TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Computer Science

TUT Centre for Digital Forensics and Cyber Security

ITC70LT

Jaan Vahtre 143653IVCM

DETECTION OF RANSOMWARE ON WINDOWS OPERATING SYSTEMS

Master thesis

Jaan Priisalu

MSc

Early stage researcher

Tallinn 2016

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Jaan Vahtre

23.05.2016

Abstract

According to the overview of the statistics of ransomware portrayed in chapter 2.1, the number of new ransomware samples have been rapidly increasing, reaching out to more than 1,2 million new samples in the second quarter of 2015 [5] and therefore the need for a more efficient defensive strategy is crucial. Although a variety of different strategies on prevention and remediation techniques can already be implemented, the methods can be circumvented and the necessity to include an efficient detection method might be considered vital.

The aim of the thesis was to help improve the detection strategy against ransomware by analyzing different detection methods and finding the most accurate method for identifying encrypted files and detecting the running of ransomware on windows servers.

In this thesis, a theoretical analysis was conducted in order to find the most suitable detection methods, the minimal requirements for a suitable detection method were described, a secluded test environment was setup, PowerShell scripts were written which incorporated the most suitable detection methods, live samples of ransomware were gathered and a series of practical tests were implemented to be able to evaluate the different detection methods and choose the most suitable method.

As a result of the analysis, it was determined that the most accurate method for identifying ransomware on windows file shares is the Chi-square statistical data analysis algorithm. Furthermore, it was determined that the developed PowerShell script alongside with the incorporated Chi-square statistical data analysis algorithm can be used to identify encrypted files and detect the running of ransomware on a windows file share.

This thesis is written in English and is 65 pages long, including 7 chapters, 24 figures and 2 tables.

Annotatsioon

Lunavara tuvastamine Windows operatsioonisüsteemides

Vastavalt peatükis 2.1 väljatoodud statistikale, on uute lunavaranäidiste arv kiirelt tõusnud, esitades 2015. aasta teises kvartalis rohkem kui 1,2 miljonit uut lunavaranäidist [5] ning seetõttu on aktuaalne leida lahendus, piiramaks lunavara massilist levikut. Kuigi on olemas erinevaid lunavara eemale hoidmise ja andmete taastamise strateegiaid, ei ole piisavalt rõhku pandud lunavara avastamise strateegiate välja kujundamisele ning seetõttu jääb tihtipeale märkamata olukord, kus lunavara on juba eemal hoidmiseks mõeldud abinõudest mööda pääsenud.

Käesoleva diplomitöö eesmärgiks oli parandada lunavaravastast tuvastusstrateegiat, analüüsides erinevaid avastamise meetode ja valides välja parimad lahendused, mis aitaks tuvastada krüpteeritud faile ning avastada lunavaraga nakatunud serverit.

Käesoleva diplomitöö raames kirjeldati tuvastusmeetodi minimaalsed funktsionaalsed ja mittefunktsionaalsed nõuded, teostati teoreeritiline analüüs, mille tulemusena valiti välja 5 kõige sobivamat kandidaati, loodi eraldatud testkeskkond, arendati välja töötav programm, mis sisaldas endas väljavalitud parimaid praktikaid ja testiti väljavalitud praktikate tõhusust, kasutades selleks eelnevalt kogutud lunavara näidiseid.

Diplomitöö tulemusena nenditi, et diplomitöö käigus püstitatud eesmärk sai täidetud, kõige tõhusamaks meetodiks valiti Chi-square statistiline andmeanalüüsi algoritm ja tõestati, et diplomitöö käigus arendatud programm suudab tuvastada krüpteeritud faile ja lunavaraga nakatumist windows operatsioonisüsteemides.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 65 leheküljel, 7 peatükki, 24 joonist, 2 tabelit.

Acknowledgement

I would like to extend my gratitude to my supervisor Jaan Priisalu who provided me with a lot of good ideas and guided me to the right direction in building my thesis. In addition, I'd like to give my recognition to Jozwiak, I., Kedziora, M. and Melinska, A. for conducting a solid overview of the possibilities of detecting encrypted files with statistical data analysis algorithms, Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., Kirda, E. for a concisive look under the hood of ransomware attacks, and John Walker, for building a program that provided me with the necessary calculations of the statistical data analysis algorithms. Furthermore, I'd like to give my appreciation to Bart Kuppens and all of the other contributors that are looking for ways to improve our systems by developing different powershell scripts and last but not least, I would also like to thank my family and friends for providing me the necessary support in taking on this endeavour.

Table of abbreviations and terms

RSA algorithm	Rivest Shamir Adleman algorithm
CTU	Counter Threat Unit
DGA	Domain Generation Algorithm
HTTP	Hypertext Transfer Protocol
AES	Advanced Encryption Standard
NTFS	New Technology File System
MFT	Master File Table
AV	Antivirus
API	Application Programming Interface
USB	Universal Serial Bus
OU	Organizational Unit
RAM	Random Access Memory
UAC	User Account Control
LAN	Local Area Network
WAN	Wide Area Network
NAT	Network Address Translation
SMB	Server Message Block
CSV	Comma Separated Value

Table of contents

List of figures

List of tables

Introduction

The development of new variants of ransomware and the devastating affect it might bring to important documents held in computers, is making it essential for businesses worldwide to figure out a comprehensive solution in order to battle the threat. Although a variety of different knowledge on prevention and remediation techniques can already be observed and implemented, those methods are not always efficient enough and the presence of an efficient ransomware detection method could be the decisive factor whether the company will be affected by the attacking ransomware.

The main focus of the thesis is to improve the detection strategy of ransomware, by finding an efficient detection method in order to protect windows systems against ransomware and provide a valuable addition to the overall defensive strategy.

In this thesis, the author introduces different patterns found in various number of ransomware, in order to perform a proper analysis on detecting it. The thesis describes the challenges in detecting ransomware and proposes the minimal functional and non-functional requirements of a suitable detection method. During the analysis different detection methods are evaluated and the most suitable methods are chosen. In the practical study part of the thesis, a program is developed which incorporates the chosen detection methods and the chosen methods are tested against real samples of ransomware. Furthermore, results are analyzed and illustrated and proposals for further development are portrayed.

The author of the thesis chose the topic due to personal experience in dealing with clients who have been affected by ransomware and due to the raising demand of making a more efficient defensive strategy against the threat.

The topic of the thesis is well-timed as the growth of various ransomware samples is on rapid increase which is also portrayed in the chapter 2.1 and new variants of ransomware are becoming more evasive and destructive than before.

Research Question: Which detection method helps in identifying ransomware on windows operating systems in the most accurate way?

The expected outcome of the thesis is a fully implemented solution, capable of the following detection activities:

- The detection method must be able to distinguish encrypted files from the ordinary files with the maximum amount of 5% of false positives.
- The detection method must be able to detect the running of ransomware on windows systems for at least 90% of the times.
- The detection method must contain the ability to make alterations in its configuration.

In order to find a suitable solution for the research question, the author of the thesis suggests the following methodology, illustrated in Figure 1.

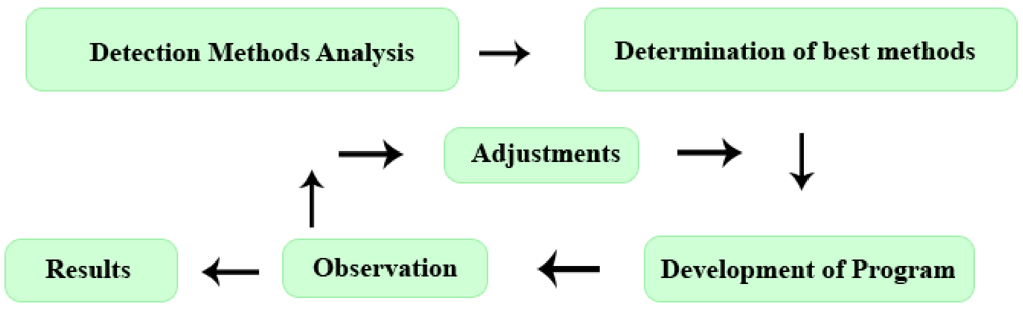


Figure 1. The methodology of the thesis

The background information and the history of ransomware are an essential part in understanding the importance of building an efficient defensive strategy against the threat. In order to battle the challenges of the thesis, an analysis on the structure of a ransomware sample is concluded and found patterns are described and inspected. Before moving to seek out the possible detection methods, the problem analysis is conducted and the minimal functional and non-functional requirements for a suitable detection method are described. Taking the aforementioned patterns as the underlying foundation, an analysis on various detection methods is completed and the most promising methods are determined taking into consideration the problem statement and the previously set minimal functional and non-functional requirements of a viable detection method.

In the practical study part of the thesis a program is developed in order to implement the chosen methods, examine and illustrate the outcome. A collected set of ransomware examples are released in a secluded test environment, and the outcome of each implementation is analyzed taking into consideration the research question and the setup of the minimal functional and non-functional requirements. The results are portrayed and discussed and most suitable method is chosen. In addition, recommendations for further development are presented.

The thesis consists of the following chapters:

Background Information – an overview of the history and the definition of ransomware is explained and also the statistical illustration on how ransomware has spread over the years is presented. In addition, a deeper insight on one of the most popular ransomware families is given.

Patterns – A list of patterns that ransomware families are dependent on which have been found in various samples of ransomware is composed.

Problem Analysis - The problem to which the thesis is trying to find a viable solution is identified and described. Furthermore, the minimal requirements of a viable solution is set on which the analysis of the practical study and choosing of the viable solution will be dependent on.

Detection Methods – Based on the list of patterns found, the possible detection methods are portrayed and analyzed. In addition, the most suitable methods, which follow the minimal requirements set in the problem analysis chapter are chosen and will be implemented in the practical study part of the thesis.

Practical Study – A secluded test environment is setup and a detection program is written, which incorporates the best chosen detection methods. Live ransomware samples are gathered and a series of tests are implemented in order to analyze the detection capabilities of each of the chosen methods. Results are analyzed and compared to the problem statement and the minimal functional and nonfunctional requirements as the most accurate detection method is chosen.

Conclusion – The goals and the outcome of the thesis is presented.

Further development – Recommendation for further development are portrayed.

1. Background information

In this chapter the definition and a brief history of ransomware is displayed. This chapter focuses on portraying what is meant by ransomware, how it is built and where did the idea come from. This chapter gives an overview of the importance to battle the threat, as it displays how quickly new ransomware samples have evolved and spread throughout the world.

1.1. Ransomware definition and statistics

Stated in the free dictionary, the definition of ransomware is: “an illegal computer software that disables a computer or blocks access to data until a payment is received [1].” Looking into the aforementioned definition, it’s visible, that ransomware can be categorized into two different types of malware. Described in a knowledgebase article by Sophos “Information on malware knows as Ransomware” the first type of ransomware is the “file encrypting” ransomware which encrypts personal files and folders, deletes the originals and displays an informative instructions for payment and getting back the files [2]. The second type of ransomware described in the article is the ‘WinLocker’ ransomware which purpose is to lock the screen and demand payment without actually encrypting any of the files [2].

In this thesis, the main focus has been set solely on the file encrypting ransomware and therefore most of the detection methods analyzed will serve the purpose of detecting encryption based processes or data.

Although the history of ransomware dates back to 1989, it wasn’t until 2011, before ransomware moved in big time [3]. According to McAfee Labs Threats Report in the first quarter of 2013, ransomware became an increasing problem during the last 2 quarters of 2011, and the situation continued to worsen [4]. In Figure 2, the number of new ransomware samples ranging from 2010 until 2013 according to the McAfee Labs threats report is shown [4]:

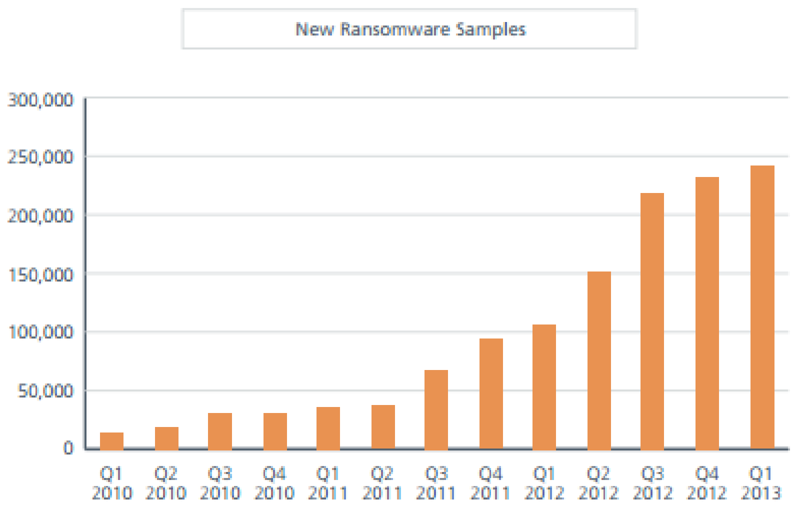


Figure 2. New ransomware samples 2010-2013

In addition to the previous figure and according to the newest McAfee Labs Threat Report in August 2015, the number of new ransomware samples have been rapidly increasing, and there isn't any indication of getting the situation under control [5]. In Figure 3 and Figure 4, the number of new ransomware samples and also the estimated total number of ransomware samples according to McAfee Labs threat report is shown [5]:

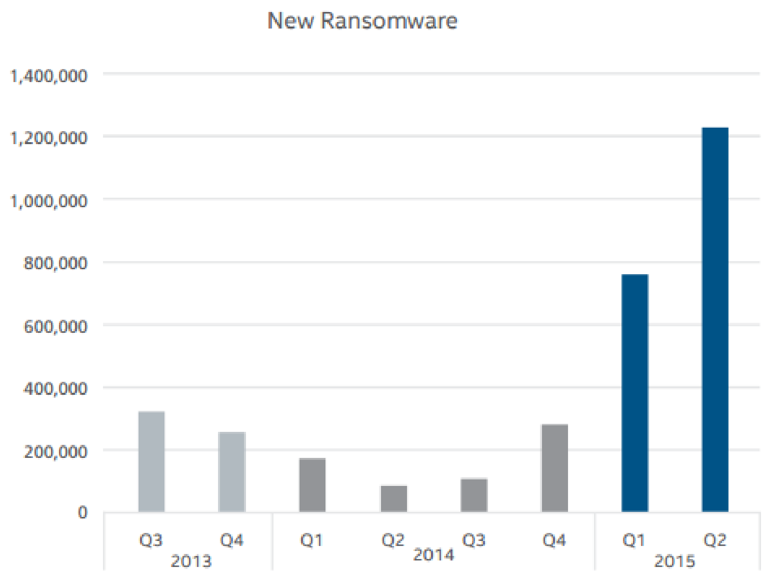


Figure 3. New ransomware samples 2013-2015

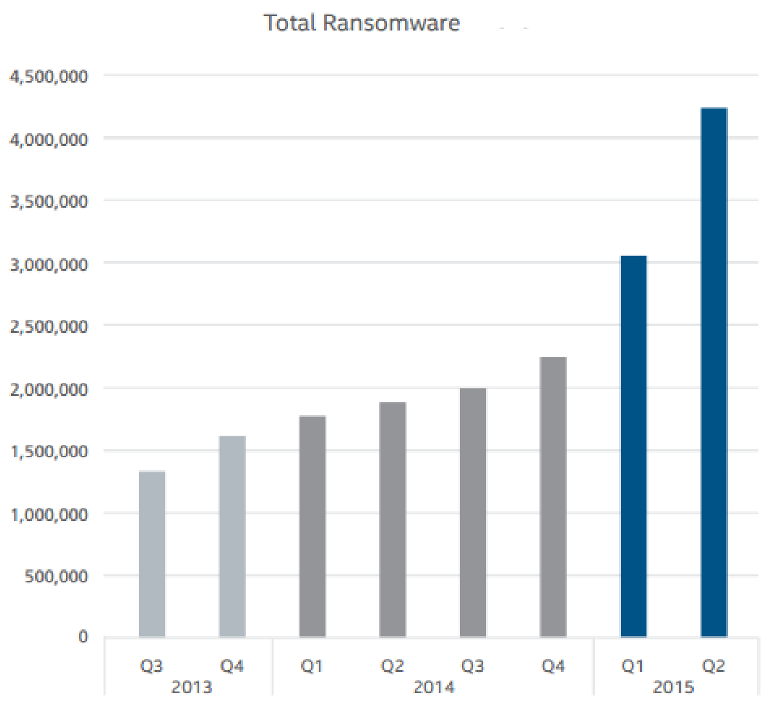


Figure 4. Total ransomware 2013-2015

Taking this into consideration, it's clearly visible that methods taken up today to prevent and stop the spread of ransomware is not enough and more concise strategies are needed.

1.2. History of ransomware

1.2.1. First documented examples

In an article by Babu Nath Giri and Nithin Jyoti "The Emergence of Ransomware", the first successful documented example of ransomware is described. According to the article, it was called PS Cyborg, also known as AIDS Trojan in 1989 [6]. The article states that the Trojan was mailed in a socially engineered package containing a floppy disk to deceive the recipients. Once installed on the system, the program replaced autoexec.bat file with one that counts the number of times the system reboots, and after 90 reboots, the Trojan began hiding directories and encrypting all the file names in the system root directory. After the successful encryption of files, a message was displayed asking the user for $378 in order to recover the lost files and directories [6].

Although this was the first documented occurrence of ransomware, it didn't take long for cyber criminals to understand the potential of the ransomware concept. According to the article, in May 2005 due to the wide use of internet, which made possible for the authors of ransomware to deliver the ransomware to a broader audience, the GP-Coder emerged [6]. The article describes GP-Coder as a Trojan, which searches and encrypts files with predetermined extensions. After the encryption process, it places a random note in each directory it had previously encrypted and asks users to e-mail to a certain address, ultimately leading to the part of asking for the ransom money in exchange for the decoder [6]. Although GP-Coder had many similarities with ransomware PS Cyborg, the focus which made it unique, was put towards enhancing the encryption algorithm [6]. As stated in the article, the variants of GP-Coder improved continuously their method of encryption, ranging from the author’s own algorithm to a much more complex RSA encryption [6].

1.3. CryptoLocker

1.3.1. Overview

The success of the continuous improvement of the encryption algorithm can be overviewed in one of the most popular families of ransomware called CryptoLocker. In an article by Keith Jarvis “CryptoLocker Ransomware” the ransomware is examined. According to the article the earliest samples of CryptoLocker appeared on September 5, 2013 and although the details about its’ initial distribution phase is unclear, the samples were downloaded from a compromised website located in the United States [7]. The article continues by describing the main distribution method of CrytoLocker which is described as spam e-mails with ZIP archives which contained a single executable with the same filename as the ZIP archive. [7] The article comprises a table with observed examples of CryptoLocker inside a distributed ZIP archive [7]. Table 1 lists several examples of CryptoLocker observed by CTU researchers [7]:

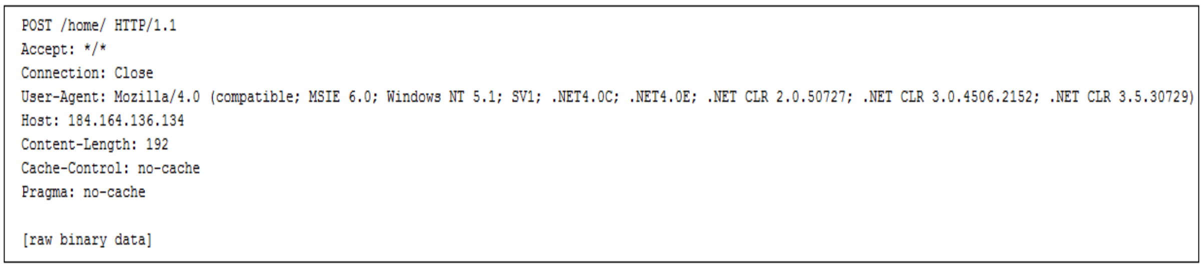
Table 1. Filenames of e-mail-delivered malware samples

Compressed archive	**Included executable file**
Jcgnbunudberrr.zip	Jcgnbunudberrr.exe
Lmpjxmvheortt.zip	Lmpjxmvheortt.exe
Icmcobxksjghdlnnt.zip	Icmcobxksjghdlnnt.exe
Gfaiqhgtqakbxlbf.zip	Gfaiqhgtqakbxlbf.exe

The article then shifts its’ focus on describing the execution and persistence phase of CryptoLocker. According to the article the ransomware hides its presence until it has a successful connection to a command and control server and is able to encrypt all the files on the connected drives [7]. In addition, the malware ensures that it remains running on the infected system and that it persists across reboots [7]. In order to achieve that, the malware creates a copy of itself in either %AppData% or %LocalAppData% folder and creates an “autorun” registry key [7]. Furthermore, the malware creates new registry keys to store the VersionInfo value, which contains configuration data and PublicKey value, which in turn contains the RSA public key received from the command and control server during the initial connection [7].

1.3.2. Network

The network connectivity between the command and control server and the ransomware is done with a domain generation algorithm (DGA) that produces 1000 potential domain addresses per day [7]. Although the article states that in some cases the ransomware also used static command and control servers embedded inside the malware, it’s not a common sight, and CryptoLocker using DGA cycles indefinitely until it finds a suitable command and control server, which it can connect to over HTTP [7]. After the initial connection is established, CryptoLocker sends an encrypted message with an RSA public key embedded within the malware [7]. Stated in the article, only servers with the corresponding RSA private key can decrypt the message and establish a successful communication with the malware [7]. In Figure 5, CryptoLocker’s initial phone-home traffic is observed [7]:



```
POST /home/ HTTP/1.1
Accept: */*
Connection: Close
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)
Host: 184.164.136.134
Content-Length: 192
Cache-Control: no-cache
Pragma: no-cache

[raw binary data]
```

Figure 5. CryptoLocker’s initial phone-home traffic

1.3.3. Encryption

CryptoLocker uses a strong third-party certified cryptography offered by Microsoft’s CryptoAPI, which makes it a robust program that is difficult to circumvent [7]. According to the article, CryptoLocker uses the “Microsoft Enhanced RSA and AES Cryptographic Provider” to create keys and to encrypt data with RSA and AES algorithms. The encryption process begins after an established connection between the malware and the command and control server [7]. The article continues by describing the encryption process. The encryption process begins by using the GetLogicalDrives() API, to get a full list of disks that have been assigned a letter, which also include USB thumb drives and external hard disks. After the successful selection of disks to attack, the malware lists all files on the selected disks that match the file patterns compiled by the attackers [7]. According to the article, the list of file patterns have changed over time, and more and more file extensions are added, in order to be able to encrypt more data [7]. After the file list has been created, each file is encrypted using a unique AES key, which in turn is encrypted with the RSA public key received from the command and control server [7]. In addition, the article states that the encrypted key, a small amount of metadata and the encrypted file contents are then written back to disk, replacing the original file. The last step according to the article is that the malware will store the location of every encrypted file in the registry, to be able to present the list of encrypted files to the user [7]. Furthermore, the article also explains that decrypting the data is not feasible and can only be recovered by obtaining the RSA private key unfortunately held in the command control server [7].

1.4. Summary

In this chapter, the background information concerning ransomware was explained. This chapter introduced the importance and necessity of finding the right answers to battle the threat. The chapter illustrated the history of ransomware, described the process of infecting a machine and encrypting users’ data and provided an illustrative statistical analysis on how ransomware has spread and become the threat it is known to be today. The next chapter will give an overview of found patterns used in ransomware that includes vital information in order to be able to build up a thorough detection strategy.

2. Patterns

In order to be able to research and evaluate the possible detection methods of ransomware, it’s essential to comprise a list of patterns that ransomware is dependent on. Furthermore, using the patterns which the ransomware depends on, provides us with valuable insight on concluding a solid detection method that the authors of ransomware cannot easily evade. In this chapter the author introduces different patterns, which will be taken under consideration and thoroughly analyzed in the upcoming chapters of the thesis.

2.1. File iteration and modification

One of the patterns that ransomware depends on, which was also described in chapter 1.3.3 is file iteration and modification. In order to perform the encryption process on a file share, ransomware makes a list of available disks and iterates through all of the files on the disks, to comprise a list of files that will be going through the encryption process [7]. As the iteration and modification of files is done in a rapid pace, it might be possible to detect the modifications of files as well as detect the making those modifications in the short period of time.

2.2. Predefined extension

In an article by Keith Jarvis, which has also been mentioned in chapter 1.3.3, it is discussed that the process of concluding the list of files to encrypt is done by iterating the file share and matching the files to a predefined list of extensions [7]. According to the article, the initial predefined number of extensions was 72, but it has been adjusted over time [7]. Taking this into consideration, the knowledge of understanding which extensions the ransomware is aiming at might be of great importance in building up a solid detection strategy.

2.3. Registry

Furthermore, according to article in chapter 1.3.1 ransomware creates different registry keys in order to automatically run the malware, ensure that the malware executes even if the system is restarted in “safe mode” and also to store additional configuration data [7]. Taking a closer look into this pattern and using known keywords with an active scanner, it’s possible to scan the registry and make notice of ransomware before it’s able to encrypt the entire file system.

2.4. File density and randomization

In addition, the type of ransomware this thesis focuses on, is built dependent on the encryption of data, which offers a crucial pattern that can be used against it. As encrypting a file also means that the density of the contents in the file rises and the randomness of data inside the file increases, it might be possible to build a detection solution with a built in calculator that would look through file shares with the intent to find files with high density. In addition to the change of density in encrypted files, it might also be possible to calculate the randomness of data inside the files using various mathematical algorithms.

2.5. Summary

In this chapter the importance of patterns in ransomware was discussed and an overview of found patterns in ransomware were given. These found patterns will be used as a foundation in finding a solid detection method and will be more thoroughly analyzed in the upcoming detection methods chapter.

3. Problem analysis

In this chapter the problem statement is taken under observation. If we look at the strategies that have been formed against ransomware today, they can mostly be summarized into four categories: prevention, detection, disruption and remediation [8]. Prevention category is clear, meaning a way to prevent the ransomware from reaching a system. Detection and disruption often work together meaning that if ransomware is detected, a disruptive action is also taken into effect. Remediation strategy could be interpreted as the art of performing proper backups. Although there's a lot of information and focus put towards the prevention and remediation strategies, the detection strategy is mostly left upon installed AV products or left completely unattended.

In this thesis, the main focus is put towards the detection strategy of ransomware, as although there is a variety of knowledge on prevention and remediation techniques, there's also new kind of ransomware being refined, that might be able to elude the preventative measures and encrypt the backup, which makes an efficient ransomware detection method the decisive factor.

The overall strategy in the defense against ransomware is lacking a proper detection method that would accurately identify encrypted files and detect the running of ransomware. This thesis focuses on finding a solution for the stated problem.

3.1. Detection method minimal requirements

In order to be able to determine viable solutions it's necessary to setup the minimal functional and nonfunctional requirements.

Functional requirements:

FR1: The detection method must be able to distinguish encrypted files from the ordinary files with the maximum amount of 5% of false positives.

FR2: The detection method must be able to detect the running of ransomware on windows systems for at least 90% of the times.

FR3: The detection method must contain the ability to make measurement alterations in its configuration.

NFR1: The detection method must not interfere with other system processes or resources.

NFR2: The detection method must support Windows operating system.

NFR3: The detection method must be freeware with an open source code.

3.2. Summary

In this chapter the problem statement was analyzed and presented. The chapter described the importance of finding a solution for the stated problem, pinpointed the focus points of the thesis and also defined the minimal functional and nonfunctional requirements for a suitable detection method. In the next chapter, viable detection methods for the stated problem will be thoroughly analyzed and the suitable solutions will be chosen.

4. Detection methods

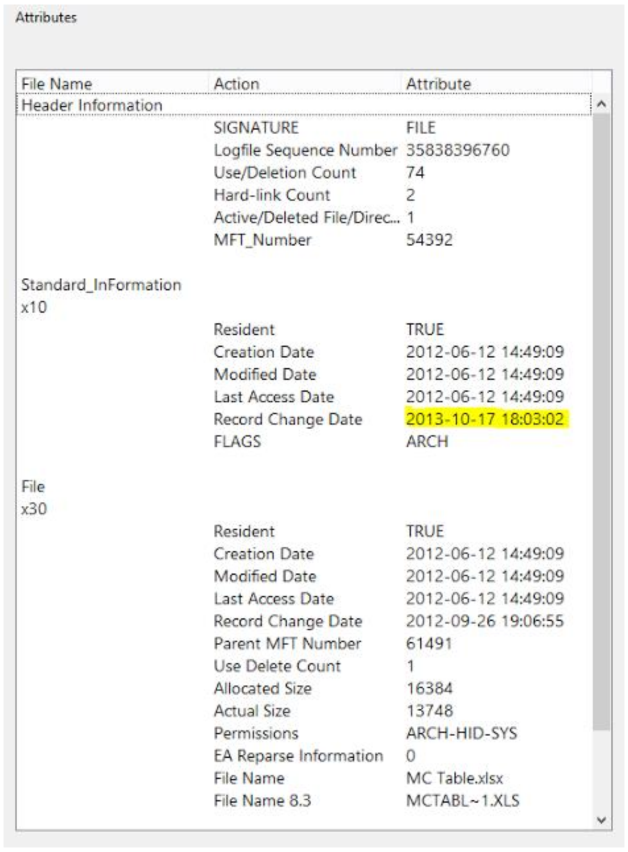
In this chapter the detection methods are portrayed and analyzed. Taking into consideration the patterns discussed in chapter 2, the thesis focuses on finding and performing a thorough analysis on suitable detection methods which use the aforementioned patterns. This chapter is portrayed as the understructure which the upcoming chapters of the thesis will depend on.

4.1. File iteration and modification

As stated in the chapter 2.1, one of the patterns that ransomware depends on is file iteration and modification. In a paper by Amin Kharraz, William Robertson, Davide Balzarotti, Layla Bilge and Engin Kirda "Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks" a long-term study of ransomware attacks has been conducted [9]. The article describes the results of analyzing 1359 samples collected between the years of 2006 and 2014 that belong to 15 different ransomware families [9]. Although the main focus of the article was to perform a thorough behavioral analysis on the collected samples, the article also provides a valuable input into the feasibility of implementing defense mechanisms against destructive ransomware attacks [9]. According to the article the analysis of the ransomware samples suggest that when an infected system is under attack, a significant change in the file system activity can be observed due to the malicious program generating a large number of similar file system access requests [9]. The article continues by stating that based on their analysis, detecting and stopping a large number of destructive ransomware attacks is not as complex as it has been previously portrayed and deploying practical defense mechanisms is feasible due to the engineering of the NTFS file system [9]. Furthermore, the article suggests that by closely monitoring the MFT table, it is possible to detect the creation, encryption and deletion of files and based on their analysis, a significant number of status changes occur in a very short period of time in MFT entries of the deleted files [9]. The article also brings out an example of their findings regarding encrypted files, where they observed a large number of MFT entries with encrypted content in the $DATA attribute of files that do not share the same path [9].

4.1.1. MFT

According to the article, in the NTFS file system, each file has an entry in the MFT that reflects the changes of the corresponding file or a folder [9]. The core file's attributes in each MFT entry are located in $STANDARD_INFORMATION attribute, and also in the $DATA attribute that contains the contents of the corresponding file [9]. The status of a file is determined both by a flag and a $BITMAP in an MFT entry [9]. $BITMAP attribute manages information about allocation status of clusters within the disk [9]. When ransomware attack occurs, the malware lists the non-system files and initiates a delete operation for each file which updates the MFT entry by changing the status flag value of the file from 0x01 to 0x00 and the $BITMAP attribute value to zero [9]. The article continues by suggesting that although when a file is deleted, the MFT entry is updated and the $BITMAP attribute is set to unallocated, the actual content of the file is not deleted immediately and is therefore salvageable [9]. The article concludes that according to their analysis, ransomware can be detected by observing the changes made in MFT table and also the content of the associated deleted files is recoverable due to the engineering of the NTFS file system [9]. Observing the results of the analysis and taking that into consideration, one possible detection method could definitely be the monitoring of changes in the MFT. The idea is also supported by another article by SecurityBrainDump „Finding Cryptolocker Encrypted Files using the NTFS Master File Table" where the author explains that during an analysis on an encrypted file shares' NFTS master file table, the $STANDARD_INFORMATION attribute's creation and modified values remained the same while the MFT entry date or the MFT record itself was modified [10]. In Figure 6, the MFT attributes of an encrypted file is shown and the changed timestamp in $STANDARD_INFORMATION is marked with yellow [10]:


Attributes

File Name	Action	Attribute
Header Information		
	SIGNATURE	FILE
	Logfile Sequence Number	35838396760
	Use/Deletion Count	74
	Hard-link Count	2
	Active/Deleted File/Direc...	1
	MFT_Number	54392
Standard_InFormation x10		
	Resident	TRUE
	Creation Date	2012-06-12 14:49:09
	Modified Date	2012-06-12 14:49:09
	Last Access Date	2012-06-12 14:49:09
	Record Change Date	2013-10-17 18:03:02
	FLAGS	ARCH
File x30		
	Resident	TRUE
	Creation Date	2012-06-12 14:49:09
	Modified Date	2012-06-12 14:49:09
	Last Access Date	2012-06-12 14:49:09
	Record Change Date	2012-09-26 19:06:55
	Parent MFT Number	61491
	Use Delete Count	1
	Allocated Size	16384
	Actual Size	13748
	Permissions	ARCH-HID-SYS
	EA Reparse Information	0
	File Name	MC Table.xlsx
	File Name 8.3	MCTABL~1.XLS

Figure 6. Encrypted file MFT attributes

Using that observation from the previous author, a PowerShell tool has been developed by the user PowerShell bear and thanks to the Get-FileTimeStamp function written by PowerShell MVP Boe Prox [11], which aim is to utilize the observation and put up an alert about a possible encrypted file when a file is found with a MFT changed time value which doesn’t match the create or modified time value [12]. Although according to the author, the program might produce false positives [12], it still gives valuable insight whether the theory of observing MFT File changes can be used with the idea of detecting ransomware on windows systems.

4.1.2. File iteration

According to the article [9], the analysis of ransomware samples concluded that the attack strategies adopted to encrypt or delete user files proved to be very similar amongst ransomware families and one of the possible approaches to detect ransomware is to develop a monitoring solution which would monitor all the file system requests that user-mode processes generate [9]. However, the article continues by implying that cybercriminals could try to evade detection by mimicking a normal user behavior and therefore using file system monitoring alone as detection method might not be a viable solution [9].

4.1.3. Decoy resources

In a paper by Brian M. Bowen, Shlomo Hershkop, Angelos D.Kermytis and Salvatore J.Stoflo "Baiting Inside Attackers Using Decoy Documents" a thorough analysis of using decoys resources also known as honeypots in a file system is introduced. Although the authors imply that the papers main focus is on human insiders attempting to exfiltrate sensitive information, the techniques described may also be utilized against external attackers [13]. According to the article honeypots are considered to have low false positive rates as they are designed to capture only malicious attackers and only some occasional mistakes triggered by an innocent user may trigger a false positive alert [13]. Nevertheless, the article proposes to use multiple overlapping signals embedded in the decoy documents to ensure proper detectability [13]. The utilization of honeypots might not be a sufficient method by itself, but understanding the potential gain in combining it with other detection methods is imperative. Using honeypot files alongside with other detection methods is most valuable, as it shortens the time it takes to detect ransomware as well as allows us to use more resource consuming detection methods, as there is no need to traverse through the whole file share.

4.2. Registry and file scanning

As described in an article by Lawrence Abrams „CryptoLocker Ransomware Information Guide and Faq" the information about the configuration of ransomware and the files it has encrypted are stored in the Windows registry by generating the necessary registry keys and values [14]. Although the article brings out only the known file paths and registry keys used by the ransomware called CryptoLocker, the statement itself provides valuable insight and a pattern which could be used to build a ransomware detection method. Furthermore, two PowerShell scripts have been conducted by Dane Kantner which purpose is to retrieve a list of machines in specified OU-s and probe each machine for evidence of CryptoLocker or CryptoWall activity [15] [16].

4.2.1. Active alerter/scanner

Although the scripts are both written by Dane Kantner and the author himself implies that the first script named “Cryptolocker/cryptowall OU scanner report” is no longer kept current for the latest variants of ransomware and the second script named “Cryptowall active alerter/scanner“ should be used [16], the script is still valuable as it uses registry scanning to identify ransomware activity. The second script from the author Dane Kantner is an upgrade for the first script, which scans through all the shares on a list of given servers and looks for files left by know variants of ransomware [16]. The second script also has an e-mail alert ability if a suspicious file is found and can be scheduled to run with overlapping times such, that one instance does not impact the other and new alerts will be synchronized between the running instances [16]. Although both of those scripts might provide the ability to detect ransomware, they are still strictly bound to certain variants of ransomware as in both cases, the variables of both the paths in registry and known filenames, have previously been set up in the aforementioned scripts and therefore they could be easily evaded by new upcoming variants of ransomware. The other reservation of this kind of approach is time, as it might take a lot of time for an infection to be discovered because of the nature of the approach. Scanning registries of remote computers or traversing through file shares definitely takes up time and as an infection might occur while the script is scanning other remote computers, it might already be too late for successfully detecting an infection.

4.3. Signature analysis

In the paper by Ireneusz Jozwiak, Michal Kedziora and Aleksandra Melinski “Theoretical and Practical Aspects of Encrypted Containers Detection” the popular and accurate methods of detecting encrypted files are discussed. One of those methods according to the paper is file signature and extension analysis [17]. According to the paper, signature analysis is the process of identifying and comparing extensions, headers and footers of files [17]. The paper gives a thorough overview of signature analysis, stating that this type of method is not based on finding encrypted data inside a file, but rather on file metadata [17]. The paper continues by describing the methods used in signature analysis, implying that searching files by specific extension is the easiest way in finding specific encrypted files but encrypted containers may have different extensions without any signature in order to mislead the detection [17]. In addition, the paper implies that although signature analysis is a powerful tool to discover encrypted files in operating systems, it does not detect the usage of encryption rather than point to a file that was created by a cryptographic tool [17]. Furthermore, the paper suggest looking into statistical algorithms, in order to confirm that a file has encrypted data inside [17].

4.4. Statistical data analysis algorithms

The paper [17] continues by explaining the importance and the accuracy of using statistical data analysis algorithms in the detection of encrypted data [17]. The paper introduces Entropy based detection and Chi-square based detection more thoroughly and also introduces Arithmetic Mean, Serial Correlation Coefficient algorithm and Monte Carlo Value for Pi algorithm.

4.4.1. Entropy

The paper [17] states that the definition of the term entropy is a measure of the disorder or randomness of the constituents of a thermodynamic system which was later adopted into computer science where it presents the measure of the uncertainty associated with a random variable or more logically defined as the information density of the contents of the file, expressed as a number of bits per character [17]. Equation 1 presents the definition of entropy H of a discrete random variable X with possible values $\{x_1, \ldots, x_n\}$, where I is the information contest of X, I(X) is a random variable and E is the expected value [17]:

$$H(X) = E(I(X)) \tag{1}$$

Equation 2 presents the equalization of entropy where p denotes the probability mass function of x and b is the base of the algorithm [17]:

$$H(X) = \sum_{i=1}^{n} p(x_i) I(x_i) == -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) \tag{2}$$

According to the paper [17] entropy value will be close to max value when the input is random data and any sign of data order will lower the entropy value [17]. The paper comprised a list of entropy tests on different file types and concluded that in their experiment the range of encrypted data was between 7,99984 to 7,99999 [17]. In Figure 7, the entropy test values of the results in the paper is displayed [17]:

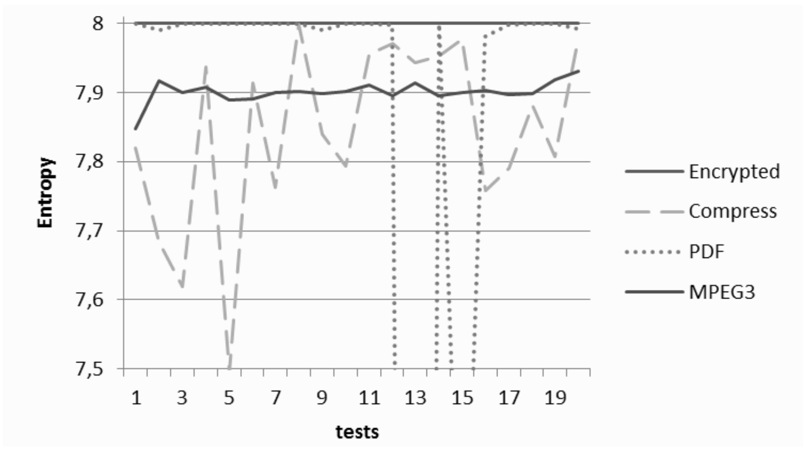


Figure 7. Results of the Entropy's test values

The article concludes, that although this kind of method might produce false positives with some compressed files, the authors of the paper didn't observe false positive hits with other file types [17]. Although it is not clear, whether their tests included .JPG format, which is known to have high density value and might produce more false positives, the idea of using entropy based detection on ransomware can also be seen by AV companies, as there is outstanding patent on detecting file encrypting malware [18]. According to one of their claims in the patent, the entropy value of a new file data is calculated after which a second entropy value for the current file data is also calculated and those results are compared to obtain a measure of difference between the current and the new file data [18]. If the difference exceeds a threshold a file encryption attack is identified [18]. Even though false positives were observed with compressed file types in the paper [17], using entropy based tests to detect ransomware seems as a viable solution, backed up by authoritative AV companies.

4.4.2. Chi-square

Another statistical data analysis algorithm named Chi-square is thoroughly discussed in the book by Donald E. Knut "The Art of Computer Programming, Semi numerical Algorithms [19] ". According to the book chi-square tests can be summarized as follows [19]: A large number of independent observations are made and the number of observations falling into each of k categories are counted and the quantity is computed, after which the value of computed quantity is compared to the values in a predefined table with v= k-1 [19]. In Appendix 1, the predefined table with the selected percentage points of the Chi-square distribution is presented [19]. According to the author, if the value is less than the 1% entry or greater than 99%, the numbers are not considered random. If the value lies between 1% and 5% entries or between 95% and 99% entries, the number are "suspect". If the value lies between 5% and 10% entries or the 90% and 95% entries, the numbers are considered "almost suspect" [19]. A good summarization of the chi square tests is also described in paper [17] as chi-square test is described as a statistical hypothesis test in which the distribution of the test is a chi-square distribution when the null hypothesis is true [17]. Furthermore, according to John Walker [20], the chi-square distribution is calculated for the stream of bytes in the file and expressed as an absolute number and a percentage which indicates how frequently a truly random sequence would exceed the value calculated [21]. The author continues by stating that the percentage is interpreted as the degree to which the sequence tested is suspect of being non-random and also provides the following chi-square test equations for calculations [21]. In equation 3, the statistics of the chi-square test for an experiment with k possible outcomes, performed n times in which Y_1,Y_2...Y_k are the number of experiments which resulted in each possible outcome, with probabilities of each outcome p_1,p_2,...p_k is presented [20]:

$$X^2 = \sum_{l<s<k} \frac{(y_s - np_s)^2}{np_s} \qquad (3)$$

According to the statement from the author [20], X^2 will be larger to the extent that the observed results diverge from those expected by chance [20]. In equation 4 the probability Q computation using the X^2 sum for the test with d degrees of freedom is presented [20]:

$$Qx^2, d = \left[2^{d/2}\Gamma\left(\frac{d}{2}\right)\right]^{-1} \int_{x^2}^{\infty} (t)^{\frac{d}{2}-1} e^{-\frac{t}{2}} dt \qquad (4)$$

Where Γ is a factorial function to complex and real arguments and is presented in equation 5 [20]:

$$\Gamma_x = \int_0^{\infty} t^{x-1} e^{-t} dt \qquad (5)$$

Taking a look at the analysis of chi-square tests done in the paper [17], the authors imply that they have performed several tests with different encrypted, compressed MPEG and PDF files, and the analysis concludes that encrypted files had chi square constant value of near 256, as other files had the value thousands or millions higher [17]. The paper concludes that Chi Square is extremely sensitive and it's an accurate way to detect encrypted files and distinguish them from ordinary files [17]. In Figure 8, the Chi-square test values of the results in the paper [17] are presented:

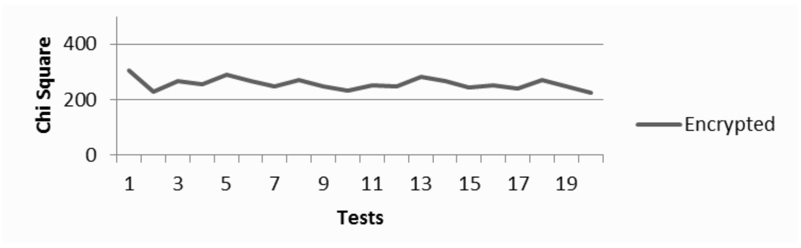


Figure 8. Results of the Chi Square encrypted files values

4.4.3. Serial Correlation Coefficient

The next statistical data analysis algorithm is Serial Correlation Coefficient, described by John Walker as a quantity that measures the extent to which each byte in the file depends upon the previous byte [21]. According to the author of the book [19], Serial Correlation Coefficient is a measure of the amount $U_j + 1$ depends on U_j [19]. In equation 6, the correlation coefficient between n quantities of $U_0, U_1 \dots, U_{n-1}$ and n quantities of others $V_0, V_1, \dots, V_{n-1}$ are defined [19]:

$$C = \frac{n\sum(U_jV_j) - (\sum U_{j)})(\sum V_j)}{\sqrt{\left(n\sum U_j^2 - \left(\sum U_j\right)^2\right)(n\sum V_j^2 - (\sum V_j)^2)}} \qquad (6)$$

The author continues by implying that all summations are to be taken over the range of 0 ≤ j ≤ n and the correlation coefficient always lies between -1 and +1 [19]. According to the author, if the value of correlation coefficient is zero or close to zero the quantities of U_j and V_j are independent of each other which can be interpreted as more random, but when the correlation coefficient is $\pm$ 1, the quantities are in total linear dependence [19]. Even though the observed number of false positives using the Serial Correlation Coefficient algorithm is not revealed in the studies, the detection method seems as a viable solution showing promising results in early pretests.

4.4.4. Arithmetic Mean

According to the paper [17] Arithmetic Mean is a simple implementation of a frequency test summing up all the bytes and dividing it by the file length in bytes [17]. The paper continues by stating that the result values should be about 127.5 for byte stream or 0.5 for bit stream for random data and any other values mean that data is not random [17]. The paper conducted a series of experiments on encrypted, compressed and other file types, and the results show that encrypted data is in conformity with random data frequency test with the value 127,5 $\pm$ 0,5, and compressed values are divergent but rarely in detection threshold of encrypted files [17]. The paper [17] also adds that MPEG files had mean values much below encryption mean with values ranging between 124 and 125. In figure 9, the arithmetic mean test values of the result are presented [17]:

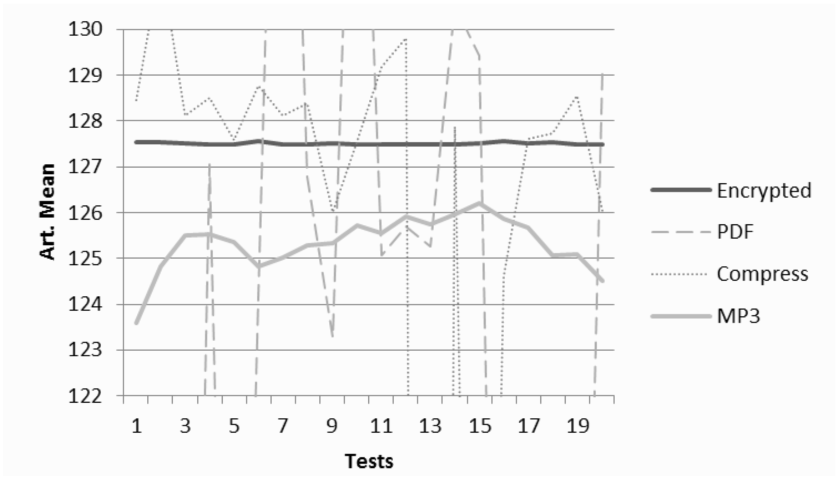


Figure 9. Results of the Arithmetic Mean test values

As the results of the analysis in paper [17] show, arithmetic mean can be used to find encrypted data and therefore might be considered as a viable solution in the detection of ransomware.

4.4.5. Monte Carlo Value for Pi

Stated in paper [17], Monte Carlo Value for Pi is an algorithm in which each consecutive 6 bytes sequence is used as 24 bit X and Y co-ordinates within a square [17]. If the distance from random point is less than the radius of a circle placed in the square, the sequence is called a “hit” as the percentage of hits can be used to calculate the value of Pi [17]. According to the article a random sequence value should be equal to Pi value 3,14159265 [17]. The article continues by implying that they have conducted a series of tests and can conclude that Monte Carlo Pi algorithm is efficient in detecting encrypted data [17]. According to their results, all encrypted files had values near Pi value of 3,14 as compressed files had clearly lower or higher values [17]. In addition MPEG and PDF files were much outside encrypted file value [17]. In Figure 10, the results of testing the Monte Carlo Pi algorithm to find encrypted data is presented [17]:

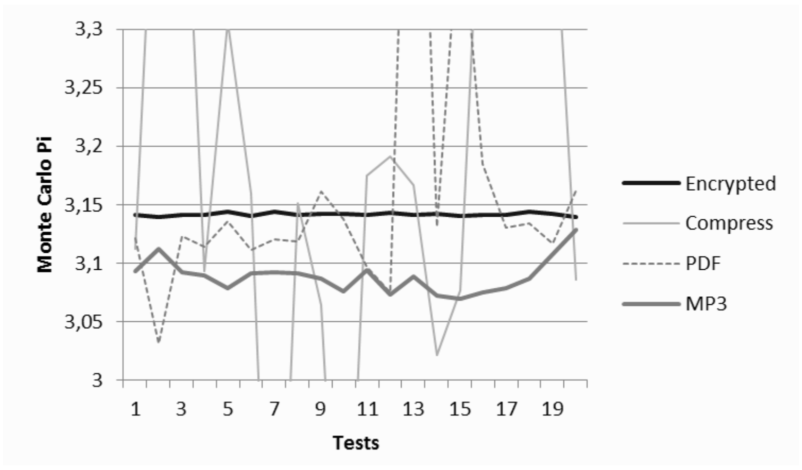


Figure 10. Results of the Monte Carlo for Pi algorithm tests

As shown in Figure 9, the test results of paper [17] in using Monte Carlo Pi algorithm to find encrypted data are promising and although the tests does not reveal the outcome of other file types, the authors of the paper concluded that Monte Carlo Pi algorithm is efficient in detecting encrypted data [17].

4.5. Evaluation of detection methods

In chapter 4, the most promising solutions of detecting ransomware on windows file shares, which used the patterns mentioned in chapter 2, were analyzed and portrayed. Chapter 4 analyzed 10 different detection methods and in Table 2, the detection methods are compared against the minimal functional and non-functional requirements which were set in chapter 3.

Table 2. The comparison of detection methods

Detection Methods	**Distinguish Encryption**	**Detect ransomware**	**Adjustable Configuration**	**Supports Windows OS**	**Free**
MFT	No	Yes	Yes	Yes	Yes
File Iteration	No	Yes	Yes	Yes	Yes
Decoy Resources	No	Yes	Yes	Yes	Yes
Registry Scanning	No	Yes	Yes	Yes	Yes
Signature Analysis	No	Yes	Yes	Yes	Yes

Entropy	Yes	Yes	Yes	Yes	Yes
Chi-Square	Yes	Yes	Yes	Yes	Yes
Arithmetic Mean	Yes	Yes	Yes	Yes	Yes
Monte Carlo for Pi	Yes	Yes	Yes	Yes	Yes
Serial Correlation Coefficient	Yes	Yes	Yes	Yes	Yes

According to the comparison in Table 2, the author of the thesis chose 5 solutions to be implemented in the practical part of the thesis: Entropy, Chi-square, Arithmetic Mean, Monte Carlo for Pi and Serial Correlation Coefficient. According to the author of the thesis, the determination of the chosen detection methods are justified with meeting the minimal requirements set in chapter 3.

Although MFT, File Iterations and Registry Scanning could also be valuable in detecting ransomware, the solutions lack the possibility of distinguishing encrypted files from clean files and therefore do not meet the minimal requirements of a proper detection method.

In addition, the signature analysis is also discarded from the practical part of the thesis, due to the fact that it does not detect the usage of encryption inside files, but it can only be used to point out that a file was created by a cryptographic tool.

4.6. Summary

In this chapter, different detection methods based on the patterns found in chapter 2 were introduced, analyzed and portrayed. This chapter introduced 10 different detection methods from file iteration, MFT and registry scanning to signature analysis and statistical data analysis algorithms. The methods found were thoroughly analyzed, evaluated and compared against the minimal requirements stated in chapter 3. Furthermore the most suitable detection methods were chosen, which will be implemented in the practical study part of the thesis.

5. Practical study

In this chapter the chosen detection methods from the previous chapter are implemented, tested and the results are analyzed. In order to implement and test the chosen detection methods, a test environment is set up, a PowerShell program is written which incorporates the chosen detection methods and live samples of ransomware are gathered.

5.1. Test environment setup

For convenient testing purposes the author of the thesis ran the testing environment on Hyper-V virtualized server which contains the possibility of making checkpoints from configurations and reverting the machine after the test have been completed. From the Hyper-V console, a separated virtual switch and a network card was assigned in order to keep the test environment secluded from the rest of the network. In addition, a virtual machine was created from the Hyper-V console, with the purpose of imitating a file server as well as a workstation and on which the further testing would be conducted on.

5.1.1. Virtual machine setup

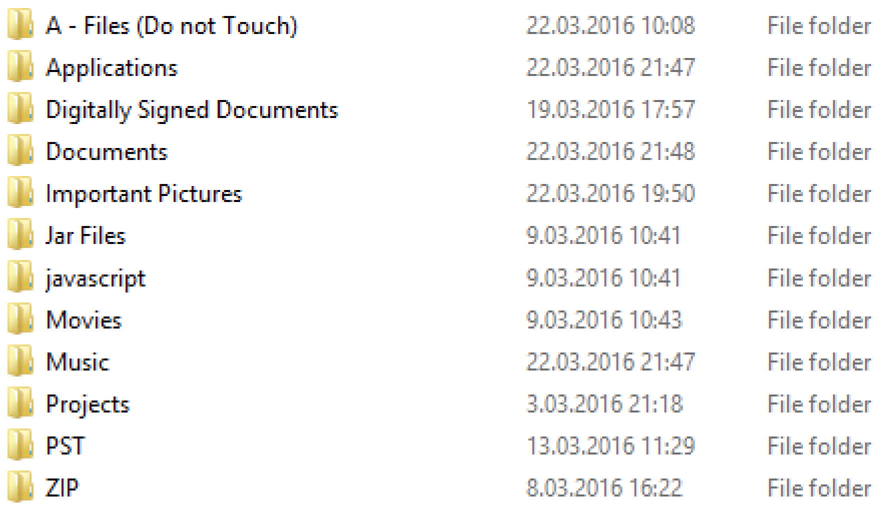
A workstation was set up on Hypev-V virtualized server, using a 64-bit Windows 8.1 image. The workstation was assigned 12GB of RAM, 200GB of data storage and 4 virtual processors and although the implementation of the mentioned detection methods do not require 12GB of RAM nor that many virtual processors, it was necessary, in order to be sure that the performance of the detection methods did not falter because of insufficient system resources. After the initial setup of the workstation basic programs were installed, in order to be able to identify the file types in the file server. Furthermore, UAC, Firewall and SmartScreen filter were disabled to make sure that nothing interferes with the live testing of ransomware. After completing the setup of the virtual machine, a separated network on a router was configured.

5.1.2. Router setup

The separated network was configured on Mikrotik CCR1009-8G-1S-1S+ model router, with a personal LAN subnet and a WAN address. As implied in article [9], internet access was controlled via NAT and network traffic was allowed to enable command and controls communication.

5.1.3. File server setup

The file share was initiated in a separate folder share and added to the workstation as a mapped drive to imitate the usage of a real file share. The file share consisted of 24697 files with various file types: bdoc, cdoc, docx, pdf, ppt, raw, xlsx, exe, jpg, png, mp4, jar, js, avi, mov, mp3, pst, zip, txt, wav. The file types were chosen by the author of the thesis, by observing the most common file types used in the windows servers on which regular maintenance is conducted by the author of the thesis and also by looking over the file types that are commonly targeted by ransomware samples. The directory structure was built combining similar file types under one folder and adding a separate decoy folder to the start of the file share. For example, the documents folder contained pdf files, docx files, xlsx files and ppt files. In Figure 11, the directory structure of the file share is presented:



A - Files (Do not Touch)	22.03.2016 10:08	File folder
Applications	22.03.2016 21:47	File folder
Digitally Signed Documents	19.03.2016 17:57	File folder
Documents	22.03.2016 21:48	File folder
Important Pictures	22.03.2016 19:50	File folder
Jar Files	9.03.2016 10:41	File folder
javascript	9.03.2016 10:41	File folder
Movies	9.03.2016 10:43	File folder
Music	22.03.2016 21:47	File folder
Projects	3.03.2016 21:18	File folder
PST	13.03.2016 11:29	File folder
ZIP	8.03.2016 16:22	File folder

Figure 11. Directory structure of the file share

A large picture file was placed inside the decoy folder, as the early tests showed promising results in detecting it with all of the aforementioned detection methods. All of the other folders were filled with ordinary files, most of them downloaded from [22] and others added from finding free downloadable documents i.e. user manuals, on the internet.

5.2. Statistical data analysis program development

In order perform the implementation of testing out the possible detection methods, a program was needed that would traverse through the file share, calculate the values of statistical data analysis algorithms, detect encrypted files based on the results, block SMB access to the file share, disconnect the SMB sessions, disable the offending user as quickly as possible and portray the results of the tests. Although there were programs that met one or another requirement, the author of thesis was unable to obtain a viable solution and therefore wrote the required program as a PowerShell script, which included a premade program named ENT, written by John Walker, to calculate the values of the statistical data analysis algorithms [21]. The full outline of the script is downloadable from the following page - https://github.com/jvahtre/RDS.

5.2.1. ENT

According to the author, ENT is a program which applies various tests to sequences of bytes stored in files and reports the results of the tests [21]. It's useful for evaluating pseudorandom number generators for encryption and statistical sampling applications, compression algorithms, and other applications [21]. The program includes the possibility of using terse mode, which presents the output in a CSV format that is easily read by most of the programming languages [21]. In this thesis, ENT is embedded in the statistical data analysis program to calculate the values and present the results.

5.2.2. Logoff-DisconnectedSessions

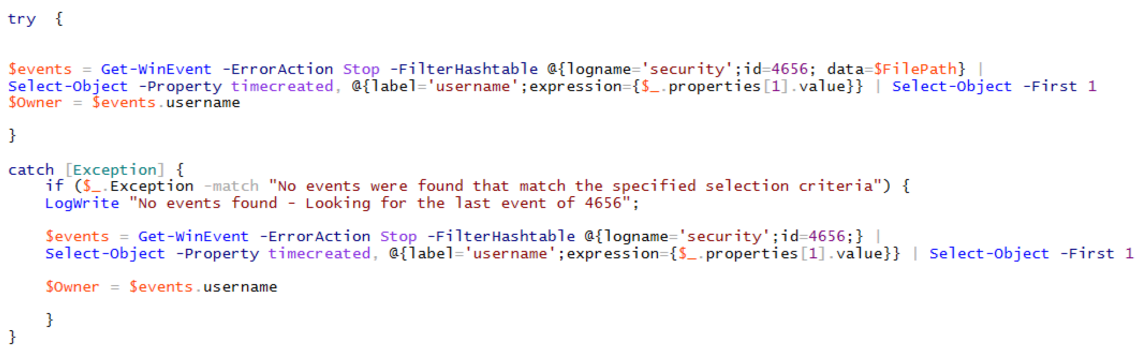
In order to be able to disconnect the detected offending users, the session of the user has to be identified. Bart Kuppens has written a PowerShell script, which aims to terminate all found disconnected sessions [23]. The author of thesis used the aforementioned script and modified it, to be able to block the SMB access, disconnect the SMB sessions and disconnect the offending user if possible. To identify the offending user, the author of the thesis enabled file system auditing and captured the event with a PowerShell Get-WinEvent command.

5.2.3. File system auditing

Unfortunately the author of thesis was unable to find an attribute which could be used to identify the user who last accessed a file and early tests showed that the owner attribute was kept unchanged when a file encryption occurred and therefore an alternative solution was deemed necessary. In the article [24] enabling file access auditing in Windows operating system is explained. In order to identify the offending user, the author of the thesis enabled file access auditing, made a new group named audited users, added the ordinary users into the specified group and changed the advanced security settings of the files to audit any alterations made to the files by the users inside the audited users group. As the script in the later stages only monitored certain folders not the whole file share, file access auditing was only enabled for the folders being monitored.

5.2.4. Get-WinEvent

After the security log was filled with events indicating alterations to the files on the file share, the next important part was identifying the important logs from the rest and also obtaining the user who last modified a certain file. As a solution to the challenge mentioned above, the author of the thesis used Get-WinEvent. In earlier stages of the program, a variable was used to identify the folder from which encrypted files were sought and after traversing through the folder, each found file’s full path and name was separately saved as a variable. As the name of the file and the full path was in a variable, it could be used alongside with the Get-WinEvent command, looking for specific events in the security log, which had an id of 4656 which according to [24] is the first event logged when a user attempts to access a file [24], and contained data with the full path of the saved file variable. After a list of events on a specific file was identified, the username responsible for the alteration was determined, the most recent event was chosen and the username affected with the specific event was saved as a different variable. In cases where the ransomware completely changed the filename and no events for particular filename were found, an exception was caught and another search was launched, without the specific filename and only the last event with the id of 4656 specified. In Figure 12, the Get-WinEvent portion of the code is shown:



```
try  {


$events = Get-WinEvent -ErrorAction Stop -FilterHashtable @{logname='security';id=4656; data=$FilePath} |
Select-Object -Property timecreated, @{label='username';expression={$_.properties[1].value}} | Select-Object -First 1
$Owner = $events.username

}

catch [Exception] {
    if ($_.Exception -match "No events were found that match the specified selection criteria") {
    LogWrite "No events found - Looking for the last event of 4656";

    $events = Get-WinEvent -ErrorAction Stop -FilterHashtable @{logname='security';id=4656;} |
    Select-Object -Property timecreated, @{label='username';expression={$_.properties[1].value}} | Select-Object -First 1

    $Owner = $events.username

    }
}
```

Figure 12. Get-WinEvent portion of the code

5.2.5. Excluded files

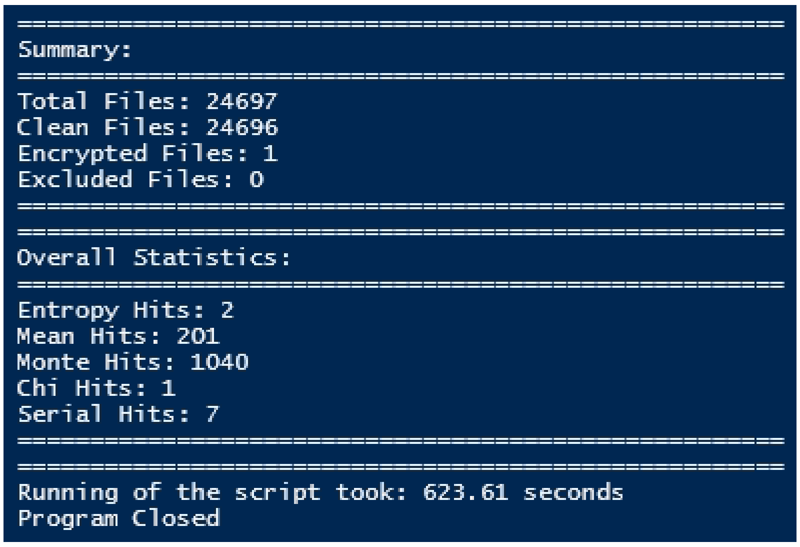
Another addition to the script is the possibility to exclude files. In chapter 3.1 the FR2 states that the detection method must be able to detect the running of ransomware on windows systems for at least 90% of the times. As a complication might develop when launching the program on a clean folder but false positives are detected, the program includes the possibility to exclude files from being analyzed. With that addition in place, the program can be tested and false positives removed before launching the tests with live samples of ransomware.

5.2.6. Building the data analysis program

The program itself consists of 3 PowerShell script files, where the first script FindEncryptedFiles.ps1 is the variable script, the second script functions.ps1 is where most of the functions are stored and last one Ent.ps1 is the script where the actual analysis of the files are performed. In the first script named FindEncryptedFiles.ps1 it's possible to change the folder where encrypted files are looked for, the path where the ENT program [21] is downloaded, choose which algorithm will be used to evaluate the files and also setup the filenames for different logs. In addition, it's possible to set whether the program will try to disconnect the offending user when an encrypted file is found or just provide the analysis results and also it's possible to provide the measurement values for the statistical data analysis algorithms. These values will be taken into consideration, when evaluating whether a specific file is encrypted or not. At the end of the first script, the program is launched in a while loop which can be modified to run for an elapsed period of time, starting the FindEncrypted function from the Ent.ps1 script again automatically in every 5 seconds after it has finished one sequence.

The second script, functions.ps1 contains EncryptedFileFound function, which purpose is to portray the encrypted file found message, timestamp how long the script ran before an encryption was found and send the results data to different logs. In addition, the script includes a Get-Sessions function, which purpose is to get the session of the offending user which was also portrayed previously in chapter 5.2.2. Additionally, the script includes the DisableUser function with the idea of identifying the offending user by looking through the security log and after a match is found, the SMB access will be blocked, the sessions will be closed and the user will immediately be disconnected.

The third script, Ent.ps1 is the main script where the actual analysis is done. The script includes one function named FindEncrypted. At the beginning of the script, path variables are set alongside with different counters. The SearchDirectory variable which includes the path and the directory for which the analysis is going to take place is set and traversed recursively, setting up a new variable for the file in question, launching the ENT program with the stored file name variable and sending the analysis result to a separated data file. From the data file, the results are imported and stored as separate objects in order to be able to evaluate each of the results. The result values are truncated accordingly and evaluated against the measurement values set in the first script FindEncryptedFiles.ps1. When an encryption is detected, the EncryptedFileFound function, Get-Sessions and the DisableUser function is launched, after which the SMB access for the user will be blocked, SMB session will be closed and the offending user will be disconnected from the file share and the data of the encryption will be sent to a separate log file. At the end of the script, a summary will be displayed, with the number of clean, encrypted and excluded files portrayed and also with the number of encrypted files that each of the statistical data analysis algorithm was able to detect. Furthermore, after an encryption is found, an e-mail is sent to the administrator, indicating that an encryption has been found and portraying the log file with the measurements taken. In Figure 13, the summary of a detection with one encrypted file on the file share with default values of the algorithms is presented:



```
====================================================
Summary:
====================================================
Total Files: 24697
Clean Files: 24696
Encrypted Files: 1
Excluded Files: 0
====================================================
====================================================
Overall Statistics:
====================================================
Entropy Hits: 2
Mean Hits: 201
Monte Hits: 1040
Chi Hits: 1
Serial Hits: 7
====================================================
====================================================
Running of the script took: 623.61 seconds
Program Closed
```

Figure 13. Summary of detection with one encrypted file and default values

5.2.7. Constraints

Altough the developed data analysis program addresses many arisen complications, in order to be able to fully understand and successfully implement the developed program with different detection methods chosen in chapter 4.5, the constraints of the developed program have to be examined and considered.

As the developed program alongside with the different statistical data analysis algorithms detect encryption inside the files, one of the complications arisen is ordinary users encrypting their own files and getting disconnected for their action. In order to circumvent the mentioned constraint, rules must be applied, determining the folders where encryption can be used and the folders on which the monitoring will take place in.

The second constraint considered is users having different shared folders connected. As portrayed in chapter 1.3.3, ransomware obtains a full list of disks and shared folders that have been assigned a letter and due to the users being connected to different shares, the share under the programs observation might not be connected to the user and will be left untouched by the ransomware. In order to circumvent the mentioned constraint, a public folder shared with every user inside the organization should be enforced, set under observation by the program and filled with different files by the users.

The third constraint of the program examined is the process of determining the offending user. In order to determine the offending user, the developed program takes the events from the security log in the event viewer, but as by default the file access auditing is not enabled, the security log does not contain the vital information in order to determine the offending user.To circumvent the aforementioned constraint, file access auditing must be enabled for the folder under observation.

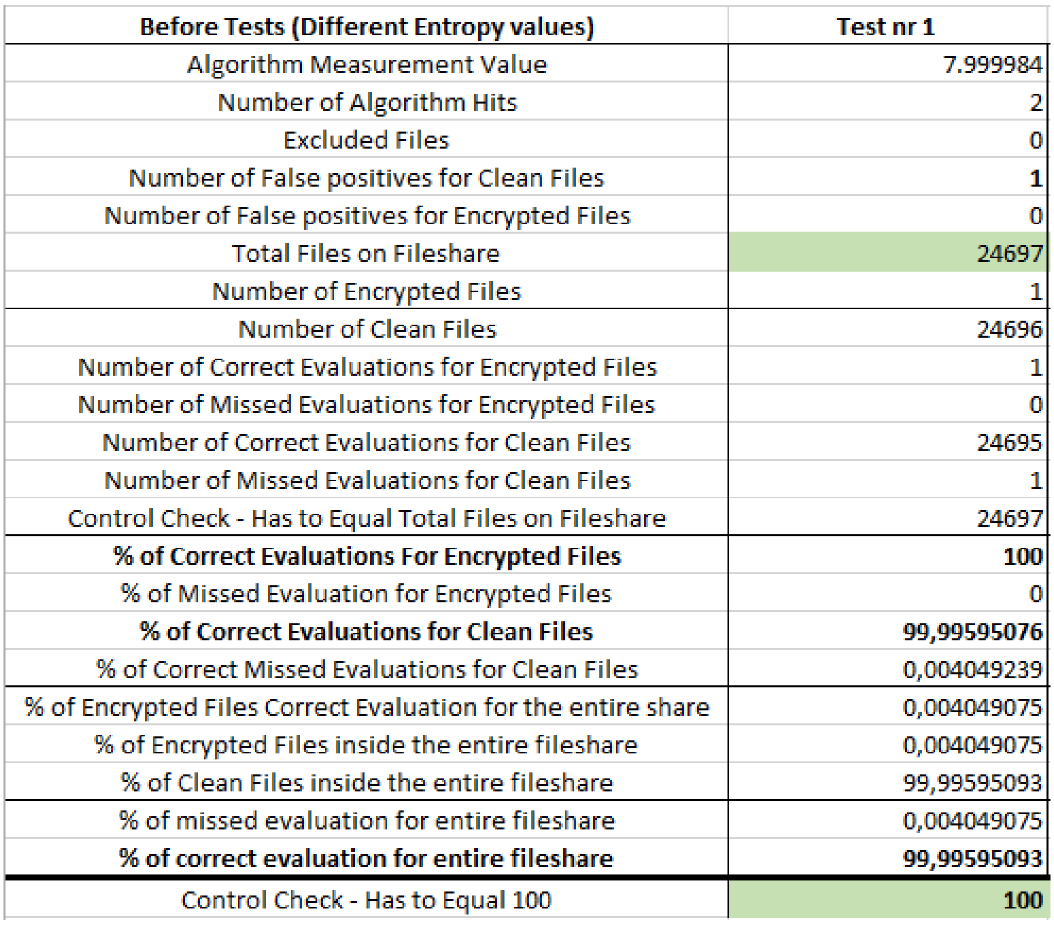
The fourth constraint of the program considered is implementing the developed program on an earlier environment than windows 8.1 environment. Although the PowerShell commands used in the developed program did not produce any complications on the selected windows 8.1 environment, the PowerShell commands are not meant for earlier releases and complication might arise when using them in alternate environments. In order to circumvent the aforementioned constraint, newer versions of windows server environments are required or equivalent PowerShell commands should be used to block and close SMB session of the user.

5.3. The overview of testing process

To be able to evaluate the chosen detection methods and their efficiency, various tests are performed, which include testing on live samples of ransomware as well as implementing tests before encryption and on different folders to understand whether the running of ransomware could be detected. Below, the author of thesis describes the different implementations, their descriptions and also the different ransomware samples used in implementing different tests.

5.3.1. Adjustment test

Before implementing the tests with live ransomware samples, it is essential to understand the number of detected false positives on the file share without the encrypted files and in addition, the limits to alterations that can be made to the measurement values that determine whether a specific file is encrypted. In light of these necessities and taking into consideration the minimal functional requirement 1 stated in chapter 3.1, a series of tests are performed with different measurement values for each of the statistical data analysis algorithm. The results of the tests portray the percentage and the number of correct and missed detections and from which the most suitable measurement values for the algorithms can be chosen and implemented in further experiments. In Figure 14, the results of the Entropy's adjustment test with default value is portrayed.



Before Tests (Different Entropy values)	**Test nr 1**
Algorithm Measurement Value	7.999984
Number of Algorithm Hits	2
Excluded Files	0
Number of False positives for Clean Files	**1**
Number of False positives for Encrypted Files	0
Total Files on Fileshare	24697
Number of Encrypted Files	1
Number of Clean Files	24696
Number of Correct Evaluations for Encrypted Files	1
Number of Missed Evaluations for Encrypted Files	0
Number of Correct Evaluations for Clean Files	24695
Number of Missed Evaluations for Clean Files	1
Control Check - Has to Equal Total Files on Fileshare	24697
% of Correct Evaluations For Encrypted Files	**100**
% of Missed Evaluation for Encrypted Files	0
% of Correct Evaluations for Clean Files	**99,99595076**
% of Correct Missed Evaluations for Clean Files	0,004049239
% of Encrypted Files Correct Evaluation for the entire share	0,004049075
% of Encrypted Files inside the entire fileshare	0,004049075
% of Clean Files inside the entire fileshare	99,99595093
% of missed evaluation for entire fileshare	0,004049075
% of correct evaluation for entire fileshare	**99,99595093**
Control Check - Has to Equal 100	**100**

Figure 14. Results of the Entropy's adjustment test with default value

After a series of tests with different measurement values have been completed, most suitable measurement values are chosen and will be implemented in live sample tests alongside with the live samples of ransomware.

5.3.2. Live sample test

Live sample tests include various live samples of ransomware executed on the test environment. Within an elapsed period of time, the detection script is triggered and when an encrypted file is detected, the SMB access of the user is blocked, the SMB session is disconnected and the user is logged off from the server, disabling the encryption process. After disabling the encryption process, the whole file share is evaluated to determine the number and the percentage of correct and missed detections with each of the statistical data analysis algorithms. The results are stored and process will be started again with a different sample.

5.3.3. Different folder test

After the end of the live sample testing, different folder testing will be implemented to understand the effectiveness of using each of statistical data analysis algorithm alongside with the composed detection program. For this type of testing, 10 random folders were selected. – A - Do Not Touch, XLSX, 015, 130, J2K, PNG, JS, AVI, MP3, PST. The security settings of the selected folder are changed to audit the users' behavior, in order to be able to distinguish the offending user, and as ransomware begins the encryption process, the program will automatically sign off the offending user after a successful detection. The results for each implementation will be stored and an overall analysis will be concluded.

5.4. Ransomware samples

The live samples used for testing the statistical data analysis algorithms were gathered from the [25] website. The author of thesis tested out various number of samples in order to make sure the samples started the encryption process and chose out 10 samples to be used in the live implementations. 2 samples were chosen because it changed the filename completely and added a new extension. The author of thesis chose those samples due to the fact that it was important to understand whether the offending user was identified if the filename was completely changed. 3 samples were chosen that changed the file extension after the encryption process and 5 samples were chosen which did not alter the filename at all. The chosen samples and their descriptions are portrayed below.

The first sample is a zipped executable named setup.exe which according to [26], was analyzed at 24.March and was detected by 4 AV products out of 56. The file is 244.9KB of size and the MD5 value of the sample is dd2ccf90555f375fb46d699432a08099 [26]. The overview of the sample from [25] is presented in Appendix 2.

The second sample is an executable with various different names. According to [26], it was analyzed in 4.April and was detected by 43 AV products out of 57. The file is 666KB of size and the MD5 value of the sample is 97512f4617019c907cd0f88193039e7c [26]. The overview of the sample from [25] is presented in Appendix 3.

The third sample is an executable with the filename 59.exe. According to [26], it was analyzed in 2.March, and was detected by 43 AV products out of 56.The file is 376KB of size and the MD5 value of the sample is b4c370efce46e7abfec0b147f3118b6e [26]. The overview of the sample from [25] is presented in Appendix 4.

The fourth sample is an executable with the original file name of Recalls1.exe. According to [26], it was analyzed in 23.March and was detected by 7 AV products out of 56. The file size is 368KB and the MD5 value of the sample is d8ff1d1e84a30d521a3f2bbbbee68492 [26]. The overview of the sample from [25] is presented in Appendix 5.

The fifth sample is an executable with the filename of 3476grb4f434r.exe. According to [26], it was analyzed in 28.March and was detected by 40 AV products out of 58. The file size is 172KB and the MD5 value of the sample is 81e85dcaf482aba2f8ea047145490493 [26].The overview of the sample from [25] is presented in Appendix 6.

The sixth sample is an executable with the original filename Bitingl.exe which according to [26] was analyzed in 23.March and was detected by 7 AV products out of 57. The file is 386KB of size and the MD5 hash value of the file is ec10753a4162dc1e0a39cae36d9a6873 [26]. The overview of the samples from [25] is presented in Appendix 7.

The seventh sample is an executable with the original filename of Anointsl.exe which according to [26] was analyzed in 27.February and was detected by 35 AV products out of 57. The file is 396KB of size and the MD5 value of the file is 15667babdcdd88ee08174a39c86b00ad [26]. The overview of the sample from [25] is presented in Appendix 8.

The eighth sample is an executable with the original filename of Proportionalityl.exe which according to [26] was analyzed in 5.March and was detected by 44 AV products out of 56. The file is 408KB of size and the MD5 value of the file is c1c6416c7f9b1a3eb260333b2f548ca2 [26]. The overview of the sample from [25] is presented in Appendix 9.

The ninth sample is an executable with the original filename of Yrsl.exe which according to [26] was analyzed in 6.March and was detected by 42 AV products out of 56. The file is 376KB of size and the MD5 value of the file is b09aca00a8dcded70eeac6ec2b497e60 [26]. The overview of the sample from [25] is presented in Appendix 10.

The tenth sample is an executable with the original filename of Reshufflingl.exe which according to [26] was analyzed in 27.March and was detected by 49 AV products out of 58. The file is 642KB of size and the MD5 value of the file is 34016905603c92a45b2de5810c3cc92c [26]. The overview of the sample from [25] is presented in Appendix 11.

5.5. Entropy implementation

The implementation of Entropy statistical data analysis algorithm was concluded in the manner stated in chapter 5.3. The first implementation was conducted with default measurement values which were set in chapter 4.4 by the paper [17] and in order to choose the right measurements for live sample tests, the author of the thesis conducted a series of adjustment tests with different measurement values.

5.5.1. Adjustment tests

The first measurement value tested was 7.999984 and the result of the test is presented in Figure 14, chapter 5.3.1. Although the test result portrayed a 99,99% accuracy when evaluating a file share with a single encrypted file, it might not might be the case when dealing with live samples of ransomware. In chapter 5.3 it was stated that the most suitable measurement values will be included in the live sample tests an therefore, taking into consideration the FR1, stated in chapter 3, the author of the thesis altered the measurement values and conducted further tests in order to improve the detection results in further live sample implementations. Although it was stated in FR1 that the margin of errors must be less than 5%, the author of the thesis decided not to go over 2% of false positives in the algorithms' adjustment tests and leave at least an extra of 3% margin for live sample testing. In Figure 15, the results of conducted adjustment tests with different measurement values is presented:

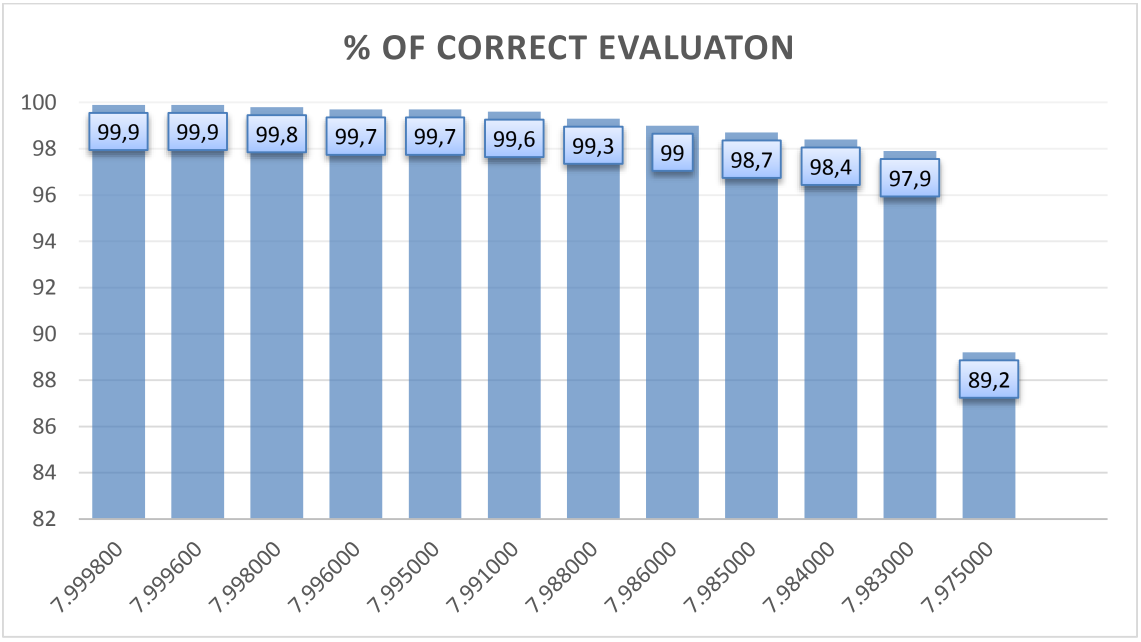


Figure 15. Results of the conducted adjustment tests with different Entropy measurement values

As shown in Figure 15 and taking into consideration the restraints set, the last possible measurement that can and was chosen to be implemented in the live tests of the thesis is 7.984000. Although the default value suggested by paper [17] provided a solid detection in the adjustment tests, the results of adjustment tests between the first and the third measurement values didn't show much difference and therefore the measurement value of 7.999800 was taken as the second measurement to be implemented in the live sample tests of this thesis. The third measurement chosen was 7.995000, which was the center measurement of the concluded adjustment test, providing 60 false positives for the entire file share, and after which the number of false positives started to escalate in a more rapid pace. The fourth and the fifth measurement value chosen was 7.986000 and 7.985000 to understand the difference in the results from the last possible measurement values. As the measurement values for the specific statistical data analysis algorithm was chosen, the next implementations were conducted which included the chosen measurements as values for the specific statistical data analysis algorithm.

5.5.2. Live sample tests

The completed live sample testing with the specified algorithm included 50 tests with 5 different measurement values and 10 different ransomware samples. After choosing out the best identified results for each sample, where the detection of clean files did not fall below the preset functional requirement 1, the live sample testing presented an average of 79% correct detection rate in identifying encrypted files and 88% correct detection rate in identifying both encrypted and the ordinary files from the entire files share. In Figure 16, the most accurate results for each of the sample in identifying encrypted files from the file share is presented:

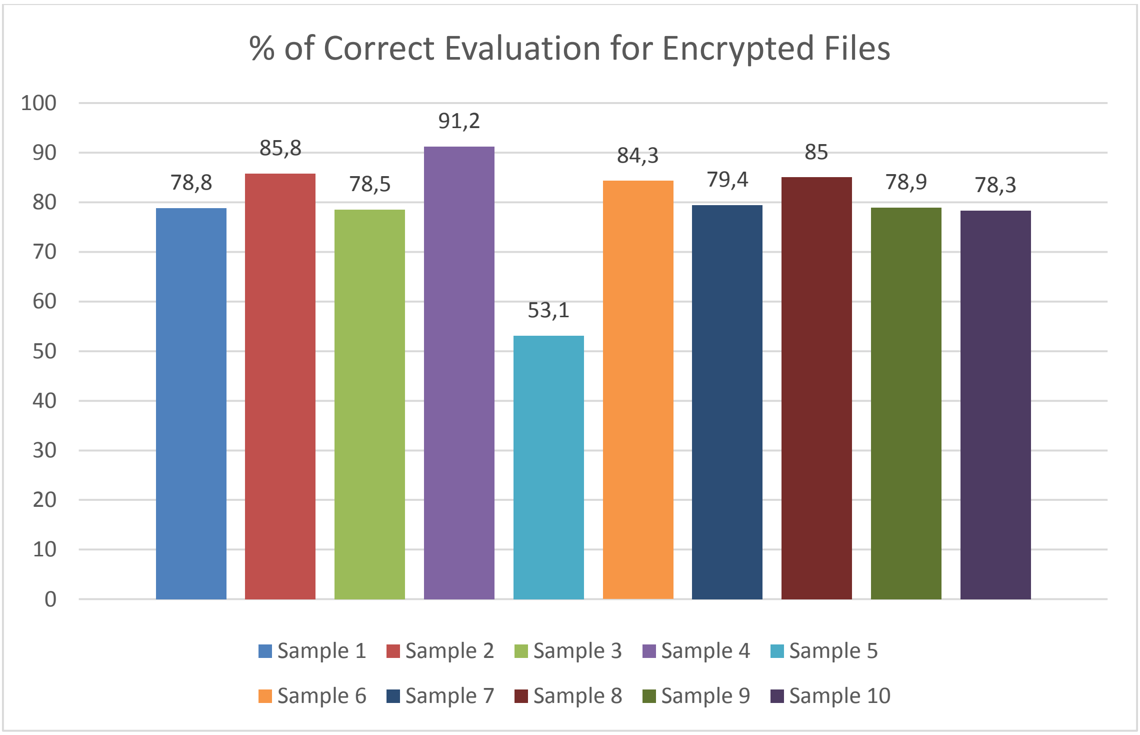


Figure 16. Results of identifying encrypted files from the file share with Entropy algorithm

As it is portrayed in Figure 16, the most accurate result in detecting encrypted files was 91%, whereas the worst result portrayed was 53%. Although the tests with bigger measurement values showed even better results when modified, the accuracy of differencing ordinary files from the encrypted, became increasingly difficult and started to fall below the present FR1 requirement.

In addition, testing out samples with different measurement values portrayed that the default value of 7.999800 proposed by paper [17] was insufficient in detecting encrypted files made by different ransomware samples and in this thesis the measurement value of 7.984000 was proven to be the most accurate measurement value which fell under accordance with the FR1 requirement in terms of evaluating clean files on the file share.

5.5.3. Different folder tests

In order to implement different folder testing with the Entropy's statistical data analysis algorithm, random live samples of ransomware from the previous implementation were chosen. Using different samples, 10 implementations were launched on the previously selected folders, to understand whether the algorithm and concluded program can be used to detect the running of ransomware on a windows file share. The results for detecting ransomware with the Entropy's statistical data analysis algorithm portrayed a 100% success, which resulted in the offending user being disconnected and the encrypted files being observed.

5.6. Chi-square implementation

The implementations with the next statistical data analysis algorithm was concluded in the similar manner than the previous one, launching the first implementation with default measurement values set in chapter 4.4 by the paper [17] and altering the measurement values in order to improve the detection results in the next phase of conducted tests.

5.6.1. Adjustment tests

The first suggested measurement value tested was 256 which portrayed a 100% accuracy in detecting encrypted files in a file share with only a single encrypted file. The next implementations with the values of 300 and 400 produced the same results, and the first difference was found when testing with the measurement value of 500, which produced an overall of 3 false positives. Choosing the measurement value of 500 as the first value to be implemented in the live sample testing, the next values chosen, 560 and 600 produced a slight increase of false positives, until a higher increase was introduced with the value of 690, which was chosen as the fourth measurement value to be tested in the live sample phase of implementations. In Figure 17, the results of the adjustment tests with different measurement values is presented:

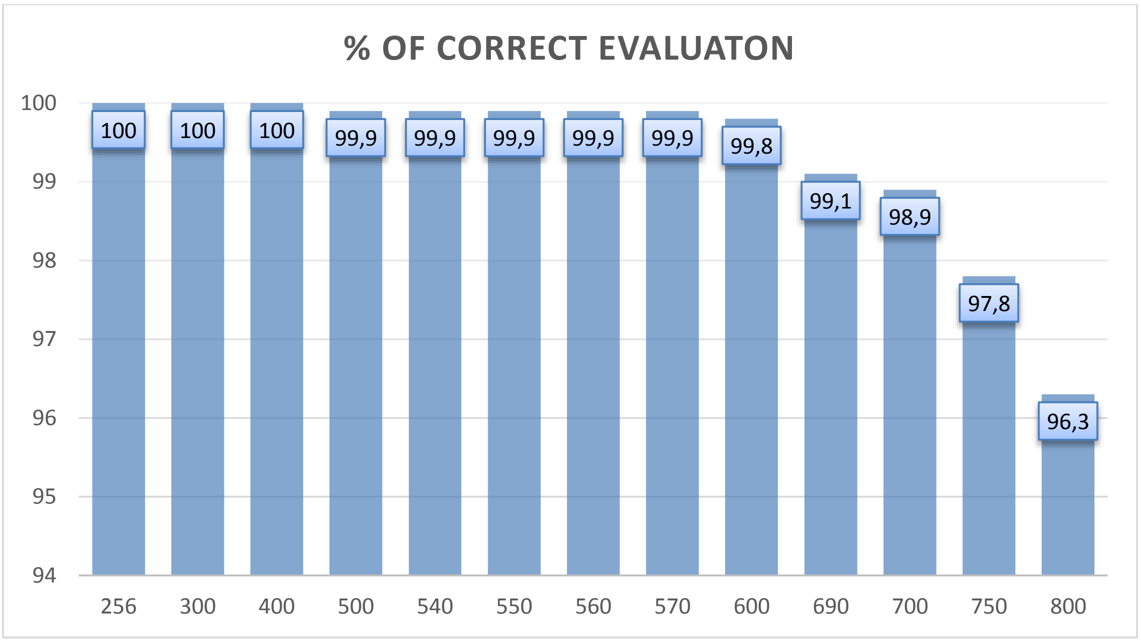


Figure 17. Results of the conducted adjustment tests with different Chi-square measurement values

As shown in Figure 17, the decline in correctly differencing encrypted files from the ordinary started with the measurement value of 690 and with the measurement value of 800, the percentage had declined significantly. As shown in Figure 17, the number of false positives with the measurement values of 750 and 800 exceeded 2% and therefore the author of thesis did not include the specified values into the live sample tests. The last measurement value included in the next phase of implementations was 700.

5.6.2. Live sample tests

The live sample tests with the Chi-square statistical data analysis algorithm included 50 tests with 10 different ransomware samples and 5 different measurement values preset in the previous chapter. After collecting the best results from each sample, where the detection of clean files did not fall below the preset functional requirement 1, the tests portrayed an average of 98,5% correct detection rate in identifying encrypted files and 99% correct detection rate in identifying both encrypted and the ordinary files from the file share. In Figure 18, the best chosen result for each of the sample in identifying encrypted files from the file share is presented:

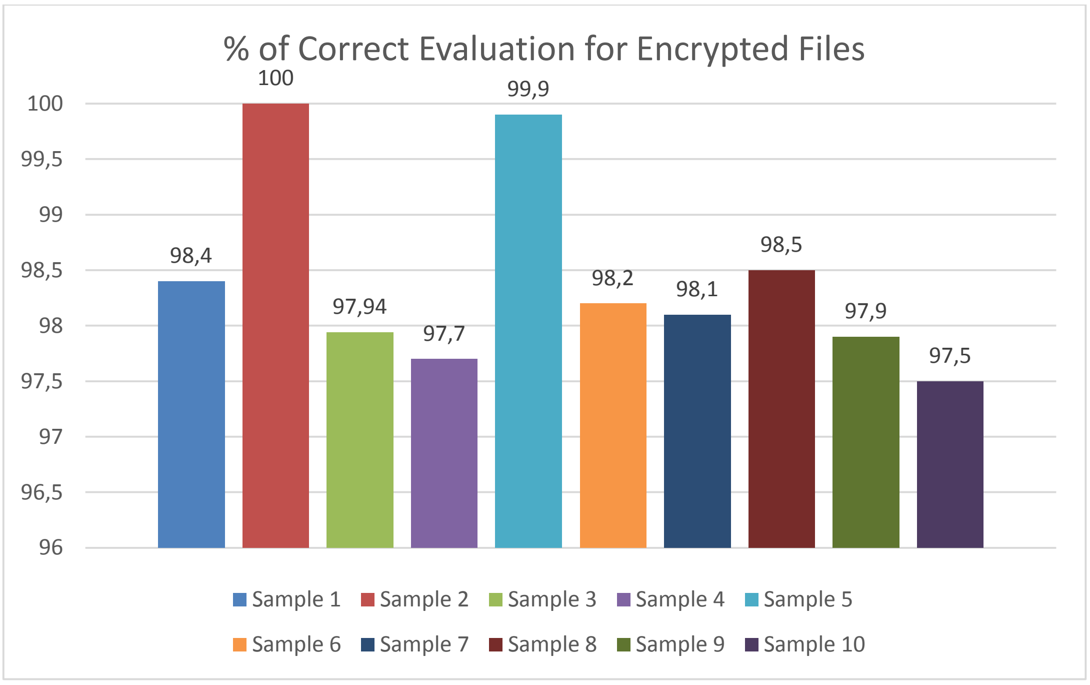


Figure 18. Results of identifying encrypted files from the file share with Chi-square algorithm

As shown in Figure 18, the most accurate result in detecting encrypted files was sample 2, with a 100% accuracy and the worst result was portrayed in sample 10, with 97,5% of accuracy. Even though the author of thesis concluded additional tests with even higher measurement values, the results of detecting encrypted files did not improve and the overall accuracy significantly decreased. Furthermore, although all the implemented tests with different measurement values showed a high detection rate, the best values were given with the measurement values of 690 and 700.

5.6.3. Different folder tests

Different folder tests with the Chi-square statistical data analysis algorithm were conducted in the similar manner as the Entropy's different folder implementations. Using different samples, 10 tests were launched on the folders selected in chapter 5.3.3 and the results showed a 100% success, which ultimately resulted in the offending user being signed off and the encrypted files being observed by the program.

5.7. Serial Correlation Coefficient implementation

The implementations with Serial Correlation Coefficient statistical data analysis algorithm were launched with the measurement value of ±0.000020, altering the measurement value in order to determine the best possible measurement values for the live sample tests concluded in the next phase of implementations.

5.7.1. Adjustment tests

The analysis of the first measurement value presented a 99,9% accuracy when evaluating a file share with a single encrypted file. As the next implementation made with a measurement value of ±0.000050 provided similar results, the first measurement value chosen to be implemented in the live sample tests was ±0.000050. The analysis of the next measurement value of ±0.000100 displayed a slight increase on the number of false positives, as a bigger change was observed when evaluating the file share with the ±0.000400 measurement value. As a result, the measurement value of ±0.000400 was chosen as the second measurement value to be tested in the live sample phase of implementations. In Figure 19, the results of the adjustment tests with different measurement values is presented:

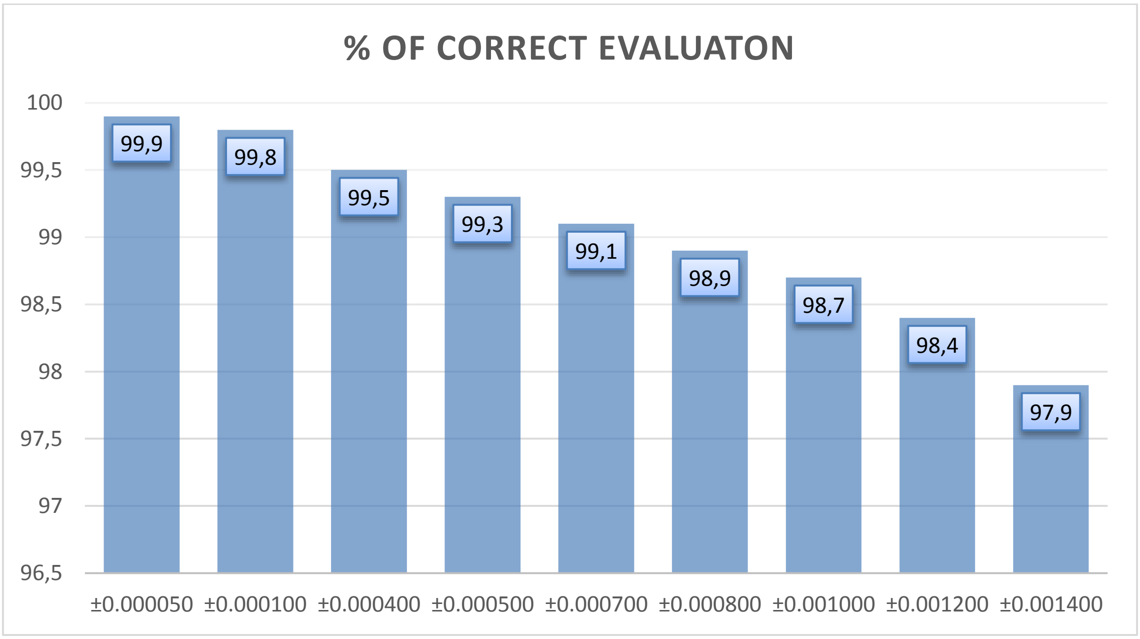


Figure 19. Results of identifying encrypted files from the file share with Serial Correlation Coefficient algorithm

As shown in Figure 19, and taking into consideration the restraints set, the last possible measurement value chosen to be implemented in the live sample tests of the thesis is ±0.001200. In addition, the values of 0.000800 and 0.001000 were also chosen to be implemented in the live sample tests of the thesis.

5.7.2. Live sample tests

The live sample tests with the Serial Correlation Coefficient statistical data analysis algorithm included 50 tests with 10 different ransomware samples and 5 different measurement values selected previously. After collecting the best results from each sample, where the detection of clean files did not fall below the preset functional requirement 1, the tests portrayed an average of 22,2% correct detection rate in identifying encrypted files and 64.4% correct detection rate in identifying both encrypted and the ordinary files from the file share. In Figure 20, the chosen result for each sample in identifying encrypted files from the file share is presented:

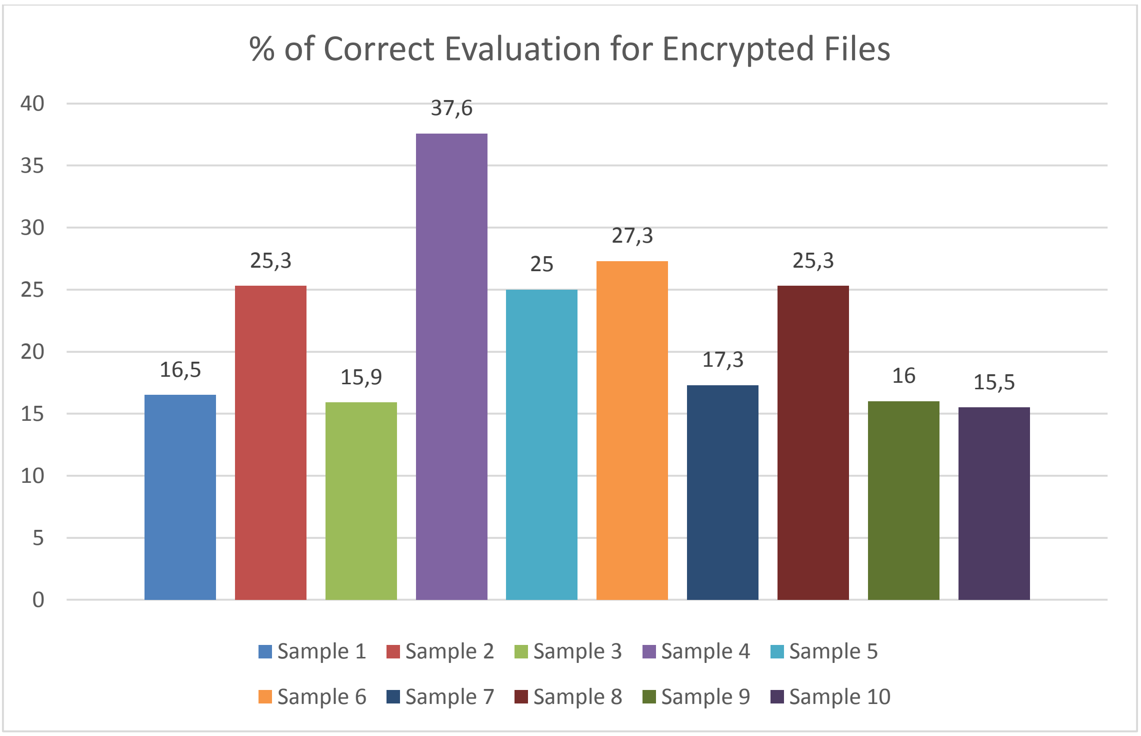


Figure 20. Results of identifying encrypted files from the file share with Serial Correlation algorithm

As shown in Figure 20, the most accurate results obtained in detecting encrypted files was sample 4, with a 37,6% accuracy and the worst results were obtained in sample 10, with 15,5% of accuracy. As the live sample tests were concluded, the author of the thesis observed the possibility of altering the measurement values even further, without violating the FR1 set in chapter 3. As the percentage of falsely identifying clean files reduced slowly, it was possible to significantly increase the accuracy of identifying encrypted files, although unfortunately the alterations were not enough as the best accuracy reached without the violation of FR1 was 55% of accuracy.

In addition, although the measurement values could have been altered even further, the best measurement value observed with live testing phase of Serial Correlation Coefficient statistical data analysis algorithm is ±0,001200.

5.7.3. Different folder tests

Different folder tests with Serial Correlation Coefficient statistical data analysis algorithm were conducted according to the setup in chapter 5.3.3 where 10 live sample tests on previously selected folders were launched and the results for each implementation was stored and analyzed. The results portrayed a 90% success in detecting the running of ransomware on windows file share and although the results in the previous chapter presented a low detection rate on identifying encrypted files, the algorithm alongside with the developed program was able to detect the running of ransomware in 90% of the tests implemented.

5.8. Arithmetic Mean implementation

The implementations with Arithmetic Mean statistical data analysis algorithm were started with the measurement value of 127.5, which was proposed by the paper [17]. After the initial test, the measurement values were altered to determine the best possible measurement values for the live sample tests in the next phase of implementations.

5.8.1. Adjustment tests

Using the first chosen measurement value of 127.5, the analysis portrayed a 99,1% accuracy of evaluating a file share with a single encrypted file. The accuracy with the next measurement value of 127.4 – 127.6 remained the same, and the first impact on the accuracy of correctly distinguishing encrypted files from the ordinary was introduced with measurement value of 127.3 – 127.7, portraying a 98.1% accuracy. In Figure 21, the results of the adjustment tests with Arithmetic Mean algorithm and different measurement values is presented:

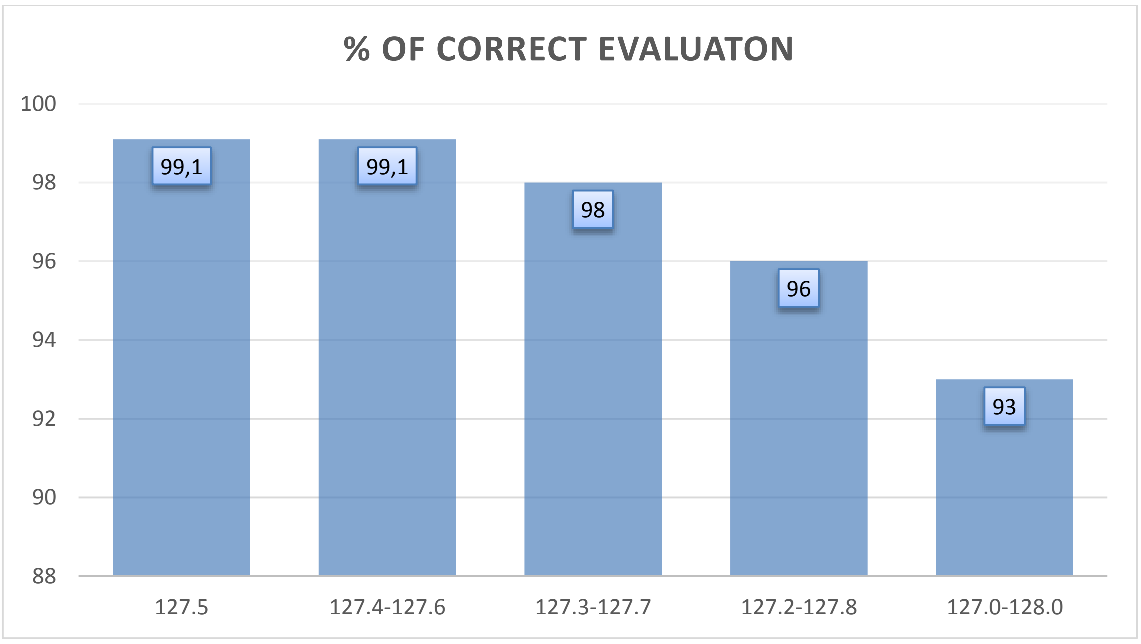


Figure 21. Results of identifying encrypted files from the file share with Arithmetic Mean algorithm

As shown in Figure 21, the decline with the measurement values of 127.2-127.8 and 127.0-128.0 was more than 2%, and therefore the author of thesis did not include the specified values into the live sample tests. The last measurement values included in the next phase of implementations were 127.4-127.6 and 127.3-127.7.

5.8.2. Live sample tests

The live sample tests with Arithmetic Mean statistical data analysis algorithm included 30 tests with 10 different ransomware samples and 3 different measurement values selected in the previous chapter. Following the selection of the most accurate results for each sample, where the detection of clean files did not fall below the preset functional requirement 1, the analysis of the results presented an average of 25,1% correct detection rate in identifying encrypted files and 68,2% correct detection rate in identifying both the encrypted and the ordinary files from the file share. In Figure 22, the selected most accurate results for each of the sample in identifying encrypted files from a windows files share is portrayed:

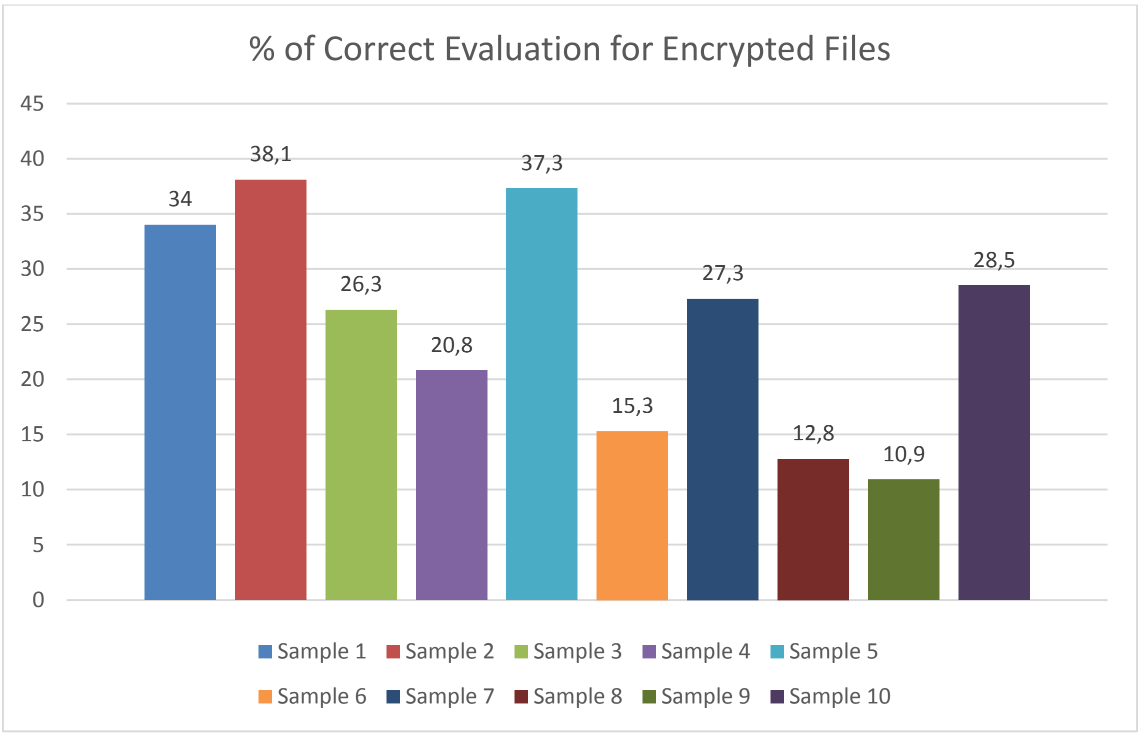


Figure 22. Results of identifying encrypted files from the file share with Arithmetic Mean algorithm

As shown in Figure 22, the most accurate results in detecting encrypted files on a windows file share was obtained in sample 2 with a 38,1% of accuracy and the worst result was encountered with sample 9, providing 10,9% of accuracy. Even though additional tests were also concluded with alternative measurement values and the result obtained showed improvement of identifying encrypted files to some extent, the analyzed results did not portray a remote possibility of meeting the FR1 requirement set in chapter 3.1 as the maximum accuracy of detecting encrypted files remained around 40%.

5.8.3. Different folder tests

Different folder tests with Arithmetic Mean statistical data analysis algorithm were conducted in the similar manner as with the previous algorithms. Using different samples, 10 implementations were launched on the folders selected in chapter 5.3.3. The results of implementing different folder test with Arithmetic Mean statistical data analysis algorithm portrayed a 100% success for every implementation, which resulted in the offending user being signed off and the encrypted files being observed.

5.9. Monte Carlo for Pi implementation

The implementations with Monte Carlo for Pi statistical data analysis algorithm were launched with the default measurement value of 3.14 which was proposed by the paper [17], after which a series of adjustment tests were conducted in order to improve the detection results in the live sample phase of implementations.

5.9.1. Adjustment tests

The first measurement value tested was 3.14, which portrayed a 95,7% accuracy when evaluating a file share with a single encrypted file. Although the accuracy was already lower than the expected 2% set by the author of the thesis, tests were carried on to understand, whether it was possible to alter the measurement values without violating the FR1, preset in the chapter 3.1. The second measurement value tested was the value between 3.13-3.15 which portrayed exactly the same results of 95,7% accuracy. In Figure 23, the results of the adjustment tests with different measurement values using Monte Carlo for Pi algorithm is presented:

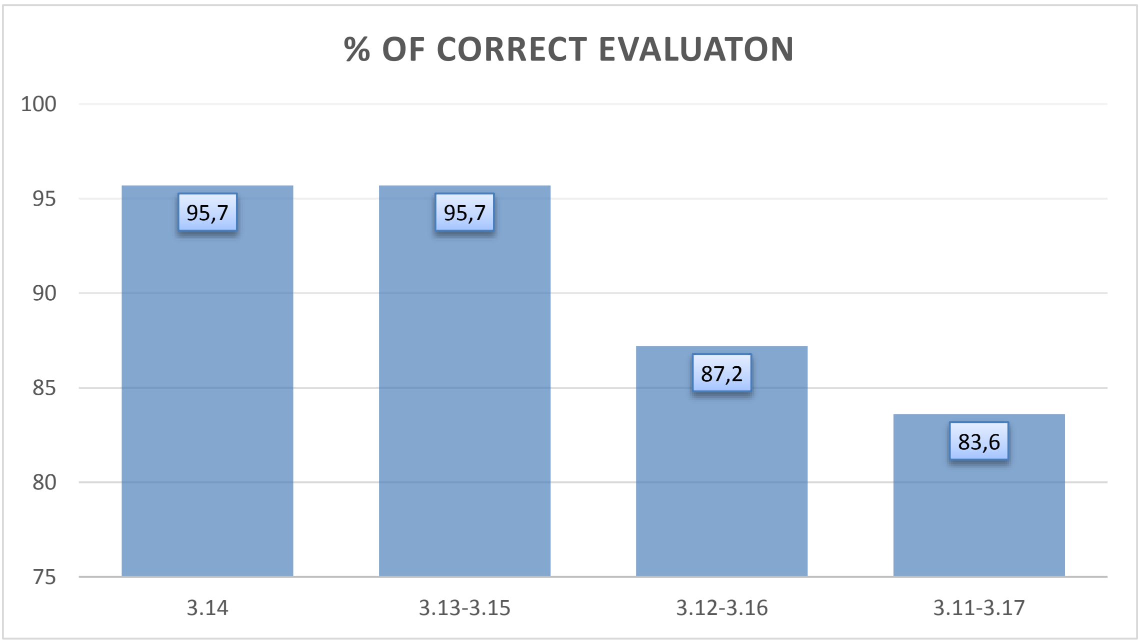


Figure 23. Results of the conducted adjustment tests with different Monte Carlo for PI measurement values

As shown in Figure 23, using the measurement values of 3.12-3.16 and 3.11-3.17 produced more than 5% of false positives and therefore violated the FR1 preset in chapter 3. As those values cannot be included in the live sample implementations, only two measurement values were selected to be included into the next phase of implementations.

5.9.2. Live sample tests

The live sample tests with the Monte Carlo for Pi statistical data analysis algorithm included 20 tests with 10 different ransomware samples and 2 different measurement values preset in the previous chapter. After the collection of the best results from each sample, where the detection of clean files did not fall below the preset functional requirement 1, the analysis portrayed an average of 27% correct detection rate in identifying encrypted files and 65,2% correct detection rate in identifying both the encrypted and the ordinary files from the file share. In Figure 20, the best chosen results for each sample in identifying encrypted files from the file share is portrayed:

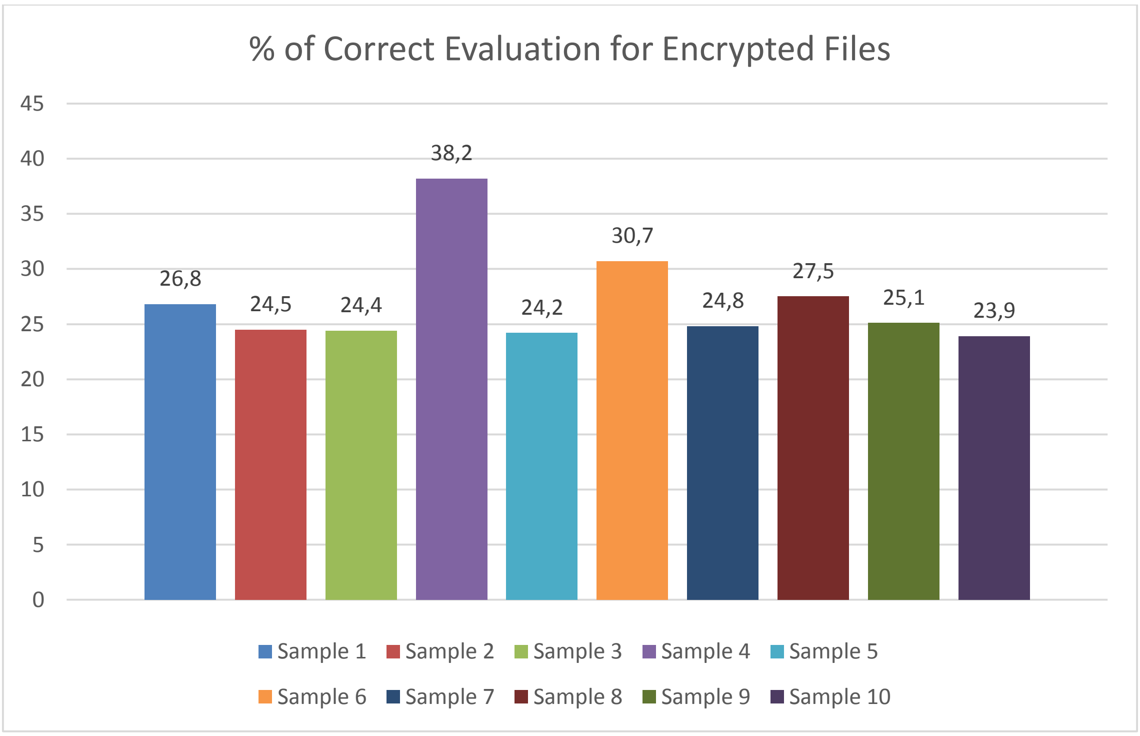


Figure 24. Results of identifying encrypted files from the file share with Monte Carlo for Pi algorithm

As shown in Figure 24, the most accurate results were portrayed by sample 4, with a 38,2% accuracy and the worst result was presented by sample 10 with a 23,9% accuracy. As the percentage of accuracy was low, the author of the thesis tried to alter the measurement values even further to understand whether it would be possible to improve the results and although the % of correctly evaluating encrypted files increased, the number of falsely identified ordinary files increased further, violating the FR1 set in chapter 3. In addition, the author of the thesis was unable to pinpoint the measurement value which produced the best results, as with many samples both of the measurement values of 3.14 and 3.13-3.15 provided exactly the same results.

5.9.3. Different folder tests

Different folder tests with the Monte Carlo for Pi statistical data analysis algorithm included 10 tests on different folders selected in chapter 5.3.3., and the analysis portrayed a 90% success in detecting the running of ransomware on the file share. Although in the previous chapter the results showed a low detection rate on identifying encrypted files on the file share, the algorithm alongside with the developed program was still able to detect the running of ransomware in 90% of the tests conducted.

5.10. Results

In the practical study part of the thesis, the author of the thesis set up a test environment, wrote a detection program, gathered live samples of ransomware and implemented series of tests in order to analyze the detection capabilities of 5 different statistical data analysis algorithms.

Taking into consideration the problem statement and the minimal requirements set in chapter 3, the author of the thesis recommends using Chi-square statistical data analysis algorithm as a viable detection method. According to the analysis done in the study part of the thesis, Chi-square statistical data analysis algorithm was the only algorithm capable of meeting every single functional and nonfunctional minimal requirement set in chapter 3. The analysis with Chi-square statistical data analysis algorithm portrayed a 98,5% correct detection rate in identifying encrypted files from a windows file share. In addition, the analysis of the results of different folder tests showed a 100% success in detecting the running of ransomware on a windows file share.

The second closest statistical data analysis algorithm was Entropy, which results portrayed a 79% correct detection rate in identifying encrypted files on a windows file share. Although the analysis of different folder tests portrayed a 100% success in detecting the running of ransomware, the algorithm was unable to meet the requirements set for FR1 and cannot be recommended to be used as a viable detection method.

The analysis of the results of the other three statistical data analysis algorithms portrayed a failure to reach 30% correct detection rate in identifying encrypted files from the ordinary and barely reached the goal in detecting the running of ransomware on windows file shares with a 90% of success rate.

In addition, after the selection of the best detection method, the author of the thesis conducted a series of validity tests, to be able to understand whether the chosen detection method could be circumvented. According to the conducted analysis, the easiest way to circumvent the chosen method, was to append 25 numbers of 0-s for 25 sequences separated by a single space in the beginning of the encrypted file which gave the file the necessary structure to produce a false analysis by the chosen detection method. In Appendix 12, the results of validity test of appending 25 numbers of 0-s for 25 sequences separated by a single space in the beginning of the encrypted file is portrayed. Although a false positive was produced by the chosen algorithm, the appending of numbers in sequences to the beginning of the file did not considerable alter the results of Entropy and Monte Carlo for Pi algorithms detection capabilities.

6. Further development

In this chapter the recommendations for further development for improving the detection strategy is presented.

As the chosen detection method provided high accuracy in identifying and detecting the running of ransomware on windows file shares, further development should focus on building a more solid program independent from windows systems.

Additionally, further development should focus on improving the performance of the program, by looking into the computational complexity and the speed of different statistical data analysis algorithms and the possibility to bind them together.

Also, as the analysis of the validity tests showed that the chosen detection method can be circumvented by adding a structure to the encrypted files, a more comprehensive program might be needed to protect against this kind of circumvention.

Furthermore, as in this thesis, the focus was merely on detection strategies and disruptive strategies were incorporated only to prove the efficiency of the detection methods, further development should focus on incorporating the detection method with different and more efficient disruptive strategies.

In addition, further development should focus on building a binding solution which would incorporate all of the best practices found in prevention, detection, disruption and also remediation strategies.

Although it might be difficult to solve the aforementioned recommendations, the author of thesis is certain that if such a binding solution is compiled, it will be more than difficult for the authors of ransomware to keep evading the binding solution and keep on spreading their composed malware.

7. Conclusion

The goal of the thesis was to help improve the detection strategy against ransomware by analyzing different detection methods and finding the most accurate method for identifying and detecting the running of ransomware on a windows file share.

The main focus of the thesis was to find an efficient detection method, in order to help protect windows systems against ransomware and provide valuable addition to the overall defensive strategy.

The problem of the thesis was analyzed and patterns that could be used for identification of ransomware were portrayed. The author of the thesis defined the functional and non-functional minimal requirements for a viable detection method and the best methods of detecting ransomware were analyzed. The author of the thesis chose 5 best methods which met the minimal functional and non-functional requirements to be implemented in the practical study part of the thesis.

In the practical study part of the thesis, a secluded test environment was setup, a detection program was written which incorporated the 5 best chosen detection methods, 10 live samples of ransomware were gathered and a series of tests were implemented in order to analyze the detection capabilities of the chosen methods. As a result of the analysis, the most accurate detection method was chosen and suggestions for further development were composed.

The goal of the thesis was to help improve the detection strategy against ransomware by finding the most suitable method for identifying encrypted files and detecting the running of ransomware on a windows file share. The problem was described, analyzed and solved in a secluded test environment through a series of tests with live samples of ransomware and a written detection program which incorporated the viable detection methods.

Thus, the goals set in this thesis are met.

References

[1] Ransomware – Definition of ransomware by The Free Dictionary. [WWW] http://www.thefreedictionary.com/ransomware (08.04.2016)

[2] Information on Malware known as Ransomware. [WWW] https://www.sophos.com/en-us/support/knowledgebase/119006.aspx (08.04.2016)

[3] A Brief History Of Ransomware. [WWW] http://blog.varonis.com/a-brief-history-of-ransomware/ (08.04.2016)

[4] McAfee Threats Report: First Quarter 2013. [WWW] http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2013.pdf (08.04.2016)

[5] McAfee Labs Threats Report: August 2015. [WWW] http://www.mcafee.com/us/resources/reports/rp-quarterly-threats-aug-2015.pdf (08.04.2016)

[6] Giri, N., Babu, Jyoti, N. The emergence of ransomware – McAfee Avert, 2006, 2. [Online] citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.169.5881&rep=rep1&type=pdf (8.04.2016)

[7] Jarvis, K. CryptoLocker Ransomware. – Threat Analysis, 2013. [Online] https://www.secureworks.com/research/cryptolocker-ransomware (08.04.2016)

[8] Buckbee, M. Using Powershell to Combat CryptoLocker. [WWW] https://blog.varonis.com/using-powershell-combat-cryptolocker/ (08.04.2016)

[9] Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., Kirda, E. Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks, 2015,1-10 [Online] http://seclab.ccs.neu.edu/static/publications/dimva2015ransomware.pdf (08.04.2016)

[10] Finding CryptoLocker Encrypted Files [WWW] http://www.securitybraindump.com/2013/11/finding-cryptolocker-encrypted-files.html (08.02.2016)

[11] Get MFT (ChangeTime) Timestamp of a file. [WWW] http://gallery.technet.microsoft.com/scriptcenter/Get-MFT-Timestamp-of-a-file-9227f399 (08.04.2016)

[12] Find-PCBCrypto: Function to aid in finding CryptoLocker encrypted files. [WWW] http://www.poshcodebear.com/blog/2014/5/2/function-to-aid-in-finding-cryptolocker-encrypted-files (08.04.2016)

[13] Bowen, B. M., Hershkop, S., Keromytis, A. D.,Stoflo, S. J. Baiting inside attackers using decoy documents. Springer, 2009

[14] CryptoLocker Ransomware Information Guide and FAQ. [WWW] http://www.bleepingcomputer.com/virus-removal/cryptolocker-ransomware-information (08.04.2016)

[15] Cryptolocker / Cryptowall OU scanner report (with auto remote registry start) [WWW] https://gallery.technet.microsoft.com/scriptcenter/Cryptolocker-report-8155ac6b (08.04.2016)

[16] CryptoWall active alerter / scanner [WWW] https://gallery.technet.microsoft.com/scriptcenter/Cryptowall-active-file-ad91b701 (08.04.2016)

[17] Jozwiak, I., Kedziora, M. and Melinska, A.: Theoretical and Practical Aspects of Encrypted Containers Detection. Digital Forensics Approach, Dependable Computer Systems. – Springer Berlin Heidelberg, Berlin, Germany, 2011; 75-85.

[18] Thure, M., Suominen, M. Detecting File Encrypting Malware: US20150058987, 2015. [Online] https://www.google.com/patents/US20150058987 (08.04.2016)

[19] Knuth, D.E. The Art of Computer Programming, Seminumerical Algorithms, vol. 2. Addison-Wesley, Reading (1969) [Online] http://e-x-a.org/stuff/books/knuth/Addison.Wesley.Donald.E.Knuth.The.Art.of.Computer.Programming.Volume.2.pdf (08.04.2016)

[20] Walker, J. Introduction to Probability and Statistics. A Pseudorandom Number Sequence Test Program, Fourmilab, 2008. [WWW] http://www.fourmilab.ch/rpkp/experiments/analysis/chiCalc.html (08.04.2016)

[21] Walker, J. Introduction to Probability and Statistics. A Pseudorandom Number Sequence Test Program, Fourmilab, 2008. [WWW] http://www.fourmilab.ch/random/ (08.04.2016)

[22] Filetypes 1. [WWW] http://digitalcorpora.org/corp/nps/files/filetypes1/ (10.03.2016)

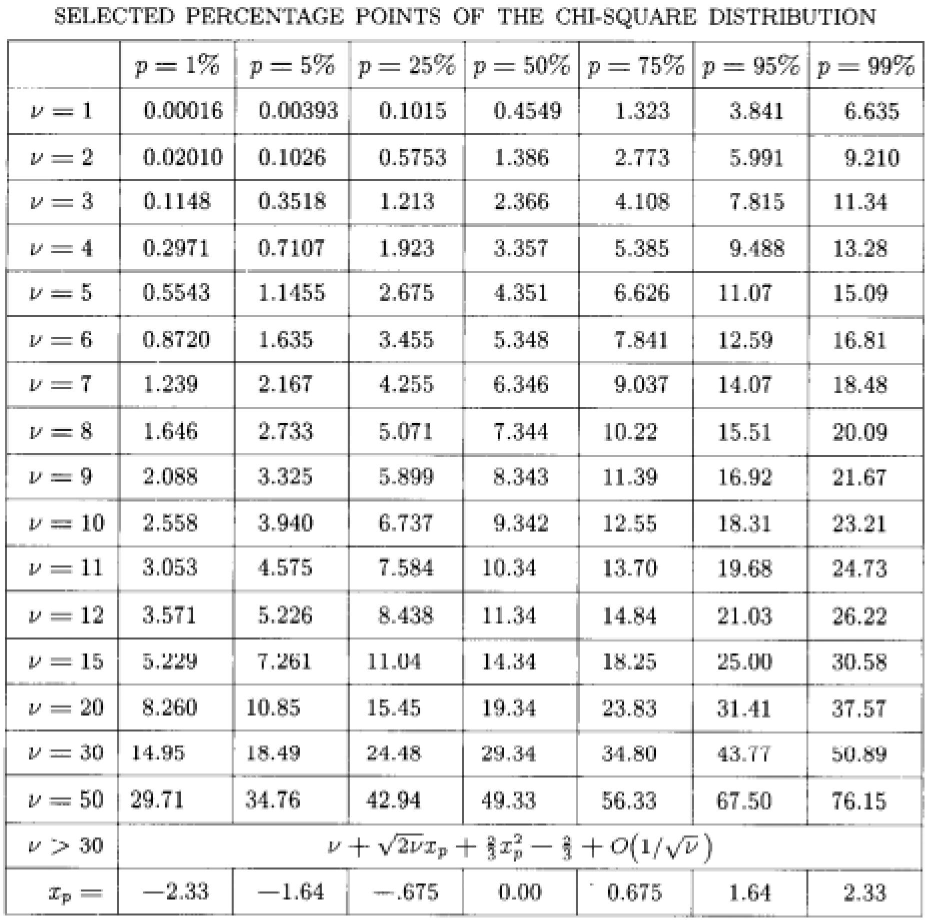
[23] Close disconnected remote session using PowerShell. [WWW] http://blog.kuppens-switsers.net/it/close-disconnected-remote-sessions-using-powershell/ (10.03.2016)

[24] Enable File Access Auditing in Windows. [WWW] http://www.morgantechspace.com/2013/11/Enable-File-System-Auditing-in-Windows.html (10.03.2016)

[25] VirusShare.com – Because Sharing is Caring. [WWW] https://virusshare.com/ (10.03.2016)

[26] Virustotal. [WWW] https://www.virustotal.com/ (08.04.2016)

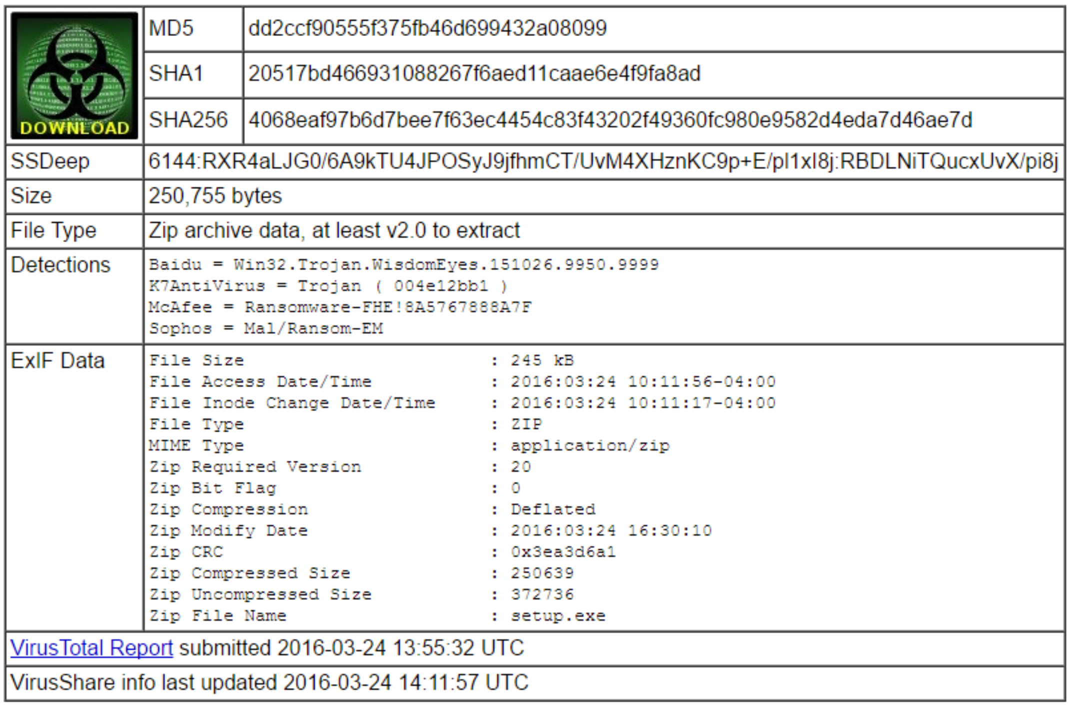
Appendix 1 – Selected percentage points of the Chi-square distribution



SELECTED PERCENTAGE POINTS OF THE CHI-SQUARE DISTRIBUTION

	$p = 1\%$	$p = 5\%$	$p = 25\%$	$p = 50\%$	$p = 75\%$	$p = 95\%$	$p = 99\%$
$\nu = 1$	0.00016	0.00393	0.1015	0.4549	1.323	3.841	6.635
$\nu = 2$	0.02010	0.1026	0.5753	1.386	2.773	5.991	9.210
$\nu = 3$	0.1148	0.3518	1.213	2.366	4.108	7.815	11.34
$\nu = 4$	0.2971	0.7107	1.923	3.357	5.385	9.488	13.28
$\nu = 5$	0.5543	1.1455	2.675	4.351	6.626	11.07	15.09
$\nu = 6$	0.8720	1.635	3.455	5.348	7.841	12.59	16.81
$\nu = 7$	1.239	2.167	4.255	6.346	9.037	14.07	18.48
$\nu = 8$	1.646	2.733	5.071	7.344	10.22	15.51	20.09
$\nu = 9$	2.088	3.325	5.899	8.343	11.39	16.92	21.67
$\nu = 10$	2.558	3.940	6.737	9.342	12.55	18.31	23.21
$\nu = 11$	3.053	4.575	7.584	10.34	13.70	19.68	24.73
$\nu = 12$	3.571	5.226	8.438	11.34	14.84	21.03	26.22
$\nu = 15$	5.229	7.261	11.04	14.34	18.25	25.00	30.58
$\nu = 20$	8.260	10.85	15.45	19.34	23.83	31.41	37.57
$\nu = 30$	14.95	18.49	24.48	29.34	34.80	43.77	50.89
$\nu = 50$	29.71	34.76	42.94	49.33	56.33	67.50	76.15
$\nu > 30$	$\nu + \sqrt{2\nu}x_p + \frac{2}{3}x_p^2 - \frac{2}{3} + O(1/\sqrt{\nu})$						
$x_p =$	−2.33	−1.64	−.675	0.00	0.675	1.64	2.33

Appendix 2 – Overview of sample 1



	MD5	dd2ccf90555f375fb46d699432a08099
	SHA1	20517bd466931088267f6aed11caae6e4f9fa8ad
DOWNLOAD	SHA256	4068eaf97b6d7bee7f63ec4454c83f43202f49360fc980e9582d4eda7d46ae7d
SSDeep	6144:RXR4aLJG0/6A9kTU4JPOSyJ9jfhmCT/UvM4XHznKC9p+E/pl1xl8j:RBDLNiTQucxUvX/pi8j	
Size	250,755 bytes	
File Type	Zip archive data, at least v2.0 to extract	

Detections

```
Baidu = Win32.Trojan.WisdomEyes.151026.9950.9999
K7AntiVirus = Trojan ( 004e12bb1 )
McAfee = Ransomware-FHE!8A5767888A7F
Sophos = Mal/Ransom-EM
```

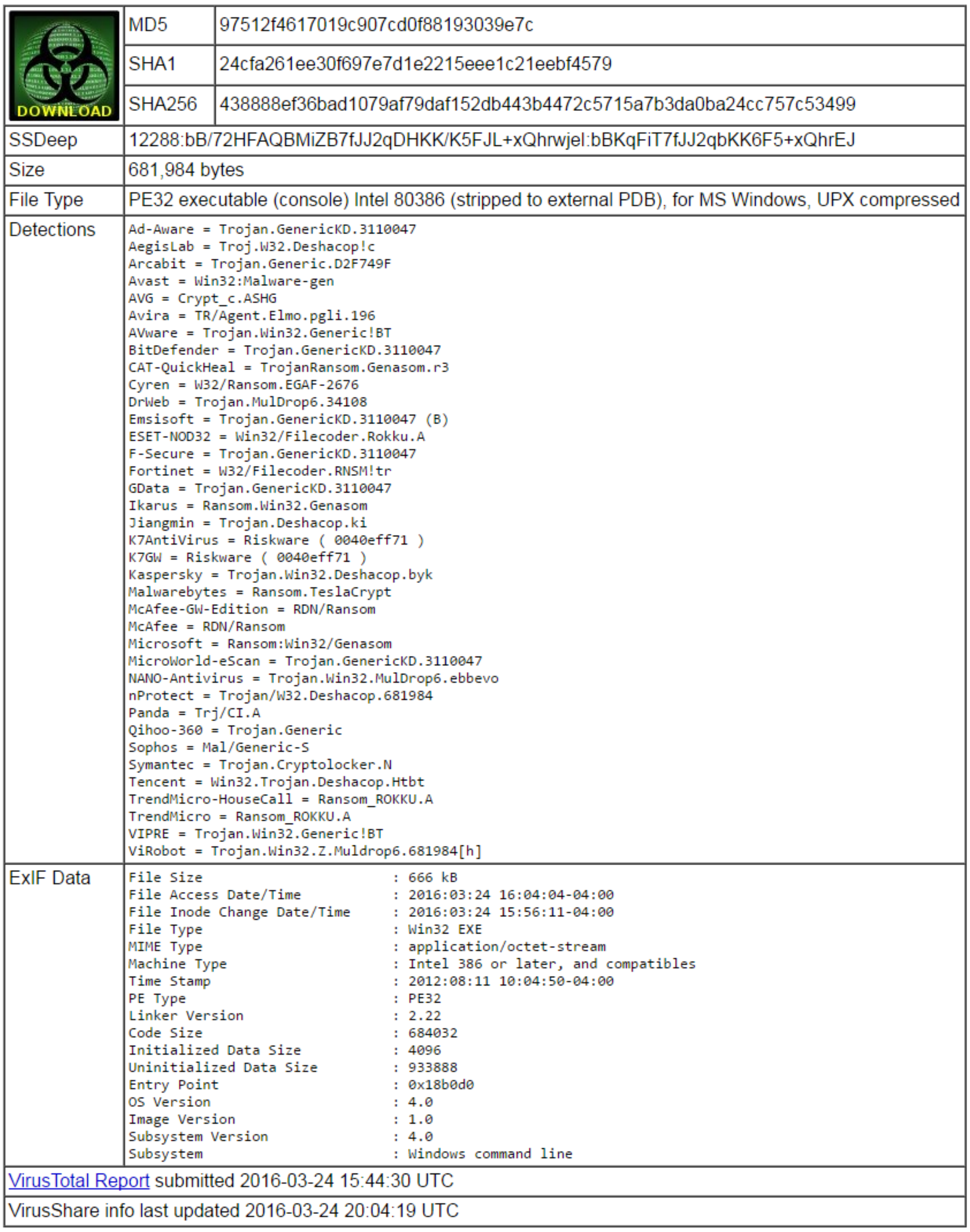
ExIF Data

```
File Size                       : 245 kB
File Access Date/Time           : 2016:03:24 10:11:56-04:00
File Inode Change Date/Time     : 2016:03:24 10:11:17-04:00
File Type                       : ZIP
MIME Type                       : application/zip
Zip Required Version            : 20
Zip Bit Flag                    : 0
Zip Compression                 : Deflated
Zip Modify Date                 : 2016:03:24 16:30:10
Zip CRC                         : 0x3ea3d6a1
Zip Compressed Size             : 250639
Zip Uncompressed Size           : 372736
Zip File Name                   : setup.exe
```

VirusTotal Report submitted 2016-03-24 13:55:32 UTC

VirusShare info last updated 2016-03-24 14:11:57 UTC

Appendix 3 – Overview of sample 2



DOWNLOAD	MD5	97512f4617019c907cd0f88193039e7c
	SHA1	24cfa261ee30f697e7d1e2215eee1c21eebf4579
	SHA256	438888ef36bad1079af79daf152db443b4472c5715a7b3da0ba24cc757c53499
SSDeep	12288:bB/72HFAQBMiZB7fJJ2qDHKK/K5FJL+xQhrwjel:bBKqFiT7fJJ2qbKK6F5+xQhrEJ	
Size	681,984 bytes	
File Type	PE32 executable (console) Intel 80386 (stripped to external PDB), for MS Windows, UPX compressed	

Detections	Ad-Aware = Trojan.GenericKD.3110047
	AegisLab = Troj.W32.Deshacop!c
	Arcabit = Trojan.Generic.D2F749F
	Avast = Win32:Malware-gen
	AVG = Crypt_c.ASHG
	Avira = TR/Agent.Elmo.pgli.196
	AVware = Trojan.Win32.Generic!BT
	BitDefender = Trojan.GenericKD.3110047
	CAT-QuickHeal = TrojanRansom.Genasom.r3
	Cyren = W32/Ransom.EGAF-2676
	DrWeb = Trojan.MulDrop6.34108
	Emsisoft = Trojan.GenericKD.3110047 (B)
	ESET-NOD32 = Win32/Filecoder.Rokku.A
	F-Secure = Trojan.GenericKD.3110047
	Fortinet = W32/Filecoder.RNSM!tr
	GData = Trojan.GenericKD.3110047
	Ikarus = Ransom.Win32.Genasom
	Jiangmin = Trojan.Deshacop.ki
	K7AntiVirus = Riskware (0040eff71)
	K7GW = Riskware (0040eff71)
	Kaspersky = Trojan.Win32.Deshacop.byk
	Malwarebytes = Ransom.TeslaCrypt
	McAfee-GW-Edition = RDN/Ransom
	McAfee = RDN/Ransom
	Microsoft = Ransom:Win32/Genasom
	MicroWorld-eScan = Trojan.GenericKD.3110047
	NANO-Antivirus = Trojan.Win32.MulDrop6.ebbevo
	nProtect = Trojan/W32.Deshacop.681984
	Panda = Trj/CI.A
	Qihoo-360 = Trojan.Generic
	Sophos = Mal/Generic-S
	Symantec = Trojan.Cryptolocker.N
	Tencent = Win32.Trojan.Deshacop.Htbt
	TrendMicro-HouseCall = Ransom_ROKKU.A
	TrendMicro = Ransom_ROKKU.A
	VIPRE = Trojan.Win32.Generic!BT
	ViRobot = Trojan.Win32.Z.Muldrop6.681984[h]	

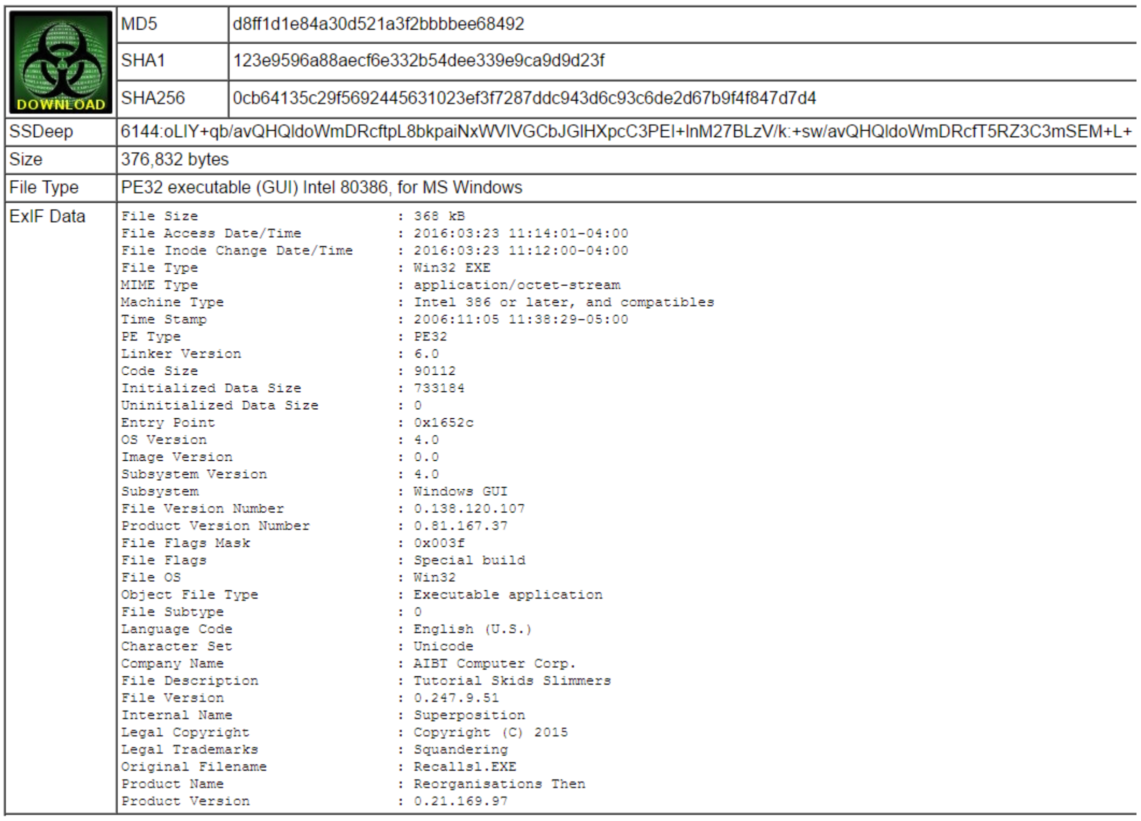
ExIF Data	File Size : 666 kB File Access Date/Time : 2016:03:24 16:04:04-04:00 File Inode Change Date/Time : 2016:03:24 15:56:11-04:00 File Type : Win32 EXE MIME Type : application/octet-stream Machine Type : Intel 386 or later, and compatibles Time Stamp : 2012:08:11 10:04:50-04:00 PE Type : PE32 Linker Version : 2.22 Code Size : 684032 Initialized Data Size : 4096 Uninitialized Data Size : 933888 Entry Point : 0x18b0d0 OS Version : 4.0 Image Version : 1.0 Subsystem Version : 4.0 Subsystem : Windows command line	
VirusTotal Report submitted 2016-03-24 15:44:30 UTC		
VirusShare info last updated 2016-03-24 20:04:19 UTC		

Appendix 4 – Overview of sample 3

MD5	b4c370efce46e7abfec0b147f3118b6e
SHA1	d08babb7e379e05f24270b005efd42c57834d6f1
SHA256	ce70e4c500aa39d8d43b1fc93909894c87b68843420b359f4017bec77292fa7d
SSDeep	6144:ie3rNhMeYq4CGRTs4kadSoKVStcmTVn57CpSCwsUbg62oXd:iY5hMfqwTsTKcmTV5kINEx+d
Size	385,024 bytes
File Type	PE32 executable (GUI) Intel 80386, for MS Windows
ExIF Data	

```
File Size                          : 376 kB
File Access Date/Time              : 2016:03:24 00:45:58-04:00
File Inode Change Date/Time        : 2016:03:24 00:43:32-04:00
File Type                          : Win32 EXE
MIME Type                          : application/octet-stream
Machine Type                       : Intel 386 or later, and compatibles
Time Stamp                         : 2008:11:30 16:44:22-05:00
PE Type                            : PE32
Linker Version                     : 6.0
Code Size                          : 249856
Initialized Data Size              : 1757184
Uninitialized Data Size            : 0
Entry Point                        : 0x3d15a
OS Version                         : 4.0
Image Version                      : 0.0
Subsystem Version                  : 4.0
Subsystem                          : Windows GUI
File Version Number                : 0.108.23.182
Product Version Number             : 0.124.246.245
File Flags Mask                    : 0x003f
File Flags                         : Special build
File OS                            : Win32
Object File Type                   : Executable application
File Subtype                       : 0
Language Code                      : English (U.S.)
Character Set                      : Unicode
Comments                           : Battlements
Company Name                       : OLYMPUS Corporation.
File Description                   : Beached Bacterial Acidrain
File Version                       : 0.247.220.105
Internal Name                      : Centralise
Legal Copyright                    : Copyright (C) 2014
Legal Trademarks                   : Assegai
Original Filename                  : Bestirl.EXE
Product Name                       : Arthur Aspersion
Product Version                    : 0.218.166.256
Special Build                      : 0.107.129.145
```

Appendix 5 – Overview of sample 4



	MD5	d8ff1d1e84a30d521a3f2bbbbee68492
	SHA1	123e9596a88aecf6e332b54dee339e9ca9d9d23f
	SHA256	0cb64135c29f5692445631023ef3f7287ddc943d6c93c6de2d67b9f4f847d7d4
SSDeep	6144:oLIY+qb/avQHQldoWmDRcftpL8bkpaiNxWVlVGCbJGlHXpcC3PEI+InM27BLzV/k:+sw/avQHQldoWmDRcfT5RZ3C3mSEM+L+	
Size	376,832 bytes	
File Type	PE32 executable (GUI) Intel 80386, for MS Windows	

ExIF Data

```
File Size                        : 368 kB
File Access Date/Time            : 2016:03:23 11:14:01-04:00
File Inode Change Date/Time      : 2016:03:23 11:12:00-04:00
File Type                        : Win32 EXE
MIME Type                        : application/octet-stream
Machine Type                     : Intel 386 or later, and compatibles
Time Stamp                       : 2006:11:05 11:38:29-05:00
PE Type                          : PE32
Linker Version                   : 6.0
Code Size                        : 90112
Initialized Data Size            : 733184
Uninitialized Data Size          : 0
Entry Point                      : 0x1652c
OS Version                       : 4.0
Image Version                    : 0.0
Subsystem Version                : 4.0
Subsystem                        : Windows GUI
File Version Number              : 0.138.120.107
Product Version Number           : 0.81.167.37
File Flags Mask                  : 0x003f
File Flags                       : Special build
File OS                          : Win32
Object File Type                 : Executable application
File Subtype                     : 0
Language Code                    : English (U.S.)
Character Set                    : Unicode
Company Name                     : AIBT Computer Corp.
File Description                 : Tutorial Skids Slimmers
File Version                     : 0.247.9.51
Internal Name                    : Superposition
Legal Copyright                  : Copyright (C) 2015
Legal Trademarks                 : Squandering
Original Filename                : Recallsl.EXE
Product Name                     : Reorganisations Then
Product Version                  : 0.21.169.97
```

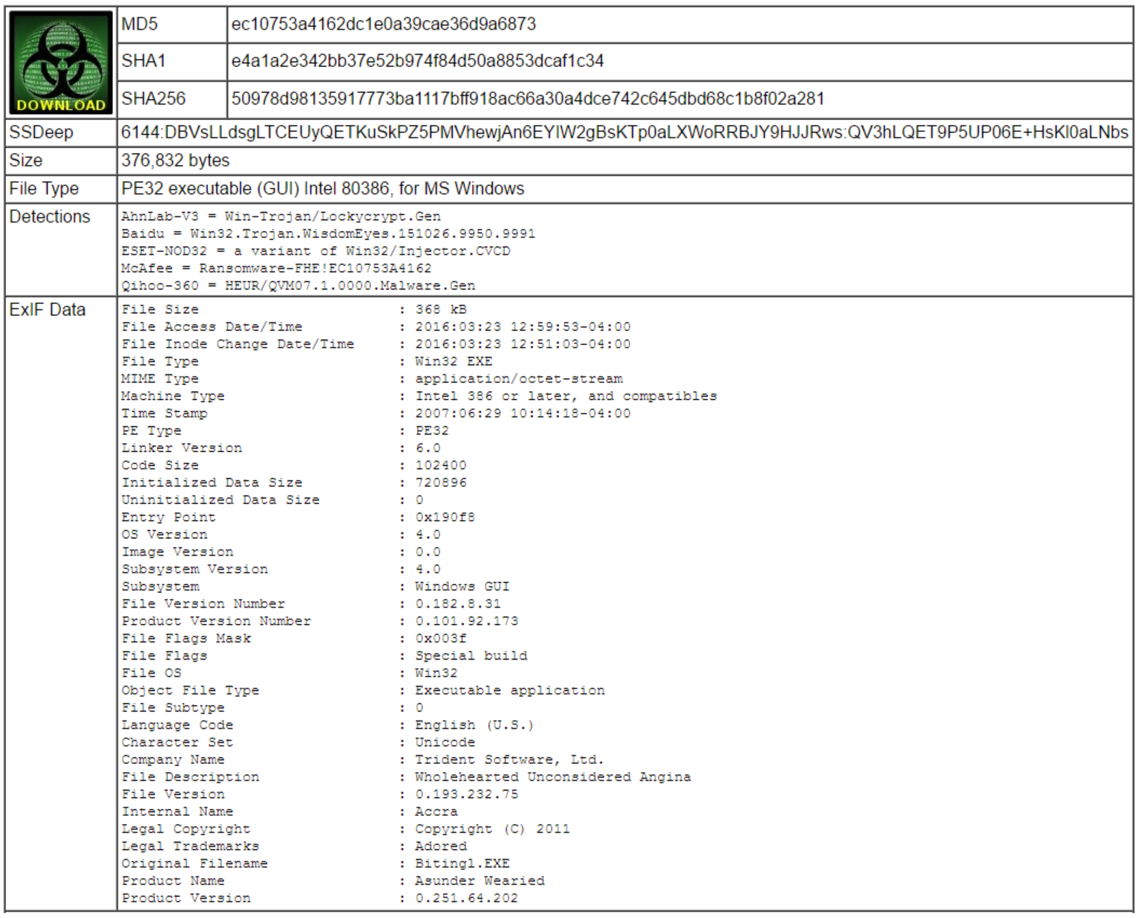
Appendix 6 – Overview of sample 5

	MD5	81e85dcaf482aba2f8ea047145490493
	SHA1	244f4f2eb2b83ff7c2672bd9b721aa02b3172869
	SHA256	9afb127e733b01952f00a8174291d39f740b6df2c90d0422b4d6f2e2e6bc7d1a
SSDeep	3072:U7Xsylq83i2mSEsSKFs+auzc5XUBOAQaH5svGX+rlaWVG4zG8vH0eH:gXsylq8y6EX+Pw5kBO5aZs+6MGCHPH	
Size	176,128 bytes	
File Type	PE32 executable (GUI) Intel 80386, for MS Windows	

Detections	AegisLab = W32.Troj.Ransom!c
	Baidu = Win32.Trojan.WisdomEyes.151026.9950.9998
	GData = Win32.Trojan-Ransom.Locky.AE
	Kaspersky = UDS:DangerousObject.Multi.Generic
	Qihoo-360 = HEUR/QVM07.1.0000.Malware.Gen
	Rising = PE:Malware.XPACK-HIE/Heur!1.9C48 [F]
	Tencent = Win32.Trojan.Raas.Auto	

ExIF Data	File Size : 172 kB File Access Date/Time : 2016:03:24 13:58:23-04:00 File Inode Change Date/Time : 2016:03:24 13:56:27-04:00 File Type : Win32 EXE MIME Type : application/octet-stream Machine Type : Intel 386 or later, and compatibles Time Stamp : 2007:09:03 17:28:52-04:00 PE Type : PE32 Linker Version : 6.0 Code Size : 77824 Initialized Data Size : 217088 Uninitialized Data Size : 0 Entry Point : 0x135a0 OS Version : 4.0 Image Version : 0.0 Subsystem Version : 4.0 Subsystem : Windows GUI File Version Number : 0.136.141.161 Product Version Number : 0.176.48.87 File Flags Mask : 0x003f File Flags : (none) File OS : Windows NT 32-bit Object File Type : Executable application File Subtype : 0 Language Code : English (U.S.) Character Set : Unicode Company Name : MicroSmarts LLC. File Description : Illusionist File Version : 241, 67, 184, 172 Internal Name : Implosion Legal Copyright : Copyright © 2018 Product Name : Groundwater Enticed Product Version : 107, 71, 55, 22	

Appendix 7 – Overview of sample 6



	MD5	ec10753a4162dc1e0a39cae36d9a6873
	SHA1	e4a1a2e342bb37e52b974f84d50a8853dcaf1c34
	SHA256	50978d98135917773ba1117bff918ac66a30a4dce742c645dbd68c1b8f02a281
SSDeep	6144:DBVsLLdsgLTCEUyQETKuSkPZ5PMVhewjAn6EYIW2gBsKTp0aLXWoRRBJY9HJJRws:QV3hLQET9P5UP06E+HsKl0aLNbs	
Size	376,832 bytes	
File Type	PE32 executable (GUI) Intel 80386, for MS Windows	

Detections	AhnLab-V3 = Win-Trojan/Lockycrypt.Gen
	Baidu = Win32.Trojan.WisdomEyes.151026.9950.9991
	ESET-NOD32 = a variant of Win32/Injector.CVCD
	McAfee = Ransomware-FHE!EC10753A4162
	Qihoo-360 = HEUR/QVM07.1.0000.Malware.Gen	

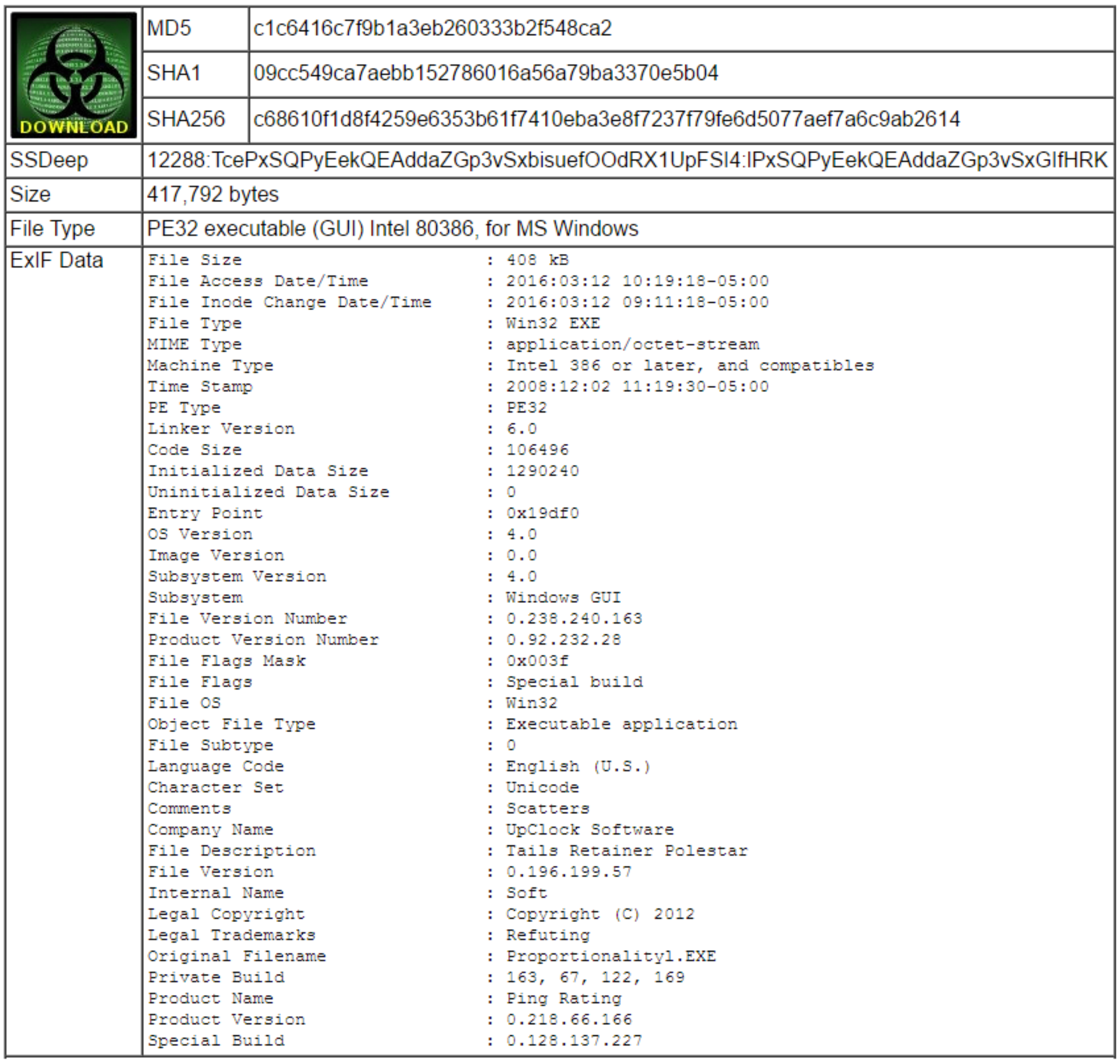
ExIF Data	File Size : 368 kB File Access Date/Time : 2016:03:23 12:59:53-04:00 File Inode Change Date/Time : 2016:03:23 12:51:03-04:00 File Type : Win32 EXE MIME Type : application/octet-stream Machine Type : Intel 386 or later, and compatibles Time Stamp : 2007:06:29 10:14:18-04:00 PE Type : PE32 Linker Version : 6.0 Code Size : 102400 Initialized Data Size : 720896 Uninitialized Data Size : 0 Entry Point : 0x190f8 OS Version : 4.0 Image Version : 0.0 Subsystem Version : 4.0 Subsystem : Windows GUI File Version Number : 0.182.8.31 Product Version Number : 0.101.92.173 File Flags Mask : 0x003f File Flags : Special build File OS : Win32 Object File Type : Executable application File Subtype : 0 Language Code : English (U.S.) Character Set : Unicode Company Name : Trident Software, Ltd. File Description : Wholehearted Unconsidered Angina File Version : 0.193.232.75 Internal Name : Accra Legal Copyright : Copyright (C) 2011 Legal Trademarks : Adored Original Filename : Bitingl.EXE Product Name : Asunder Wearied Product Version : 0.251.64.202	

Appendix 8 – Overview of sample 7

DOWNLOAD	MD5	15667babdcdd88ee08174a39c86b00ad
	SHA1	19ed09bbe8711e7e0b9a6b7664538559a86d312d
	SHA256	5061395e96ddf44be20b37f12ab25da2ee84f9c8ec2dd0b5db4f11cfdb14b2a0
SSDeep	12288:LVaauWatLv/kjWaesK3YSYJmlzFZ3lHmMr:L03DkjtLS5hVq	
Size	405,504 bytes	
File Type	PE32 executable (GUI) Intel 80386, for MS Windows	
ExIF Data		

```
File Size                       : 396 kB
File Access Date/Time           : 2016:03:12 10:57:35-05:00
File Inode Change Date/Time     : 2016:03:12 09:14:31-05:00
File Type                       : Win32 EXE
MIME Type                       : application/octet-stream
Machine Type                    : Intel 386 or later, and compatibles
Time Stamp                      : 2006:04:16 22:36:29-04:00
PE Type                         : PE32
Linker Version                  : 6.0
Code Size                       : 245760
Initialized Data Size           : 2846720
Uninitialized Data Size         : 0
Entry Point                     : 0x3bd38
OS Version                      : 4.0
Image Version                   : 0.0
Subsystem Version               : 4.0
Subsystem                       : Windows GUI
File Version Number             : 0.238.104.153
Product Version Number          : 0.28.233.78
File Flags Mask                 : 0x003f
File Flags                      : Special build
File OS                         : Win32
Object File Type                : Executable application
File Subtype                    : 0
Language Code                   : English (U.S.)
Character Set                   : Unicode
Comments                        : Striped
Company Name                    : SMART Technologies Inc.
File Description                : Astronomically Tabulations Boneless
File Version                    : 0.231.119.30
Internal Name                   : Viewings
Legal Copyright                 : Copyright (C) 2016
Legal Trademarks                : Umbra
Original Filename               : Anointsl.EXE
Product Name                    : Tomtom Agitating
Product Version                 : 0.70.81.155
Special Build                   : 0.72.128.245
```

Appendix 9 – Overview of sample 8



	MD5	c1c6416c7f9b1a3eb260333b2f548ca2
	SHA1	09cc549ca7aebb152786016a56a79ba3370e5b04
	SHA256	c68610f1d8f4259e6353b61f7410eba3e8f7237f79fe6d5077aef7a6c9ab2614
SSDeep	12288:TcePxSQPyEekQEAddaZGp3vSxbisuefOOdRX1UpFSl4:lPxSQPyEekQEAddaZGp3vSxGlfHRK	
Size	417,792 bytes	
File Type	PE32 executable (GUI) Intel 80386, for MS Windows	
ExIF Data		

```
File Size                       : 408 kB
File Access Date/Time           : 2016:03:12 10:19:18-05:00
File Inode Change Date/Time     : 2016:03:12 09:11:18-05:00
File Type                       : Win32 EXE
MIME Type                       : application/octet-stream
Machine Type                    : Intel 386 or later, and compatibles
Time Stamp                      : 2008:12:02 11:19:30-05:00
PE Type                         : PE32
Linker Version                  : 6.0
Code Size                       : 106496
Initialized Data Size           : 1290240
Uninitialized Data Size         : 0
Entry Point                     : 0x19df0
OS Version                      : 4.0
Image Version                   : 0.0
Subsystem Version               : 4.0
Subsystem                       : Windows GUI
File Version Number             : 0.238.240.163
Product Version Number          : 0.92.232.28
File Flags Mask                 : 0x003f
File Flags                      : Special build
File OS                         : Win32
Object File Type                : Executable application
File Subtype                    : 0
Language Code                   : English (U.S.)
Character Set                   : Unicode
Comments                        : Scatters
Company Name                    : UpClock Software
File Description                : Tails Retainer Polestar
File Version                    : 0.196.199.57
Internal Name                   : Soft
Legal Copyright                 : Copyright (C) 2012
Legal Trademarks                : Refuting
Original Filename               : Proportionality1.EXE
Private Build                   : 163, 67, 122, 169
Product Name                    : Ping Rating
Product Version                 : 0.218.66.166
Special Build                   : 0.128.137.227
```

Appendix 10 – Overview of sample 9

	MD5	b09aca00a8dcded70eeac6ec2b497e60
	SHA1	9247ba9335b88b4fc1d8febed66e92e4aad8317c
	SHA256	b45ae8dabc0e3d299a47425c624d526ce6668728499307d77acb6266f4c4ae29
SSDeep	6144:J+lMnaN9yLmfyoZjcbxstF8clxnTYI4LVmKJ7t2AQeRi:8TN9xyomFstF8conTCLVzTZRi	
Size	385,024 bytes	
File Type	PE32 executable (GUI) Intel 80386, for MS Windows	

ExIF Data

```
File Size                       : 376 kB
File Access Date/Time           : 2016:03:12 10:07:32-05:00
File Inode Change Date/Time     : 2016:03:12 09:13:05-05:00
File Type                       : Win32 EXE
MIME Type                       : application/octet-stream
Machine Type                    : Intel 386 or later, and compatibles
Time Stamp                      : 2008:01:10 12:33:25-05:00
PE Type                         : PE32
Linker Version                  : 6.0
Code Size                       : 245760
Initialized Data Size           : 4632576
Uninitialized Data Size         : 0
Entry Point                     : 0x3c8be
OS Version                      : 4.0
Image Version                   : 0.0
Subsystem Version               : 4.0
Subsystem                       : Windows GUI
File Version Number             : 0.104.250.202
Product Version Number          : 0.61.130.188
File Flags Mask                 : 0x003f
File Flags                      : Special build
File OS                         : Win32
Object File Type                : Executable application
File Subtype                    : 0
Language Code                   : English (U.S.)
Character Set                   : Unicode
Comments                        : Virulence
Company Name                    : Panda Security S.L.
File Description                : Crock Crunchier Connectives
File Version                    : 0.5.37.16
Internal Name                   : Alacrity
Legal Copyright                 : Copyright (C) 2013
Legal Trademarks                : Adder
Original Filename               : Yrsl.EXE
Product Name                    : Vivisected Baluster
Product Version                 : 0.187.109.76
Special Build                   : 0.69.8.144
```

Appendix 11 – Overview of sample 10

DOWNLOAD	MD5	34016905603c92a45b2de5810c3cc92c
	SHA1	8e8586264904455ade7a9557e4fc6fb5e3939ab0
	SHA256	29ffc26ba24154f9dcb0f391cac5e2bd69acaa4ba58932faeab3769829324304
SSDeep	12288:wGCQZWc5/0vYucd5jEfuTW1RSeQ/VWKvYucd5jEfuTW1R:NCQkc5/cYuE5ECWriWaYuE5ECW	
Size	638,976 bytes	
File Type	PE32 executable (GUI) Intel 80386, for MS Windows	
ExIF Data	see below	

```
File Size                       : 624 kB
File Access Date/Time           : 2016:03:12 11:19:22-05:00
File Inode Change Date/Time     : 2016:03:12 09:14:42-05:00
File Type                       : Win32 EXE
MIME Type                       : application/octet-stream
Machine Type                    : Intel 386 or later, and compatibles
Time Stamp                      : 2006:12:25 05:31:08-05:00
PE Type                         : PE32
Linker Version                  : 6.0
Code Size                       : 372736
Initialized Data Size           : 2248704
Uninitialized Data Size         : 0
Entry Point                     : 0x5b3b6
OS Version                      : 4.0
Image Version                   : 0.0
Subsystem Version               : 4.0
Subsystem                       : Windows GUI
File Version Number             : 0.21.196.238
Product Version Number          : 0.211.9.19
File Flags Mask                 : 0x003f
File Flags                      : Special build
File OS                         : Win32
Object File Type                : Executable application
File Subtype                    : 0
Language Code                   : English (U.S.)
Character Set                   : Unicode
Comments                        : Quiches
Company Name                    : Net Transmit & Receive
File Description                : Picnics Pounces Reserves
File Version                    : 0.193.145.169
Internal Name                   : Loomed
Legal Copyright                 : Copyright (C) 2014
Legal Trademarks                : Redress
Original Filename               : Reshuffling1.EXE
Private Build                   : 130, 137, 130, 108
Product Name                    : Reconciliations Naval
Product Version                 : 0.244.41.138
Special Build                   : 0.230.231.193
```

Appendix 12 – Result of validity test with 25 numbers in pattern and 25 numbers of sequences

Validity Test - Test nr 8	Entropy	Chi-Square	Mean	Monte	Serial
Number of Letters in Pattern	25	25	25	25	25
Number of Sequences	25	25	25	25	25
Analysis Result Value	7.999203	719.796275	127.278235	3.143044	0.003858
Measurement Threshold	7.984000	690	127.3-127.7	3.12-3.16	±0.001200
Success/Failure	Success	Failure	Failure	Success	Failure