but overall they should be quite the same. This can be used to determine which computer is the most anomalous within the company.

2.1.5.7 Summary

All of the tool and techniques described above are focused on a part of the information available on a system. When an attacker has malware in memory the tools that look at the hard drive will not find it. It can be reversed as well, when malware resides on the

disc and is not active, the memory tools will not find evidence of the malware. These tools are generally fast to search through their data sets but can only look at the small part of the picture. The tools that look at all the information available are generally slower because of the large amount of information.

2.1.6 Anti-Forensics

A last interesting topic within computer forensics is the use of anti-forensic measures by attackers. Anti forensics is the practice of hindering a forensic analyst in his attempts to uncover evidence from a digital device. The literature identifies the following anti forensic techniques[30, 42]:

Anti-Forensic technique:	Section:
Hiding the data	2.1.6.1
Prevention of data generation	2.1.6.2
Destroying data	2.1.6.3
Direct attacks against computer forensic software	2.1.6.4
Counterfeiting evidence	2.1.6.5

2.1.6.1 Data Hiding

This technique relies on the fact that there is a lot of unimportant information on the disk. An attacker can hide information within other files: Compressing JPEG files in such a way that information can be hidden[53, 54]. This practice is called steganography. Another approach is to hide data in the slack space of a drive: some tools do not check this space because it is usually not used[42]. Encrypting data is another way to hide the data: even if an analyst discovers the hidden container, a large enough key will stop the analyst from reading information from it[56, 36]. Data hiding can also be applied on malware or on the programs the attacker uses, this is the use of packers.

2.1.6.2 Prevention of data generation

This technique includes things like disabling logging systems, wiping log files and altering creation dates.[42, 30, 27]. One of the things that can be done to detect these kinds of anti forensics is to look at the reporting systems and see if they are disabled. In case these systems are disabled this can be seen as an anomaly.

2.1.6.3 Data Destruction

This technique is simply wiping or destroying hard drives that have evidence on them[27]. Wiping hard drives means that everything is overwritten at least once. Because of the density of modern hard drives no previous information can be read from the drive. SSD drives are somewhat different, clusters within these drives have to be set back before new data can be written to these clusters. In practice this means that a garbage collector is present on these drives. Every file that is marked as deleted will be reset to the default state by this garbage collector. There are some limitations to this garbage collecting, the OS system must support his behaviour and should send a "trim" command to the drive to start the garbage collecting. The wiping and garbage collection of the drives means that no information can be retrieved[5]. If evidence of this wiping is found then this may be an indication that someone wanted to hide something. This evidence does not always mean any malicious intent, thus it may not be a good indicator for the research problem.

2.1.6.4 Direct attacks against computer forensic software

There are some ways to attack the computer forensics itself: One way an attacker can use to hide for forensic software is to employ techniques that work against the forensic tools that are used. One of the examples of this is a ZIP bomb attack. This is a zip file that contains 16 other files, those contain another 16 files and so on, if a forensic tool tries to unpack this zip file it requires about 4TB of storage. Other ways are undermining the data collection process or undermining the way the analyst is preserving data[17, 56].

2.1.6.5 Counterfeiting Evidence

This is the process of creating fake evidence of intrusions to put forensic techniques on a wrong trail, this can include techniques like making fake log files or putting in evidence that points at another type of exploit.

2.1.6.6 Summary

The use of anti-forensics may prove to be a viable indicator to see if some computer is compromised: only attackers will use these techniques whereas normal applications will not. Thus when evidence of anti-forensics can be found, it can be a clear indication that the system is infected and / or compromised by an attacker[50]. See Table 2.1 for an comparison of triage and anti-forensic techniques. In the table the type means

Authors/Name	Type	What?	Where?
Kim et al.[32]	Signature	Specific indicators	File system, network
Berte et al.[6]	Signature	Programs, browser history	File system
Marturana et al.[40]	Signature	Programs, number of certain files	File system
Bulk_extractor[19]	-	Extraction of files	File system
Spektor[28]	Human	Provide assistance of additional analysts	Entire system
ADF triage[28]	Signature	Most accessed files	File system
CBR[28]	Signature	Map current case to known case	Entire system
Redline[68]	Anomaly	Scan for malicious processes, hooks	Memory
Data hiding[56, 36]	Anti-Signature	Hiding data in unused places, encryption	File system
Prevention data generation[42, 30, 27]	Anti-Analysis	Disabling logging activities	Registry
Data destruction[5]	Anti-Analysis	Wiping harddrives	File system
Anti-forensic software[17, 56]	Anti-Signature	Attacking the forensic tools	File system, memory
Counterfeiting Evidence	Anti-Analysis	Creating false evidence	File system, memory

TABLE 2.1: Comparison triage and anti-forensic techniques

what kind of detection or anti detection mechanisms the methods use. Signature based detection uses the features or structure of the to be analyzed files or indicators and checks them against a database of known bad signatures. One of the other types of detection is Anomaly: Anomaly based detection looks at what normal behaviour is and in some cases what malicious behaviour looks like and tries to map current behaviour to either one of these. Anti-Signature is the behaviour of attackers to hide their traces or for executables to change themselves in a way that the signature will no longer work. Anti-Analysis is to make sure the traces can not be analyzed: this can be done by for example wiping certain files or by encrypting thing to make sure it is unreadable. The "What?" column describes what the analyst is looking for or how the attacker avoids detection. Lastly the "Where?" column describes where within the computer the traces can be found.

2.2 Research questions

The research questions are as follows:

- Which features can be used to baseline and identify a possible breach in a computer?
- To what extend can these features be used to reduce the number of computers that have to be investigated in a large data set?

The questions contain all of the problems described above: feature selection and if these features can reduce the number of computers that have to be investigated. One last question can be derived from the problem statement: whether it is feasible to reduce the workload of forensic analysts using this method. When this system will run for a couple of hours to determine if the system is breached it might be faster to just manually inspect the system. However when the tool takes a couple of seconds to minutes this

system can effectively be used in a company wide search.
Additional question:

- How feasible is such a system to be used in a company wide search?

Chapter 3

Method

The Chapter describes the method used to answer the research questions. The first Section (3.1) is about how long a forensic analyst will take to investigate a single computer. This can be used to say something about how much time will be saved. Section 3.2 explains why anomaly detection is used instead of for example signature based detection. The testing method is described in Section 3.3. Section 3.4 is about certain Windows specific features that can be used to determine if a computer is infected. The selected features are given in Section 3.5. Section 3.6 describes how the models are created and how the best suiting machine learning algorithm is selected. Lastly the testing procedure is described in Section 3.7.

3.1 Human analysis

Two global types of analysis can be done by forensic analysts: quick scan and full review. The first type is when an analyst looks at a couple of features in a computer to quickly try to determine whether the computer is interesting or not. When doing a full review the analyst will look at more features of a computer. The number of features looked at dictates the time it will take to complete the full review. Seven analysts were asked to make an estimation of the duration of a quick scan and of the full review. The average estimate of the quick scan is 4 hours of time or about half a workday. The full review type of analysis includes checking the things of the quick scan and also checking the log files and other traces of the computer. This type of analysis takes a lot longer and the average estimate of this method is 24.5 hours or about 3 workdays.

3.2 Type of detection

Generally speaking there are two types of detection, anomaly and signature based. Signature uses a database of known bad features and compares any found features to this database. If there is a match with the database then a malicious file is found. Anomaly based detection uses known good features and, in some cases, known bad features to classify a new file into one of these categories. Signature based detection usually has very few false positives but cannot detect any malicious features that are not in the database. Anomaly based detection has more false positives but is able to detect any new malicious features[13, 43, 44]. In this method anomaly based detection is used because not only known features of malware must be detection but also new malware. A company can already have a anti-virus or intrusion detection system that has the latest signatures and if those systems could not detect the malware then this system is also not able to detect it. Some of the features that may be used do not have a signature database because it is not yet widely used for detecting malware, thus setting up a database with all of the current malware will take too much time. Therefore a set of known good and bad features will be used to see whether other (new) features can be reliably classified as either infected or clean.

3.3 Testing method

In the execution of this method we assume two things. First we assume that computers from a the same company use more or less the same hardware and software and the secondly not all of the computers within the company are infected. The second assumption is no problem within our problem statement: if every computer is infected, they all have to be investigated. Thus this does not interfere in this method. We can use this assumption to filter out some false positives from different computers: when a normal executable is flagged in all computers we can assume this not to be malicious: even the clean computers have this executable.

The method to classify computers has three phases:

1. Extract information
2. Classify information
3. Use the information from step two to classify the computer

The result of step three will be a subset of computers that are more likely to be infected than the rest of the computers. This subset can then be investigated by forensic analysts.

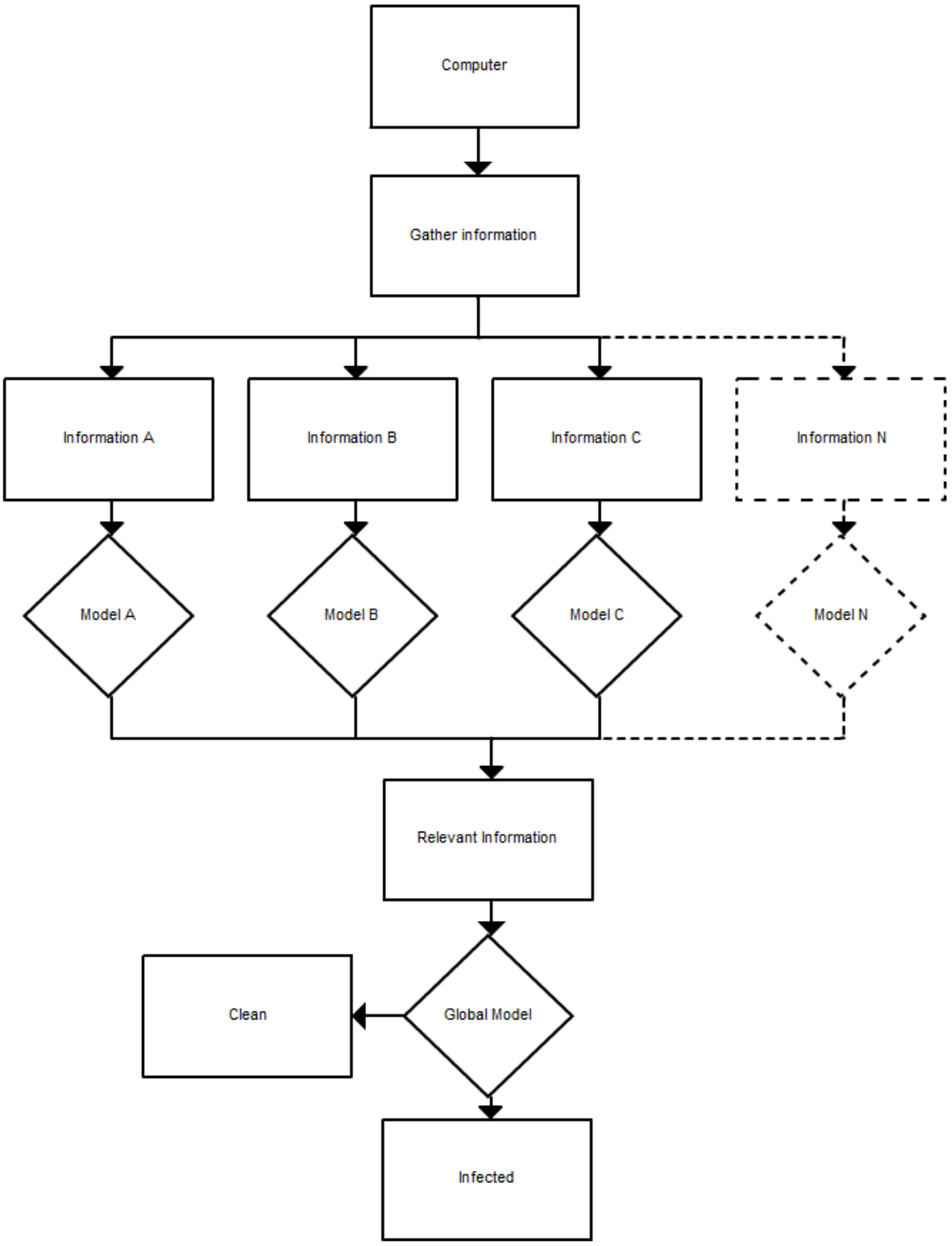

FIGURE 3.1: The method for classifying a computer

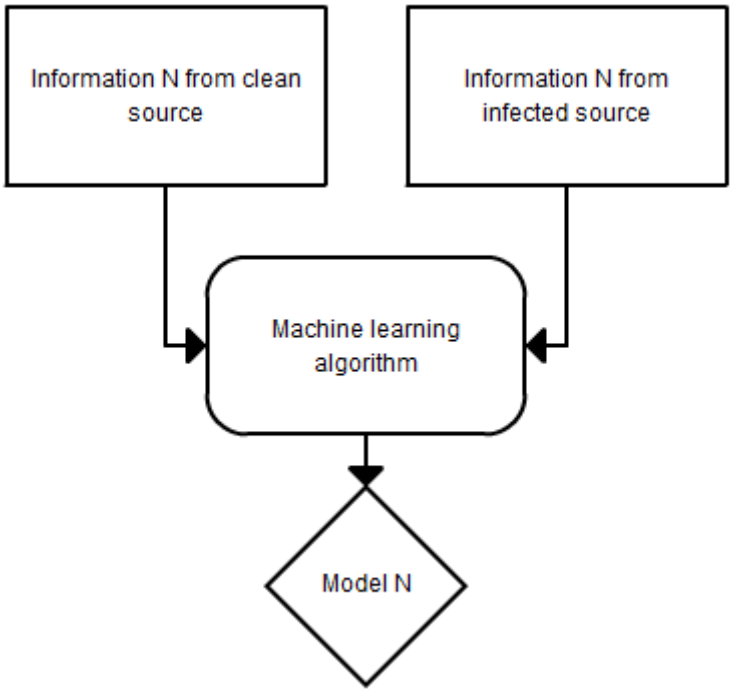

FIGURE 3.2: The process for making a model

See Figure 3.1 for an overview of the used method. First information about certain features are gathered on a computer, this results in a couple of groups of information, information A, B and C represents these groups of information. These groups of information are for example all of the executables on the hard drive. The information groups are then classified by their respective model. Only the information deemed relevant by these classification models are used to classify the entire computer. The output from these models is then used to see if the computer is likely to be infected. As a result there will be two subsets of computers: the infected and the clean computers. How large each set is and how much infected computers are missed (false negatives) will show if this method can be used to save time by forensic analysts. The time it takes to run these tests can be used to answer the question of the feasibility of such a system in a company wide search. There could be any number of information groups, not just three. Also not all information requires a model to separate relevant information from not relevant.

The creation of model A, B and C requires data of both infected and clean computers. In the case of executables both benign and malicious executables are required. This data is used to build the classification model. See Figure 3.2 for an overview of the creation of a model. These models are build before the actual testing and the same models can be reused for multiple tests.

The research will focus on Windows because it is the most commonly used OS according to Netmarketshare [69] and Statcounter [72]. Because of the focus on Windows, we can

use some Windows OS specific features.

3.4 Windows specific features

Four Windows specific features are described, equivalent features may exists in other OSes as well. The four features are file time (3.4.1), file system (3.4.2), registry (3.4.3) and system restore point (3.4.4).

3.4.1 File Time

File time analysis is an important way to determine which files where used. Some times are recorded, for example NTFS stores the following MAC times: Modified, Accessed, Create. The Modified is updated each time the file is changed, the accessed is updated when the file has been read and the create timestamp is updated at the creation of the file[11]. The accessed timestamps are disabled by default on Windows 7 and 8. These timestamps can be manipulated by attackers: if an attacker wants to cover his tracks he can backdate a certain file. This way it will seem that the file has not been altered and hopefully the forensic analyst will not notice that the file has been altered. The same can be done by changing a file but keeping the original and placing back this original file. However because Windows computers and certain file systems (NTFS, FAT) keep separate records, the records can conflict each other and this information can alert the analyst to the changed file.
Kim et al. [32] used somewhat a similar approach to this, they looked at whether the creation time is greater than the last modified time, this could indicate an attempt to backdate the file. An addition to this approach can be to look for anomalies in the MFT information, for example the sequence number and timestamps can be used to determine even better whether backdating took place.
Log2Timeline is a tool that can generate time line from log files. It can be useful for a forensic analyst to see a timeline of events around a certain time. This tool has the advantage that a clear timeline can be build for the forensic analyst which the analyst can use to see what events happened when. However it is not meant to do an automatic analysis of a system.
AnalyzeMFT is a python based tool that can extract the MAC times and other values from a MFT of a NTSF drive and outputs it in a .csv file format. The use of MFT times can be a great way to establish a timeline and determine which files were changed at a time. The advantage is that in some cases the use of backdating can be detected. The disadvantages of this kind of technique is that anomalies can be generated in a normal situation as well this is because the timestamps are not as precise and can be

updated different in some situations. Thus this technique has to be used in combination with other techniques to filter out the 'noise' and to supplement it to determine whether something is a false positive or if it actually is an anomaly.

3.4.2 Windows file system

Windows has two ways of loading a Dynamic Link Library (DLL) available: By specifying the full path or giving the filename. When a program only specifies the file name Windows will first looks at the local directory of the process that wants the .dll, when the file is not located here Windows will then search the system directory, next the 16 bit system directory, the Windows directory, the current directory, and then the PATH environment variable directories. Because Windows only checks whether the .dll file has the same name as the requested file an attacker can hijack this search by placing a .dll file with the same name in a directory that is searched first by Windows. If a .dll file is discovered in a strange directory while it should be in another directory, the .dll could give an indication that something is wrong with that system. The same can be done for other kinds of files: some .exe files should always be a in a certain directory. Finding this file somewhere else, for example the /temp/ directory, could also indicate suspicious behavior [34, 25, 57]. Another way is to record which files should be grouped together in a directory, after this baseline is created it is used to check another system for the same group of files. If the files deviates from this pattern it can be seen as an anomaly worth investigating [49].
Kwon et al.[34] made a tool that can detect unsafe loading of .dll libraries on windows machines. The tool can see whether an application provides the full path name of the .dll or that it just gives a name of the .dll it wants to find. They describe that a lot of applications are using the unsafe method of loading .dll libraries. This tool is especially useful in detecting which applications are vulnerable for .dll hijacking but not that useful for forensics uses because there are a lot of applications vulnerable for .dll injection. It can thus generate too much false positives to be useful.

3.4.3 Windows registry

The main thing the windows registry holds is configuration information of programs but it can contain all kinds of things: Recently accessed files, information about user activity, in some older versions code can be stored here as well. A forensic analyst can check this registry to figure out whether something is wrong; a certain value of the registry has a strange value or a different value then one might expect: an example of this is that a feature of Windows that reports the status of the firewall is disabled[58].Not

only anomalous information from the registry can be used: the windows registry also contains logs of devices that had been connected to the system, recently run programs and used wireless connections [10, 15, 14].
Farmer gives an overview in [15] of useful registry location for forensic purposes. These include auto-run locations, these locations can make a specific malware to be persistent: the malware will keep on infecting the computer after reboot, Most Recently Used (MRU) locations of keys and the location of value of wireless networks and USB keys. For a forensic triage point of view some of the keys can be useful in telling if the computer is infected. However the keys in these locations can vary a lot between different users and different installed applications. Some keys may be a good indicator across systems to see if the computer is likely to be infected while others may not be as useful. Dolan et al. [14] looks at the difference between the registry on the harddrive and the (part of the) registry in memory. They show that one can detect changes in the memory registry that could otherwise be unnoticed in a forensic analysis. The values in the registry in memory can be changed by an attacker and it would not be persistent, thus the next reboot could put the values back to normal. However the disadvantage is that the registry can be changed in memory as normal behaviour, when the system is shut down the registry changes are committed to disc. Thus this approach can lead to false positives of genuine differences between the registry in memory and the registry on the harddisk.

3.4.4 System Restore Point

System restore point is a feature in windows that gives an opportunity to restore the system to a point in time. A restore point contains information about the registry, file snapshots (critical .dll, .exe files) and file metadata. This information can be very useful for a forensic analyst. [26]

3.5 Features

3.5.1 Requirements

There are a couple of requirements on the features that can be used:

1. The features must be independent of the installed programs.
2. The features can be gained easy and quick enough for a live analysis.

3. The features should not produce a lot of false positives and negatives.

The first requirement makes sure the same feature can be used in different companies, all of these companies can have different programs installed. There are too much programs used by all of the companies that it is not possible to include them all into the baseline. Therefore it is better to have features that are not depended on the installed software. The second requirement is to make sure it can actually be used to quickly identify infected computers within a set because the goal is to save time. If the feature takes a lot of time to complete then it could cost more time than it saves. The third requirement means that the features should be representative of whether the computer is clean or infected. If this is not the case then the feature will not have a good enough ability to differentiate between clean and infected computers and it will be not as useful.

3.5.2 Scope

Features that have to do with networking, even though this can be a valuable source of information, will be left out of scope. This is because it also gives a lot of new problems. Some malware will only communicate every couple minutes or every couple of hours. How will a forensic analyst know how long to observe the traffic before the infected computer will communicate? The infected computer could also hide the malicious traffic into other normal traffic: this is an example of the stenography anti-forensic technique. Another problem can be that the normal traffic of a company depends on the programs that the employees use. Thus it can be too hard to make a baseline of normal office traffic because the normal office traffic is different everywhere. For these reasons the features that relate to networking will be left out of scope.

3.5.3 Features previous work

Kim et al[32] used the following features:

- Timeline analysis
- DNS Analysis
- Correlation Analysis between Network, the Connection and Processes
- ARP Analysis

These features can be helpful however networking is currently out of scope of our problem thus DNS, the correlation and ARP analysis cannot be done. Timeline analysis can and

probably will give a lot of false positives, also this analysis is more useful in combination with a forensic analyst. Thus none of these features will be used.

3.5.4 Selected Features

The focus will lie upon features and characteristics that are mostly used by malicious programs. They all mostly rely on certain anti-forensic techniques. Hiding from the user, hiding from inspection and virus scanners and hiding in memory.

The use of encryption and packers are chosen to detect malicious executable files. Both the entropy and malware features of executables are run both on the entire system and separately on the C:\Users folder. It is expected that more malware is positioned in the User folder than on the rest of the drive. Scanning this part of the drive is also faster because it is a subset of the entire drive. The information gained by this approach can be used to see whether scanning the Users folder is enough for high accuracy or the entire drive has to be scanned.
The detection of backdating is harder to do: Windows will generate anomalies in the MFT in normal situations. This is because of errors or because the file is moved or copied from another drive. The false positives that are generated this way can be easy to see for a forensic expert but learning a computer whether something is a false positive or a trace of malicious activity is somewhat harder to do and will take more time. Thus the automatic detection of backdating is probably not a good way to go.
Hiding in memory is also something normal programs will not do. It may be detected by dumping the memory of the system and analyzing it. However the dumping of the memory is the constraint in this case: the file representing the memory will be as large as the memory itself. Thus if a computer has 8 GB of memory, the file will also be the same size. This can give some problems if not one but 100 computers have to be investigated. The total space required for the analysis is 100 times the amount of memory available. Thus dumping the memory may not be a good way to obtain data about the computer. One of the ways of solving this problem is to use live memory analysis tools, these tools run on a live computer and do not require the memory to be dumped. Redline is such a tool, the MRI option of Redline can give an indication of whether the process is malicious.
Another information source is the registry; there is a lot of information available. However it will take too much time to analyze what parts of the registry each malware family changes. However we can look at keys that a lot of malware uses: the autorun locations. This can be used to provide malware with the ability to survive reboots (persistence). The autorun keys are easy enough to obtain and malware can easily be extracted from these values.

One last thing that will be checked is the CPU usage during the scanning process. Arnold [4] shows that rootkits and malware can have an impact on the CPU of a computer. This information is easy enough to obtain. The average CPU value of each of the other steps will be captured, analyzed and taken into account in the model.

Table 3.1 shows the features that are selected. In the next couple of sections the chosen features are explained more in dept.

Name	What?	Where?	Section
Entropy of executables	9 features of executables (focused on entropy)	File system	3.5.5.1
Malware features of executables	7 features of executables	File system	3.5.5.1
Redline	MRI score, path and if the process is hidden	Memory	3.5.5.2
Autorun	Description, publisher and path	Registry	3.5.5.3
CPU usage	CPU usage during above phases	Computer hardware	3.5.5.4

TABLE 3.1: Overview of the selected features

3.5.5 Modeling data

3.5.5.1 Executables

For the detection of potential malicious executables and .dll files two existing approaches are used. A high number of malware use packers (around 80%) to prevent detection by anti virus. The detection method of Santos et al.[51] is used to see if there are programs with packers on the system. However if this method works perfectly up to 80% can be detected. Therefore the use of another method is used to compensate for this limit. The method of Raman [46] is used to detect malware on the system. Only 7 features are collected instead of 42+, thus the time to collect this data will take less time. Also because of the anomaly based detection only a training set of malware is required and not a database of signatures for the entire malware population.

Packer detection The method of [45] is used as a basis for the packer detection. From the website of one of the writers of [45] the python script used to get the values of executables and the dataset is obtained. They use two datasets, the testset and the trainingset for this approach the two are combined. In the combined testset there are 3267 packed executables and 2231 not packed executables.

The 9 features that are collected are:

1. Number of standard sections
2. Number of non standard sections
3. Number of RWX sections
4. Number of executable sections
5. Number of entries in the IAT
6. Entropy of PE header

7. Entropy of code sections
8. Entropy of data sections
9. Entropy of the entire file

Malware detection For the detection of malicious executables the method of [46] is used. The python script pefile is used to obtain the required information from the executables. The dataset from training contains 35122 normal executables (both .exe and .dll files) from three clean computers that have a windows 7 installation. The malicious dataset was not limited to packed executables but also includes non packed executables. The set contains 106503 malware samples from various sources.

The 7 features collected are:

1. Debug size
2. Image version
3. RVA of the IAT
4. Size of exports
5. Size of resources
6. Size of the virtual section
7. Number of sections

Some interesting features are the image version, this is the version number of the executable, normally this is filled in by the creator of the executable. Thus a new version of a program will have a new image version. Attackers who create malicious programs usually skip the step of giving versions their malware, making the image version a good indication of whether the executable is malware or benign.

3.5.5.2 Memory analysis

Some tools are available for memory analysis. The two which are considered are Redline and Volatility. Redline is able to do live memory analysis and able to give an indication if a process is malicious. The result of Redline are considerable smaller than an memory image, the combination of these advantages are the reason that Redline is used in this method. Volatility on the other hand is currently not able to do live memory analysis and thus requires a memory image. Gathering a memory image for every computer that is investigated is too much data to gather, for this reason Volatility is not used.

A collector for Redline can be run at the target computer or memory image and extracts data. After this step the information is fed into Redline. Redline then processes this information and outputs this information into a SQLite database. The following information is extracted from this database.

- Process name, The name of each process, this is only used to identify a process.
- Hidden, whether Redline found that this process was hidden.
- Arguments
- SID
- Path, location of the executable.
- MRI, Malware Risk Index: this is the likelihood that this process is malicious. As determined by Redline.

The dataset to model the result of the Redline tool is made by running Redline on a couple of clean computers, this gave a total of 32 normal processes. After this computers were infected with known malware. The information that Redline gives on the malicious processes are used as the malicious dataset. This set contains 28 malicious processes.

3.5.5.3 Registry

The information we look at in the registry is the location of the autorun keys, these locations can provide malware and other programs with persistence; the ability to survive reboots. This information is selected because of the easy access to the registry and the wealth of information that is contained within this registry. The autorun keys are used a lot by malware because it is an easy way to provide the malware with persistence. Thus finding a malicious entry in this location is a good indication that the computer is infected. Therefore this information from the Registry is used.

The following information is gathered about the autorun registry keys:

- Description: this is a description of the executable.
- Publisher: this is the name of the company that provided the executable.
- Image path

The description and publisher should be filled in by the creators of the executable. Similar to the research of Raman[46] we suspect that an attacker will not fill in this information.

The main detection strategy is to check whether the executable is within the user directory because malware is usually placed there. The cause of this is that administrator access is needed before one can install something in the Program Files, Program Files

(x86) and System32 directories. Information about where the executable is located will by provided by the image path. The other values that are gathered to check whether malware entries have other values for these keys or if they even include these values.

The key values are extracted from the registry places using a tool called autorunsc.exe[70]. This tool is first run on a clean machine to see which programs are normally in the autorun location. After this the same machine is infected with malware samples and the same tool is executed to see which programs are modified and added to the list. The information from the clean and infected machines is used as a dataset for the autorun classifier. The dataset contains 21 benign autorun keys and 51 malicious keys.

Some values from this source are discarded: .lnk files do not contain the description and publisher and this could case incorrect results. lnk files are shortcuts to an executable, this is usually done by the user by placing an entry in the autorun folder. This causes the problem that the information about the description and publisher cannot be found, this because the file points to the executable and the lnk file itself does not have these values. This is different from other entries in that none of the .lnk files provide these values. Thus for the 'normal' entries this information can be used to see if the entry is malware while for the .lnk it is always absent.

3.5.5.4 System resources

Arnold in [4] looks at the impact of spy ware on the system resources. He shows that when malware or spyware is present on a system the CPU can have a higher than normal load. This method may also be able to detect malware on the systems. While the tool that collects the entropy and malware values the CPU usage is measured. We expect that the CPU usage will be higher on system that are infected than on clean systems. Even though tools are running the higher average CPU usage may be detected using this method. The values during the collection and processing of all of the other steps are measured.

3.5.5.5 Traces

The usage of these features will create traces of them on the computer. This can be a problem in a computer forensics context when the forensic experts do not want to create new traces on the computer. In the situation of emergency response when the objective is not to obtain evidence of a crime but to make sure the company can return to a working order while removing malicious activity from the network. In this case a forensic expert wants to find the infected computers as soon as possible and the creation

of extra traces is not as problematic. The method used here is best suited for the second case. However one can use this same features and method in a way that will not generate any traces on the computer. This way is to image the entire computer (both hard drive and memory), use a write blocker and after these steps use this method to analyze the computers. This way will however use a lot more time than analyzing the computer live. Which will reduce the effect of this system to save time and money.

3.6 Classification models

This next section is about the classification models, how they are created and which algorithms are suitable to generate these models. Section 3.6.1 gives the requirements for the algorithms. Section 3.6.2 shows the performance of the selected algorithms.

3.6.1 Requirements

The algorithms should be able to do the following:

1. High accuracy
2. Ability to modify the classifier after making it.
3. Time to classify the data

Lastly the type of the machine learning algorithm should be supervised learning. This because the data for the clean and malicious features is already classified and thus uses a supervised learning algorithm to make a model. Table 3.2 gives an overview of the categories of machine learning algorithms. For the validation of the models the best performing algorithms of the categories where a ✔ sign is in the suitable column. The values in the table are adapted from Kotsiantis et al.[33]. The categories that score at least two stars on each category are included within the model validation. Some other variants of machine learning algorithms that do not fit in these categories are also tested.

3.6.2 Model validation

All of the classifiers are created using Weka [23]. Weka is an open source machine learning tool that is written in java. The information that is extracted from the computers is read into Weka. For all of the features a separate model is created. The output from these models is then used to determine whether a machine is more likely to be infected

Algorithm type:	1	2	3	Suitable?
Decistion Trees	**	**	****	✔
Neural Networks	***	***	****	✔
Naïve Bayes	*	****	****	✗
kNN	**	****	*	✗
SVM	****	**	****	✔
Rule-learners	**	****	****	✔

TABLE 3.2: Suitable machine learning algorithms

or clean. See Table 3.3 for an overview of the performance of different machine learning algorithms per category of the models that are created during step one. The percentages in the table are the result of the 10-fold cross validation done by Weka. The percentages in **bold** are the best performing algorithms for that feature. The models created by these algorithms are used for predicting whether something is malicious or benign. The output from each of these models per category is used to predict whether the entire computer is infected or clean. The models are selected based on the requirements above. Some of them are the best performing within their category, others are also used to test these methods before.

	Malware	Entropy	Redline	Autorun
J48	99.0701	99.4725	96.6667	95.8333
RandomForest	**99.3003**	**99.8181**	95.0000	97.2222
RandomTree	99.1866	99.4907	**100**	98.6111
JRip	98.4946	99.5089	96.6667	95.8333
BayesNet	94.9832	98.9451	96.6667	**100**
SVM	95.1425	99.1451	97.1154	98.6111

TABLE 3.3: Performance of machine learning algorithms

3.7 Testing procedure

The features described in Section 3.5 are gathered from the computers in the data sets. This information is then classified using the models. The output from the different classification models is then used to see whether it can be used to determine which computers are most likely to be infected.

Two tests will be executed, the first test is executed on an ESXi server. There are two machines active on that server: one is the virtual machine (VM) that is to be tested, the other one observes the testing VM and runs the extraction and analysis program. When a test is completed the testing VM is set to the next testing stage by using a configured snapshot. After this the analysis program is executed again. This process is repeated until the 50 VMs are tested. The second test is executed on images from

companies that were investigated. Only the features that are suitable from the first test are gathered in the second test. After each test the results from the features are used to determine whether a computer is likely to be infected or clean.

3.8 Data sets

Two distinct data sets are be used to validate the research. The first data set mimics an organization. This to produce results of how good this will perform in an organization. No such set is available for testing thus it will be generated. There will be 50 machines, divided into four groups. There is software installed on the computers from these groups. Section 3.8.1 describes what programs are installed, Section 3.8.3 shows an overview of the data set. The second data set is a real life dataset that contains forensic images that are gathered from government institutions and companies. Some of them contain malware, others were used to detect fraud within a company. The clean images can check if there are a lot of false positives and the images with malware on them can be used to check how much false negatives this method generates. These forensic images are not all from within the same company thus it can be expected that every image is quite unique with different programs on them. It can be interesting to see if the same approach works with images from different companies. Section 3.8.3 shows an overview of the data sets.

3.8.1 Installed Programs

The analysis of a computer should be independent from the installed programs, this requirement is used to remove the need to baseline all of the existing programs. To see whether (unknown, new) programs skew the results of the analysis some programs will be installed. The following types of programs are assumed to be on the normal computer within an average company:

- Anti Virus
- Web-browser
- Chat program
- E-mail program
- Office
- Programs to open certain files (.pdf, media and .rar/.zip files)

The programs purely related to business are taken from the highest ranking business application of download.com[64]. The more general programs are chosen from the most popular downloads of download.com[65]. The two highest ranking programs are used for each category.

- Avast & AVG
- Firefox & Chrome
- Microsoft Outlook & Mozilla Thunderbird
- Skype & Yahoo Messenger
- Microsoft Office & OpenOffice
- KMPlayer & VLC
- WinRar & WinZip
- Adobe Reader & PDFCreator

These programs are (randomly) divided into groups with the exception of Microsoft Office and Microsoft Outlook as they are usually installed together. Facebook messenger is more popular than Yahoo messenger however the installer does not work, thus Yahoo messenger is chosen. The groups of programs are put in opposite categories, thus if Avast is place in group A then AVG is placed in group B to prevent multiple anti virus and browsers to be installed on the same system while others may not have these programs.

A Microsoft Outlook, Microsoft Office, VLC

B Mozilla Thunderbird, OpenOffice, KMPlayer

C Avast, Chrome, Yahoo Messenger, WinRar, PDFCreator

D AVG, Firefox, Skype, WinZip, Adobe Reader

These groups are combined as follows:

1. A, C
2. A, D
3. B, C
4. B, D

The applications are divided this way to prevent a computer from having two browsers but no anti virus and other combinations. Applications of group 1 and 3 are installed on 13 computers and group 2 and 4 on 12 computers. This way every application is installed on half of the total number of computers.

The applications on the computers will show up in the results of the executable analysis and autorun because these methods look at the information at the hard drive and registry. This information is added to the registry and hard drive at the time of installation. The programs will not show up in the Redline features when it is not running therefore some of the programs are executed before starting the Redline analysis. This way the programs will also show up in the Redline analysis.

3.8.2 Malware used

In total eight computers are infected with rootkits and 22 computers are infected with malware. The malware samples are obtained from a site that packs malware for attackers. From a set of about 3000 samples the 22 samples are chosen at random. Five of the eight samples of rootkits are current Zeus droppers obtained from ZeuSTracker[62]. The other three samples are SpyEye droppers obtained from the SpyEye Tracker[61].

3.8.3 Data sets

Table 3.4 contains an overview of the first data set, there are 50 computers in total. 20 clean computers, 8 with rootkits and 22 with malware. Normally only one computer can be infected in a company, however we will test on a data set with a larger number of infected computers to see whether different infections can be reliably detected instead of only one specific sample. All computer run the same version of Windows 7, Windows 7 is used because it is the most commonly used Windows based OS according to netmarketshare[69] and statcounter[72].

Group	Clean	Rootkits	Malware	Total
1	5	2	6	13
2	5	2	5	12
3	5	2	6	13
4	5	2	5	12
Total	20	8	22	50

TABLE 3.4: Overview of the data set

Table 3.5 contains an overview of the second data set. The data set contains four different Windows OSes: Windows XP, 7, Server 2003 and Server 2008. These images are obtained on different times from a diverse group of companies. Three of the images have malware on them, others were gathered for a fraud investigation or other goals. Most of the Server 2003 and 2008 images were breached by an attacker and they contain some traces of this attacker. However these images are classified as clean because this

method looks for traces of malicious programs. They will be reclassified if malicious programs are found on these servers.

OS	Clean	Malware	Total
7	8	1	9
XP	20	2	22
Server2003	16		16
Server2008	10		10
Total	54	3	57

TABLE 3.5: Overview of the second data set

3.9 Overview of datasets used

Table 3.6 gives an summary of the datasets used for building the models. The numbers behind Entropy and Malware correspond to the number of executables used. The numbers behind Redline and Autorun correspond to the number of entries found in clean and malicious computers. The number behind first test and second test correspond to the number of computers in each set.

Name	Malicious	Normal	Total
Entropy	3.267	2.231	5.498
Malware	106.503	35.112	141.625
Redline	28	32	60
Autorun	51	21	72
First test	30	20	50
Second test	3	54	57

TABLE 3.6: Overview of the datasets used within this research

Chapter 4

Results

4.1 First dataset

4.1.1 Results of first dataset

The results for Redline are missing, this is because Redline did not give reliable results for most of the test. Only at a couple of tests did Redline provide any valid results. Thus Redline does not seem reliable enough for using it in this method.
The results are in Table 4.2. The features are in the top row. When a classification model gave a positive hit or multiple hits for a computer it is counted in the True Positive (TP) if the computer was actually infected or in the False Positive (FP) if the computer was not infected. When a classification model did not give any positive hits for a computer it is counted as a True Negative (TN) if the computer was not infected and as a False Negative (FN) when the computer was infected. The total score the combination of any of the features. The total suitable score is the combination of the suitable features (Autorun, Usermalware & User Entropy). See Section 5.1 for why these features are deemed suitable. The tests that failed to give any results are included in the total and total suitable scores.
The results are filtered on a value of 5. This value is the median number of how much computers actually contain a single file. This median value was selected because the files that are present in a lot of machines are not interesting for the investigation. See Table 4.1 for an overview of the number of files similar over multiple computers. The top row shows the number of computers that contain the same file. The bottom number is how many files are in that category. Thus there are two files that are in 45 computers and there are 17 unique files within the data.

Computers	45	24	23	21	14	13	10	7	5	4	1	Total
Files	2	4	1	1	2	1	3	1	4	1	17	37

TABLE 4.1: Number of files similar per computer for the first dataset

The classification of the first dataset is quite good: 90% of the computers were classified correctly. There were some problems however: failed tests and false negatives. First the failed tests: five of the tests did not complete within four hours. After these four hours the tests were aborted. During the tests that would take too long to be finished the CPU percentages for the testing process was constant at 0%. Thus it can be concluded that it did not run correctly or did not receive any CPU cycles. This only happened in the malicious part of the test: three within the malware and two within the rootkit set. An explanation can be that remote executed programs were blocked by the malicious programs running on the computer. Another explanation is that something went wrong with the test.
The false negatives are hard to explain because the precise working of the malware is unknown. Looking at the raw data provided by the extraction and comparing it to the clean counterpart revealed no different files on the false negative computers. Perhaps the malware did not perform correctly on Windows 7 or it hid itself within the volatile memory instead of the hard drive and registry.

	TN	TP	FN	FP	DR	Remaining
Autorun	20	16	14	0	72%	32%
Malware	20	20	10	0	80%	40%
UserMalware	20	18	12	0	76%	36%
Entropy	20	12	18	0	64%	24%
UserEntropy	20	11	19	0	62%	22%
Total	20	25	5	0	90%	50%
Total Suitable	20	23	7	0	86%	46%

TABLE 4.2: Results per feature for the first data set

4.1.2 Collection time

See Table 4.3 for an overview of the duration of the collection of the different features. The clean set contains the computers that are not infected, the infected set contains the computers infected with either rootkits or malware. The times in the table are in seconds. Redline is missing from this table because it did not provide any reliable results. To measure the entropy characteristics of all PE files on a computer will take about 1 hour. To gather all of the malware characteristics it will take about 26 minutes. Based on these times the Entropy and Malware features of the entire computer may not be suitable features.

Set	Malware	User Malware	Entropy	User Entropy	Autorun
Clean	1592	10	3598	19	6
Infected	1620	11	3571	19	6

TABLE 4.3: Duration of the collection of information in seconds

4.1.3 CPU times

Table 4.4 contains the average CPU (in %) of the computer per different feature. The average CPU in the Redline feature is a lot higher for the malware and rootkit computers than for the clean computers. The CPU percentages are also available for Redline because it did run on most of the computers however it did not provide reliable results.

	Entropy	Malware	Autorun	Redline
Clean	13,0	12,0	5,3	19,0
Malware	13,5	13,5	12,1	25,8
Rootkit	13,7	14,3	7,9	31,1

TABLE 4.4: Average CPU % per feature

The average CPU times are a lot higher in the infected sets as compared to the clean set. This can indicate that it is a lot more likely that a computer that has a high CPU percentage is more likely to be infected. Some things have to be noted about this approach. The CPU times should be done on the same type and speed of the processor. Otherwise it could lead to some false conclusions. For example a weaker CPU with less cores will show more CPU percentage during the tests. This could also lead to false positives. Another way is to measure the average CPU on an idle computer. This could lead to better results because the only processes that are using CPU cycles will be the malicious programs. This will not work for machines that are always in use and using much CPU cycles. Thus on machines that are never idle this test will not work.

4.2 Second dataset

4.2.1 Results of second dataset

The results of the second test are in Table 4.5. The best result is obtained using the autorun feature. This feature managed to detect all of the malicious instances. There were however some false positives. Nine computers are also flagged as malicious by this feature. In the end a detection rate of 84% was achieved by the autorun feature. The potential malicious set consists out of 21% of the total set. Thus a reduction of workload of about 79% can be achieved. The other features got a higher remaining percentage, which indicates that these features are less effective in reducing the workload.

	TN	TP	FN	FP	DR	Remaining
Autorun	45	3	0	9	84%	21%
Malware Exe	7	3	0	47	18%	88%
Malware Dll	14	3	0	40	30%	75%
Entropy	29	1	2	25	53%	45%
Total Suitable	7	3	0	47	18%	88%

TABLE 4.5: Results per feature for the second data set

Table 4.6 gives an overview of the number of files that are similar on computers. In this case the median number will be 1. This is not a good choice because the filter will not leave any files: each file is at least positioned at one computer. Therefore the results were not filtered. Table 4.6 also shows that there are a lot of unique files that are classified as malicious. Most of these files are false positives. There are however some positives on clean computers that could also be regarded as a right hit: three programs on two computers are identified as adware. Adware is a program that is usually included with genuine software that shows ads to users. These kinds of programs are usually unwanted by the user of the computer and may point to other malicious programs. However these genuine hits are buried by other false positives and because it is adware and not 'real' malware it is not included as infected in the analysis.

Computers	25	19	9	8	6	5	4	3	2	1	Total
Files	1	3	3	1	4	4	22	22	56	1221	1337

TABLE 4.6: Number of files similar per computer for the second dataset

4.2.2 False positives

There are some false positives encountered within the data. Most of them are from the malware and entropy characteristics and some of the autorun keys were deemed malicious by the model. A couple of autoruns that were flagged by the model are simple .url files that link to some webpage. Perhaps the user of the computer wanted to open a certain webpage every time the computer started. However this .url file does have a publisher and description. Other autorun keys seem to be programs specific to the organization while these also did not contain any information about the publisher and description. Most of these are false positives were those specific to the organization or a certain link file that linked either to an address or to a program on the computer. To mitigate these false positives in the future a better model could be used or there could be other variables that can be included within this analysis such as if the executable is signed. Other ways to reduce the false positives are white listing of certain entries within a company or filtering these values when a lot of computers report having the same autorun entry.

The false positives of the malware and entropy characteristics include a lot of installers. These are usually flagged as malicious because they have similar features. One of the things to mitigate these is to improve the model by including more installers and other files that are usually false positives. Perhaps the model will classify things better if there are more categories. Another category could include the installers. Some new file could be then classified as an installer instead of a malicious file because it is closer to the installer class than the malicious class. A problem that could arise is that most malware droppers can also be seen as an installer of the malicious file.

4.2.3 False negatives

There were a few false negatives as apposed to the false positives. One of the reasons is that the malicious set and the malicious traces are only a small percentage of the total set. Thus one can only have so much false negative signals as apposed to the false positives. The false negatives in this case were caused because the two malicious files were changed by the organization to render them useless. However this also made sure that it could no longer be detected. In this case the malicious file was already found and it did not need to be detected anymore. Thus it should not be a problem in the case of a company which does not known where the infection is because disabling requires knowing where and what the infection is.

4.3 Reducing false positives

Filtering was used to reduce the false positives. The selection of the filtering value can be an issue: how do you know if the infection was also filtered away or if there was no infection to begin with. The filtering value can be adjusted when nothing is found the first time around. However this would increase the time it takes to actually find it. There are some other ways: as a filtering value someone can just pick two, this would mean that only the unique files will remain. This has the same problem as before but it can be helpful in finding small scale infections.

A whole other way of reducing false positives is the combination of signals. In this case there are two signals for malicious executables: malware and entropy characteristics. Instead of flagging something malicious when either one of them classifies them as such, one can also count it as malicious when both classify them as such. This will result in less positives but can also increase the false negative rate because only about 80% of malware uses packers thus about 20% could go undetected.

4.4 Detection rate vs Reduction rate

Detection rate is the number of machines that are classified correctly, the reduction rate is the ratio between the actual number of computer and the number of computers that have to be investigated based upon this method. These two rates are related, for example if there is a company with 100 computers and one of them is infected and the method gives a 100% detection rate then the reduction rate is the maximum achievablee rate: only one computer has to be investigated. However if the detection rate is not 100% it can have a negative impact on the reduction rate. In the same situation as before the company has 100 computers and one infected. If the method classifies everything as not infected the detection rate would be 99%. However the reduction rate will be 0%: if none of the computers are selected, all of them will have to be investigated. In a situation where 10 computers are classified as infected including the one true positive then the detection rate would be 91%, this case will however have a way larger reduction rate of 90%. Therefore the optimal values for detection rate and reduction rate should be found. In the first dataset the best performance is given by the total of all of the features. It gives a detection rate of 90%: this is the highest value. However the remaining set is 50% of the original which is not the lowest value. The user entropy feature gives a remaining set of 32% of the original set. However the combination of features would still be the best case because the difference between the 32% and 50% is caused by the higher number of false negatives. Thus more infected machines are in the total set as compared to the user entropy set. The second set has the opposite behaviour. The highest detection rate also reduces the infected set the most. Therefore this value is both the best in terms of detection and reduction rate. There are a lot of situations in which the highest detection rate will also lead to the greatest reduction of computers, however this does not have to hold up in every situation.

4.5 Time saved

Section 3.1 gives an estimate of how long an analysts needs to check a computer for an infection. In case of a large company the forensic analyst will most likely use a quick scan. If we apply these numbers to the two datasets then it will take approximately 200 hours to check the first set for infections and about 228 hours for the second set. If the system described is used this can reduce this time of the first dataset to 100 hours and the time of the second dataset to 36 hours. This shows that a lot of time can be saved with prioritizing computers based upon these features.

Chapter 5

Conclusion

The research questions are:

1. Which features can be used to baseline and identify a possible breach in a computer?
2. To what extend can these features be used to reduce the number of computers that have to be investigated in a large data set?
3. How feasible is such a system to be used in a company wide search?

5.1 Suitable features

To answer the first question we have to look at the requirements of the features:

- The baseline must be independent of the installed programs.
- The information can be gained easy and quick enough for a live analysis.
- The features should not give a lot of false positives and negatives.

See Table 5.1 for an overview which features conform to the selected limitations. User malware and entropy are both quite independent of the Installed software: normally software is installed in the program files folder and not in the user folder. There are some exceptions to this rule but generally speaking these function are quite software independent.

The malware and entropy functions seem to generate a lot of false positives, this can be expected because the models that are used to classify executables into malicious or

	Time	Installed software	FP	FN	Suitable?
Malware	✗	✗	✗	-	✗
User Malware	✔	✔	✗	✔	✔
Entropy	✗	✗	✗	-	✗
User Entropy	✔	✔	✗	✔	✔
Redline	-	-	-	-	✗
Autorun	✔	✔	✔	✔	✔

TABLE 5.1: Overview of suitable features

clean have a detection rate of 99.8181 and 99.3003 %. However if the large number of executables is taken into account there are a lot of executables misclassified.

One feature which has all of the requirements is the Autorun feature. The malware and entropy characteristics have most of the requirements but still having the problem of false positives, besides this downside the first test shows that these false positives can be mitigated in a homologous environment. For example by selecting only the malware hits that are only on a few computers. The second test shows that this will not work in every situation, in such a case it is better to use the autorun feature: the autorun keys gave generally the best performance during the two test. It gave no false positives in the first test and some false positives in the second test.

Thus the features that can be used to baseline and identify a possible breach in a computer are autorun keys located in the registry and to some extend the malware and entropy characteristics of the user folder.

5.2 Reduction in data

The first dataset contains 30 infected computers. After testing only 25 computers were selected by this method. This would be a reduction of 50% in the number of computers. Of these 25 computers in this set all of them contained malware. This means that 10% of the computers were not identified correctly.

The second data set contains less malware than the first: about 5% (3 computers) of the computers have some sort of malware on them. After testing them the best method gave a reduction of 79% of data which contained all of the computers with malware. So instead of all of the computers only a small percentage of 21% (12 computers) will have to be investigated.

During the two tests a maximum of 79% reduction was achieved. Perhaps when including other methods a higher reduction can be achieved. It can be noted that the maximum reduction is dependent on the number of infected computers in the set. If we look

purely to the reduction of the clean computers a reduction of about 81% is achieved in the second set and 100% in the first test. Thus this method can be used to reduce the workload of the forensic analysts.

5.3 Feasibility

The best situation will be when the time to scan takes at most a couple of minutes. If the scan will take more than an hour the forensic analysts will have to wait too long for a result. Table 4.3 contains the collection time of the first dataset. The entropy and malware characteristics take too long and it will not be feasible for a company wide search. Even though it can be done in parallel. The malware and entropy characteristics of the user folder is a lot more feasible. In the first test it only took about 30 seconds to collect both features. The collection time of the autorun feature is also acceptable, it only took about 6 seconds to collect this information. During the second test the longest collection of information of a computer was around 5 minutes. This time is still feasible to use in a company wide search. One thing to note is that these features can be extracted parallel on each computer. Thus the time does not take 10 times as long for 10 times more computers. The two parts of the process that have impact on the total duration of the method are the time it takes to process the results and the longest extraction time of the features. During the second tests it took the processing program less than a second to classify all of the data from the features. Thus it seems that the longest time it takes for the computers to collect the information will influence the total time. In the first test this is about 30 seconds and in the second test it would take around 5 minutes. Thus it is very feasible to use such a system in a company wide search.

5.4 Future work

There are some things that can be done to improve this research: Networking was left out of scope because of the many problems that networking introduces (Section 3.5.2). If the network is included as a potential information source there are some new possibilities. One of the things that can be done is comparing the ports that the computer reports are open versus the ports a port scanner sees. This could reveal ports hidden by malware or rootkits and provide valuable information.
The use of memory images was also deemed not viable for this research. Dumping the memory of computers and sending them to the computer that will analyze the results would take too much time because all of the test run on the computers at the company,

this means that they are done in parallel. In case of analyzing memory images on a system that the analysts brought then all of the images has to be processes in sequence. This will change the time requirement from a maximum of the longest time to all of the times added together. If these factors are not a problem, for example by dumping the memory contents to an USB drive and enough processing power, then it could provide the forensic analysts with valuable information. Some other possibilities can emerge as well: for example the method of bianchi et al.[7]: they try to cluster similar memory images together where the smallest clusters of computers are more likely to be infected with rootkits.

The only files that were looked at are PE files because these files are easy to abuse: an exploit can just execute such a file or the user can be tricked to execute the malicious executable. There are some other files that can be of interest. For example pdf files can be rigged with a malicious payload, an user only has to open the file for the payload to activate. These kinds of attacks are very effective when they are combined with social engineering. Similar to pdf files are rigged Microsoft Office files. These files work in the same way. When a document is opened the payload is activated. It could be possible to detect these kind of files on a computer. For example an anti-virus application can be used to detect such malicious files

The prioritization is done by putting computers in one of two possible groups: clean or infected. If a forensic analyst wants to quickly find an infected computer and work from there it may be a good idea to prioritize computers in more than two groups. A sliding scale such as likely clean to likely infected can be used instead of a binary scale. The likely infected computers can be investigated first and the likely clean computers can be investigated last or ignored entirely. The same thing can be done for the different signals. The autoruns give for example results which have few false positives but also will not detect everything. The malware feature will give a lot of false positives but will detect almost everything. These signals can be given different weights. Thus autoruns are given a higher weight: when it detects something it gives a lot more impact on the score as apposed to a hit from the malware feature. This could lead to a better prioritization of computers.

Bibliography

Peer reviewed literature

[1] C. Abad, J. Taylor, C. Sengul, W. Yurcik, Y. Zhou, and K. Rowe. Log correlation for intrusion detection: a proof of concept. In *Computer security applications conference, 2003. proceedings. 19th annual.* IEEE, 2003, pp. 255–264.

[2] M. M. K. Al-Anezi. Generic packing detection using several complexity analysis for accurate malware detection. *International journal of advanced computer science and applications,* 5(1):7–14, 2014.

[3] A. R. Arasteh, M. Debbabi, A. Sakha, and M. Saleh. Analyzing multiple logs for forensic evidence. *Digital investigation*, 4:82–91, 2007.

[4] T. M. Arnold. A comparative analysis of rootkit detection techniques. PhD thesis. UNIVERSITY OF HOUSTON, 2011.

[5] G. B. Bell and R. Boddington. Solid state drives: the beginning of the end for current practice in digital forensic recovery? *Journal of digital forensics, security & law*, 5(3), 2010.

[6] R. Bertè, F. Marturana, G. Me, and S. Tacconi. Data mining based crimedependent triage in digital forensics analysis. In *Proceedings of 2012 international conference on affective computing and intelligent interaction (icacii 2012) and ieri lecture notes in information technology issn*, 2012, pp. 2070–1918.

[7] A. Bianchi, Y. Shoshitaishvili, C. Kruegel, and G. Vigna. Blacksheep: detecting compromised hosts in homogeneous crowds. In *Proceedings of the 2012 acm conference on computer and communications security.* ACM, 2012, pp. 341–352.

[8] G. Cantrell, D. Dampier, Y. S. Dandass, N. Niu, and C. Bogen. Research toward a partially-automated, and crime specific digital triage process model. *Computer & information science*, 5(2):29–38, 2012.

[9] B. Carrier. The sleuth kit. *Tsk). http://www.sleuthkit.org/sleuthkit/*, 2010.

[10] H. Carvey. The windows registry as a forensic resource. *Digital investigation*, 2(3):201–205, 2005.

[11] K.-P. Chow, F. Y. Law, M. Y. Kwan, and P. K. Lai. The rules of time on ntfs file system. In *Systematic approaches to digital forensic engineering, 2007. sadfe 2007. second international workshop on.* IEEE, 2007, pp. 71–85.

[12] S. Davidoff. Cleartext passwords in linux memory. *Massachusetts institute of technology*:1–13, 2008.

[13] D. E. Denning. An intrusion-detection model. *Software engineering, ieee transactions on*, 13(2):222–232, 1987.

[14] B. Dolan-Gavitt. Forensic analysis of the windows registry in memory. *Digital investigation*, 5:S26–S32, 2008.

[15] D. J. Farmer. A forensic analysis of the windows registry:1–17, 2007.

[16] F. C. Freiling and B. Schwittay. Towards reliable rootkit detection in live response. In *Imf*, 2007, pp. 125–144.

[17] S. Garfinkel. Anti-forensics: techniques, detection and countermeasures. In *2nd international conference on i-warfare and security*, 2007, p. 77.

[18] S. L. Garfinkel. Digital forensics research: the next 10 years. *Digital investigation*, 7:S64–S73, 2010.

[19] S. L. Garfinkel. Digital media triage with bulk data analysis and bulk_extractor. *Computers & security*, 32:56–72, 2013.

[20] S. Garfinkel, A. J. Nelson, and J. Young. A general strategy for differential forensic analysis. *Digital investigation*, 9:S50–S59, 2012.

[21] L. Gomez. Triage in-lab: case backlog reduction with forensic digital profiling. In *Proceedings of the argentine conference on informatics and argentine symposium on computing and law*, 2012, pp. 217–225.

[22] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: cold-boot attacks on encryption keys. *Communications of the acm*, 52(5):91–98, 2009.

[23] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *Acm sigkdd explorations newsletter*, 11(1):10–18, 2009.

[24] Y. Al-Hammadi and U. Aickelin. Detecting botnets through log correlation. *Arxiv preprint arxiv:1001.2665*, 2010.

[25] U. Haritha and V. L. Naidu. A survey on windows component loading vulnerabilities. *International journal of advanced research in computer engineering & technology (ijarcet)*, 2(5):pp–1780, 2013.

[26] K. Harms. Forensic analysis of system restore points in microsoft windows xp. *Digital investigation*, 3(3):151–158, 2006.

[27] R. Harris. Arriving at an anti-forensics consensus: examining how to define and control the anti-forensics problem. *Digital investigation*, 3:44–49, 2006.

[28] G. Horsman, C. Laing, and P. Vickers. A case based reasoning system for automated forensic examinations, 2011.

[29] K. Kent, S. Chevalier, T. Grance, and H. Dang. Guide to integrating forensic techniques into incident response. *Nist special publication*:800–86, 2006.

[30] G. C. Kessler. Anti-forensics and the digital investigator. In *Australian digital forensics conference*, 2007, p. 1.

[31] H. Khan, F. Mirza, and S. A. Khayam. Determining malicious executable distinguishing attributes and low-complexity detection. *Journal in computer virology*, 7(2):95–105, 2011.

[32] A. C. Kim, W. H. Park, and D. H. Lee. A study on the live forensic techniques for anomaly detection in user terminals. *International journal of security & its applications*, 7(1), 2013.

[33] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas. Supervised machine learning: a review of classification techniques, 2007.

[34] T. Kwon and Z. Su. Automatic detection of unsafe component loadings. In *Proceedings of the 19th international symposium on software testing and analysis*. ACM, 2010, pp. 107–118.

[35] Z. Liang, H. Yin, and D. Song. Hookfinder: identifying and understanding malware hooking behaviors. *Department of electrical and computing engineering*:41, 2008.

[36] S. Lowman. The effect of file and disk encryption on computer forensics. *Http://lowmanio. co. uk/. acesso em*, 7(05):2011, 2010.

[37] R. Lyda and J. Hamrock. Using entropy analysis to find encrypted and packed malware. *Ieee security & privacy*, 5(2):40–45, 2007.

[38] D. Manson, A. Carlin, S. Ramos, A. Gyger, M. Kaufman, and J. Treichelt. Is the open way a better way? digital forensics using open source tools. In *System sciences, 2007. hicss 2007. 40th annual hawaii international conference on*. IEEE, 2007, 266b–266b.

[39] A. Marrington, I. Baggili, G. Mohay, and A. Clark. Cat detect (computer activity timeline detection): a tool for detecting inconsistency in computer activity timelines. *Digital investigation*, 8:S52–S61, 2011.

[40] F. Marturana, S. Tacconi, R. Berte, and G. Me. Triage-based automated analysis of evidence in court cases of copyright infringement. In *Communications (icc), 2012 ieee international conference on*. IEEE, 2012, pp. 6668–6672.

[41] J. McMillan. Importance of a standard methodology in computer forensics. *Information security reading room*, 2, 2000.

[42] P. Pajek and E. Pimenidis. Computer anti-forensics methods and their impact on computer forensic investigation. In, *Global security, safety, and sustainability*, pp. 145–155. Springer, 2009.

[43] A. Patcha and J.-M. Park. An overview of anomaly detection techniques: existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470, 2007.

[44] S. Peisert, M. Bishop, S. Karin, and K. Marzullo. Analysis of computer intrusions using sequences of function calls. *Dependable and secure computing, ieee transactions on*, 4(2):137–150, 2007.

[45] R. Perdisci, A. Lanzi, and W. Lee. Classification of packed executables for accurate computer virus detection. *Pattern recognition letters*, 29(14):1941–1946, 2008.

[46] K. Raman. Selecting features to classify malware. *Infosec southwest*, 2012.

[47] G. G. Richard III and V. Roussev. Next-generation digital forensics. *Communications of the acm*, 49(2):76–80, 2006.

[48] V. Roussev, C. Quates, and R. Martell. Real-time digital forensics and triage. *Digital investigation*, 10(2):158–167, 2013.

[49] N. C. Rowe and S. L. Garfinkel. Finding anomalous and suspicious files from directory metadata on a large corpus. In, *Digital forensics and cyber crime*, pp. 115–130. Springer, 2012.

[50] N. C. Rowe and S. L. Garfinkel. Finding suspicious activity on computer systems, 2012.

[51] I. Santos, X. Ugarte-Pedrero, B. Sanz, C. Laorden, and P. G. Bringas. Collective classification for packed executable identification. In *Proceedings of the 8th annual collaboration, electronic messaging, anti-abuse and spam conference*. ACM, 2011, pp. 23–30.

[52] T. Stallard and K. Levitt. Automated analysis for digital forensic science: semantic integrity checking. In *Computer security applications conference, 2003. proceedings. 19th annual*. IEEE, 2003, pp. 160–167.

[53] M. C. Stamm, S. K. Tjoa, W. S. Lin, and K. R. Liu. Anti-forensics of jpeg compression. In *Acoustics speech and signal processing (icassp), 2010 ieee international conference on*. IEEE, 2010, pp. 1694–1697.

[54] M. C. Stamm, S. K. Tjoa, W. S. Lin, and K. R. Liu. Undetectable image tampering through jpeg compression anti-forensics. In *Image processing (icip), 2010 17th ieee international conference on*. IEEE, 2010, pp. 2109–2112.

[55] D. S. Thomas and K. Forcht. Legal methods of using computer forensics techniques for computer crime analysis and investigation. *Issues in information system*, 5(2):693, 2004.

[56] C. Thuen. Understanding counter-forensics to ensure a successful investigation, 2007.

[57] S. Veeralakshmi and M. Sindhuja. A secure environment for unsafe component loading. *International journal of engineering and computer science*, 2(4):1111–1116, 2013.

[58] H. Xie, K. Jiang, X. Yuan, and H. Zeng. Forensic analysis of windows registry against intrusion. *International journal of network security & its applications*, 4(2), 2012.

[59] A. Yasinsac, R. F. Erbacher, D. G. Marks, M. M. Pollitt, and P. M. Sommer. Computer forensics education. *Security & privacy, ieee*, 1(4):15–23, 2003.

[60] Y. Yusoff, R. Ismail, and Z. Hassan. Common phases of computer forensics investigation models. *International journal of advanced computer science and information technology*, 3(3):17–31, 2011.

Online sources

[61] abuse.ch. Spyeye tracker. 2014. URL: `https://spyeyetracker.abuse.ch/`.

[62] abuse.ch. Zeus tracker. 2014. URL: `https://zeustracker.abuse.ch/`.

[63] A. Data. Forensic toolkit. 2014. URL: `http://www.accessdata.com/products/digital-forensics/ftk`.

[64] Download.com. Most popular in business software. 2014. URL: `http://download.cnet.com/windows/business-software/most-popular/3101-2010_4-0.html?showAll=nodeIds`.

[65] Download.com. Most popular windows downloads. 2014. URL: `http://download.cnet.com/windows/most-popular/3101-20_4-0.html?showAll=nodeIds`.

[66] H. Hoogenstraaten and R. Prins. Black tulip report of the investigation into the diginotar certificate authority breach. 2012.

[67] S. Inc. Splunk. 2014. URL: `http://www.splunk.com`.

[68] Mandiant. Redline. 2014. URL: `https://www.mandiant.com/resources/download/redline`.

[69] Netmarketshare. Desktop operating system market share. 2014. URL: `http://www.netmarketshare.com/operating-system-market-share.aspx`.

[70] M. Russinovich and B. Cogswell. Autoruns for windows. 2013. URL: `http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx`.

[71] G. Software. Encase. 2014. URL: `http://www.guidancesoftware.com/products/Pages/encase-forensic/overview.aspx`.

[72] Statcounter. Statcounter global stats. 2014. URL: `http://gs.statcounter.com/`.

[73] Volatility. The volatility framework. 2014. URL: `https://code.google.com/p/volatility/`.