A comparative study of CERBER, MAKTUB and LOCKY Ransomware using a Hybridised-Malware Analysis Framework

Submitted in Partial fulfilment
of the Requirements of the Degree of
Master of Science
at

Rhodes University

Veronica Schmitt

Grahamstown
January 2019

Abstract

There has been a significant increase in the prevalence of Ransomware attacks in the preceding four years to date. This indicates that the battle has not yet been won defending against this class of malware. This research proposes that by identifying the similarities within the operational framework of Ransomware strains, a better overall understanding of their operation and function can be achieved. This, in turn, will aid in a quicker response to future attacks. With the average Ransomware attack taking two hours to be identified, it shows that there is not yet a clear understanding as to why these attacks are so successful. Research into Ransomware is limited by what is currently known on the topic. Due to the limitations of the research the decision was taken to only examined three samples of Ransomware from different families. This was decided due to the complexities and comprehensive nature of the research. The in depth nature of the research and the time constraints associated with it did not allow for proof of concept of this framework to be tested on more than three families, but the exploratory work was promising and should be further explored in future research.

The aim of the research is to follow the Hybrid-Malware analysis framework which consists of both static and the dynamic analysis phases, in addition to the digital forensic examination of the infected system. This allows for signature-based findings, along with behavioural and forensic findings all in one. This information allows for a better understanding of how this malware is designed and how it infects and remains persistent on a system. The operating system which has been chosen is the Microsoft Window 7 operating system which is still utilised by a significant proportion of Windows users especially in the corporate environment. The experiment process was designed to enable the researcher the ability to collect information regarding the Ransomware and every aspect of its behaviour and communication on a target system. The results can be compared across the three strains to identify the commonalities. The initial hypothesis was that Ransomware variants are all much like an instant cake box consists of specific building blocks which remain the same with the flavouring of the cake mix being the unique feature.

Acknowledgements

The roots of education are bitter, but the fruit is sweet.

Aristotle

A special thanks to my husband Adrian and daughters Nicola and Mikayla, they have sacrificed, that I could work on my research. I would also like to thank my thesis advisor Professor Barry Irwin at Rhodes University. The door to Prof. Irwin's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it. He challenged and pushed me, he was instrumental in the growth I have had both personally and academically. He has been the wizard to magically fix all my LaTeX errors. I also want to thank the other wizards and elves for their help and support. I want to thank each and everyone that assisted me and motivated me to finish this work which has been a trying two years.

I would like to dedicate this work to my mother in law Jennifer Barker, who sadly lost her battle against cancer during the writing of this. She has always believed in me and supported me and part of this journey is due to her motivation and support. I also would like to thank my parents who have been there to support and assist with my children when I needed to focus on my writing. I would like to thank my mother Bettie Venter for raising me to shoot for the moons, she has taught me that nothing is impossible.

I would like to thank my business partner and friend Jason for giving me the freedom and time to work on my research and believing in me. I also would like to extend a special thank you to Jared Meyers from Carbon Black for the support and access to the Ransomware repository, without access to this my research would not have been possible.There is simply too many people to thank that made this research possible and who believed in me, I am forever thankful for the influences and supports I have had in my life.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction

Ransomware can be described as a type of malware which is specifically designed to lock up an endpoint from use or hold data hostage by encrypting the data. McAfee Labs (2018) published their yearly threat report, this report which was published in September 2018 indicates that there has been a surge of Ransomware attacks in the past 18 month. The recent study done by Panda Security (2015) indicated that in the preceding 23 months there has been an increase of email attachments being actual Ransomware exploit (Barkly Research, 2018). The 2017 security landscape has seen a familiar picture to previous years in that Ransomware remains a threat. IBM Security (2018) states that during 2017 the estimated cost of Ransomware to companies was approximately $8 billion globally. This was due to downtime or ransoms paid. Although Ransomware has been on a steady increase since 2014, the godfather or first known Ransomware attack took place in 1989 (IBM Security, 2018). This Ransomware was malicious code which was sent via the postal system to unknowing victims contained on a floppy disk. The sophistication level of Ransomware has also shown to grow and evolve, this can be seen in the major Ransomware outbreak of WANNACRY in 2017 (RadWare, 2017). This strain was seen to employ lateral moving techniques using the ETERNAL BLUE exploit.

Statista (2018) indicates that for SME's (small medium enterprises) Ransomware has become an increasing concern. 30% of businesses felt vulnerable to Ransomware and a further 28% have indicated that they have already fallen victim to Ransomware. The most common solution to Ransomware included the use of security software which filters out the Ransomware files and blocks them. The study which was conducted by IBM Security

(2018) showed that Ransomware attacks have quadrupled since 2016. The more worrisome statistics are that only one in three consumers (approximately 31%) of end users know or have knowledge of what Ransomware is, this leaves 69% unaware of the danger of Ransomware. A further 75% of end users feel that they can confidently protect the data on their personal devices. This shows a high level of confidence but a lack of knowledge in terms of what Ransomware is (Barkly Research, 2018).

In 2017 we have seen the spread of WANNACRY and PETYA/NOTPETYA, this has brought about a new threat to the Security industry. These attacks served as a reminder to security professionals that Ransomware is forever evolving and getting more sophisticated. Ransomware is not only a threat, but has been seen to develop into a revenue stream with Ransomware as a Service (RaaS). From 2017 to 2018 there has been a steady increase of new Ransomware variants observed in the wild, this number has gone up by 46% since 2017.

The Ransomware landscape in 2017 was seen to be dominated by two strains namely WANNACRY and PETYA/NOTPETYA Symantec Corporation (2018). Both these attacks were not the run of the mill attacks and both were seen to make use of additional exploits. The PETYA/NOTPETYA Ransomware strain was found to be a wiper[1] or software designed to remove or permanently delete data, and that it was simply masquerading as Ransomware. The WANNACRY Ransomware was seen to employ a worm like behaviour to laterally move within network shares on a network. The WANNACRY Ransomware was seen to exploit two known vulnerabilities in Windows namely `CVE-2017-0144` and `CVE-2017-0145`, which allowed the Ransomware behave much like worm and self spread across unpatched systems on a network.

In 2018 one of the best RaaS[2] to be seen was GRANDCRAB, this Ransomware strain has infected 50,000 endpoint devices located mainly in US, UK, Scandinavia and Israel and to date has collected $600,000 in revenue (Barkly Research, 2018). Thus far it has become the most popular bidder and most aggressive Ransomware of this year. The Ransomware threat still remains one of the biggest cyber dangers. Morgan (2018) predicts that the revenue raised from RaaS, will reach $6 trillion annually by 2021 (Campbell, 2016). In Costlow (2018) it is predicted that by 2018 Ransomware would peak at 250% growth. Further research is needed to understand the working of Ransomware so that it can be fought on a better more informed footing.

The fact that these attacks are actively bypassing security is a leading concern in the information security field. Barkly Research (2016) found that seven out of ten successful attacks were because the malicious payload got past the security perimeter of the organisations. The traditional methods of security which deploy firewalls and antivirus are just not sufficient in protecting against the ever evolving attacks of Ransomware operators. This is resulting in the deployment of machine learning and behavioural analytics to block Ransomware in real time. The organisations have gone as far as to stock pile bitcoin for the

[1] is a software based method to erase/delete data

[2] Ransomware as a service

payment of the Ransomware. The 2017 outbreak of WANNACRY has seen the utilisation of worm like behaviour to spread to other hosts on the same network. This is an indication that Ransomware has evolved into a self spreading mallous. Ransomware authors have packaged some of the strains to be all inclusive in that no additional payload needs to be downloaded to deploy the encryption.

There has also been an increase in the Ransomware amount in the last four years. The ransom fee which was generally asked was $373 per decryption key, this has increased to a substantial amount of $1,077 in 2016/2017 (Barkly Research, 2017a). This is also an indication that the aim is to target those individuals and organisations that can afford the ransom amount. The increase is also indicative that Ransomware authors have taken into considerations that data equates to money for organisations and these are more likely to pay the ransom. Ransomware is no longer just a threat on Windows Operating systems but has been shown to be across all platforms. Recently a South Korean web hosting company Nayana indicated that 153 of their Linux servers where infected with the Ransomware strain EREBUS.When the Ransomware has infected and found a foothold within an organisation, it can spread via network shares, devices, servers and cloud applications. Riley (2017) suggests that the Ransomware operators have come to the realisation that there is more revenue to be made from RaaS, this market is set to grow during the remains of 2018-2019. They further use CERBER Ransomware as one of the most successful RaaS service streams.

1.2 Problem Statement and Research Question

There has been a significant increase in Ransomware attacks over the period of 2014-2018. This indicates that the battle has not yet been won against this form of malware. The researcher proposes that by identifying the similarities between the behavioural traits of the Ransomware families, a better understanding can be gained. This in turn can aid in faster detection of the infection. With the average Ransomware attack taking 2 hours to be identified or found, it shows that there is not a clear understanding of how the Ransomware infects and compromises the victim machine. The examination of malware is described by Yusirwan *et al.* (2015) as the examination of software which is suspected to be malicious. The research into Ransomware has not yet been fully explored and this has offered limitation into academic journal papers on the topic. The research goals are as follows:

1. The primary goal of the research is to identify the similarities between the three Ransomware families in terms of static, dynamic and behavioural traits. In order to achieve this the three Ransomware families were examined in three phases, the static analysis phase, the dynamic analysis phase and network communications.

2. The secondary goal is to identify the basic differences between the three families which make each family unique.

3. The final research goal is to identify the method used by each family to communicate on the network, this is achieved by intercepting the network communication post exposure.

1.3 Research Scope

A definition of research, as given by Booth *et al.* (1995), is "gathering the information you need to answer a question and thereby help you solve a problem". The art of academic research is to identify a problem which interests you and with the use of information gathering and experimentation find a solution to the problem. The sources from which you should gather the information should be trusted. This means that you should mainly use sources from articles or referenced sources first and foremost. A technical examination was conducted across three Ransomware families namely CERBER, MAKTUB and LOCKY. These three variants where chosen for the success rate and longevity. The collection of samples was done by utilising the malware suppository supplied by Carbon Black, which is a company in the United States specialising in malware analysis. There was 220 samples obtained from varied families of Ransomware. This was done to ensure that a qualitative research could be done instead of a quantitative.

The aim of the research is to follow the Hybrid-Malware analysis framework (Roundy and Miller, 2010). This framework consists of static malware analysis, the dynamic malware analysis and the digital forensic examination of the compromised system. This allows for the examination of the malware based on a multi disciplinary approach allowing for the maximum varied data collection surround the signature, behaviour and interaction of the malware (Malin *et al.*, 2008).

With this information the researcher will gain a better understanding of how this malware is designed and how it implements itself on the system. The operating system which was chosen is the Microsoft Windows 7 operating system, the reason for this is that this is still the preferred used operating system used by the majority of the users globally. According to the statistics from StatCounter (2018) Microsoft Windows 7 is still the most popular operating system for desktops and laptop computer topping the charts at 36,22%. The experiment process was designed to enable the collection of static information of the malware strain, behavioural analysis and iteration with the Microsoft Windows 7 operating system. The hypothesis for this research is that the basic design of Ransomware is similar to one another and simply the execution and attack method differ between families. The analogy of an instant cake mix can be used, in that the basic ingredients remain the same and the differences are in flavouring and decoration.

1.4 Scope and Limitations

The research is limited to include only the most relevant information. Due to in depth detail which was examined on each strain, the scope was limited to three prevalent fami-

lies. This was done to present an experiment which is qualitative in nature and contains comprehensive detail on each family.

1.5 Research Contribution

The main aim of good research is to contribute towards the improvement of the academic field. This section identifies the contribution which will be made by the researcher in terms of the research conducted. The research conducted by the researcher follows a scientific approach using an experimental process. This allows for a deeper technical understanding on the way that these three Ransomware strains have been compiled and the process they follow to encrypt and exploit the Window 7 operating system. A methodology is presented as how to examine Ransomware to gain a deeper understanding of it, along with the fact that there has not been research done in depth to compare these three families. The contribution would offer others a deeper technical understanding into the inner workings of malware families. The process can be repeated multiple times across various strains, this will be done in future research.

1.6 Terms and Definitions

The research is conducted in a field where there is technical definition which the reader may not be familiar with. This section specifies the technical terminology and the definition thereof, used in the remainder of this work.

- **Ransomware** a type of malicious software designed to block access to a computer system until a sum of money is paid
- **Malware** software which is specifically designed to disrupt, damage, or gain unauthorised access to a computer system.
- **Command and Control servers (C&C)** are computers that issue commands to members of a botnet.
- **Malware analysis** is the study or process of determining the functionality, origin and potential impact of a given malware sample such as a virus, worm, Trojan horse, rootkit, or backdoor.
- **Reverse Engineering** the reproduction of another manufacturer's product following detailed examination of its construction or composition
- **Digital Forensics** is the process of uncovering and interpreting electronic data. The goal of the process is to preserve any evidence in its most original form while performing a structured investigation by collecting, identifying and validating the digital information for the purpose of reconstructing past events.

1.7 Document Conventions

The naming conventions which have been used specifically to highlight or ensure that the reading of the document is made easier on the reader. Due to the fact that there area lot of system specific items and system paths these are changed to make reading better.

Malware a strain as shown using small caps, for example CERBER

System system related activity and or system related artefacts as shown using the font TypeWriter , for example `vssadmin.exe`

Documents document names and file names as shown using the font Sans Serif, for example Terms of Reference.doc

Functions system functions as shown using the font TypeWriter, for example `CreateFile`

URL links in text are shown using the font TypeWriter, for example `www.google.com`. In many cases these are also provided as footnotes, to aid the reader.

All Figures and Tables unless explicitly mentioned otherwise, have been produced by the researcher during the course of this work.

1.8 Document Structure

The remainder of this document is structured as follows. Chapter 2 details the Literature review on related topics associated with the work. Chapter 3 details the new proposed methodology and the data set collection. Chapters 4, 5 and 6 details the findings from the examination of CERBER, MAKTUB AND LOCKY using the framework detailed in Chapter 3. Chapter 7 details and compares the findings across the three Ransomware strains. Chapter 8 detail the conclusion and evaluation of the work presented in this document.

2

Literature Review

2.1 Introduction

This chapter provides a review of literature conducted by the researcher. The literature review has been limited due to the fact that on the topic of Ransomware there has not been significant academic papers published. This means that the researcher had to broaden the topical scope to be more encompassing. This also means that there was more focus on technical reports than academic papers. The researcher attempted to be as thorough as possible to ensure that all aspects of the literature review has been covered. The researcher has focused on areas which fall within the examination methodology that will be used. The literature on actual Ransomware has been limited to online reports and even though this is a new field there has not been formal publication made and as such the majority of the resources on the various strains which were looked at were limited to online reports and technical content online.

This includes a comprehensive review on Ransomware in Section 2.2, the next section includes information relating to the operation of Command and Control Servers (C&C) in Section 2.3, the following section covers the types of encryption used by the malware strains in Section 2.4 and 2.5. The remainder of the literature review is on malware analysis including Static Malware Analysis, Dynamic Malware Analysis in Section 2.6. Furthermore the literature covers the structure of the Portable executable file in Section 2.6.3.

2.2 Ransomware

Liska and Gallo (2016) describes the term Ransomware as a blanket term, which is used to

describe the class of malware that is used in the digital extortions of victims. This variant of malicious software, denies the user access to their data, and threatening to destroy or release the data until a ransom has been paid. There are two definitive kinds, there is the encrypting Ransomware that includes encryption algorithms , this malware is specifically designed to block system files until the demand for ransom has been paid to which the user will receive a key to decrypt these files. The second variant is called locker Ransomware this locks the victim out of the operating system as a whole (Zaharia, 2016). There are some locker variants, like SATANA Ransomware, which do not only encrypt your data. This type of Ransomware also alters the master file table on a NTFS file system, to boot into a custom loader, thus ensuring that the user cannot gain access to the Windows operating system. According to MalwareBytes (2016), this type of locker has had a bigger impact on users. The use of Ransomware as a service has become more accessible and easier to obtain by criminals.

The Ransomware-as-a-Service model has been a big factor in the contributing factors making it easier for cyber-criminals with limited technological knowledge to facilitate their own attacks. There are not specific targets when it comes to victim selection. There are some industries which allow for themselves to be bigger targets than others. In recent attacks it has been seen that the UK National Health services is one of the main industries which have been targeted (Vincent, 2017). There have however also been victims in the Education, IT/Telecoms and Financial sector hits (Laves, 2018). The Ransomware successfully bypassed their security controls and encrypted their data. This means that Ransomware has a 70% success rate in bypassing security measures of their victims. There are currently two schools of thought in the combat of Ransomware one is to employ machine learning to identify and block the Ransomware in execution phase and the stock piling of bit coins that they are prepared to pay the ransom. Barkly Research (2017c) predicts that the financial damages to organisation and individuals hit by Ransomware for 2017 is approximately $8 billion dollars.

2.2.1 Key Characteristics

Meskauskas (2017) indicates that the Ransomware variants across families make us of mainly two encryption protocols namely RSA or AES-256. Ransomware makes use of the public key encryption process. The other encryption process utilised by Ransomware is the AES-256 (advanced encryption system) which is based on the design principle known as the substitutions permutation network. Townsend Security (2016) describes the AES-256 encryption system as the combination of both the substitution and permutations, this means that it is both fast in software and hardware. The AES-256 encryption is a variant of the Rinjindael algorithm, which is a fixed block size encryption of 256 bits. The encryption which is employed by the Ransomware is generally RSA or AES-256; both of these are extremely strong encryption algorithms meaning that decryption can only be achieved with the corresponding decryption key. The RSA encryption protocol is one of the first encryptions ecosystems which used a public-key. Both of these algorithms are strong and

very difficult to brute force decrypt without the corresponding key. Both are widely used to disseminate data securely. The RSA encryption system is based on the practical difficulty of factoring the product of two large prime number. This is an asymmetrical encryption process. The public key is used to encrypt the data and the private key is used to decrypt the data. This algorithm is fairly slow and not used to directly encrypt user data. This protocol is used to pass encrypted shared keys for symmetric key cryptography this in turn can perform bulk encryption operations at much higher speeds.

The file types which are encrypted vary from Ransomware strain but generally include documents, pictures, videos, audio file and other files, most do however exclude system files and executables. The Ransomware relies on its ability to confuse the user to not know which files are which, by scrambling the filenames and replacing them with random strings. Each Ransomware strain has its own extension protocol, which it will apply to the encrypted files this can be used to identify the strain which has infected the system. A window pops up which cannot be closed that displays the specific image and or message which details the process of paying the ransom. Crypto currency is widely used by different malware strains, the reason being that is a form of digital or virtual currency. This form of currency employs anti-counterfeiting methods. The reason that this is so popular, according to Vincent (2017) is because it is fast, global, secure and anonymous. This currency has applied to different groups of criminals and those distrustful of the government. The stability of the fiat currency also plays a big part. The old days of collecting ransom money, involved the proverbial suitcase full of cash and wire transfers which can be tracked is long gone. Criminal organisations now have the resources to collect and clean large sums of money without the fear of being caught. The ransom payment has a time limit which exerts pressure on the user by adding a psychological element to the extortion. This does not allow the user enough time to source another option and the fear of losing personal and confidential data. Ransomware is now marketed as a service which can be sold on the deep web. The authors of the Ransomware have started marketing on-demand versions of the original code to suit the needs of their clients offering customisation (Campbell, 2016). The author would then collect the ransom and share it with the distributors. The most used methods of distribution is with spam campaigns and drive-by downloads which have been infected.

According to Patterson (2018), the Ransomware market has scaled up quickly partly because of the offering of RaaS. These services can be obtained on the deepweb using TOR[1] hosted websites and payment being made. The CERBER for example employs advanced evasion techniques; this malware makes use of anti-sandbox and anti-detection technology such as using a new loader which hollows out legitimate processes and inputting the CERBER code. This way the process is legitimate on the surface, however internally it is compromised code. Arghire (2017) states that whilst the dll is not packed or encrypted, it reads the configuration file and decrypt a portion of it that contains the loader and the configuration settings. The loader proceeds to run a check to determine whether it is running

[1]The TOR browser facilitates communication across distributed networks or relays which ensure that you browse anonymously

in a virtual environment. The Ransomware would also check to determine within the processes whether there are any analysis tools run and would proceed to kill those processes. This is however not been noted in all variants. One of the main methods used by various Ransomware strains is to have the initial downloader infect the machine; the main purpose of the downloader is to download the secondary exploit onto the compromised system. There has also been a noted instance whereby a Trojan botnet was used to download Ransomware onto computers which were infected. This is usually the final step in monetising the infected machine which they have control over according to (O'Brien, 2017). Proofpoint (2017) found that the CRYPTXXX Ransomware employs an exploit kit, STILLERX, which acts as a standalone credential stealing exploit.

STILLERX mainly targets credentials of a wide range of software such as download managers, FTP software and email clients (Szmigielski, 2016). The fact that Ransomware has infected your machine does not indicate the end of the exploit and it the background your credentials may be exfiltrated for further monetising. The Ransomware included sophisticated geographical targeting and exclusion. This means that certain campaigns will exclude countries which are included in a module within the binary. The malware campaigners have been seen to have diverted their attentions from the hardened network attacks to targeting the users themselves. This makes Ransomware more dangerous because even if you attempt to protect your network perimeter the real danger is that of your users. Protection against Ransomware should be protecting your network from the inside out. The attacks that have been seen with regards to spam mail campaigns are becoming more professional and have certainly improved their execution. There has been correlation seen in the attack vectors which have been used.

2.2.2 The Attack Vectors of Ransomware

The attack vectors often used by Ransomware has been identified and explained below. These are not the only attack vectors but the ones currently used the most. These include targeting the human factor and in terms of this work the scope was limited to the four most popular.

Social Engineering The art of social engineering is the ability to manipulate people into giving up their personal and confidential information. The type of information which is looked for by the malware campaigners can vary on the individual which is being targeted. The information can range from user credentials or access to your computer. Social engineering is the process of exploiting the human psychology rather than attempting other sophisticated methods. Ransomware relies on the unsuspecting user to download the file or respond to the phishing email by clicking on the link. This part of the malware campaign is the most important as stated by Goodchild and Hulme (2017). The malware campaigners are known for doing reconnaissance on their victims, and design the spam email campaigns to mimic a known source that is trusted to the victim to deliver the malware email to. Once control has been gained on an email account the malware campaigners

have the ability to access all social networks of the victim and contact lists.

Baiting is described by Babic *et al.* (2011) as the process in which an attacker leaves an USB device which someone could then pick up. The device is then used within the network, as would be the case dealing with the human factor. Once the device has been introduced onto the network or machine the malware takes over the spread and compromise of the environment.

The use of phishing attacks to deliver Ransomware has been one of the main methods used. The malicious campaigners send out fraudulent emails disguised as legitimate emails, often visually from a trusted source. These messages are often well thought out and professionally designed to trick the victim into opening the attachment which in turn would install the malware. The other technique employed by malware campaigners is known as scareware. This process is whereby the victim is deceived into thinking that their computer is infected by malware and unknowingly downloads content containing malicious software. This aim is for the victim to install the malware whilst playing on the fear element of being infected. Ransomware relies on the user to open the attachment that contains the initial payload. This is done by employing social engineering techniques.

Layered Attacks The malware campaigners will sell access to the resources which have already been compromised and infected. The undetected malware on the zombie machine will be able to download the Ransomware and remain on the system until after the Ransomware is paid or lay dormant in wait for the next extortion. The encryption of data is not always the only end game according to Zaharia (2016).

Embedding Ransomware can be embedded into legitimate software downloads much in the way of a Trojan horse. Jennings (2016) reported that Adobe flash issued and emergency update due to the fact that the security flaws within the Adobe Flash download was used to deliver Ransomware to Windows PC. The drive by download would install the CERBER strain when the software was downloaded from the compromised website. The fake pages which are used are notoriously convincing stated by Zaharia (2016).

Self-propagating Ransomware is the threat from within once it is executed from within the network it can communicate with a command and controls severs downloading further exploits. This can then exploit different computers on the local network. This can be done by sending emails, files or communications.

2.3 Malware Command and Control Servers

One or more action requires some form of command and control to effectively take their next action. This is the same process which is followed with traditional cyber warfare and attacks. Ransomware makes use of what is known as command and control server.

These are also known as a C&C. These servers are used by the attackers to maintain the communication with the compromised machines. The terms command and control is a term taken from the military. The compromised machines form part of a larger network known as zombies communicating to the botnet C&C server. The communication can be as simple as to maintain a time beacon or heartbeat which keeps the running attack alive and keeps an inventory of the systems which have been compromised.

The connection which is kept alive serves the purpose to have access to the compromised machine to be able to send exploits or ex filtration data. The C&C servers are often hosted on compromised web servers or on the `.onion` websites which offer more obfuscation options (Gross, 2015).

Malware is described by Bullguard (2017) as short for malicious software which is contained within as single binary file, which is unpacked on the system. This uses the kernel of the operating systems as a device driver. This allows for persistence on the system and to evade detection. When the malware has been executed the malware code sends out a beacon to the command and control beacon to download the next intrusion instructions from the C&C. Malware mostly contain pre-programmed autonomous commands which will attempt to set up and maintain the persistent communication line this is in contrast to some Ransomware such as CERBER which can encrypt offline. Within the malicious code is a phone book which has domain names and IP address for the malware to cycle through to evade tracing. When the compromised network firewall has very liberal outbound or egress firewall rules the malware will establish a communication channel and maintain the connection (RadWare, 2017). With the increase of organisation having tighter controls implemented in relation to the traffic which is sent out on the network the malware operators have adjusted their method of execution and communication. Most companies allow unfiltered outbound communication on the well known HTTP, HTTPS, FTP and SSH ports and themalware operators have been seen to use these ports for their communication.

This makes it easier for the malware operators to hide amongst the network chatter of the Windows operating system. Once the malware has successfully secured the channel of communication to the C&C server, the instruction to download further rootkits, exploits or remote access tools will be done. The compromised machine once fully infected will start to send beacons home to the C&C server (Trend Micro, 2017). A popular communication technique that is used by Ransomware is to make use of DNS services which are publicly available which assists in hiding the C&C within valid websites avoiding detection. Advanced persistent threat actors will try and utilise public DNS services so as to avoid the logging within a private network which can risk the detection. The malware operators will also leverage off dynamic DNS hosting sites. This method allows for the infection to be done on a larger scale. This is one of the stealthiest and difficult to catch if you do not have any other Indicators of Compromise (IOC). Indicators of Compromise can be described as forensic artefacts which are left behind after an intrusion has taken place that can aid in the identification of the host or network compromised (Jacobson, 2013). To be able to identify these malicious programs there is systems and network analysis needed to determine the interaction and communication of the malware (Jacobson, 2013). The use of dynamic

DNS services allows for the C&C servers to be moved to a new IP address fairly easily and regularly. These DNS services used allow for the use of time to live domains which ensures for a computer to check back to the IP address when it attempts to resolve this part of the communication.

The malware campaigner's leverages on the vast number of dynamic DNS solutions meaning that if the C&C was found it can be moved quickly (Liska and Gallo, 2016). Ransomware makes use of domain generation algorithms; these are the portions of the code that enable the creation of various numbers of communication channels on the fly. These domains only need to be available for a brief period of time and can be rotated through a series of IP address. An example of this can be seen in Figure 2.1. Ransomware attacks once the malicious code has been executed and installed the first initial communications beacons are sent out to the control and command server. The communication can range from identifying the file types which should be targeted for encryption or how long the delay should be for the process to begin. Some of the Ransomware strain variants they send statistics, and environmental information of the system and its location. C&C servers vary between variants and families of malware strains. This can range from web based communications leveraging from unencrypted HTTP protocols to more advanced ones embedded in the TOR web based services according to Liska and Gallo (2016).

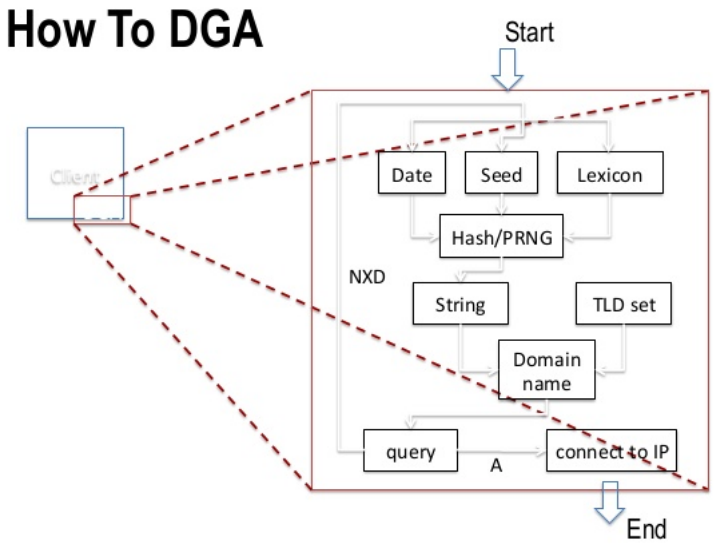


Figure 2.1: DGA algorithm example
Source :Source:https://www.slideshare.net/OpenDNS/using-algorithms-to-brute-force-algorithms-a-journey-through-time-and-namespace

2.4 Symmetric Key Encryption

Symmetric Key Encryption otherwise known as the Secret key cryptography is described as generally consisting of a stream cipher or a block cipher. The stream cipher is operated on a

single bit at a time and implements some sort of feedback mechanism which allows the key to be constantly changing (Arntz, 2016a). The block cipher is the scheme which encrypts one block of data a time, making use of the same key. Ransomware across most families make use of symmetric key encryption, this method is used as it leverages off the device itself to generate the key that is used in the encryption process. The use of symmetric key encryption process aids in the obfuscation process as limited system resources are used during the encryption process. This information is furthermore detailed in Section 2.2.1 and the use of these in Ransomware.

This aids in preventing the detection of the Ransomware through process monitoring. Ransomware effectively uses the CPU resource of the system which it is infecting. Using a small key generated on the device can minimise performance overhead whilst maximising the volume of files which it can encrypt in relatively small amount of time. The one main advantage of this method is that a unique key is generated for each device and the extortionist can determine which systems were successfully infected and which were not. This also allows for the encryption to take place offline and online. However it would require the system once online to still communicate with the C&C to send the adversary key that would initiate the ransom clock. The key is not stored on the device and is sent to the malware campaigners and kept to ransom once the key on the system has been destroyed.

2.5 Asymmetric Key Encryption

Proofpoint (2017) describes Asymmetric Key Encryption is the ability for anyone who is in possession of the public key of Alice is in a position to send her a message only she can view with her public key no matter the algorithm used. The other method also employed by Ransomware is the use of asymmetrical key encryption wherebythe malware campaigners would have a public and private key that is used for the encryption process. An example of this would be WANNACRY where there was no unique key issued to victims, this poses a problem in that there is no way to distinguish between who has paid the ransom and who has not. The public key is utilised to encrypt the files on the infected machine and the private key is used to decrypt the files. When using the asymmetric key encryption there are mainly two types embedded public key and downloadable public key. The embedded public key methodology is basic in that within the Ransomware binary file there is a hard coded public key which enables the encryption of data online or offline. The downloadable public key methodology means that the actual encryption cannot be done until the public key has been downloaded. This means that communication is an essential part of the encryption process. This method of encryption carries high prime numbers starting at 2048 bit encryption and higher meaning brute force attacks are difficult.

2.6 Malware Analysis

Dalziel and Crosby (2014) states that for a researcher to understand malware, they have

to observe how the malicious code is executed on the host and on network level. The behaviour of malware is studied through a process called malware analysis. This is a process in which a piece of malware is examined to determine how it behaves once it has been executed. There are two methods of conducting malware analysis; static malware analysis and dynamic malware analysis. Zeltser (2017) states that the analysis of malware is important to gain a deeper insight into the tangible inner behaviours of the malware. The analysis of malware is a notable manual and automated process which in itself Section a significant amount of patience and time. Khouzani and Karyotis (2016) explain that there are two phases which aim to gain an overview of malware and its inner workings. The first phase is specifically there with the goal to gain information on the malware binary and to be able to anticipate the behaviour of the malware, the authors further state that the second phase is there to analyse the features of the malware whilst observing the behaviour of the malware in a live or vitalised environment (Gibson, 2011). To be able to be in a position to stop malware from spreading, there needs to be an understanding on how the malware operates. This is achieved by examining and analysing the malware strains. When a clear understanding is achieved of the malware a solution for better protection a prevention can be achieved. Even though malware analysis holds benefits which can aid in the detection and prevention of the spread of malware the opposing side is that it has limitations as well. In the static analysis phase of malware analysis the data collection is only effective if the data is free of encryption and data obfuscation. When malware is packaged with encryption decryption is needed for the analysis to be completed. This is why the malware sample will be subjected to a various selection of tools to gain the maximum information whilst minimising the risk of accidental infection during the examination process.

2.6.1 Static Malware Analysis

The first step to malware analysis is called the static analysis phase. This phase is the process in which the malware binary is examined in terms of the code to determine the structures and functions which it calls. The aim of the static analysis phase is to extract as much information out from the malware binary as possible without executing it. There are different tools used in this phase to examine the internal structures of the code. The information gathered in this phase range from the file type to the more complicated strings which can be found in the unencrypted portion of the code. This phase is less risky than the dynamic analysis; the malware binary is simply subjected to various static analysis tools to extract information. There is no risk of infection in this phase. Elisan (2015) states that the consideration needs to be given that static analysis is a low-risk analysis phase but offers low-rewards in return. The information gathered in this phase is limited and does not reveal more that the malware's directive. The true nature of the malware can only be known by observing the malware in action.

2.6.2 Dynamic Malware Analysis

The second process of malware analysis is the dynamic malware analysis phase. This phase is the process of extracting information from the malware whilst it is running. This phase offers an in depth view of the inner workings of the malware whilst it is in a live state. The malware test environment should be a system where the malware is executed for the purpose of review which is designed to satisfy the conditions to run. The malware test environment should also be safe and isolated as to ensure that the malware does not spread. The dynamic analysis tools are the general tools used by system administrators to monitor their systems these include *Microsoft Sysinternals* [2] tools such as *Process Monitor, RegShot*. These tools are used to monitor the changes made by the malware binary whilst it proceeds to run its course. The changes which are generally recorded in this process are:

- File System changes
- Modifications in the configuration files
- Registry changes
- Process changes

The aim of this phase of the analysis is to also monitor the inbound and outbound communication on the network. An ideal tool for the capturing of the network communication is the built in tool within *Wireshark*[3] called *Tshark*. Elisan (2015) states that the process of dynamic analysis is a high-risk process which has high rewards. The risk of infection is high if the malware is not contained and controlled. There needs to be precautions taken to isolate the malware test environment from infecting other machines, this means that your test environment should be isolated from your productions systems and network.

2.6.3 The Portable Executable File

The Windows PE (Portable Executable) file is part of the Microsoft Windows operating system. The vision behind this file according to Elisan (2015), is that it wanted to create a single format file which could be used across different flavours of Windows family. There are two common known file types which fall under the classification of PE files. These are the `exe` and the `dll` files. The `exe` file is an executable stand alone file whilst the `dll` file is a dynamic link library file. The `dll` file forms part of the Microsoft's code library which contain data that can be used by one or more program within the operating system. This method promotes the reuse of code and efficient memory usage. The way that this works is that when a program is executed instead of having to load the functionality it needs in a separate memory address space it simply invokes the `dll` file from the code library of Windows.

[2] https://docs.microsoft.com/en-us/sysinternals/
[3] https://www.wireshark.org/

2.6.4 PE File format

The Windows PE file format was developed/derived from the older Common Objects File Format (COFF) (Shaaban and Sapronov, 2016). The Windows NT development team designed this file format to reuse code which they were familiar with from previous version. The aim was to come up with a single format file which can be used across multiple flavours of Windows. As can be observed in Figure 2.2 the 'PE files' contain certain sections within the file itself. The first section is known as the DOS MZ header this is located at offset 0. This section is purely placed there in case the file is executed on a system running the Disk Operating System (DOS). Originally when the PE format was developed by Microsoft many systems were running DOS operating systems. This header's sole purpose is to enable DOS systems to recognise the file as a PE file and a valid executable (Elisan, 2012). In Figure 2.2, the second section of the PE file is known as the DOS stub. this section assists the DOS MZ header to allow DOS systems to be able to recognise PE format files as valid executables. The standard for this is to display the following piece of string "program cannot be run in DOS mode". The PE header is the third section within the PE file format this section contains information regarding information needed by the PE loader. . That location is reserved for the DOS MZ Header and the DOS stub. The start of the PE header is found at offset 0x3C which is relative to the start of the PE file. Upon the execution of a PE file within Windows the PE loader refers to the PE header, it bypasses the first two sections. Contained within the PE header as can be seen in Figure 2.2 there are two structures namely the IMAGE_FILE_HEADER and the IMAGE_OPTIONAL_HEADER. The section IMAGE_FILE_HEADER contains the information the regarding the following:

Figure 2.2: PE header structure

The IMAGE_OPTIONAL_HEADER is the second substructure which is contained within in the PE File header as can be seen in Figure 2.2. This section contains information which allows the individualised files to be implemented. This contains critical information which is needed for the PE file to run successfully. The information contained within the section is shown in Table 2.1, for the purpose of this research only the most important were listed: The IMAGE_DATA_DIRECTORY enables the PE loader to identify specific sections without needing to process the image sections. The Sections table within the

PE file contains the information needed by the PE loader to load the sections. The IMAGE_SECTION_HEADER also contains common fields, for the purpose of the research only the most relevant have been listed in the Table 2.1 and Table 2.2 .

Table 2.1: IMAGE_OPTIONAL_HEADER information

Offset	Field Name	Description
0x18	Magic	This indicates whether the file is an executable file or not
0x1A	MajorLinkerVersion	The version of the linker used to produce the executable file.
0x1B	MinorLinkerVersion	This contains the minor version number meaning the value after the dot.
0x1C	SizeOfCode	The rounded-up size of all the code sections combined.
0x28	AddressOfEntryPoint	This is the address that the loader will execute the file. This is a relative address which can be found in the .text section.
0x34	ImageBase	This is the address which will be used to map the file when it has been executed.
0x70	LoaderFlags	This is the field which contains information regarding the debugging of the code.

Source:https://docs.microsoft.com/en-us/windows/desktop/debug/pe-format

Table 2.2: IMAGE_SECTION_HEADER - File Content

Offset	Field	Description
0x00	Name	Contains the 8 character value of the name of the section
0x08	Misc	The virtual size which would be loaded into memory for each section.
0x0c	Virtual Address	This is the address of the first byte which will be loaded into the memory which is relative to the image base.
0x24	Characteristics	Contains the flags of the characteristics of the image.

Source:https://docs.microsoft.com/en-us/windows/desktop/debug/pe-format

2.7 Malware

The term malware has been the subject matter, which has become the new word in the Information Security field. The definition of this is perhaps harder for individuals to agree upon. The definition which is considered the most acceptable is the definition published by Savage *et al.* (2015) as in the NIST SP 800-83 “Guide to Malware Incident Prevention and Handling for Desktops and Laptops, in which the term malware is described as follows: “Malware, also known as malicious code, refers to a program that is covertly inserted into another program with the intent to destroy data, run destructive or intrusive programs,

or otherwise compromise the confidentiality, integrity, or availability of the victim's data, applications, or operating system." According to Arntz (2017) there has been a drastic increase in malware related incidents from 2007 which still saw minimal malware incidents to 2017 which showed a drastic increase of malware incidents. This is an indication that the security has not developed in equal proportion to the malware. One could deduce that malware has grown sharply whilst the security in organisations has not kept up.

The researchers from Arntz (2017) indicated that in 2016 they collected worldwide approximate 127 million different malware strains. This was the first time that there was a decline in new malware strains. There has been a notable 22 million new malware strains collected in the first quarter of 2017. The forecasted rated calculated in millions shows that 2017 is the highest in the last year for new malware strains. This shows that this trend is set to only increase unless more research is done to understand and better fight the information security battle against malware.

2.8 MITRE ATT&CK Framework

The detection of post-compromise intrusion is an important part of examining the effects of malware on a network. The MITRE ATT&CK framework has evolved methods for mapping malware behaviour. This is done by using a behavioural based threat model which identifies methods used by adversaries. Malware authors are often part of advanced persistence threat actors and rarely act on their own (Johnson, 2018). Strom *et al.* (2017) indicates that contemporary network security approaches have had short comings in the detection of advanced persistent threat tactics. The use of anti virus is not reliable in the detection of custom malware or tools being used to exploit the victim's machine. The use of code obfuscation and techniques used to evade detection by malware authors furthermore make the use of anti virus as a detection tool for malware inefficient. Malware operators have adopted the approach of *'living off the land*'. This framework makes use if a threat based security approach. Using a behavioural methodology which is guided by five principals which weredeveloped after extensive research.

2.8.1 Principles of the threat based security

The first principle of the framework is the detection of post-compromise indicators. This is the post mortem analysis of artefacts which were left behind after access or exploitation has taken place on a victim machine. This allows for a broader picture to be painted of what happened during the incident. This can be used in relation to malware analysis to examine what happened post encryption. The second principle is to focus on the behaviour to create indicators of compromise. This is done by focusing on the signature of an incident which include the method of exploitation and tools used. This principle focuses on the known signature of an attack and evolves this with each new post-compromise examination. The third principle is the creation of a threat based model which can be evolved and

ensure that the information gathered in the previous principles can be collated into a single model which is realistic and relevant to adversarial behaviour. The fourth principle is the iteration by design which is the mapping of the tools and techniques used by the adversary which allows for the evolution and expansion after each new attack. The success of this approach for successful mitigation against these threats is that with each iteration there is evolution and refinement of the model to account for changes in adversarial conduct. To defend against malware or advanced persistent threats one has to understand the methods and processes used by them. The fifth and final principle of the framework is to develop and test these in a realistic environment. This principle is the analytical development which can be simulated and tested in a production environment to test the environment's ability to protect against these attacks (Strom *et al.*, 2018).

2.8.2 Tactics

This framework represents a high level mapping of tactics used by advanced persistent threats and malware. The techniques which is included in this framework are not only relevant to exploitation by advanced persistent threats but can be used to map the behaviour of malware in terms of tactics used. The inclusion of this framework is to further support the hypothesis that there are similarities within the methods used by malware authors to exploit Windows (Strom *et al.*, 2018).

1. **Persistence :** The access, action or configuration changes which allows the adversary to maintain persistent access on the system.
2. **Privilege Escalation** : This is a technique used to obtain a higher level of permission on a system. This is done for an adversary to be able to execute system level processes which need root access.
3. **Defence Evasion :** The purpose of exploitation and malware is to evade detection this can be unique to each adversary and is part of their defence strategy.
4. **Credential Access :** This is the process of using credentials to gain access to an enterprise.
5. **Discovery :** Techniques used by the adversary to gather information which allows him access to the internal network.
6. **Lateral Movement :** The lateral movement within a network allows the adversary access and remote control on a system within a network.
7. **Execution :** This is the methods used by the adversary to execute controlled code on a local or remote system.
8. **Collection :** This is the identification of methods which are used to gather information which can include sensitive information which is ear marked for exfiltration.

9. **Exfiltration :** The attributes and techniques used which aids the adversary to exfiltrate information by removing it off the network locally or remotely.

10. **Command and Control :** This shows how the adversary communicates to the systems which are under their command within a network. These can include legitimate protocol such as HTTP which is used to carry the information to the command server.

2.9 Previous work on Ransomware Families

This section is work which has been identified on the Ransomware families CERBER, MAKTUB and LOCKY. The primary aim was to identify publications or previous work done in an academic domain on these strains. The works which have been identified are however not academic in nature but industry tech reports. There is a lack of notable academic publications which are strain specific. The information which were examined is information relating to the static and dynamic analysis of the malware strains.

2.9.1 CERBER

Hasherezade (2016a) describes the CERBER as a new but mature variant of Ransomware. The author also explains that this strain brings justice to the name chosen as a mythical creature. CERBER is one of the first Ransomware strains that have been actively sold on the underground Russian forums - a Ransomware as a service model. The CERBER has since 2016 become a contentious adversary for the top Ransomware family taking the place of LOCKY for a brief period. According to Barkly Research (2016) therehas been a new release of CERBER every 2.4 days. The developers of CERBER instead of taking on the approach to distribute the attacks themselves, they have taken the approach to sell this as a service. This is done by means of an affiliates program whereby the authors receive a portion of the income generated. This allows for them to focus their efforts on building newer and better variants. The success in this is that CERBER is one of the most prolific Ransomware families (Barkly Research, 2017b). According to Barkly Research (2017b) there was 150,000 Windows users were infected in July 2016 alone. This was during the month of July and Check Point managed to track 161 active CERBER campaigns. These were all delivered via exploit kits which managed to infect the 150,000 Windows users.

CERBER has infected approximately 150,000 victims with the total profit estimated at $195,000 monthly until April 2018. These malware operators ensure that each campaign which is run is evolved from the previous campaign meaning, a new different distribution method and a unique packer. The most notable CERBER campaign run targets users in China and South Korea and is distributed using the MAGNITUDE EXPLOIT KIT (Anubhav and Ellur, 2016). This information was detailed in the Section 2.9.1. This would indicate that CERBER is certainly no longer an emerging threat however it is the leader in the RaaS ecosystem and this is the reason that it has been chosen for inclusion into this study. CERBER has shown that it has a model which has been proven multiple times to be one of the

biggest in the world, furthermore they are sophisticated in the methodology which is used for the money to flow (Barkly Research, 2017b).

According to statistics collected by Barkly Research (2017b) CERBER accounted for a fourth of all Ransomware attacks in late 2016 and early 2017. This shows that CERBER has surpassed LOCKY by 15% becoming the poster child for Ransomware. The theory is that by employing the Ransomware as a service model they are able to be widely spread while still making 40% profit on each campaign run. This is a very sound and smart business model that these actors are following to maximise infection and maximise profits (Barkly Research, 2016). The CERBER is said to have raised roughly $195,000 through the campaign and the cut for the developers was $78,000 (Barkly Research, 2017a). Segura (2017) explains that most CERBER was delivered by use of the MAGNITUDE exploit kit. This is but one of many different ways CERBER is distributed.

The CERBER employs a standard attack process. This process is explained by Anubhav and Ellur (2016) in eight steps:

1. The target receives the malicious document and opens it.
2. The Macro within the document is invoked using Powershell in hidden mode.
3. The control is passed to the Powershell console which then connects the Ransomware to the disk of the victim.
4. Once the connection is successful the Ransomware is written to the victim's disk.
5. The Powershell executes the Ransomware in hidden mode.
6. The malware then proceeds to configure the multiple concurrent persistence mechanisms to ensure full control of the victim's machine.
7. The executable exploits the native Windows utilities such as the `vssadmin` and `WMIC` utilities.
8. The victim's files are encrypted and the ransom note is displayed

CERBER comes packed with a Crypter/FUD which is used to obfuscate the code. Even when the researcher Hasherezade (2015) unpacked the code it was found that the specific strain of CERBER employs encrypted strings and will decrypt these as they are needed. CERBER has a specific format which is used to decrypt these strings and the format of this can be seen in Figure 2.3. CERBER comes with a resource which is encrypted and stored as RC Data this is then decrypted using a dedicated function which can be seen in Figure 2.3. Once the resource has been decrypted it is found to be a JSON file which contains the configuration file (Hasherezade, 2016c).

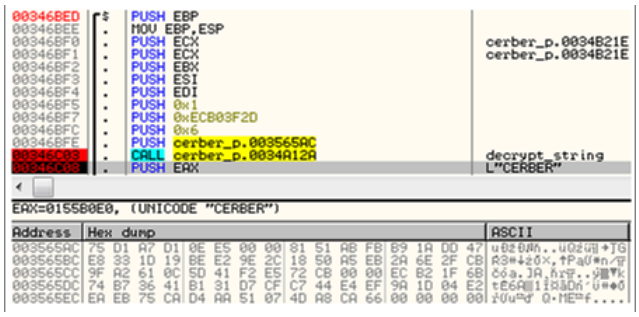


Figure 2.3: CERBER Assembly code portion showing decryption keys (Hasherezade, 2016a)

The first part of the configuration file which was extracted by Hasherezade (2016a) contains a list of countries which have been blacklisted . This means that CERBER will not execute in the following countries as can be seen in the listed countries below. It can be noted that the countries on the blacklisted list are all situated in the Eastern European countries. This could indicate that the Ransomware which has been famed to be connected to the APT 28 group which originates from Russia is linked to the CERBER (Davis, 2016). The following countries are excluded from attacks; Armenia, Azerbaijan, Belarus, Georgia, Kyrgyzstan, Kazakhstan, Moldova, Russia, Turkmenistan, Tajikistan, Ukraine, Uzbekistan (Hasherezade, 2016a) this is detailed in Appendix C.5. CERBER configuration file contains a list of folders which the Ransomware is set to target this list is detailed in Appendix C.1. These are generally paths where users will save data and where Windows settings data is stored. This shows a design for Windows infection. This CERBER may encrypt your files but it does leave you access to your system in some form and does not lock you out (Barkly Research, 2017b). Also contained in the CERBER configuration file is a list of domains to use to do geolocation of the victims IP address. This list is detailed in Appendix C.2. The three geolocation services used are legitimate services which allow for the location of the external facing IP address to be determined by the CERBER Ransomware these can be seen as detailed in Appendix C.3.

AS CERBER has evolved significantly from CERBER V4, which was first seen in 2016 , to CERBER V6 currently there have been leaps made in better execution. CERBER 6 has shown to employ methods in which the Ransomware does check the affected system for firewalls, antivirus or spyware. This version goes beyond just checking for these indicators it can now configure the Windows firewall configurations, this is done to specifically ensure that these tools are not able to communicate with the specific vendor databases online. This further displays the evolution of this specific malware to circumvent the machine learning.

CERBER comes packed with a hard-coded public RSA key which is detailed in Appendix C.4 which is taken from the configuration file. This is decrypted using Base64. The key is imported using the `CryptImportPublicKeyInfo` , this is the native Windows function that converts and imports a public key's information and returns the handle for the public key. The configuration file of the CERBER Ransomware makes mention of a 'rsa_key_size:576', however the researcher Hasherezade (2016a) indicates that it is in fact a 2048 bit key with a blob size of 276 Bytes.

CERBER has a rather unique method of installation in that it is able to bypass the Windows User Access Control (UAC). The UAC as it is called was introduced by Windows to enable users who are not administrators by allowing a user to be able to escalate their privileges (Cobalt Strike, 2014). The CERBER Ransomware when executed will rename itself to a randomly generated filename which is calculated in the following format: ‘‘`[random char] * [random char]exe, id ‘‘p*hexe]`. To be able to prevent a user from finding this file by its creation date it, changes the timestamp of the malicious file to inherit the date created from the `Kernel32.dll` file. Once the sample has been executed and installed it terminates it original file and deploys the dropped copy (Barkly Research, 2017b). The UAC is bypassed by using the UAC and Windows against itself.

CERBER searches for an executable within the path `C:\Windows\System32` as these can auto elevate their privileges this is also a method used in pretesting. The table for the executable is imported and the dll is hollowed out and replaced with the malicious code. This allows the malware to execute malicious code while it is masked as a legitimate executable. The dll which has been compromised is copied into the %TEMP% folder and patched. This now gives CERBER auto elevated privileges on the victim machine. The code is then also injected into the `explorer.exe` which is responsible for executing the UAC and afolder is then created containing both files. Once CERBER is successful in bypassing the UAC the property of the sample is set to `Cerber_uac_status` and added into a `Shell-TrayWnd` the original file is then deleted along with the dropped samples and exists(Hasherezade, 2016c). This process can be observed in Figure 2.4.

Figure 2.4: Assembly code of UAC bypass used by the CERBER Ransomware(Hasherezade, 2016a)

Anubhav and Ellur (2016) states that CERBER has been observed to delete and disable the safe boot options of Windows. This is not done with any injected code but manipulating the native Windows executables. CERBER is most definitely one of the most powerful written malware with a specific attention to detail. It offers the authors and their clients a wide variety of customisation options which makes the analysis of this harder as it for-

ever changing. According to Hasherezade (2016a) the malware authors behind CERBER are not new kids on the block but have been in the business of developing malware for a while. With the tempo that this Ransomware has evolved it is likely to become more sophisticated.

2.9.2 MAKTUB

There were limited comprehensive papers found on the examination done on the MAKTUB Ransomware family and two blogs were found which contained technical aspects of an analysis and as such they were the only two references. MAKTUB is described to have a beautifully designed graphical user interface. The name of this malware is Arabic in origin and the meaning is *"this is written" or "this is fate"*. This is said to be a play of adding psychological meaning to the Ransomware by indicating to the user that it isinevitable tthat encryption will occur.(Hasherezade, 2016c). Abrahams (2016) describes MAKTUB as a standard variant of Ransomware in that it encrypts the data and asks for bitcoin payment. This specific malware family places a time limit on the initial payment amount. Once the allotted time has passed the ransom amount is increased. The malware as can be seen in Figure 2.5 is spread as an executable which is disguised as a document intended to be a Terms of Service document named `TOS-update-2016-March-18.scr` (Hasherezade, 2016c). The mode which this Ransomware follows is that the document with legitimate text opens. Whilst the Ransomware is busy working in the background the victim is reading the rather lengthy note as can be seen in Figure 2.5.

MAKTUB once finished with its encryption process will display a very well graphically designed ransom note indicating a time limit and details on payment methods. This can be seen in Figure 2.6. The MAKTUB strain has a unique feature in that it does not require communication with the command and control server related to the malware. This variant of Ransomware has the ability to be able to encrypt a machine which has no connection to the internet. To better understand these features the author Hasherezade (2016c)conducted a malware analysis on the strain of MAKTUB.

Hasherezade (2016c) presented analysis looking into the inner workings of the MAKTUB to try and obtain relevant information without executing the code dynamically. The finding was that at first glance the MAKTUB sample was packed using s very well written crypter/-FUD. A Crypter/FUD is described as an additional built-in defensive technique which protects the malware from being detected by anti-virus software's Hasherezade (2015). This is done by deceiving pattern-based or even behaviour-based detection engines by masquerading as a harmless program before unpacking. The methods used by MAKTUB Ransomware is to make arbitrary calls to meaningless API's for example to read random registry keys, once this process is completed the call is made to the function which allocated virtual memory which is either `VirtualAlloc` or `VirtualAllocEx`. The following execution which takes place is to assign this address for the decrypted payload to use (Abrahams, 2016).

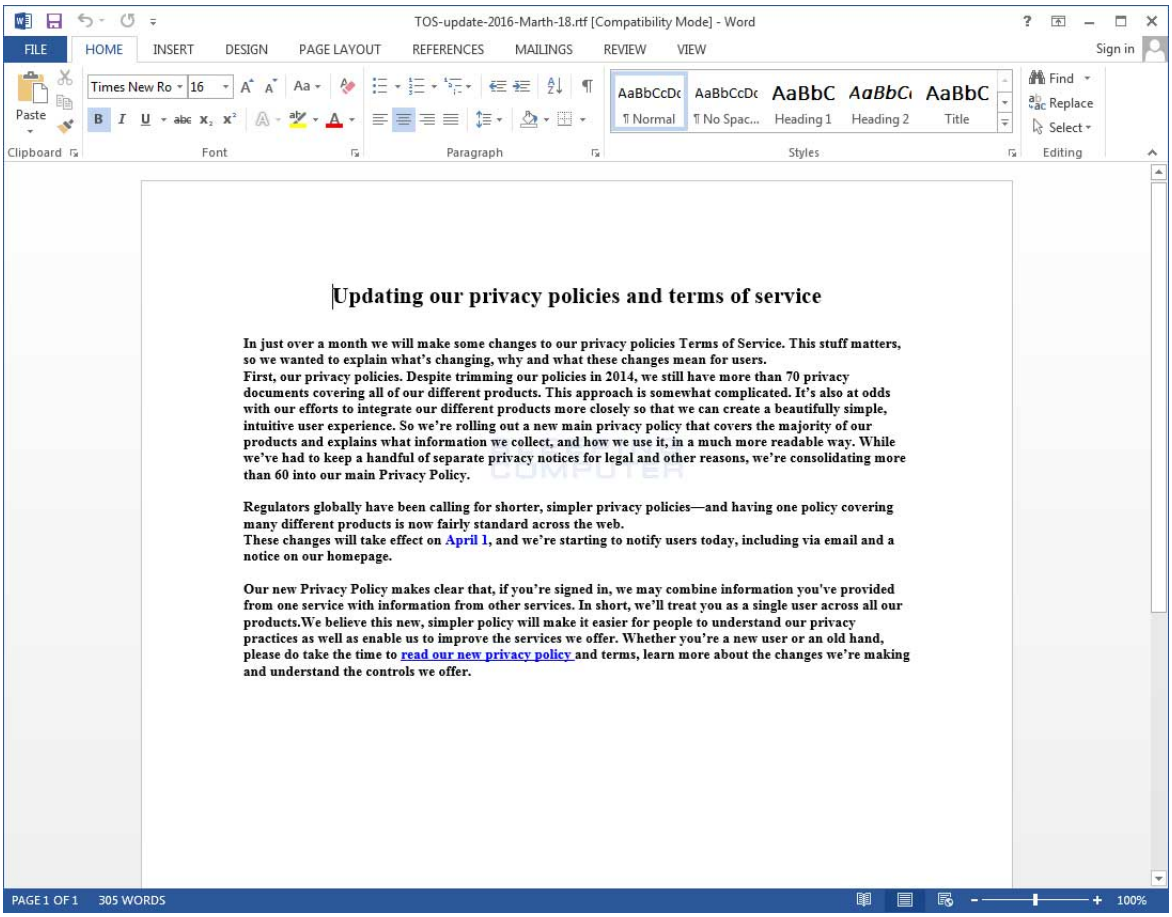


Figure 2.5: Malicious MAKTUB Ransomware file
Source: https://www.bleepingcomputer.com/news/security/the-art-of-the-maktub-locker-Ransomware/

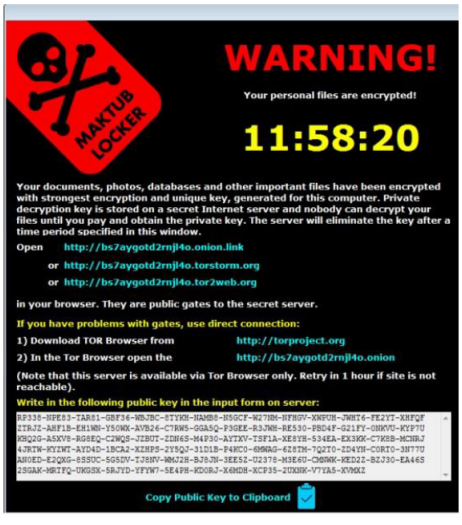


Figure 2.6: MAKTUB Ransomware note (Hasherezade, 2016c)

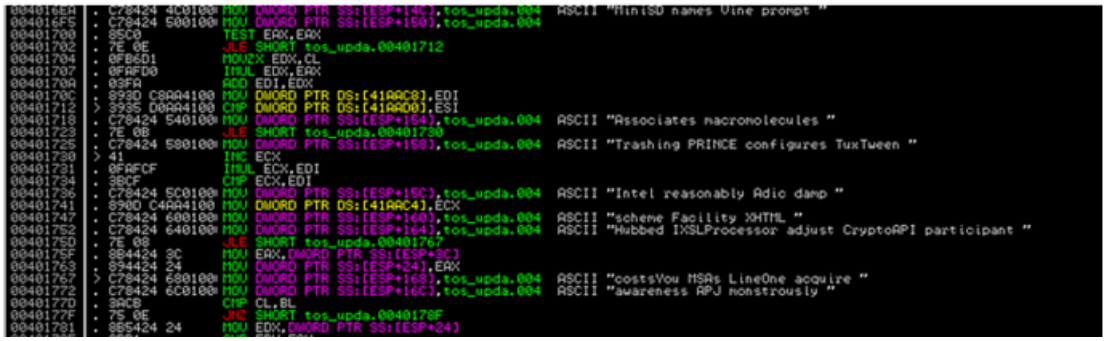


Figure 2.7: Initial harmless code (Hasherezade, 2016c)

The first code which is executed has the main purpose to deceive malicious tool detectors. Once the code is executed the code is completely overwritten by the extracted code. This is also from research done not the actual code but yet again a deception technique. In Figure 2.7 the code responsible for the unpacking and executing of the non-malicious TOS document, this is done by unpacking it first and placing it in the %TEMP% folder as a cabinet file (Hasherezade, 2016c).

The actual malicious code portion is unpacked dynamically in memory and will not be visible until this process has been completed. The method of the crypter/FUD is to unpack the PE File is loaded into a continuous area of memory and dynamically allocated and then used as a complete new virtual section. This trick is often used to defeat the process of dumping the original payload. This process also protects the malware from automated dumping tools Hasherezade (2016c).The actual malicious code portion is unpacked dynamically in memory and will not be visible until this process has been completed. The method of the crypter/FUD is to unpack the PE File is loaded into a continuous area of memory and dynamically allocated and then used as a complete new virtual section. This trick is often used to defeat the process of dumping the original payload as can be seen in Figure 2.8. This process also protects the malware from automated dumping tools (Hasherezade, 2016a).

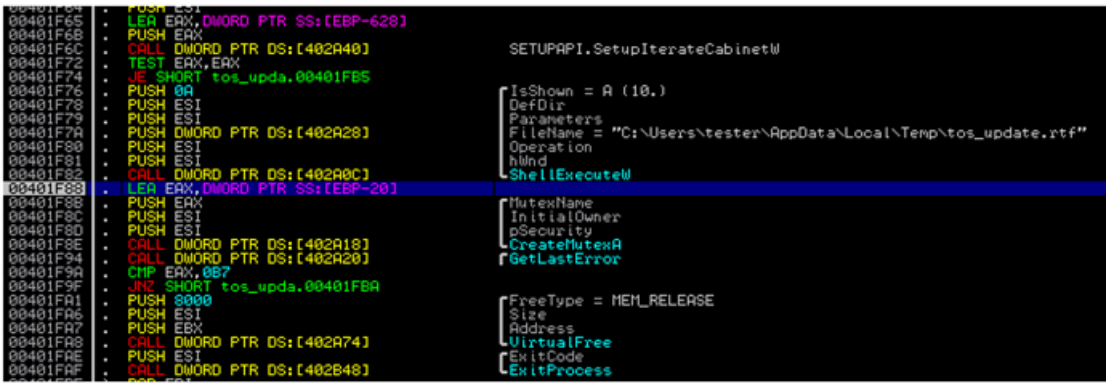


Figure 2.8: Code portion responsible for unpacking and executing the TOS document (Hasherezade, 2016c)

MAKTUB follows a distinct process in how it behaves. This strain compiles a list of each

file on the victim's machine this is done one by one. Each of these are first checked against the list within the configuration file that contains the list of restricted paths and attacked executables. The code which was extracted by Hasherezade (2016c) can be seen in Figure 2.8. This shows that when a new file is created a call is made to the function `CreateFileA` which is loaded into the `EAX` register dynamically. Once the function is complete the new file is created and assigned with an extension.

At this stage both files exist on the victim's machines, the new file which was created in Figure 2.8 has the size value of 0. After this process the original file is compressed and encrypted. This will be added into the newly created file which was created in Figure 2.3. When the process of compression and encryption is done the original file is delete from the host machine (Hasherezade, 2016c). After the process in Figure 2.8 has been completed for all the files on the victim's machine, the MAKTUB Ransomware will display the ransom note. This is done by displaying a graphical user interface which contain the ransom note.

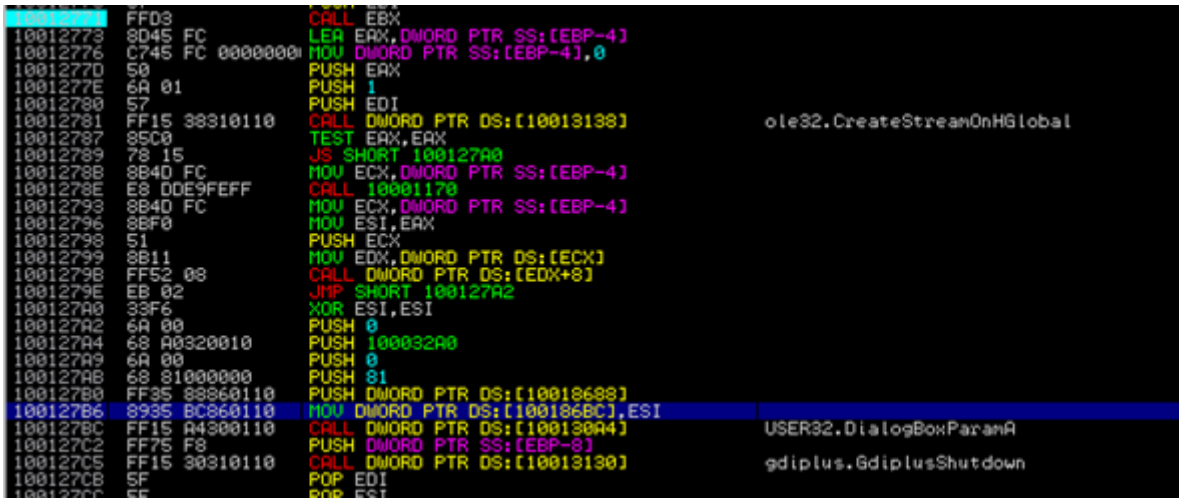


Figure 2.9: MAKTUB Ransomware Assembly Code responsible for the ransom note (Hasherezade, 2016c)

Most malware has a list of countries which are blacklisted. This means that these countries will not be infected or does not farm part of a specific campaign that is run. This MAKTUB strain was found to check the keyboard locale list from registry and will not execute if the "*Russian {value 0x419=1049*" is found and the malware is then designed to exist without infecting the machine. This means that if this value is there the malware has been programmed to exist this can seen in Figure 2.10. MAKTUB locker according Abrahams (2016) only uses one key which is hard-coded key and it does not create unique key for each victim. The authors further explain that the key which is used for the encryption process is locally generated. The first initialised relationship between his key and the key which will be used for encryption uses the `PROV_DH_SCHANNEL`. In Figure 2.11 Hasherezade (2016c) explains that the MAKTUB Ransomware strain firstly makes use of the `CryptGenRandom`. which is a Windows function that fills the buffer with random cryptographic bytes .

MAKTUB then proceeds to make a MD5 hash value of the randomly generated bytes using the `CryptCreateHash` windows function. The `CryptCreateHash` function allows the hashing

of a stream of data. The final process is to call the function `CryptDeriveKey` which then takes the MD 5 hash value and converts that to a 256 bit AES key. The `CryptDeriveKey` is a function to create cryptographic session keys from base data values. These processes can be observed in Figure 2.12. Once this process has been completed the AES key along with the random extension is concatenated together and the buffer prepares to encrypt it using the RSA encryption (Hasherezade, 2016c). In conclusion the Hasherezade (2016c) states that the MAKTUB strain is developed by authors that are professional in that, the complexities of the code design and execution is well polished. This family of Ransomware is done is designed with precision and well thought out protections.

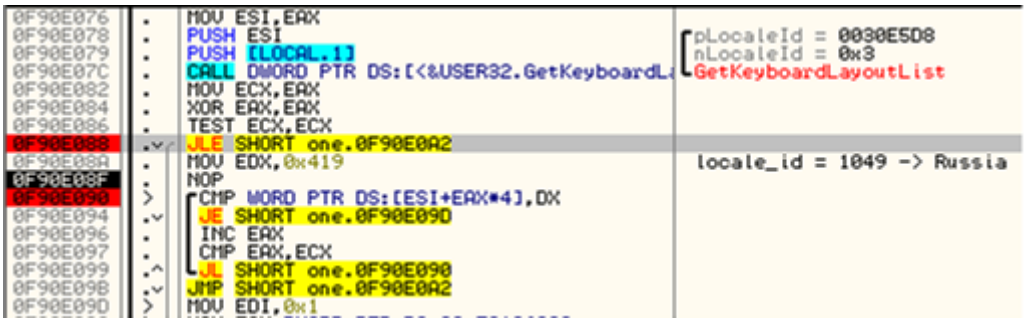


Figure 2.10: Assembly code showing blacklist protocol for Russia (Hasherezade, 2016c)

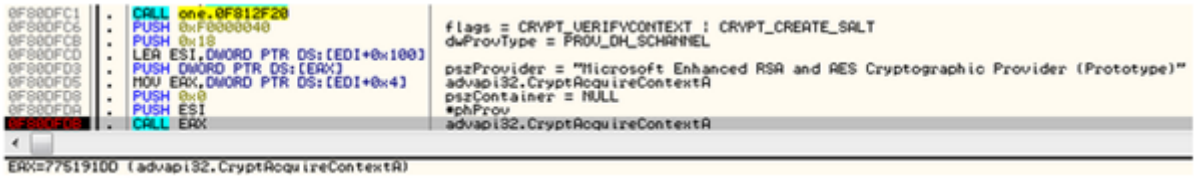


Figure 2.11: Assembly code portion that gets the `CryptGenRandom` function (Hasherezade, 2016c)

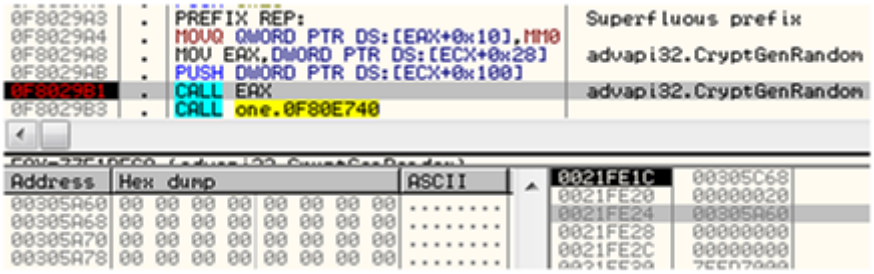


Figure 2.12: LOCKY Ransomware assembly code portion showing the use of `CryptDeriveKey` which it will convert the MD5 hash into a 256 bit AES key (Hasherezade, 2016c)

2.9.3 LOCKY

The LOCKY Ransomware has made a big comeback in 2017 showing some new sophisticated methods of bypassing and evading sandboxing. According to Vlad (2016) there has

been a 64% increase of LOCKY Ransomware related encryption since the first quarter in 2015. In one month according to Barkly Research (2017a) there was a spike in LOCKY activity in March 2016 which showed an infection of 56,000 machines and was seen to have collected $209 million been paid to the ransom campaigners. In the second quarter of 2016, it was estimated that 7 out of every 10 emails contained an attachment which delivered the LOCKY Ransomware. These were typically hidden in malicious word document (Barkly Research, 2017a). LOCKY Ransomware has been seen to leverage off of macros and has been seen as one of the most successful Ransomware families (Barkly Research, 2016). They are pretty much the Big Mob family of Ransomware. The LOCKY Ransomware on an average demands on average 0.5 – 1 Bitcoin for the release of individuals' machine. This amount varies for bigger organisations. The infection rate in 2016 has been 90,000 devices per day. The attack vector according to county the following were most affected (Balaban, 2016).

There has been a return of LOCKY Ransomware since 2016 (Griffin, 2016a). This shows a level of sophistication's in their methods in evading sandboxes (Dell Secureworks, 2015). The new LOCKY variant which has taken the cyber world by storm does no longer trigger by running the macro itself. This variant idles until the user closes the word document to start to invoke the set of commands to start the infection process (Segura, 2017). This process can be seen in Figure 2.12. The payload is the downloaded and then launched from the %APPDATA%, once the command below has been successfully executed the ransom note will be displayed which can be seen in Figure 2.14.

```
C:\Windows\System32\WindowsPowerShell\v1.0\powershellexe –nop –exec Bypass –Command (New–
    Object System.Net.WebClient).DownloadFile(http://newhostrcm[.]top/admin.php?f=1, $env:
    APPDATA + \sATTfJYexe); Start–Process $env:APPDATA\sATTfJYexe.
```

The payload is executed by using Poweshell scripts which attempts to facilitate contact with a predetermined set of domains. Griffin (2016a) explains that there is a similarity between the LOCKY Ransomware and DRIDREX TROJAN in the method of delivery. These similarities according to the researcher show a possible teaming up of malware operators.

```
Sub AutoClose()
  Dim Flying_Squirrel As String
  Dim Duqu As String
  Duqu = "Shell"
  Flying_Squirrel = "pqwgsuhglm1hzf""0qos!-G{ef""Bzpbvs .Eooobod *Ney0Rdmfcw!Tzutep/
      Pfv0ZecClleov,1DrxqopafFilg*)jtws:20oexkrvvrco0urr1bgnip0pjp?h>4*- &hpw;BPPDCVB#.#(]sAVVfLZ1fzg'+;
      Twbrw/Ssqehvv""'hpw=DPQFAWC(^vBTTgJ[1h{h)=#"
  Application.Run "Octopus", Flying_Squirrel, "WScript" & "." & Duqu
End Sub
```

Figure 2.13: Idle process before invocation of commands LOCKY (Hasherezade, 2016b)

Segura (2017) states that even though the technique is not sophisticated it highlights the constant change that the malware authors go trough to evade more security methods. The authors liken it to a cat and mouse game between malware campaigners and security professionals (Segura, 2017). LOCKY variants are all packed with some form of crypter which means the code is unreadable in its native form. Hasherezade (2016b) indicated that

once the external defences have been unpacked the core dll was not obfuscated and could be read. The researcher further found evidence which explained the encrypted network traffic. The RSA key as well as the ransom note is fetched from the server by using a HTTP based protocol. The sample which was examined by McNicholas and Cunningham (2017) contained three hard-coded IP addresses :

Table 2.3: Hard-coded IP addresses contacted by LOCKY

IP ADDRESS	COUNTRY	ASN	REVERSE DNS
31.41.47.37	Russia	AS56577	oiwerpil.qw
188.138.88.184	Germany	AS8972	xray730.startdedicated.net
85.25.138.187	Germany	AS8972	echo509.dedicatedpanel.com

LOCKY employs a Domain Generation Algorithm which is described by Arntz (2016a). This technique is used by malware that is dependent on a fixed domain or IP address. Because these are easily blocked rather rapidly, however by switching between domains at a regular interval it delays the finding of these domains. An alternative name for this process is Domain Fluxing which refers to an older method which was used to abuse the DNS load balancing system (Arntz, 2016a). LOCKY Ransomware has three main attack types which can be seen in the assembly code in Figure 2.16. The three areas of attach are the fixed drive within the machine, any removable drives you may have plugged in at the time and ram disks. LOCKY also has the ability to map network shares using the `WNetAddConnection2` function that allows a connection to be make to a network resource and which can redirect a local device to a network resource. The static analysis done by Vlad (2016) indicates that the LOCKY Ransomware employs code checks for MMX floating points of SSE instructions which is used to hypothetically break out of Heuristics based AV emulators. The second interesting section of code is the introduction of the INT3 used in a loop which causes the debugger to stop at the break point for the INT3. This causes the AV emulator simply to keep running in a loop (Vlad, 2016).

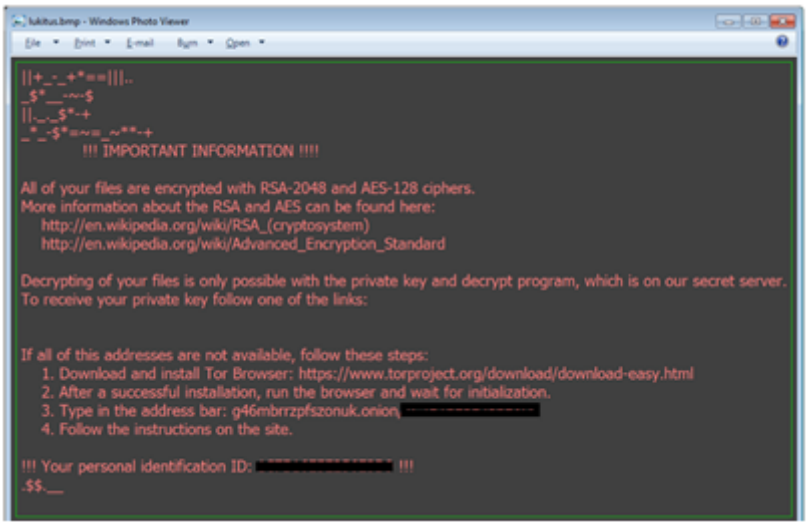


Figure 2.14: Ransomnote LOCKY

LOCKY is most commonly delivered via a MS office document or JavaScript phishing campaign. The initial payload is a 32-bit executable binary which contains the encrypted core packed dropped file which is responsible for downloading the payload (Hasherezade, 2016b). Once the file is opened the initial file disappears and runs a dropped copy of the file, this file is however renamed to `svchost.exe` which is dropped in the %TEMP% folder this is done to aid in masking the infection and avoid detection(Hasherezade, 2016b).

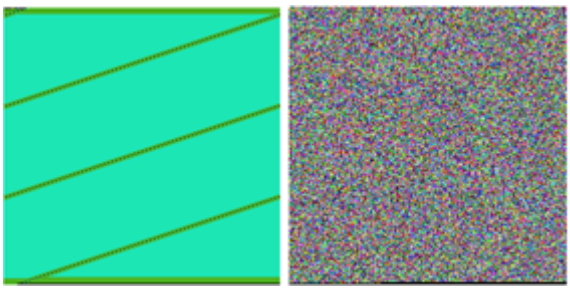


Figure 2.15: Visualization of encryption entropy between an encrypted and non encrypted file

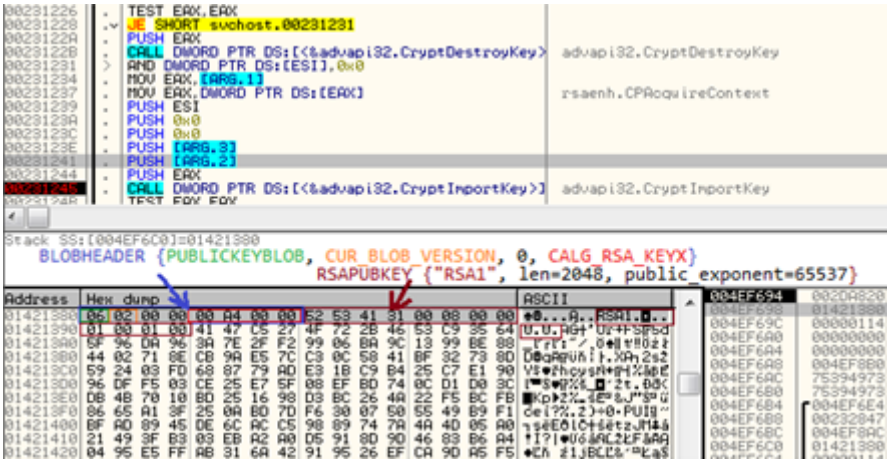


Figure 2.16: LOCKY fetches the RSA key which will be used to encrypt the AES key

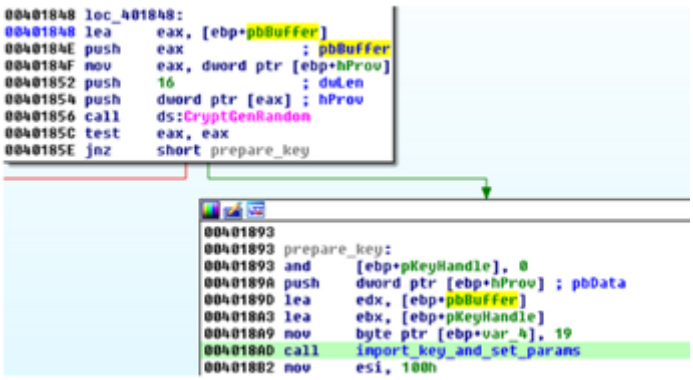


Figure 2.17: LOCKY randomly generates a key for each file and imports them

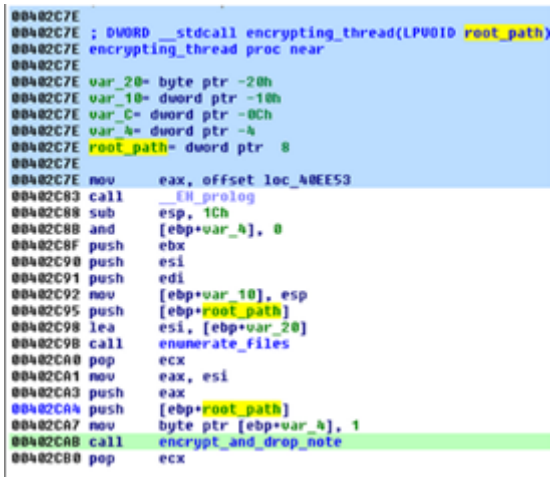


Figure 2.18: Enumerated list for files to be encrypted

The encryption process which is followed by the LOCKY Ransomware is those files which are encrypted are fully renames unlike other Ransomware strains. The extension used by the LOCKY is a custom extension .LOCKY and this has been a constant since the first sighting of this malware in the wild (Hasherezade, 2016b). These encrypted files contain high levels of entropy as can be seen by in the Figure 2.15. In these figures the author Hasherezade (2016b) shows the entropy of the un-encrypted file of the left and the entropy levels of the file which has been encrypted (Griffin, 2016b). The claim is made by the attackers that LOCKY uses both the RSA and AES algorithms for encryption. This is done by using the native Windows Crypto API, the first part of the encryption process is fetching the RSA key which is 2048 bits and this is imported.

The following step is to use the RSA key to encrypt the asymmetric AES which is randomly generated for every individual file. A unique element of LOCKY is that it compiles a list of file to be encrypted with their AES 128 bit key and then proceeds to encrypt files according to the list this can be seen in Figures 2.16 and 2.17. LOCKY collects the list of statistics which contains the summary of the total files which have been encrypted and the path of the files. This information is encrypted and sent to the command and control server (Hasherezade, 2016b). LOCKY creates specific registry keys , the most relevant key which is added is the autorun key which contains data specific to the victim such as the individual ID, public RSA key and text of the ransom note to be displayed. The key which is created in autorun will ensure that when the machine is rebooted the malware is set to automatically restart the malware (Hasherezade, 2016b). This information can be seen in Figure 2.19 and Figure 2.20.

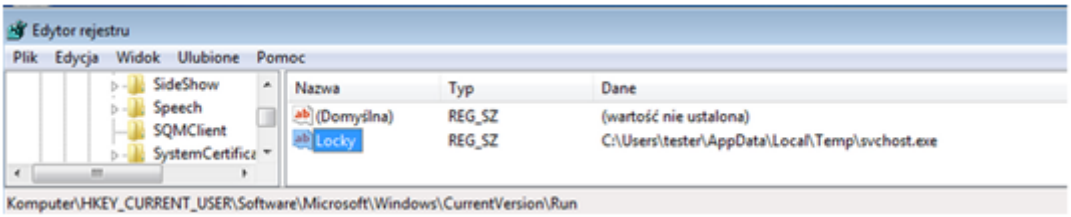


Figure 2.19: Data stored in the registry key by LOCKY

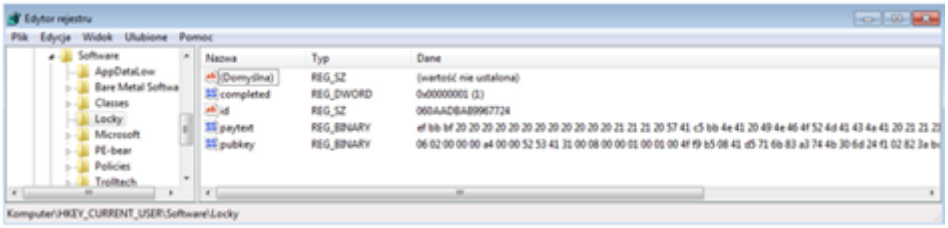


Figure 2.20: User specific info contained in Registry

The analysis done by Hasherezade (2016a) indicated that the network portion of the analysis had constraints as the full communication is encrypted. This can be seen in Figure 2.21. LOCKY has a simple communication protocol which consists of a POST request which contain parameters which are typical key=value format. The unique element is that the text is wrapped and then encrypted this is done by filling the parameters and calculating an MD5 hash value, both these values are then wrapped and encrypted together in Figure 2.22 the data can be observed before the encryption takes place. Once the response is received back it is also encrypted, for this to be parsed the MD5 hash value is validated and the packet is parsed (Hasherezade, 2016b). LOCKY has shown that it has managed to keep evolving. This shows a level of commitment to the attack in that the authors of this Ransomware employ new techniques with every new attack wave. Even though they have shows adaptability the maintain to do what other families do and follow the standard methods. This is a well written and even better executed piece of malware.

Figure 2.21: Locky Network Analysis TCP Packet

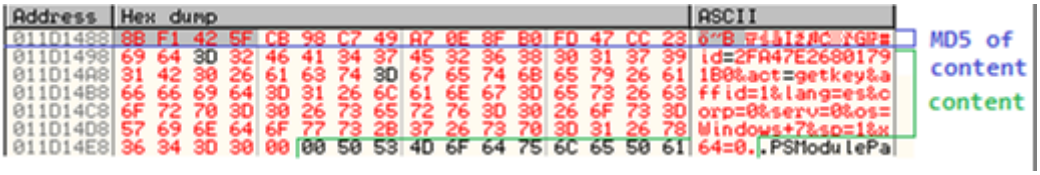


Figure 2.22: Example of wrapped request before it is encrypted

2.10 Summary

The previous work identified to have been done on the three Ransomware families, did not include any academic publications. The majority of the work that has been published has been done so online. This has been identified as a gap within the academic field that there has not be strain specific publications. The publication which have been cited in the sections which follow are all tech reports which have been published online by other malware analysts. The previous work specifically only focused on following a standard approach to malware analysis which included the static analysis and the dynamic analysis. There were no instances found where the approach detailed in Chapter 3 was followed. The work surrounding Ransomware in specific were associated or detailed statistics and the topic in a broad sense. The process of the MITRE ATT&CK framework which is relatively new yielded detailed references to the application of this framework to APT attacks (Strom *et al.*, 2018).

3

Methodology and Data

3.1 Introduction

This chapter sets out the methodology that the researcher used to complete this work. This section starts with proof of concept and system specification used during the examination of the three Ransomware strains. This follows an overview of the analysis approach which will be taken and the various elements used for the new proposed Hybridised-Malware analysis framework. The chapter also highlights the three strains which were chosen and how the selection of the data set was done.

This chapter is structured as follows. Section 3.2 details the set-up of the test machine which was utilised during this research. Section 3.3 details the analysis process which is proposed and details the extension of the current Hybridised-Malware analysis framework. Section 3.4 details the proposed methodology for the Static Analysis phase. Section 3.5 details the proposed methodology for the Dynamic Analysis phase. Section 3.6 details the approach to the examination and interception of the network communication phase. Section 3.7 details the forensic process to be followed during the collection and examination of the post-mortem artefacts left behind from the Dynamic Analysis phase. Section 2.8 details the details surrounding the identification and scoring using the MITRE ATT&CK framework in relation to the behavioural traits of the Ransomware strains. Section 3.9 details the collection of the data and the methodology used to refine and narrow down the selection of the data sets. The chapter concludes with a summary in Section 3.10.

The goal of this research was to investigate three strains of Ransomware from a forensic point of view. This was done by exposing a stand alone Microsoft Windows 7 environment, to the three different malware families to determine the commonality between the three

families. This chapter covers the definition of malware analysis, previous work was done by other researchers, techniques and tools used. Roundy and Miller (2010) describe the process of malware analysis as the analysis of malicious software that infects a computer or systems. They further explain in the research that one of the main goals for malware authors is to make the analysis of the malware as difficult as possible and make the analyst exhaust as many resources as possible. The other main goal for malware authors is to have the maximum income for their campaign (Ferrante *et al.*, 2016). This could potentially explain the reason why 90% of all malware binaries employ some analysis resistance techniques. An analysis resistant technique is a technique used to obfuscate information contained in the binary (Sikorski and Honig, 2012). These methods include run time unpacking the binary itself. Baier (2017) explains that security companies are facing an increase of new malware strains being released continuously.

They have to cope and defend against these malicious programs in order to protect their clients. To be able to cope with this influx of malicious software there has to be research done in the behaviour of this malicious software. Baier (2017) states that there has been an increase in the number of malware appearing each day. The behaviour of malware is studied through a process called malware analysis. This is a process in which a piece of malware is examined to determine how it behaves once it has been executed. There are two methods of conducting malware analysis; static malware analysis and dynamic malware analysis.

Cho *et al.* (2016) further explains that the analysis of malware is important to gain a deeper insight into the tangible inner behaviours of the malware. The analysis of malware is a notable manual and automated process which in itself requires a significant amount of patience and time. Akkas *et al.* (2017) further explains that there are two phases which aim to gain an overview of malware and its inner workings. The first phase is specifically there with the goal to gain information on the malware binary and to be able to anticipate the behaviour of the malware, the author's further state that the second phase is there to analyses the features of the malware whilst observing the behaviour of the malware in a live or virtualised environment. To be able to be in a position to stop malware from spreading, there needs to be and understanding on how the malware operates. This is achieved by examining and analysing the malware strains. When a clear understanding is achieved of the malware a solutions for better protection and prevention can be achieved. Even though malware analysis holds benefits which can aid in the detection and prevention of the spread of malware the opposing side is that it has limitations as well. In the static analysis phase of malware analysis the data collection is only effective if the data is free of encryption and data obfuscation. When malware is packages with encryption the need to decrypt is needed. This is why the malware sample will be subjected to a various selection of tools to gain the maximum information whilst minimising the risk.

3.2 System Specification and Proof

The hardware used for the purpose of the research was a physical laptop. The specifications were kept at a minimum to ensure that their data set which needed to be examined in terms of hard drive capacity and random access memory (RAM) was limited. The research made use of an HP Compaq 610, with an Intel Celeron, with 2 GB of memory and hard drive capacity of a 150 GB.

The laptop which was used contained a single hard drive which was initialised by replacing each sector with a zero value. This process was done using Paladin Edge[1] which is a bootable Linux environment that has the ability to zero out each sector of the hard drive disk. This is an important element to ensure no cross-contamination between experiments, as this ensures that the hard drive disk is clean and can be validated. The consideration was taken that the Ransomware variants are predominately written for the Windows operating system, and many of the tools used to monitor the processes are Windows based. The Microsoft Windows 7 operating system 32-bit due to the fact that most static analysis tools are designed to run on 32-bit (Elisan, 2012). A fresh installation image was created on the erased disk using Windows builtin recovery disk option to ensure that the process can be replicated. The replication is important to validate that the baseline has been followed with each new experiment. The operating system was loaded on an erased disk which has been validated. The machine was prepared to identify as a user machine. This was done by loading preloaded user files on the system such as documents, pictures. The purpose of adding user documentation to the test machine was to be able to simulate a real world attack where there was user data present. The aim was to keep these files unified and standard across the experiments.

3.3 Analysis Overview

Malware analysis has traditionally been the process of determining the underlying purpose and functionality of a given sample of code or binary (Dalziel and Crosby, 2014). The process of malware analysis traditionally has had specific steps which enabled the analyst to detect malicious code or binaries before the execution of this. The process which has been followed in malware analysis is that the code or binary would be examined in a dead state which refers to the binary as it is without dynamically executing it to determine the static or dead information which can be gained from it without executing it (Elisan, 2015). This would include file header information and strings. The next step which traditional malware analysis has followed is to execute the binary dynamically which in most situations would mean to execute it through as sandboxed environment would monitor the interaction with the virtualisation operating system. The main downfall of the process of using a sandboxed environment is that it runs for a short period of time, and with most environmentally aware malware they will not maliciously execute if it is determined that they are in a sandboxed environment.

[1] https://sumuri.com/product/paladin-edge-32-bit-version-6-08/

The process which is proposed is to use the traditional Hybrid-Analysis framework for the Static and Dynamic analysis of the malware (Bethencourt *et al.*, 2009). For the purpose of this work the researcher has chosen to extend two additional disciplines to the existing Hybrid-Model to be more comprehensive and include maximum analysis success. The two additional disciplines which have been identified are **Network Analysis** which is the analysis of network based communication and the forensic science discipline of **Digital Forensics**, this is important to be able to identify the data artefacts which have been left after the fact as this aids in the understanding of the interaction with the various areas of the operating system. The process which was followed in the examination of the malware samples CERBER, MAKTUB and LOCKY, is based on the new proposed Hybrid-Model.

3.4 Static Analysis

The first phase of the proposed extended Hybrid-Framework is the Static Analysis. The Static analysis section of the research would focus on the analysis of the malware without the execution of it. When a source file is compiled into a binary executable it has a PE Structure which is further elaborated on in the literature review section which is where the executable information is contained, think of it as the road map to how the executable is compiled to run Zeltser (2017). This process of the examination of this structured file is often done manually using various tools to parse out specific data or information from the executable file. The information from the original source code can get lost or obfuscated during the compiling of the executable.

Event Tracing is done using a built in tool named ETW which is used to do Kernel-Level tracing within Microsoft Environment. This allows you to log specific events. It allows the user to decide which events to register and which to consume in real time to assist with debugging processes, identify performance issues or manage log sizes. ETW allows the user to enable or disable event tracing this is done dynamically to ensure that the machine does not have to be reset.

There are various techniques used for the examination of static information contained within malware. The examination of malware in a static environment allows for a comprehensive analysis of the malware in a safe environment where there is no execution. The main pitfall or negative in relation to static analysis is that it is a manual process which is time consuming and when binaries are packed or obfuscation used it can alter or hinder analysis (Rocha, 2015).

The different disciplines within the Static Analysis phase can be broken down into Sub-Phases of the main discipline of Static Analysis. These sub-disciplines are as follows; and are discussed in detail below.

1. **File Fingerprinting:** The process of hashing a malware samples to accurately identify the sample to a known sample.

2. **Hard-coded Strings:** The examination of strings which are contained within the binary executable file.

3. **File Format Examine:** Is the examination of the format of the malware sample as part of the characteristics of the binary executable.

4. **Packer Detection:** Identification of the possibility of a packer which is manner of onfuscation used during the compiling process.

5. **Disassembly of the binary:** Dissembling the executable binary to assembly code to read the instructions and flow process of the malware.

3.4.1 File Fingerprinting

File Fingerprinting is done by examining the obvious external characteristics or fingerprint of the file itself. This would include hashing the file using a cryptographic hash calculator. Which is a tool that generates a hash value for a file using the selected SHA-256 and MD5 hashing algorithms. To determine whether the file examined matches the algorithm of known strains of this specific Ransomware. This can be done using the VirusTotal[2] online malware sandbox which calculates a hash value for the malware. This also aids to determine that the malware has not undergone metamorphosis and changed (Sikorski and Honig, 2012). SHA-256 is used for the identification of malware file names (Akira *et al.*, 2016).

3.4.2 Extraction Hard-coded strings

When software prints an output it normally draws it from the embedded strings which are hard-coded into the executable file. These strings are embedded in the compiled binary as readable text. This allows the analyst to draw certain conclusions regarding some internal aspects of the binary (Sikorski and Honig, 2012). The compiled binary of the Ransomware sample was examined using the following tool-sets to determine which one yielded the best results. The success of this will vary depending on the type of malware looked at and whether the specific malware is packed or data obfuscated.

3.4.3 File Format

The tools used were chosen on the basis of what they worked more favourable for. This means that each tool was used and only the ones which work best were utilised for each phase.When examining an executable meta data of the file can be leveraged to gather additional static information regarding the magic number of the UNIX system this file is important in the identification of the file type. This has been explained in Section 2.6.4 under the section which covers the PE Structure. This aids in the identification of the file

[2]https://www.virustotal.com/

type, compilation time, imported function, exported functions, strings, menus and symbols Elisan (2015). This is found within the PE format file, for the examination of this information the following tool-sets were used, the results varied in the amount of information extracted depending compilation factors.

3.4.4 Packer Detection

With the progression of malware, there has been an increase in the use of packers and mutation engines. This is widely used by the malware operators to evade the detection based on the signature detection. According to Jacob *et al.* (2013), 80% of the samples examined by the authors used packers or obfuscation techniques. Most malware authors would try and obfuscate or pack their code to ensure that it is not reversed or examined. When a program is packed the program displays very much different to what it would look like if not packed (Dalziel and Crosby, 2014). The detection of packers in this research was conducted using mainly using two different tool sets. The first tool which was used in the identification of whether the malware samples were packed was PEid[3] is able to detect the compiler along with whether a packer has been used, this tool has however not been updated since April 2011. The second tool which can detect the compiler and packer which was used is *DIE (Detect it Easy)*[4] was used to identify whether the malware sample was indeed packed and attempted to scan and identify the packer which was used if one was indeed used.

3.4.5 Disassembly of the Binary

One of the most important activities during the static malware analysis section is the disassembly of the executable because this allows for a more detailed look into the way the binary executable is set to run. It also allows for a sequence to be identified by the calls it makes. This is done using a tool such as *PE File Explorer*[5] which has a builtin disassembler based on the IDA disassembler. This is done by reversing the machine code to a readable assembly code language, this enables the analyst to examine the assembly code to identify the logic which the program uses (Yusirwan *et al.*, 2015). This allows for the intention of the program to be identified and can aid in the identification of possible malicious processes. The samples were also examined using an online visualisation tool VELES[6] this creates a visual representation of the binary outlay within the executable file.

3.4.6 Virus Total Static Analysis

VirusTotal[7] has the ability to perform static analysis and dynamic analysis on the malware sample. This includes scanning the malware sample performing anti-virus scans. It

[3]https://www.aldeid.com/wiki/PEiD
[4]http://ntinfo.biz/index.html
[5]http://www.heaventools.com/overview.htm
[6]https://codisec.com/veles/
[7]https://www.virustotal.com/

further more examines the malware sample to determine whether there is any entropy present in the various sections of the malware. This could indicate that there is data obfuscation in play, when high levels of entropy is identified.

3.5 Dynamic Analysis

The second phase of the new propose Hybrid-Malware framework is the Dynamic Analysis process. This section is the dynamic execution of each malware to determine the behavioural aspects of execution. The definition of dynamic malware analysis according to Zeltser (2017) is the dynamic execution of the malware in a environment that is controlled and monitored. The interactions with the operating system is done to determine which actions are malicious and allow for the analysis of these malicious actions. This process is generally done within a sandboxed environment, for the purpose of this examination and thesis it was done in a physical controlled environment and not on a sandbox (Sikorski and Honig, 2012). During initial testing during the dynamic phase of the Ransomware has detection capabilities of virtualised environments and in those situations would not unpack dynamically. There are traditionally two schools of approach dynamic malware analysis these are:

1. Analysis of differences between defined points, which is when a malware sample is executed for a period of time and the modifications which have been made after the execution is examined. This approach has an initial and post system state. This is a comparative reporting method for malware analysis (Sikorski and Honig, 2012).

2. Run time behaviour observation is the approach which is most taken my malware analysts, this is when a malware sample is run through a sandboxed environment and the malicious application is monitored during its run time this approach is usually limited to a specific time (Sikorski and Honig, 2012).

3.5.1 Sandboxed Environment

There are several online sandboxed environments which can be used to do run time based dynamic malware analysis. A sandbox is comprised of a virtualised environment that simulates an operating system with network capabilities (Babic *et al.*, 2011). The online sandboxes were used during initial testing to determine whether there was a viability for the automation of the dynamic malware analysis. The online sandboxes which were used were all Public licenses with limited scope. This was utilised to confirm finding which were identified with manual monitoring (Sikorski and Honig, 2012). The online sandboxes which were used are:

- Joe Sandbox Public Edition[8]

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

- SecondWrite Malware Sandbox[9]
- Hybrid-Analysis Public Edition[10]
- Cuckoo Online Sandbox[11]

The following are tools which form part of the above mentioned sandboxed environments.

- Androguard[12] this is mainly a tool which is used to examined Android APK, in terms of the current research this is not applicable however it is included in VirusTotal.
- Cuckoo Sandbox[13] this is an opensource automated malware analysis virtual environment.
- ExifTool[14] is a metadata extraction tool for multimedia and graphical images.
- National Software Reference Library[15] this is a collection of digital signatures of known, traceable software applications.
- PDFiD[16] is a PDF document triage tool to examine PDF files.
- pefile[17] is a Python module to read and work with PE (Portable executable) files.
- PEiD[18] detects most common packers, cryptors and compilers for PE files.
- SigCheck[19] is a command-line utility that shows file version number, timestamp information, and digital signature details, including certificate
- Snort[20] Snort is an open-source, free and lightweight network intrusion detection system (NIDS) software for Linux and Windows to detect emerging threats.
- ssdeep[21] is a program for computing and matching Context Triggered Piecewise Hashing values
- Suricata[22] is a free and open source, mature, fast and robust network threat detection engine
- Taggant packer information tool[23] developed by the Malware Working Group of ICSG (Industry Connections Security Group) under the umbrella of IEEE

[9] https://www.secondwrite.com/
[10] https://www.hybrid-analysis.com/
[11] https://sandbox.pikker.ee/
[12] https://github.com/androguard
[13] https://cuckoosandbox.org
[14] https://www.sno.phy.queensu.ca/~phil/exiftool
[15] https://www.nist.gov/software-quality-group/national-software-reference-library-nsrl
[16] https://blog.didierstevens.com/programs/pdf-tools/
[17] https://github.com/erocarrera/pefile
[18] https://www.aldeid.com/wiki/PEiD
[19] http://www.majorgeeks.com/files/details/sigcheck.html
[20] https://www.snort.org
[21] https://ssdeep-project.github.io/ssdeep/index.html
[22] https://suricata-ids.org/
[23] https://www.reversinglabs.com

- TrlD[24] is an utility designed to identify file types from their binary signatures.
- UEFI Firmware parser[25] firmware related structures: Volumes, FileSystems, Files
- Wireshark[26]is a free and open source packet analyser.
- Zemana behaviour[27] is an anti-malware tool which detects known malware on various operating systems.
- Carbon Black[28] is an endpoint security vendor.

These results are all aggregated into a single VirusTotal reports. This is all included in the dynamic analysis of malware samples which are uploaded to VirusTotal.

3.5.2 YARA Rules

YARA[29] is a widely used open source tool, which works across multiple platforms which aids in the identification and categorisation of behavioral traits of malware. This tool provides a mechanism in which multiple malware samples can be exploited to identify code similarities amongst them. The signatures which were used during the research make use of both a byte sequence and string matching along with logical operators that exclude false positives (National Cybersecurity and Communications Integration Center, 2018). YARA was used to identify whether the Ransomware samples matched any of the known YARA malware signatures. The YARA rules are used to search and identify characteristics and then classify whether the files match these rules as set out below (Deepti *et al.*, 2017):

1. **PE Module:** This module/signature allows the researcher to create parameters to identify specific known parameters of malware or binary executables.
2. **ELF Module:** This module is similar to the to the PE module but for use with ELF files.
3. **ClamAV Module:** This module identifies malicious files using a signature database from ClamAV that is a anti-virus software.
4. **PeID Module:** This module is used to identify whether a binary executable file has been packed or encrypted.
5. **WMI Module:** This module examines the processes which is called by the binary executable file.

[24]https://download.cnet.com/TrlD-File-Identifier-for../3000-2248_4-10442461.html
[25]https://github.com/theopolis/uefi-firmware-parser
[26]https://www.wireshark.org/
[27]https://www.zemana.com/release-notes
[28]https://secure.carbonblack.com
[29]https://yara.readthedocs.io/en/v3.8.1/

3.5.3 ProcMon

> *"ProcMon* is an advanced monitoring tool for Windows that shows real-time file system, registry, and process/thread activity" (Carvey, 2016).

Sikorski and Honig (2012)explain that *ProcMon* is part of the advanced Microsoft SysInternals Windows monitoring toolkit which enables the examiner to monitor the Windows Registry, file system, network, processes and thread activity. This is specifically useful in that it has the ability to capture events which occur once a malicious executable has been run on the physical Windows machine. *ProcMon* can capture basic or more detailed information, the amount that is captured is determined by Elisan (2015). This tool is very detailed in capturing the timeline of infection of the malware which it is monitoring. This information is displayed in a well design interface which is easy to use. The results can be exported as a CSV file, for further analysis.

3.5.4 RegShot

The one purpose of this section is to have two points in between which changes can be noted. To satisfy this, *RegShot* was used to monitor the changes made to the Windows registry. A baseline registry was obtained by using *Regshot* to take a snapshot of the Windows registry before the malware was dynamically executed. This is the software is known as the 1st Shot. The second portion to this was to obtain a snapshot of the Windows registry after the malicious processes have made alterations to the machine during and post-encryption. These were compared to identify the changes which have been made by the malware. This process is explained in detail by Sikorski and Honig (2012).

3.6 Network

A very important element of malware analysis is the way in which the malware communicates to a C&C server which is explained in detail in the literature review section. This is done by examining the network packets which are both sent out and received over a 24 hour period post infection. When malware communicates with their C&C server it leaves a trace on the network. The examination of the network packets can yield information regarding the domains which are being connected to Elisan, 2015 this leads to the ability to block specific IP address ranges or network behaviour. The information which can be found in the network packets can contain information regarding additional payloads being downloaded from an external source, the ports which are used to communicate with the C&C (Marak, 2015). The network communication was obtained by using *T-Shark* which is a command line version of *Wireshark* which captures network packets on a network. To capture the network packets which were issued and received post dynamic execution of the malware. The use of an automated tool such as *CloudShark* allows for the use of a signatures based on *Suricata* which is a set of known malicious indicators on a network and *Snort* is much the same as the *Surricata* rules to identify known malicious communication.

The *T-Shark* network analysis tool was used to capture network traffic in real time and displays it in a readable human format. The tool set allows the user to filter certain non relevant traffic analysis by the use of filters. The researcher took into consideration that the network needed to be isolated and secure and ensure that there was no accidental exposure to other devices whilst dynamically testing the malware. This further mimics a setup which would be done by a home user. *T-Shark* was used to capture the network packets which was initiated from the local machine. The .pcap file which was generated was saved to an external USB drive. The network communication was monitored for 24 hours post-exposure to ensure that the maximum data was recovered. The network information is as follows:

- Network Device Huawei Mifi Device Model Nr Afrihost_Mobile_a070
- Internal IP address 192.168.1.100
- External IP address 41.13.12.231

3.7 Digital Forensic Analysis

Digital Forensics can be explained as the use of scientifically and proven methods to ensure the preservation, collection and validation, identification, analysis, interpretation, documentation and presentation of digital evidence. Carvey (2016) states that the goal of Digital Forensics is to find the facts to recreate the truth to a specific event. The aim is to reveal the sequence of events which has occurred by discovering and exposing the remnants left behind by the actions taken by the malware. This is described as the Locard Principle, which Carvey (2016) explains that when two items come into contact with each other there will be an exchange which would leave remnants. The examination of malware often involve data artefacts which are volatile in nature due to the fact that malware would try and overwrites artefacts within the system, also ram is discharged when the machine is powered off. These mostly comprise of the artefacts which are contained within the ram of the target Windows machine. Contained within the ram is data of running processes, code which has been unpacked within memory and any scripts that would be running, however, if the machine is shut down the ram is cleared and as such lost (Teller and Hayon, 2014). The examination of the target machine can be described and broken down in various phases.

1. **Acquisition:** This phase refers to the collection of the digital media. This entails the forensic acquisition of the target hard drive disk. For the purposes of this research the target system was booted into a write blocked Linux environment, using *Paladin Edge*[30]. This specific environment was chosen because it allows for drives to be mounted as Read-Only which ensure that there is no cross contamination on the media being imaged.

[30]https://sumuri.com/product/paladin-edge-32-bit-version-6-08/

2. **Analysis:** This phase refers to the actual examination of the forensic image made in the previous phase. This includes identifying remnants and changes made on the target machine. This is done by examining the timeline of events from exposure until post infection over a period of 24 hours, the examination of changes within the Registry to determine which methods is used by the malware to gain persistence on the target machine, the examination of the memory of the target machine and the examination of system files for the identification of new files created or changes made to the system.

3.7.1 Preparation for Acquisition

The following step within the Digital Forensic phase is to perform a hard shutdown of the Windows machine this means pulling the plug and cutting immediate power. In this instance and there are two methods in which a forensic acquisition can be done. The hard drive disk contained within the desktop or laptop can be removed and connected to a firmware write blocker, and attached to a forensic machine which has a forensic imaging tool installed to obtain a forensic image (Elisan, 2015). The main reason a forensic image is used instead of a backup is that certain areas are not included in a back such as the firmware portion of a hard drive disk where malware can be found as well.

The second method and the one that has been used in this instance is to boot the device into a Linux write blocked environment such as *Paladin Edge* which is an evironment can fasillitate write blocking. Write blocking can be described as the process which blocks communication and writing privilege to a hard drive disk. Which is a Linux environment which does not allow for the immediate mounting a of hard drive disk and each drive needs to be mounted as Read/Write as they are automatically mounted as Read/Read or Read-Only.

The target machine was prepared for forensic acquisition by performing a controlled shut down which means that the shutdown process was performed by selecting the Shift key and shutting the target machine down which preforms a complete shut down. This allows for the machine to be booted into a forensic environment. The bootable Paladin Edge USB drive was inserted in the shutdown machine and the machine was powered on. The machine was set up during the installation phase to boot from USB drive when a USB drive is present.

3.7.2 Memory Analysis

The RAM (Random Accessible Memory) is described as the portion of memory which stores the code and data which is actively processing. This memory portion within the digital landscape if a computer is dynamically changing as it is reliant on either being in a charged state when the CPU has power and discharged state when there is no power to the CPU. The best method for acquisition of the RAM is to forensically capture the memory, whilst the machine is still in a running state. This is done by using a small footprint tool, which

run via a single executable. The reasons this is the industry acceptable standards is that the changes made to the memory is limited and extrapolated (Carvey, 2016). This means that the data is only resident whilst it is in a charged state and the machine is processed on, once the power is removed the ram will discharge and lose all the data it had contained within its pages. Windows operating system has the separation of privileges which prevents a malfunctioning or malicious user application from accessing or manipulating components of the system which functions at Kernel Mode this includes the operating system and drivers, any software which needs administrator rights to run functions at Kernel mode (Ligh *et al.*, 2014). Other user activity would function in a limited privilege capacity in what is called User mode (Teller and Hayon, 2014). Within the memory image if a malicious process has taken place it would be identified within the ram capture. The operating system is designed to aid user applications this is done when an application needs to communicate with the network resources it is assigned a system call which calls on a handle and is piped to the relevant process and instituted by the operating system (Sikorski and Honig, 2012). The memory within the operating system is responsible for assigning the system resources to a specific system call which is made.

The *Belkasoft RAM Capturer*[31] has one of the smallest footprints in that it is a single binary file. This tool allows the researcher to extract the contents of the computer's volatile memory. This tool works both on 32-bit and 64-bit processors. This tool will be used by the researcher to extract the volatile memory from the local machine, post exposure. The memory plays a vital role in extracting the whole contents of the ram memory on the local machine. When it comes to malware analysis within the memory one of the most important take a way's is that there can be remnants of code or string within the memory . This can aid in the identification of what has taken place at the point of time when the malware infected the machine. Within these strings there can be IP addresses of connections which have been facilitated by the memory. The memory dump which was take post dynamic execution the image was examined within *Mandiant Redline* [32] which is a tool to parse data from the Forensic ram dump. This image/dump was examined to determine the following:

1. **Recover command line and process paths**: Identify potential command line code run by the malware along with paths from which processes are called.
2. **Analyse Heaps:** Identify the heaps stored by applications.
3. **Inspect environment variables:** Identify search order hijacking and identify changes to environmental variables.
4. **Detect backdoors with standard handles:** Identify whether the process input and output originates and calls to valid handles. This aids in the identification of redirected by a remote network socket by malicious means.
5. **Enumerates dll:** Identify the dll used by the Operating System and track the process calling to the dll. This enables the researcher to identify hidden and malicious

[31]https://belkasoft.com/ram-capturer
[32]https://www.fireeye.com/services/freeware/redline.html

library files.

6. **Extract PE files from memory:** Extract potential PE file from memory for static analysis and disassembly. Identify changes made within memory when an binary executable is run and dynamically unpacked in memory.

7. **Detect Code Injection:** There are three types of code injections (1) **Remote dll Injection** which is when a malicious process forces the target process to load a specified malicious dll, (2) **Reflective dll injection** is when malicious process writes code into a specific memory space to target a specific process and force the execution of it (3) **Hollow Process Injection** is when a malicious process executes a new instance of a legitimate process which has been hollowed out and the legitimate code replaced with malicious code.

3.7.3 Registry Analysis

Within the Windows operating system the registry forms a core component to the successful running of the Windows operating system. The forensic analysis of the registry is the examination of the core building blocks of the Windows operating system. The registry within the Windows operating system is responsible for maintaining the configuration information about the system, maintenance settings and additional historical information of events which have taken place on the system. When a malware infection occurs on a system the malware had to arrive on the system and execute itself this would be recorded within the registry it could also be established whether it arrived via the network, a file or an external device.

When examining a Windows based operating system for indicators of malware the registry would be the first point to start at to identify whether the malware has obtained persistence on the machine. The most popular way according to Carvey (2016) for malware to obtain persistence is to include itself in auto-run keys which would execute itself with every start of the operating system. As the registry is responsible for the configuration and maintaining of the operating system, a fact which malware authors are aware of the main escalation of privileges and system changes would be recorded within the registry.

The registry hives were examined using *Registry Explorer* which is an open source tool which allows for the examination of the `NTUSER.dat`, `USSERCLASS.dat`, `SYSTEM`, `SOFTWARE` and `SECURITY` registry key. The examination of the registry analysis was based on the information collected from the Dynamic Analysis portion which made use of *RegShot* [33] this guided the examination of the registry.

3.7.4 Data Artefacts Analysis

The examination of identification and analysis of relevant data artefacts can be hard if there is only and execution timeline. When conducting malware analysis you can by using

[33] https://sourceforge.net/projects/regshot/

the information from the static analysis predispose interactions which may happen on the machine being infected and that aids in determining where the evidence of interaction would be recorded on the machine. The examination of these data artefacts can in turn return meta data information which can allow for the comprehensive understanding of all elements of the malware. The most effective way of identifying these artefacts would be to make use of what is referred to as a timeline analysis, which would identify each system file which had interaction within a specific date frame. The use of *ProcMon* which was used in the Dynamic Analysis section will also aid in the identification of the data artefacts which are important (Marak, 2015).

3.8 MITRE ATT&CK Framework

The framework was included in the research to show that there are similarities in the methods used by malware operators. This is introduced in Section 2.8 which details the framework. This framework was added into the methodology to pull together the comparisons between the different strains. Malware authors have been constantly evolving and employing new techniques. The framework allows for a matrix to be created that can be evolved as behaviour of malware or an attack changes. The framework allows the researcher to focus their efforts to observing behaviour of the Ransomware families and how these correlate to each other. This also allows for a creation of a knowledge base approach to collecting information of these families. The traditional approach which has always been taken was to depend on signature of known bad behaviour, this had many constraints when it came to malware analysis as malware could modify its code and actions to avoid signature based detection. The ATT&CK approach enables an approach to malware analysis which classifies techniques or behaviours rather than signatures. This allows for a behavioural signature to be created much like a digital footprint. Much as malware is unique the approach that it takes is unique and creating a cookbook of approaches taken by the different families enables the researcher to create a play book of expected behaviour (Johnson, 2018).

3.9 Data Sets

The samples which were examined were obtained through means of crowd sourcing. A call was put out to the IACIS list-serve which is a private listserve for Law Enforcement and Ex-Law Enforcement in the Forensic Space. The samples were obtained from Carbon Black which is an endpoint security company. The samples were saved under their family named associated with the sample. The samples were all named according to their generated SHA-256 cryptographic hash value. The samples obtained from Carbon Black consisted of approximately 2000 strains from 114 families. Due to the limitations of the research the decision was taken by the researcher to focus on only examined three samples of Ransomware originatingfrom three different families. This was decided due to the complexities

and comprehensive nature of the research. The in depth nature of the research and the time constraints associated with it did not allow for proof on concept of this framework to be tested on more than three families.

The selection was done by identifying unique families, that have had a long existence and success in the exploitation of victims. There was three factors taken into consideration the first being the longevity of the family, the gross income generated by the campaigns and whether there was a something which was unique to the specific family. The three families selected are discussed below.

3.9.1 CERBER

MD5 f5146a3bbe6c71e5a0ef2f04f955b1a1

SHA256 2562d08ffeba708fb833404d2c320ea4f29365c791d504181e08e3e9b529f5cf096

SSDeep 6144:zxXjCYxMEhcoYAOEAIDD6Y76kmDP8HNGTqKB7m4bGykJC:9JeESZaLiY70MCqiDbGyJ

There were 182 samples obtained of the CERBER Ransomware, the sample which was selected was the one sample which was identified by Virus Total to be the most occurring. This Ransomware family was introduced in Section 2.9.1. The reason that this family was one of the choices were that that according to Campbell, 2016the CERBER Ransomware was the first group to sell their "product" as a service. The name which was recorded as the filename for Virus Total and in the Structure however are inconsistent. This can be due to a parsing error or that the sample was previously saved as another file name.

These malware operators ensure that each campaign which is run is evolved from the previous campaign meaning, a new different distribution method and a unique packer. The most notable CERBER campaign targets users in China and South Korea and is distributed using the MAGNITUDE EXPLOIT KIT (Sison, 2017). This would indicate that CERBER is certainly no longer an emerging threat however it is the leader in the Ransomware-as-a-service ecosystem and this is the reason that it has been chosen (Anubhav and Ellur, 2016). CERBER has shown that it has a model which has been proven multiple time to be one of the biggest in the world, furthermore they are sophisticated in the methodology which is used for the money to flow (Barkly Research, 2017a). The CERBER examination results are detailed in Chapter 4.

3.9.2 MAKTUB

MD5 b24952857ff5cb26b2e97331800fa142

SHA256 0210a8f8e729d1b81bf81e39874af98c379f92fcdf802d6d925eb9e65186dfd3

SSDeep 6144:ZlwGzrV3k4oU1X6bxm1asEyE4laOVojtt8jAV4:LwopU4oUM01adyE4laOVQ2Ae6144:Z

There were only two samples obtained of the MAKTUB Ransomware. This strain is significant to the research due to ITSability to encrypt offline and that it is packed. The one

sample did not yield any network traffic and as such was excluded in favour of the one which did yield network communication. This Ransomware family was introduced in Section 2.9.2. This specific strain of Ransomware boasts with the best graphically designed GUI (graphical user interface) (Abrahams, 2016). This Ransomware became famous for it's attention to detail of their spam campaigns in having individually crafted emails addressed to the victim with their residential address (Hasherezade, 2016c). This added to their success that their spam campaign emails were crafted with attention to detail. The same can be said on their specific design and success. One of the interesting features is the document which the victim receives named "TOS update.rtf" this document looks legitimate and whilst the user is reading the document, the Ransomware takes the opportunity to wreck havoc on your system only letting you notice once it is too late. The other interesting fact is that pressure is added in that they want the victim to pay sooner rather than later, this has been likened to a smash and grab approach. The bitcoin ransom also increases every 24 hours (Hasherezade, 2016c). During the collection of literature a definitive revenue could not be determine. This specific family was chosen for their attention to details and method of behaviour. This strain was selected due to the fact that it has the ability to locally execute without having to be connected to the internet (Hasherezade, 2016c). The examination of MAKTUB is detailed in Chapter 5.

3.9.3 LOCKY

MD5 ec0fae82b75ee1d7ce72b49d97dec4a1ec0

SHA256 003d28f180472b832722435d27e216835a8a330f992797006d307f8f14c4a2d3

SSDeep 3072:zMmtj2jNl4ad4mH8jvIBVa7Knn+NkGVgo9UvXWF:zPtSR+Vad+Nbsv

There were 10 samples of the LOCKY Ransomware, the sample which was chosen was the on which according to *VirusTotal* was the most recent and more occurring sample. This Ransomware family was introduced in Section 3.9.3. LOCKY was first seen in the wild in February 2016 and soon became a dangerous threat to organisation and individuals. LOCKY is categorised as crypto-Ransomware which means that the main aim of this malicious code to infect and encrypt the machine holding the data ransom. There also appears to be many actors who utilises this and it can be said to form part of Ransomware-as-a-service even though there has been no active marketing done by the LOCKY creators (Vlad, 2016). The LOCKY Ransomware is spread by two methods which is either email or by an exploit kit which has been masked as legitimate software. The NECURS BOTNET has been found to be the main culprit behind the distribution of LOCKY as email spam. The exploit kits which have been used in the past has been the Neutrino exploit kit, RIG and Nuclear Exploit kits (Balaban, 2016). The LOCKY malware has been going strong since the first sighting and has shown now signs of slowing down. This strain has been chosen as it is adaptable but unlike the others have remained strong and relevant (Griffin, 2016b). The examination of LOCKY is detailed in Chapter 6.

3.10 Summary

The Hybridised-Malware model was extended to include the additional disciplines, Digital Forensics to determine the remnant left behind after the dynamic execution of the malware, the monitoring and examination of the network communication sent and received by the malware and the mapping of the code within the malware developing a DNA for each strain identified and identify the similarities within the code. Extending the current Hybridised-malware model allows for a more comprehensive approach to the understanding of malware and the interaction it has with a machine before, during and after infection. This closes the circle in terms of knowing by using a multi disciplinary approach. The examination following this approach is detailed in the Chapters 4, 5 and 6.

4

Case Study: CERBER

4.1 Introduction

This Chapter of the thesis contains the results from the process which has been used in Chapter 3. The examination of this has been done to identify the static properties of the executable, the dynamic process it follows to execute itself and how it interacts with the network. This process follows the framework as explained in Chapter 3. Check Point Software Technologies (2017) wrote up a brief history on the Ransomware family CERBER. They stated in this that CERBER Ransomware was the first group to be selling their "product" as a service. The CERBER Ransomware authors have ensured that their toolkit is available for rent by any other criminal group. The fee for their services is a 25% cut of the potential profits to be made from the campaign. Information which has been gathered by Check Point Software Technologies (2017) indicate that the CERBER authors and their clients have run approximately 161 campaigns currently from July 2016 until December 2017.

CERBER Ransomware-as-a-service uses a business model that has been proven to be effective by some of the biggest franchise businesses worldwide. Furthermore, the malware author uses a sophisticated money flow to ensure that the profits remain sealed and that its Bitcoin wallets cannot be associated with the attack operation. It is therefore little wonder that CERBER Ransomware is one of the most widespread pieces of Ransomware of our time (Antonopoulos, 2014).

This Chapter contains the finding in relation to the framework which has been set out in Chapter 3. Section 4.2 presents the findings relating to the static analysis, the section which follows onto that one is the dynamic analysis which is the execution of the malware

sample and is set out in Section 4.3 contains all the communication which was identified on the network that has been sent and received, the section which follows onto Section 4.4 which contain all the artefacts which were identified using Digital Forensic processes which is detailed in Section 4.5. This chapter is finishes with a summary which summarises the findings of the examination into CERBER in Section 4.7.

4.2 Static Analysis

This section forms the initial phase of the Hybrid-malware analysis framework introduced in Section 2.6. This process entails the analysis of the static malware file to determine what intelligence can be ascertained without dynamically running the malware. The focus is to extract as much information from the malware in its pre-execution phase.

4.2.1 PE Structure

The CERBER sample was examined using *CFF Explorer* to view the language libraries within PE Structure. The sample contains multiple language libraries, which indicates that this strain is highly customisable to be used to target different countries. This could indicate that due to CERBER being used as a Ransomware-as-a-service product the authors ensured that the product is versatile to target multiple countries. This would mean that in terms of business a larger base of potential victims. The sample which was examined had language libraries which include German, English (US), French, Italian, Korean, Russian, Chinese and Swiss. This shows the adaptability and customisation options of CERBER which is in support of this strain being sold as Ransomware-as-a-service.

The sample has the ability to be able to register a top-level exception, this can be described as the highest level of privilege on an operating system. In terms of on the Windows 7 operating system the top-level privilege is that of Kernel mode. This was identified by looking at the assembly code portion and identifying the purpose of that specific call. This indicates that the sample which was examined is able to supersede the top-level handling of a thread within a process. This would mean it has the ability to execute with Kernel mode which is the highest privilege within the Windows Operating system. This also enables the sample to run in Kernel mode with escalated privileges instead of that of User mode which is the normal mode for software and user related actions. This is shown in Table 4.1 and Table 4.2.

This information was found within the PE file examination, the second figure was found when reversing the executable sample using *PE Explorer*. The idea behind the use of exception usage as an anti-debugging is to make the static analysis of the malware sample harder than those who do not make use of it. The method used by the CERBER which was examined is to set the setting for debugging to execute as a top-level execution handler name *Kernel32*. The basic idea behind exception anti-debugging techniques is to use side effects of a debugging tool (or of decisions made by the person using the tool) to alter the

execution flow. There are several ways to do that, the classic being setting an exception handler (either for the local thread or at the process level) and implementing important parts of the logic in the exception handler before returning execution to the original EIP or changing it completely. The EIP is a register pointer within the 32 bit architecture which holds the extended instruction points. In other words this is the map which the computer will follow to execute the next control.

4.2.2 Version Information

CERBER Ransomware which was chosen in Chapter 3 was examined using the methodology and tool set which was detailed in the same chapter. The aim of this portion of the Hybrid-Malware analysis framework is to collect information regarding the file itself and what the attack surface was for distribution. The malware sample was examined using *PEStudio* and it was the following was ascertained. The file description for this software upon further research is for a tool which claims to be able to speed up the download of drivers on a system. This shows that this specific CERBER sample masquerades as legitimate software in drive-by downloads.

Table 4.1: Anti-Debugging process invoked in assembly code

IsDebuggerPresent	Determines whether the calling process is being debugged by a user-mode debugger.
SetUnhandledExceptionFilter	Enables an application to supersede the top-level exception handler of each thread of a process.
UnHandledExceptionFilter	An application-defined function that passes unhandled exceptions to the debugger, if the process is being debugged
GetCurrentProcess	Identifies the current process running.
Terminate Process	Terminates a specific process

Table 4.2: Anti-Debugging process invoked in assembly code

Process	Description
SetUnhandledExceptionFilter	Enables an application to supersede the top-level exception handler of each thread of a process.

4.2.3 Windows API Class

CERBER was examined and according to the PE header information imports some libraries/-functions from the Microsoft Window32 Link Library. The following functions allow the malware to interrogate the registry values, open keys, make changes to keys, enumerate new keys, obtain information on modules open within the memory, identify whether debuggers are run on the Microsoft Windows environments, terminate processes, create shells and many more as can be seen in the below list the malware functionality within the PE32

structure would indicate that the infection allows for maximum access to the target machine, this enables the malware to have the keys to the kingdom effectively. The import table can be seen in Table 4.3.

Table 4.3: Import Functions extracted using *PEStudio*

Function	Description
RegDeleteKeyA	This function is used to delete subkeys and it values.
RegOpenKeyA	This function opens a specified registry key.
RegCloseKey	This function handles the closure of a specified registry key.
OpenProcessToken	The OpenProcessToken function is used to open an access token associated with a process.
RegDeleteValueA	Removes the named value of a specified registry key
RegCreateKeyExA	Creates the specified registry key, if the key exists the function will open it.
RegOpenKeyExA	Opens the specified registry key.
RegEnumKeyA	This enumerates the subkeys of a specified registry key.
RegCreateKeyA	Creates the registry key which is specified.
WriteFile	Writes data to the specified file or input/output devices.
GetModuleFileNameW	Retrieves the fully qualified paths which the file which contains the specified module.
UnhandledExceptionFilter	Application-defined functions that passes the unhandled exception to a debugger.
TerminateProcess	Terminates the specified process of all the threads.
GetTickCount	This function retrieves the number of milliseconds that have elapsed since the system started.
GetVersionExA	This function retrieves the version of Windows.
LoadLibraryA	Loads a specified module into the address space of the calling process.
GetStartupInfoA	Retrieves the contents of the STARTUPINFO structure that was specified when the calling process was created.
DeleteFileA	Deletes and existing file.
GetProcAddress	Retrieves the address of an exported or variable from a DLL.
IsDebuggerPresent	Determines whether the calling process is being debugged by a user mode debugger.
CreateFileA	Creates a new file.
LockResource	Retrieves the pointer to a specific module within memory.
GetCommandLineA	Retrieves the command-line string for the current process.
GetModuleHandleA	Retrieves the module handle for the specified module.
VirtualAlloc	Reserves the state of the region specified within the memory.
ShellExecuteW	Performs shell code on a specific file.
GetLastActivePopup	This function determines the pop-up window owned by a specified windows.
SetWindowsHookExA	This function installs an application-based hook procedure into a chain.
FindWindowExW	Retrieves a handle for a specific windows whose class name and window.
FindWindowA	Retrieves the handle for the top-level windows whose class name is specified.

CERBER accesses the `vssadmin.exe` which is responsible for managing the Volume Shadow Copy process on Windows. The sample proceeds to identify whether the Volume Shadow process is enabled and where the copies of these volume shadow copies are saved. Once this process is completed the Ransomware will exploit the `vssadmin.exe` and delete these copies to ensure that the user cannot revert to a previous copy.

CERBER utilises of the `BCEDIT.exe` which is responsible for boot settings for the Microsoft Windows Operating system to control how your system boots. This is done to ensure that the victim is not able to boot into recovery mode to effectively clean the system. CERBER manipulates the ability for the user to be able to boot into any other environment than the one which has been infected and controlled by the malware. This is detailed in Figure 4.1. CERBER makes use of the Microsoft Windows Native API's and lives off the land. Each of these API's will be briefly discussed but will not be covered in depth as this falls outside the scope of the research. The Windows Native API's which were exploited by CERBER are these are in order of first exploitation:

- `Kernel32.dll`[1] handles memory management, input/output operations, and interrupts this link library file is loaded into the protected memory sections.
- `Shell32.dll`[2] is a dynamic link library that controls certain API functions of the Windows Shell.
- `User32.dll`[3] is the Windows API responsible for user interface namely anything associated with the Windows GUI.
- `GDI32.dll`[4] :functions that perform primitive drawing functions for output to video displays and printers.
- `AdvAPI32.dll`[5] provides security calls and functions for manipulating the registry.
- `COMCTL.dll`[6] implements a wide variety of standard Windows controls, such as File Open, Save, and Save As dialogues, progress bars, and list views.
- `ShlwAPI.dll`[7] is a library which contains functions for UNC and URL paths, registry entries, and colour settings.
- `Version.dll`[8] is a module that contains application programming interface (API) functions used for Windows version checking by the applications on Windows NT.
- `COMDLG32.dll`[9] the Common Dialog Box Library, implements a wide variety of Windows dialog boxes intended to perform what Microsoft deems common application

[1]https://www.webopedia.com/TERM/K/Kernel32_dll.html
[2]https://msdn.microsoft.com/en-us/library/ windows/desktop/bb776779(v=vs.85).aspx
[3]http://www.processlibrary.com/en/directory/files/user32/19597/
[4]https://en.wikipedia.org/wiki/Microsoft_Windows_library_files
[5]https://en.wikipedia.org/wiki/Microsoft_Windows_library_files
[6]https://en.wikipedia.org/wiki/Microsoft_Windows_library_files
[7]http://www.processlibrary.com/en/directory/files/shlwAPI/20075/
[8]http://www.processlibrary.com/en/directory/files/version/22668/
[9]https://en.wikipedia.org/wiki/Microsoft_Windows_library_files#COMDLG32dll

tasks. Starting with the release of Windows Vista, Microsoft considers the "Open" and "Save as" dialog boxes provided by this library as deprecated and replaced by the common Item Dialog API

- `NetAPI32.dll`[10] provides functions for querying and managing network interfaces.
- `Winspool.drv`[11]is a process belonging to Windows Printer Spooler.

unicode	11	-	x	bcdedit.exe
unicode	11	-	x	bcdedit.exe
unicode	34	-	x	vhd=[C:]\vdisks\disk01.vhd
unicode	37	-	x	\windows\system32\winload.exe
unicode	9	-	x	Debugging
unicode	4	-	x	Boot
unicode	4	-	x	Boot
unicode	6	-	x	Resume
unicode	4	-	x	Boot

Figure 4.1: Exploitation of `BCEDIT.exe`

4.2.4 Entropy

Entropy analysis is an important factor in malware analysis (Osaghae, 2015). This can give the researcher an indication on where to start the analysis. Signs of low entropy can indicate that the malware sample is not compressed nor have obfuscation sections. In the case where the entropy level is high within specific sections can indicate that the sample is packed or that obfuscation methods have been used. To understand the meaning of entropy one needs to understand the definition thereof. The simple definition for entropy is the randomness in which bytes display within a section within the PE file. The green sections of the visualisations is the high entropy sections and the varied black sections are less entropy. The binary representation in Figure 4.2 the areas of entropy is pixelated and the other sections are shaped. This means the examination of a file to determine the measurement of entropy is simply put the measurement of disorder of the bytes within a file (Osaghae, 2015). An interesting area of malware file analysis is the measurement of file entropy.

CERBER displayed high levels of entropy in the `.text` section; this section had an unusually high level of entropy compared to the other sections. The `.text` section of the CERBER malware strain which was examined contained the following information in this section namely the main code portion of the executable, which indicates the disorder within the section which indicates high levels of entropy. This means that there is a high level of binary randomness to protect the code itself. In the Figure 4.2 the section of the right of the image with the high level of entropy is the `.text` section.

The CERBER sample was examined using the methodology as detailed in Chapter 3. The sample makes reference to loading a module which has been flagged by the antivirus vendors as malicious in nature. The sample contained a high level of entropy or randomness

[10]https://en.wikipedia.org/wiki/Microsoft_Windows_library_files
[11]http://www.processlibrary.com/en/directory/files/winspool/22326/

which can be seen in Figure 4.2 which shows the `.text` section has high levels of entropy, in Figure 4.3 is a binary representation of the binary distribution.

Figure 4.2: Entropy visualisation using Veles

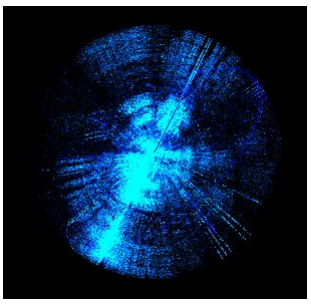


Figure 4.3: Binary visualisation using Veles

4.2.5 Hash Fingerprinting

The hashing of malware has been a practice used by malware analysts to uniquely identify these strains of malware. The sample which was examined was run through varies methods of hashing to determine the unique has fingerprint for this particular CERBER strain. This allows for signature based detection of the fingerprinted strain. The traditional method of hashing a piece of malware for signature based detection is MD5 and SHA256, this is however cryptographically a black and white approach, because if one bit of data has morphed or changed the signature will fundamentally change (Upchurch and Zhou, 2016). The suggestion is made to add into the existing fingerprinting techniques SSDeep hashing which is referred to as fuzzy hashing. This identifies similarities within a document that are similar but not the same it is also referred to as homologous files. The term homologous means to have the same or similar genes or the same order[12].

4.2.6 Strings

A Program Database File (PDB) or Specify Symbol file is a file which maps the identifier of the creator of the source files. This is discussed in Chapter 3 in Section 3.4.2. When an executable is compiled the PDB file will be created, During the compiling face the PDB information is generated and retained unless it it is stripped from the binary. The PDB file is effectively a map of the statements of the source code to the execution structure in the executable[13].

[12] http://www.dictionary.com/browse/homologous
[13] https://msdn.microsoft.com/en-us/library/ms241613.aspx

4.2.7 Code Obfuscation Techniques

There are 'six code' obfuscation techniques most popularly used. Iliev (2017) explains that there are dead code insertions which is adding ineffective or dead code within the malware which would circumvent most anti-virus. The second code obfuscation technique is reassigning registers, which is the technique of switching between registers whilst maintaining the function of the code. The third code obfuscation technique is the subroutine reorders which is when the original piece of code's subroutine reordering that changes the order in which the routines are executed.The fourth code obfuscation technique is to ensure that the original code evolves by replacing the instruction with an equivalent instruction. The fifth code obfuscation technique which reorders the sequence is which instructions is executed. The final sixth code execution technique is to integrate the malicious code into another piece of code by binding them together.

The sample examined makes use of a data obfuscation technique which is used to protect code from static analysis. In assembly code when a `call` is issues it points to a return address which is the address where the execution should take place. The CERBER sample used three functions together which is the `call`, `ret` and `jmp` functions. Dalei and Li (2016) explains that malware authors obfuscate their code with the intention to make it complicated to gain information from the code itself. Malware operators rearrange their code to make the detection process harder. The most popular method used by malware authors according to (Kumar, 2004) is that they obfuscate the call addr instruction in Table 4.4 . These CERBER code obfuscation technique can be seen in Table 4.4. This technique used is part of the sixth code obfuscation techniques described by Iliev (2017) and CERBER makes use of the Instruction reordering.

Table 4.4: `call,push,ret` CERBER

Function	Description
Call address	This function will make a push to the updated program counter. This is the pointer to the instructions after the call which should be executed.
Ret	This specific function internally pops and address off the stack and then jumps to that address. This matched with call would allow for the return to the instruction after the prior call
Jmp	Simply put this function is responsible to jump to a specific address .

4.2.8 Summary

The examination of the CERBER strain identified that this family is adaptable and that their malware is not specifically aimed at one location or country. The vast language libraries give them the opportunity to maximise their sales by being able to sell a product which is fully adaptable. This version of the CERBER malware was masked as a piece of legitimate software, which contain the malicious code. The CRC error checking did not match what was claimed by the software and the actual calculated value. The sample further more targets the Windows API responsible for booting the machine into a safe

environment. It also proceeds to access and delete the API's responsible for the Volume Shadow Copies ensuring that the user cannot revert to a previous version.

4.3 Dynamic Analysis

The following section comes from the execution of the malware on a live system. The system was monitored using the dynamic tool sets used as explained in Chapter 3. These are actual observation made while monitoring the behaviour of the Ransomware on the physical system, memory and network.

4.3.1 Sand boxed Run-time Report

The purpose of running the sample through the sandboxed environment is to identify evasion techniques, and to confirm findings made on the physical machine examination. The CERBER sample was firstly run trough VirusTotal[14] which checks various antivirus software to identify whether the file is flagged as malicious. The sample was found to be identified by 60% of all antivirus software. This means that there is still 40% of antivirus tools which do no flag this sample of CERBER as malware.

The main objective for malware besides infecting the victim's machine is to evade detection. Malware researchers make use of a virtualised environment to analyse the behavioural traits of malware. Hassan (2015) explains that malware has become environmentally aware, malware operators are able to use key characteristics of the target system which they wish to infect to identify whether they are sandboxed. This is successful because even though the virtual environments are becoming more sophisticated they still lack some characteristics of a genuine environment. The CERBER sample makes use of what is known as a sleep call. This means that the malware is set to execute the malicious code after effectively a sleep period. Even though sandboxing solution have come up with an answer to the sleep calls made by malware, the malware has adapted to check the system time against various internet time sources such as a Network Time Protocol (NTP) or other web orientated sources (Blunden, 2009).

CERBER has a sleep period which exceeds three min before the malicious portion of the code will be executed and unpacked. Abhishek and Bhu (2017) explains that with extended calls the malware refrains from being detected as malicious by waiting out the period used my sandboxed environments for analysis. In February 2013 the malware TROJAN NAP took this approach to evade detection and analysis within a sandboxed environment. This specific malware Trojan is linked to the KELIHOS BOTNET (Abhishek and Bhu, 2017).

CERBER displayed the potential ability to be environmentally aware, in that it has the ability to query the time of the host machine it is infecting. This was found during the assembly code examination to determine which static information can be drawing from the

[14] https://bit.ly/2CO8kNW

malicious executable. As can be seen in Figure 4.4 reference is made to ''GetLocalTime'' using the KERNEL32 API. This API contains top-level privileges; this means that the privileges of the malware have been escalated to Administrator. This technique is used by malware to evade or detect sandboxed environments. Dell Secureworks (2015) explains that malware use this approach to detect whether a virtual machine has been run as there is a time penalty when running a virtual machine. This can be bypassed by potentially using an external time source such as NTP.

The sandbox examination indicated that CERBER has the ability to identify that is is within a sandbox environment and delay the execution of the malicious portion of code, it also has the ability to be environmentally aware.

```
@40801c: push ebp
@40801d: mov ebp, esp
@40801f: push ecx
@408020: push ecx
@408021: push esi
@408022: lea eax, dword ptr [ebp-08h]
@408025: push eax
@408026: call dword ptr [0041D2E0h] ;GetSystemTimeAsFileTime@KERNEL32.dll
@40802c: push 00000001h
@40802e: push 00000000h
@408030: push 00000000h
@408032: push dword ptr [ebp-04h]
@408035: call 0040BB00h
@40803a: mov ecx, dword ptr [ebp-08h]
@40803d: xor esi, esi
@40803f: add eax, ecx
@408041: adc edx, esi
@408043: mov dword ptr [0043F728h], eax
@408048: mov dword ptr [0043F72Ch], edx
@40804e: xor eax, eax
@408050: pop esi
@408051: leave
@408052: ret
```

Figure 4.4: Function to get local time

4.3.2 YARA Rules

There were 182 samples collected for CERBER, however for the focus of the research only one sample was run for the purpose of the thesis as there were time constraints due to the in depth analysis needed for each sample run. The CERBER malware binary which was chosen, the sample is named using the SHA-256 hash value. The sample was run through the online *Cuckoo Online Sandbox*[15] to be in a position to generate the *YARA* rules results. The results which were identified by the online *Cuckoo Online Sandbox*. This allows the researcher to anticipate the behaviour of the malware(National Cybersecurity and Communications Integration Center, 2018). This is however only a summary of the overall examination:

- Search for randomised functions which can be hollowed out and exploited.
- Anti-debugging
- Runs a keylogger in the background
- Takes screenshots

[15]http://sandbox.pikker.ee/

- Ability to escalate privileges
- Manipulates the Windows registry
- Manipulates the Windows Token process
- Affects the User profile within Windows
- Affects User operations within Windows

4.3.3 ProcMon

The Microsoft Component Object Model (COM) is a legacy technology which was introduced by Microsoft in the late 90's. This specific allows for the development and use of binary software components across a variety of languages which can then be instantiated as COM objects in Windows. Every COM class has a unique identifier associated with it name the CLSID. Specifically for malware the COM classes is a useful way to perform certain security related system operations without the user being aware of it. This is exploited by malware, by using this as a hook point with which to place additional code which can control other objects When code is added to an existing Object class this method is adopted at run time execution and the malware operator can control the flow of execution of objects which inherent from this class. The CERBER sample makes use of the COM classes by using the functions in Table 4.5.

The sample makes use of the COM classes is exploited to enable the CERBER sample to hook onto the legitimate module is the `System32` folder `taskschd.dll` to be replaced by the malicious process which is loaded by the malware. This is done by replacing code within the class , when the class and objects associated with it is loaded the malicious code would be run instead of the legitimate code also referred to as hollowing. This is done as a method to evade detection by the malware as it seems to be a legitimate process taking place on the system (Dell Secureworks, 2015).

Table 4.5: Exploiting the COM classes

Function	Description
Colnitialize	This initializes the COM library and identifies the concurrency model as single thread apartment.
CoCreateInstance	Creates a single unutilized object of the class associated with a specific CLSID
Memset	This function sets the count characters to the character c.
PathFindFilenameW	This function searches for a specific path to a file name.
Memcpy	This function copies the count bytes from src to the dest.
CoUnitialize	This function shuts down the previous call made to the COM library.

4.4 Network Analysis

The network analysis was done as detailed in Section 3.6. The network communication was intercepted and examined using the tool sets as detailed in the Methodology chapter. The examination focuses on communication during the encryption and post encryption for a period of 24 hours.

4.4.1 Protocol Breakdown

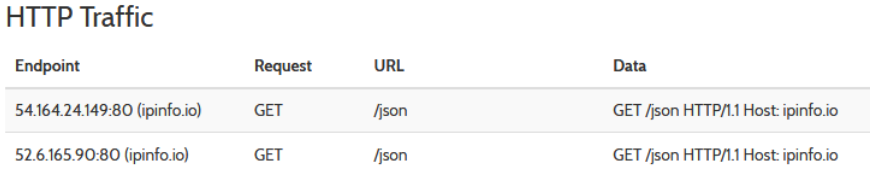
There was 35000, packets in total and the breakdown of the different protocols were explained in Table 4.6. The CERBER sample communicates mostly using UDP traffic as this noted 84.4% of the traffic captured on the network, in second was TCP communication at 6.1%, followed by ICMP at 4.4%. The CERBER Protocol breakdown table is detailed in Table 4.6.

Table 4.6: Protocol Breakdown and network traffic

Protocol	Total Packets	Total Percentage
UDP	29,540	84.40%
TCP	2,135	6.10%
ICMP	1,540	4.40%

4.4.2 Malicious Traffic

There were 33 HTTP streams identified to originate from the host machine. These were observed immediately after executing the malicious binary. These can be seen in order of most contacted. These are all legitimate websites, where some have been cleaned and did not contain any information which matched the request made from the host machine. There are three domains which will be focused on for the purpose of the research. These can be seen to be highlighted in Figure4.5. CERBER contacted three main domains which can be seen in Figure 4.5. The domain IP-API[16] and ipinfo[17] are both legitimate websites which have been utilised by the malware to identify the location of the external IP address of the machine it is infected. This is done on two separate locations to ensure that the data which is returned is similar or the same. This can be seen as CERBER sample being location aware; they identify the location of the host machine.


HTTP Traffic

Endpoint	Request	URL	Data
54.164.24.149:80 (ipinfo.io)	GET	/json	GET /json HTTP/1.1 Host: ipinfo.io
52.6.165.90:80 (ipinfo.io)	GET	/json	GET /json HTTP/1.1 Host: ipinfo.io

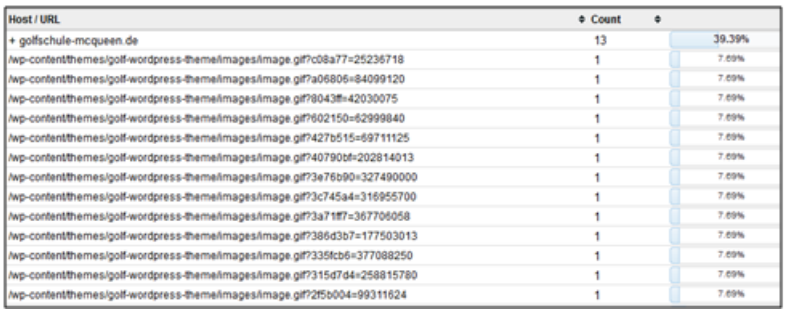
Figure 4.5: HTTP requests executed by CERBER

[16]http://ip-API.com/
[17]https://ipinfo.info/html/ip_checker.php

The next domain which is `golfshule-mcqueen.de` is the most contacted domain with 13 requests made to this domain in Table 4.7. The CERBER sample requests to download and image file which is seen to contain high levels of entropy and does not contain any visual data which makes sense or can be interpreted. This is most likely due to the data being segmented, compressed or encrypted. This domain was compromised and suspected to be run as a C&C server. This was no explored more as it fell outside the scope. This is most likely the payloads' which have been downloaded to maintain full control on the system. These were from the domain `golfshule-mcqueen.de` and there was 23 file downloaded which could not be recovered on the forensic image of the host machine in Figure 4.6.

Table 4.7: HTTP malicious packets generated by CERBER

Total	Request Code	Meaning
7	404	This indicates that there were 7 instances where the host machine was unable to connect or communicate to a specific server and or find the file it was looking for.
1	504	The server in unable or unwilling to complete the request due to versioning inconsistencies.
23	200	Indicates that the GET request has been completed and returned the requested file.



Host / URL	Count	
+ golfschule-mcqueen.de	13	39.39%
/wp-content/themes/golf-wordpress-theme/images/image.gif?c08a77=25236718	1	7.69%
/wp-content/themes/golf-wordpress-theme/images/image.gif?a06806=84099120	1	7.69%
/wp-content/themes/golf-wordpress-theme/images/image.gif?8043ff=42030075	1	7.69%
/wp-content/themes/golf-wordpress-theme/images/image.gif?602150=62999840	1	7.69%
/wp-content/themes/golf-wordpress-theme/images/image.gif?427b515=69711125	1	7.69%
/wp-content/themes/golf-wordpress-theme/images/image.gif?40790bf=202814013	1	7.69%
/wp-content/themes/golf-wordpress-theme/images/image.gif?3e76b90=327490000	1	7.69%
/wp-content/themes/golf-wordpress-theme/images/image.gif?3c745a4=316955700	1	7.69%
/wp-content/themes/golf-wordpress-theme/images/image.gif?3a71ff7=367706058	1	7.69%
/wp-content/themes/golf-wordpress-theme/images/image.gif?386d3b7=177503013	1	7.69%
/wp-content/themes/golf-wordpress-theme/images/image.gif?335fcb6=377088250	1	7.69%
/wp-content/themes/golf-wordpress-theme/images/image.gif?315d7d4=258815780	1	7.69%
/wp-content/themes/golf-wordpress-theme/images/image.gif?2f5b004=99311624	1	7.69%

Figure 4.6: Downloaded files for `golfshule-mqueen.de`

4.4.3 Post Exploitation Traffic

The network traffic which was captured post execution was examined using the online threat assessment tool Cloud Shark which was introduced in Chapter 3. Which allows for isolating the timeline used by the Ransomware to exploit a system. The data is run against major threat indicators collected. The first packet which has been identified can be seen in Figure 4.6 . This shows a data being sent to a know C&C server using UDP protocol contains the string as can be seen in Figure 4.7 (Sison, 2017). The server this data was sent to is a known FEODO TRACKER REPORTED C&C SERVER GROUP 23, which is a Trojan downloader (Check Point Software Technologies, 2017). This indicates that CERBER sent information to this to download the payload.

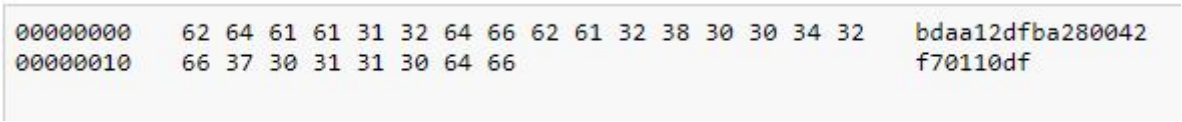


Figure 4.7: Feodo tracker packet

A ICMP error message was received back which was generated by the CERBER C&C server. This shows the same string as in figure 4.7, with additional two bytes in the front of the string. This matches the RIG-V exploitation pack and is known to be used to deliver the CERBER sample. This packet identified on the emerging threats as *"Ransomware / Cerber Checkin Error ICMP Response"* using the online pcap analysis environment *CloudShark*[18].

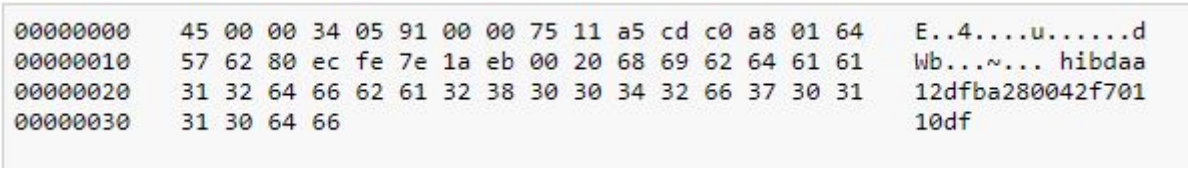


Figure 4.8: Ransomware/CERBER Checkin Error ICMP response flagged

The packet in Figure 4.8 you can observe that the host machine attempts to lookup a domain on the Onion network framework. This domain is known as a CERBER Onion domain and flagged as malicious in the emerging threat listings as "*Ransomware* / CERBER *Onion Domain Lookup*". The flags was created by *CloudShark* using *SNORT* and *Surricata* flags. Once the encryption process has taken place the CERBER sample opens a website which contains the unique URL for the for the victim machine. The communication to the domain is done through the UDP protocol and can be seen in Figure 4.9. The Domain used is `bestfordownload.click` this is a website which is known to be on the RBL lists.

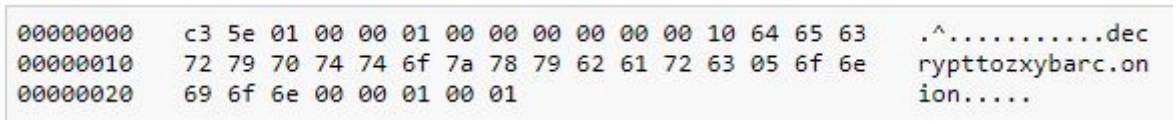


Figure 4.9: Alternate website to pay ransom

There were instances of the same string, *bdaa12dfba28dd* being transferred in fourteen bytes at a time via UDP. This is distributed via the UDP protocol to an entire netblock of IP addresses. An example of the UDP payload is shows in Figure 4.9, due to the magnitude of UDP traffic which took place during this time. This is one of the key characteristics of CERBER malware the fact that UDP is utilised to send the data out to netblocks (Check Point Software Technologies, 2017).

[18]https://www.cloudshark.org/

```
00000000   62 64 61 61 31 32 64 66 62 61 32 38 30 30 34 32   bdaa12dfba280042
00000010   66 37 30 31 31 30 64 66                           f70110df
```

Figure 4.10: UDP Payload Packets of statistics being sent out

The examination showed indication of the network communication containing W32.SALITY elements . This is a known as a file injector that infects files is replicates itself across various network shares. The infected host forms part of the peer-to-peer network which aids in the propagation of malware of the compromised computer. These additional programs are used to spam, relay proxy communications, steal private data or achieve distributed computing tasks such as password cracking. There is a synergy between the file infection and the fully de-centralized peer-to-peer networks make these threats a resilient one. This shows that CERBER malware does not only encrypt the data on the host machine. This variant also downloads a post-exploitation module meaning that there is an additional infection which occurs once the machine has been encrypted.

The CERBER communication timeline showed that the Ransomware follows a specific path in that it looks up the external IP address of the victim machine, it then sends constant UDP packets. It keeps contact with the C&C for CERBER which is done by a known FEODO TRACKER. The Ransomware proceeds to download the W32.SALITY virus.

4.5 Forensic Artefacts

When conducting a **Digital Forensic** examination it is important to note, that each interaction that is made with the system will leave a residual trail. This allows for the piecing together of information from artefacts contained on the system to combine into a timeline. The chapter which follows details the forensic artefacts which were found on the system.

4.5.1 Memory Analysis

The CERBER Ransomware downloads the file `ntdll.exe`; this executable is not contained within the Windows/System32 folder directory, this is important for processes to hide during execution as all system processes is run from this location. This specific dll file was disassembled to identify what changes it would make to the operating system. The file was flagged to fall within the time frame that the CERBER was run on the system. In Table 4.8 it shows in the assembly code portion the executable will register an `ETWRegisterSecurityProvider` which will enable this user to make changes to event tracing on the Windows system. In Figure 4.11 it shows that this specific dll is responsible for inconsistencies within the Event log.

CERBER employs `Page_Guard` which allows the malware to create guarded memory regions this can be seen in Figure 4.11. The use of `Page_Guard` means that anti-reverse engineering

methods are employed to the memory section which does not allow for data to be dumped. The creation of guarded memory regions is the creations of additional inaccessible memory during the allocation of processes. These guarded pages are basically unmapped memory pages which are placed between other mapped paged. The result of these methods is that the memory is fragmented and examination of this memory will be high if not impossible due to the nature of fragmentation[19]. Guard pages have a high degree of overhead because they fragment the kernel's memory map and can increase the amount of virtual space considerably. Their effectiveness depends on the size and pattern of allocations; they are often more effective as a debugging facility than an operational security measure.

CERBER has the ability within its functions to effectively kill or terminate other process this can be observed in Figure 4.12. This allows the Ransomware to have the ability to control the system resources and terminate processes which take away from the CPU system resource. CERBER employs hooking of dll in running processes. The process of hooking is detailed in Chapter 2.1**.** For this to be done the Ransomware needs to have administrative privileges on the host machine. This process is only followed if access and persistence has been gained on the victim's machine. This process is done to evade detection by hooking onto legitimate processes. As can be seen in the table below the CERBER sample uses the `Iexplorer.exe` to hook legitimate processes which it needs to effectively complete its execution. This is done by the following the following steps:

- Inject the dll into the address space of the process, this means that for the dll injection to work there needs to be a process running already such as the `Iexplorer.exe`.
- Modification need to be made to the Import Address Table to reflect the new address where the dll was injected to.
- Uses the proxy dll and manifest files to change the information.
- Loading the necessary drivers into the kernel address space.

There is however another option for dll hooking in malware and that is to inject a custom dll into the address space of a running process. This means that the injected dll leaves a smaller footprint on the system. This force the system to load the dll into the address space rather than the original one. The location `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\Current Version\Windows` contains a registry value `AppInit`[20] which contains the dll which are loaded in the library. The function `SetWindowsHookEx` is accessed to facilitate the hooking of dll which are contained in the registry key `AppInit`, this would prompt the `CreateRemoteThread` function to be invoked which would load the malicious code which was hooked onto.

[19]https://docs.microsoft.com/en-us/windows/desktop/memory/memory-protection-constants
[20]https://support.microsoft.com/en-us/help/197571/working-with-the-appinit-dlls-registry-value

Table 4.8: ETW Tracing

ETW Functionality	Description
ETWEventEnabled	This function verifies whether there is a specific event enabled.
ETWEventRegister	This function register a new event agent which has to be done before tracing can be done.
ETWEventUnregister	This function unregisters event agents if needed to ensure that they are not included in the trace.
ETWEventWrite	This function is responsible for writing a new event.
ETWTraceEnableFlags	This function allows you create parameter for capture tracing events
ETWEnableTraceEnableLevel	This function enables the flags which has been set.
ETWGetTraceLoggerHandle	This function is responsible for retrieving the handle of the event tracing session.
ETWRegisterTraceGuids	This function registers the provider which should be used for the trace along with the events and the providers.
ETWRegisterTraceMessage	Specifies the location for the TMF file.
ETWUnregisterTraceGuids	This function is used to deregister a provider.

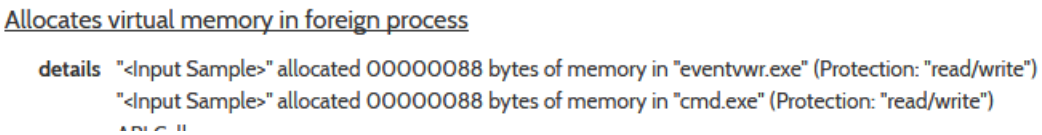


Figure 4.11: Memory Segmentation avoid memory Dumping

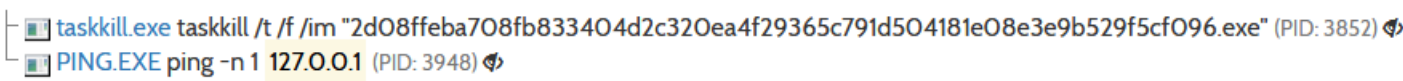


Figure 4.12: Task Kill function command found in memory

Table 4.9: Processes Hooked

Executable	Message	Written To
Iexplorer.exe	e9fda492f9	OLEAUTH32.DLL
Iexplorer.exe	e939548cf9	OLEAUT32.DLL
Iexplorer.exe	e9efb9cefa	COMCTL32.DLL
Iexplorer.exe	e98b8e92f9	OLEAUT32.DLL
Iexplorer.exe	e9b29630f9	OLE32.DLL
Iexplorer.exe	e9652bd5f7	USER32.DLL
Iexplorer.exe	e9652bd5f7	USER32.DLL
Iexplorer.exe	e9c20ae2f7	USER32.DLL
Iexplorer.exe	e9e89acef7	USER32.DLL
Iexplorer.exe	e96ff1e0f7	USER32.DLL
Iexplorer.exe	e9b943c4f7	USER32.DLL
Iexplorer.exe	e99d9a80f9	OLEAUT32.DLL
Iexplorer.exe	e937f2e0f7	USER32.DLL
Iexplorer.exe	e9e9f0e0f7	USER32.DLL

CERBER creates various mutex objects which have been listed in the list below. These mutex objects are used to ensure that the same host is not reinfected. This is done when a specimen attempts to create a specific handle but finds that the handle is already open thus showing that the host is already infected. There is also a theory by Abhishek and Bhu (2017) that the component of creating mutex objects is for the malware to be able to synchronise the injections of multiple copies of itself into various processes. This is further supported by information detailed by Elisan (2015) which details that the use of mutex files by malware is to be able to track the infection spread. The creation of the mutex files would contain the machine-GUID value which is calculated for each machine in the instance this value for the purpose of the research was *{1D92E976-C282-FCC7-9033-A7F49749543B}*, this can be seen to be repeated in various mutex creations, as a method to identify each infection uniquely. The value *[rgzmuyf]* is a random generated value for the purpose to use for the extension of the encrypted files, with each execution the malware will create a randomised string value. The mutexes created are detailed in Section B.1.

4.5.2 Registry Analysis

CERBER exhibits signs of accessing the cryptographic machine-GUID. This is done by accessing the following keys within the Windows registry of the host machine. This version acts differently than the previous versions as the extensions are no longer CERBER but rather randomised extensions. This can be observed in the creation of the mutex creation in Section 4.5.1. However, Sison (2017) theorises that the extension is the CERBER using the system information to create the encrypted file extension. The CERBER Ransomware exhibit this behaviour as can been seen in Table 4.9. CERBER tokenizes the GUID value using a hyphen the calculations are done, this can be seen in Figure 4.14. This can be used as a heuristic detection method for the CERBER sample. Theoretically the encryption process only starts after the component file was dropped onto the system. This means that

there is a early detection of the behaviour of the machine-GUID to prevent the further encryption of the files on the host machine.

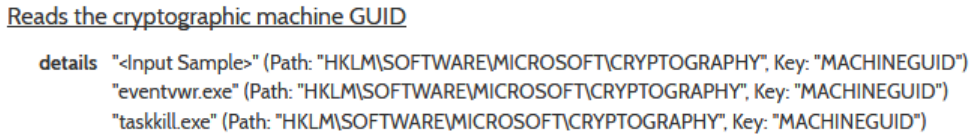


Figure 4.13: Queries the Machine-GUID

Table 4.10: Tokenisation of the Machine-GUID

1	8 Characters	This is the characters used for the folder name created by the Cerber malware strain in the %EMP% directory.
2,3	4 Characters each	These character sets are what is used for the Cerber component files.
4	4 Characters	These are the characters used as the extension for the encrypted files.

The COM objects is the platform which independent, distributed, object orientated system that a binary or software component can interact with. This was designed by Microsoft to provide an interface that allowed for developers to control and manipulate other objects of other applications. This process is often used by the malware operators to create persistence in a stealthier manner. This is done in the case of the CERBER sample by altering the following registry key:

HKLM\SOFTWARE\Classes\CLSID\{00021401−0000−0000−C000−000000000046}\InProcServer32

Rascagneres (2014) states this new way of persistence mechanism has many advantages over the other methods of persistence. As soon as the CERBER sample has infected the system it is able to bypass the anti-virus monitoring due to Windows natively executing the library which contain the process which has been infected. This also can make it harder for the victim to identify the process which is malicious. This attack according to Rascagneres (2014) has been seen to be used in conjunction with a Remote Access Trojan. In terms of the CERBER sample the initial exploit which is run on the system contains elements of a Remote Access Trojan and is discussed in Section 2.2 relating to the classification of Ransomware and their behavioural trades.

CERBER accesses the registry key as can be seen in Figure 4.15 which displays the information relating to the Windows installation date. This is done to determine the environmental variable of which operating system and service pack is installed to ensure that the malware dependencies are favourable. CERBER drops an executable `rasdial.exe` access the registry key related to the Service Control Manager of Windows. This is done to be able ensure that the CERBER is able to escalate its privileges to that of the `iexplorer.exe`, this is done by hooking the malicious executable `rasdial.exe` onto the function `iexplorer.exe`. This can be observed in Figure 4.13

CERBER attempts to access the system services which allow for it to access information on the environment which it is running on. These are all accessed via API calls on Windows. This allows for the CERBER to be able to identify the variables of the system along with searching for more information. At this stage the Ransomware has gained access to the host machine and escalated its privileges. The next step would be to maintain persistence on the host machine. This is done by accessing various registry keys and altering their values as can be seen in Figure 4.15. The changes that are made ensures that at startup the system would specifically look for the CERBER executable at start up and if not found the file will be downloaded to ensure that persistence is kept on the system.

0040330E	FF15E4534100	call dword ptr [004153E4h]	EnumWindowStationsW@USER32.DLL (Import, 2 Params)
00403314	8B75FC	mov esi, dword ptr [ebp-04h]	
00403317	3BF3	cmp esi, ebx	
00403319	744B	je 00403366h	target: 00403366
0040331B	FF15A8534100	call dword ptr [004153A8h]	GetProcessWindowStation@USER32.DLL (Import, 0 Params)
00403321	50	push eax	
00403322	56	push esi	
00403323	E82E7C0000	call 0040AF56h	target: 0040AF56 executed
00403328	59	pop ecx	executed
00403329	59	pop ecx	
0040332A	85C0	test eax, eax	
0040332C	7431	je 0040335Fh	target: 0040335F
0040332E	56	push esi	
0040332F	FF15B8534100	call dword ptr [004153B8h]	SetProcessWindowStation@USER32.DLL (Import, 1 Params)

Figure 4.14: Query of various values of services

Installs itself for autorun at Windows startup details

Type	Info
registry	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce\sdchange
registry	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\sdchange
registry	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
file	C:\Users\Virtual\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\sdchange.lnk

Figure 4.15: Registry changes to maintain persistence

CERBER makes attempts to access the sensitive stored information of the user in Internet Explorer. This also goes hand in hand with the previous steps to ensure low or no network security making it easier for the malware to establish communication channels to the outside. In Figure 4.14 the Ransomware accesses the Internet Explorer registry keys and changes the value to disabled effectively disabling any security settings put in place. Attempts are made by the CERBER to identify the supported languages. This enables the malware to create the ransom note in your supported language. This added feature makes CERBER a versatile Ransomware strain. CERBER accesses various Windows registry key values to determine which versions of software are installed on the system. This is done to detect whether there is antivirus tools or virtual machine software installed. This is purely done for gaining more information on the victim's machine it infecting.

4.5.3 Digital Forensic Artefacts

Sikorski and Honig (2012) states that malware makes use of Windows function to enumerate network shares to laterally move on a network. To access network drives the network must first enumerate the users/drives allowed on the network, this is where the function "`NetUserEnum`", is used to enumerate the users on the network. The following step once the enumeration is completed is to identify the administrator account or account with highest privilege and enumerate the information of that specific user information on the network. Once this has done the "`NetAPIBuffer`" is used to free up memory for the allocation of network management on the network. The functions used by the CERBER sample can be seen in the list below.

NetUserEnum This retrieves information regarding all the user accounts on an operating system

NetUserGetInfo Retrieves the information about a specific user account on the server

NetAPIBufferFree This function frees the memory that the NetAPIBufferAllocate can allocate the network management,

CERBER uses various native Windows functions to create a new security identifier and obtain the required level privileges. The CERBER malware obtains the `Kernel32.dll` process which is a system level process that has administrator level privileges. It then proceeds to open the process token assigned to this process. Once the information is obtained it reassigns this to a new security identifier and checks once the process is complete whether this user is indeed part of the Administrator group. This can be seen in Table 4.11. Windows makes use of security descriptors which aids in the control access of these objects. The security descriptors include the information about which process owns an object and what level of access this object has. These contain the access control list of the various objects within the Windows operating system. A Windows object can contain two types of access control lists[21]:

- **Discretionary Access List:** discretionary access control list which is responsible for identifying the users and the groups allowed or denied access.
- **Access Control List:** System access control list which controls how the access is audited.

Table 4.11: Functions used by CERBER exploiting the Token Memberships

Process	Description
AllocateAndInitializeSid	Function allocated and initializes a new security identifier with up to eight sub authorities
CheckTokenMembership	Function determines whether the specified security identifier is enabling with an access token.
FreeSid	Function is used to free a security identifier which has previously been allocated and initialized

[21] https://docs.microsoft.com/en-us/windows/desktop/secauthz/access-control-lists

The executable CERBER file, creates a folder within the %APPDATA% folder which allows it to manipulate processes, and handles. The `eventvwr.exe` file creates a handle under 3760 which is where all exploits will be run from. When a process is assigned by the operating system memory allocates a handle which will be used for all processes within memory associated with the executable. This remains to be seen whether CERBER has an influence on the handle assigned and whether it would remain the same after multiple executions. This is for further research purposes. This process is a remote process on Windows. This can be seen in Figure 4.16 where the sample places remote process into `eventvwr.exe`.

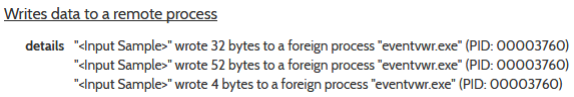


Figure 4.16: CERBER Ransomware writes to remote processes

The ETW otherwise known as the event tracing for Windows is a diagnostic tool which can be used to troubleshoot performance issues. This allows for user to perform specific troubleshooting on the system related to how it functions and performs. Some of the functionalities with ETW tracing for Windows is that is allows for the querying of events and how they are logged within the system. It allows for performance testing of the CPU and how much it can handle. Furthermore it allows for the ability to assign memory to processes. This is a valuable tool which is accessed by the CERBER sample to ensure that it has the ability to query the CPU to identify the resources available to it. The CERBER sample also uses it to register and de-register events rendering the Event Logs non reliable. The following files were identified to be deleted and zeroed out which means there is not data within them.

- EtwRTEventlog-Security.etl
- EtWRTLBPM.etl
- EtWTMsMpPSsSession7.etl

4.6 MITRE ATT&CK Map

The CERBER sample was examined using the online sandbox environment Hybrid-Analysis[22]. This online sandbox maps the behaviour of the samples which were submitted to the MITRE ATT&CK adversarial technique map. The results can be seen in Figure 4.17. This indicated that the initial point of access for CERBER is access by use of valid user accounts or removable media. CERBER infiltrates a system by stealing the credentials of specific accounts or users. This can be done via spear fishing and dedicated attacks. This indicates that CERBER relies on the weaknesses of users to divulge this information. CERBER is able to infect air-gapped networks due to fact that it can infection removable media and

[22]https://www.hybrid-analysis.com/

use this process to infect multiple machines. This is a technique to laterally move within a network and spread the infection.

The execution of CERBER is done by executing a script via a method which interacts with the Windows services which is the Service Control Manager. From earlier examination during the dynamic examination phase there was indication of interaction by CERBER with the Service Control Manager. This technique is often used by advanced persistent threat actors along side the execution of new services and modification of existing services. This is generally added in the persistence phase or escalation of privileges.

CERBER maintains persistence on the system by using a valid account and impersonating this account and reassigning the access list it belongs too. The compromise of a local or network account allows the malware to associate its processes with a valid account. CERBER adds entries to the 'run keys' within the Windows Registry or the start up folder to ensure that the infection executable is re-run at start up. These programs will be executed in the context and under the permission of the user it is associated with. To maintain the persistence on the system CERBER manipulates and modifies existing services on the system. This is done using a masquerading as legitimate services. CERBER not only has the ability to install and configure utilities or services on the system. CERBER creates services associated with the administrator account however it is executed under `SYSTEM` privileges.

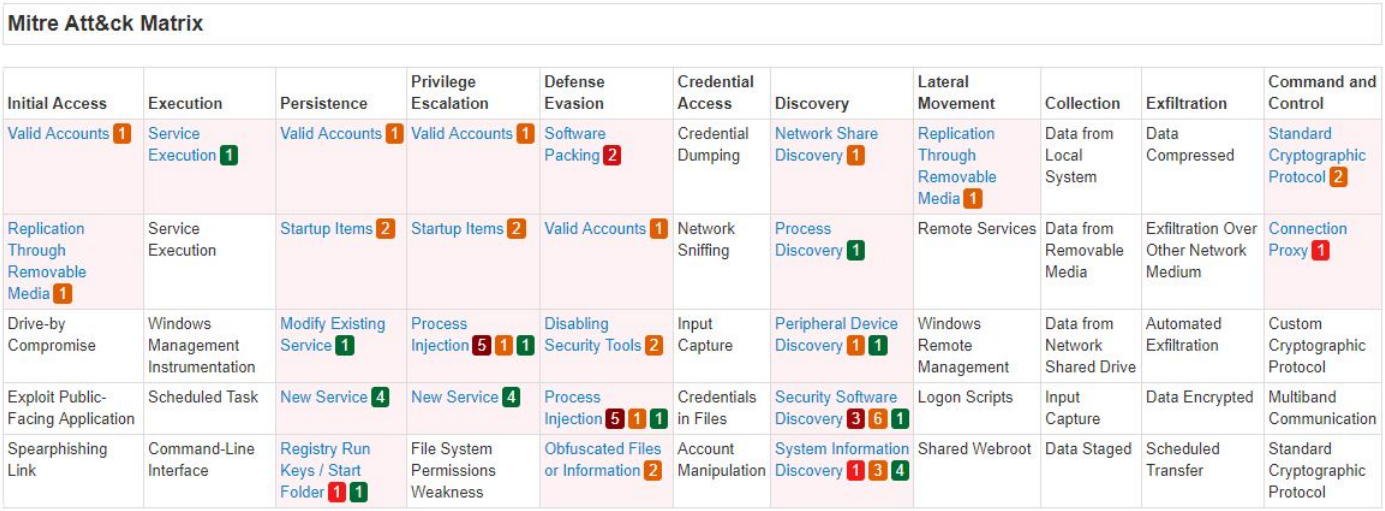
CERBER escalates its privileges by identifying the administrator account and impersonating those accounts. The process which is used to inject the malicious code in by CERBER is often a process which runs at `SYSTEM` level which is the highest level of privilege within a Windows system. This ensure that every time the code is executed the process is launched with administrator privileges. This is commonly used by malware to gain privileged access to system resources. CERBER when the operating system is booted up this Ransomware identifies a legitimate service and hooks onto that by masquerading as this service to maintain privilege persistence on the system.

CERBER does not pack the executable binary, it does however make use of entropy for the sections where the code is contained. This means that static examination can be difficult and result in false positives. The one method which is used by CERBER which is unique to this strain is that it has the ability to disable security tools by killing the process and altering keys within the registry to disable the software.By using process injection this allows for the malicious code to be masked by a legitimate process which makes detection harder.

The discovery methods used by CERBER is that it enumerates the network shares which a user has access too. File sharing on Windows is done using the SMB protocol. The identification of the level of access that a user has enables the lateral movement which can be achieved. The malware attempts to obtain information of running processes that it can identify the highest level of privilege running process it can inject malicious code into. CERBER not only identifies the network shares it identifies the peripheral devices connected to the system and infects these. The discovery phase also sees CERBER identifying the security software installed on the system which it disables. The CERBER Ransomware

attempts to gather information regarding the remote systems and IP addresses associated with a system which is a tactic associated with remote access tools.

CERBER does not have the ability to move within a network laterally, it however has the ability to replicates itself onto removable devices and use this to move laterally within a network. The movement may occur due to an infected file being accessed or malicious binary. CERBER communicates using a known standard cryptographic protocol to mask the communication with the C&C. This was discussed in previous work done. The use of standard encryption means that with enough time the protocol can be reverse engineered. CERBER used a connection to a proxy to direct traffic between systems for communication.



Mitre Att&ck Matrix

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command and Control
Valid Accounts 1	Service Execution 1	Valid Accounts 1	Valid Accounts 1	Software Packing 2	Credential Dumping	Network Share Discovery 1	Replication Through Removable Media 1	Data from Local System	Data Compressed	Standard Cryptographic Protocol 2
Replication Through Removable Media 1	Service Execution	Startup Items 2	Startup Items 2	Valid Accounts 1	Network Sniffing	Process Discovery 1	Remote Services	Data from Removable Media	Exfiltration Over Other Network Medium	Connection Proxy 1
Drive-by Compromise	Windows Management Instrumentation	Modify Existing Service 1	Process Injection 5 1 1	Disabling Security Tools 2	Input Capture	Peripheral Device Discovery 1 1	Windows Remote Management	Data from Network Shared Drive	Automated Exfiltration	Custom Cryptographic Protocol
Exploit Public-Facing Application	Scheduled Task	New Service 4	New Service 4	Process Injection 5 1 1	Credentials in Files	Security Software Discovery 3 6 1	Logon Scripts	Input Capture	Data Encrypted	Multiband Communication
Spearphishing Link	Command-Line Interface	Registry Run Keys / Start Folder 1 1	File System Permissions Weakness	Obfuscated Files or Information 2	Account Manipulation	System Information Discovery 1 3 4	Shared Webroot	Data Staged	Scheduled Transfer	Standard Cryptographic Protocol

Figure 4.17: CERBER Behavioural Matrix
Source: `https://www.hybrid-analysis.com/`

4.7 Summary

The file was not packed however it did contain sections of high entropy. The MITRE ATT&CK framework is fairly new in its application to malware, there has not been any other academic work found which uses the framework in the same way as it was used in this work. The number assignments which can be observed in Figure 4.17 is the value assigned a risk scoring from 1 being the least to 10 being the highest. The Dynamic analysis portion indicated that the CERBER sample had the ability to evade detection by the sandbox in that it is able to check for the presence of the virtualised environment and sleep for long periods of time. The CERBER sample with regards to network traffic has the ability to check the Local IP address of the machine being infected on a constant basis to determine the location this is done due to the exclusion list which has been discussed in Section 2.9.1. The Forensic analysis yielded information in the methods used by CERBER to manipulate the Windows 7 operating system, by *'living off the land'*. The weaknesses within Window's design is what is exploited most by the CERBER Ransomware. The MITRE ATT&CK map indicated that CERBER makes use of various known methods to exploit the Windows machine.

5

Case Study: MAKTUB

5.1 Introduction

This Chapter of the thesis contains the results from the process which has been used in Chapter 3. The examination of this has been done to identify the static properties of the executable, the dynamic process follows the framework as detailed in Section 3.4 of Chapter 3 as applied to the other samples. KnowBe4 (2018) describes MAKTUB as a professionally written piece of code originating from Russia.

MAKTUB was initially charging a ransom of 1.4 Bitcoins and slowly increased to 3.9 Bitcoins. Whilst the infection only lasted for 2016 multiple derivative strains have been born from MAKTUB primary (KnowBe4, 2018). There was no conclusive statistics found on the campaign for MAKTUB. This Chapter contains the finding in relation to the framework which has been set out in Chapter 3. The Static Analysis is contained in Section 5.2 which is flollowed by the follows the Dynamic Analysis which is the execution of the malware sample and is set out in Section 5.3 . The Network Analysis in Section 5.4 contains all the communication which was identified on the network that has been sent and received. The Forensic Artefacts are detailed in Section 5.5.3 which contain all the artefacts which were identified using Digital Forensic processes. This chapter concludes with a summary of the findings of the case study for MAKTUB in Section 5.7.

5.2 Static Analysis

This section reports the initial phase of the examination of MAKTUB. This process entails the analysis of the static malware file to determine what intelligence can be ascertained

without dynamically running the malware. The main limitation to the examination was the fact that MAKTUB is an executable which is packed, This means that the core information could not be extracted during this phase as they are decrypted during run time.

5.2.1 PE Structure

The MAKTUB sample was examined using *CFF Explorer*[1] to view the language libraries within PE Structure. The sample contains multiple country selections, which indicates that this strain is highly customisable to be used to target different countries. This could indicate that due to MAKTUB being used as a Ransomware-as-a-service product they authors ensured that the product is versatile to target multiple countries. This would mean that in terms of business a larger base of potential victims. The sample which was examined had a single language library which was English. MAKTUB has the ability to be able to register a top-level exception. This means that the MAKTUB is able to supersede the top-level handling of a thread within a process. This also enables the MAKTUB Ransomware to run in Kernel mode with escalated privileges, instead of that of User-Mode. This information was found within the PE file examination, the second figure was found when reversing the executable MAKTUB binary using PE Explorer.

The idea behind the use of exception usage as an anti-debugging is to make the static analysis of the malware sample harder than those who do not make use of it. The method used by the MAKTUB malware strain which was examined is to set the setting for debugging to execute as a top-level execution handler name Kernel32. The `Kernel32.dll` forms part of the default exception handlers within Windows. When an executable is run on Windows and there has been no exception handler assigned to run the debugging process, this function is taken over by in Kernel32 (Hasherezade, 2016c). The Top-level exception filters are used because that has the top-level/administrator rights on the operating system. This behaviour is done to detect the presence of a debugger on the system. Once this has been found an exception is passed and an application error is forced on the system. If this process is not being debugged it will continue on. This is a technique often used as an anti-debugging process. The basic idea behind the exception anti-debugging techniques is to use side effects of a debugging tool (or of decisions made by the person using the tool) to alter the execution flow. There are several ways to do that, the classic being setting an exception an exception handler (either for the local thread or at the process level) and implementing important parts of the logic in the exception handler before returning execution to the original EIP or changing it completely (Hasherezade, 2016a).

Malware makes use of various methods to detect whether they are run in an environment which is debugging their processes. MAKTUB makes queries to the `GetProcessHeap@Kernel32.dll` function to identify whether there are any debugging artefacts found. Pointers can be retrieved from the flags. The heap has two fields which are important to establish the anti-debugging field. The flag fields are indicative of the settings for the heap block which is used. There are specific values stored within the flags which indicate settings that will

[1] *https://ntcore.com/?tag=cff-explorer*

be used in heap manipulation or debugging. This is specifically done to protected packed malware code which is unpacked as it is executed (Abrahams, 2016).

5.2.2 Version Information

MAKTUB was examined using *PEStudio*[2] it was the following file information was ascertained. In Table 5.1 the File version information can be seen which was extracted using *PEStudio*. The MAKTUB malware file which was examined using *PEStudio* claimed to be legitimate software named `SafariBackTrack.exe` software which is a text editor. The information collected during the static phase of the analysis clearly show that this version of MAKTUB masqueraded as legitimate software. This would mean that the tool would be downloaded for legitimate processes and the victim would infect themselves. The MAKTUB malware file which was examined using *PEStudio* claims to be legitimate software name `Tandberg Constellation.exe` which is an archiving tool. The information collected from during the static phase of the analysis clearly shows that this version of MAKTUB masquerades as legitimate software. This would mean that the tool would be downloaded for legitimate processes and the victim would infect themselves.

Table 5.1: File Version Information from PEStudio

Version Number	6.5.8.7
Internal File Name	SafariBacktrack
Company Name	Emurasoft, Inc.
Legal Trademark	(C)Emurasoft, Inc. 2007-2015
File Description	Formula Interpret MoocsQmac
Purpose of software	Text Editor
Attack Vector	Infected software

5.2.3 Windows API Class

MAKTUB was examined and according to the PE Header information imports some libraries/functions from the dynamic link library files. The following functions allow the malware to interrogate the registry values, open keys, make changes to keys, enumerates new key, obtain information on modules open within memory, identify whether debuggers are run on the Microsoft Windows environments, terminates processes, create shells and many more as can be seen in the below list the malware functionality within the PE32 structure would indicate that the infection allows for maximum access to the target machine, this enables the malware to have keys to the kingdom effectively. The creation of shells can be described as malicious code which is run in the background by the malware. The shell gives the malware the ability to run custom designed code to perform specific tasks running at a system or administrator level. This is a known method which malware uses is explained as the malware starting the `WSAStartup` which initialised the Win32

[2]https://www.winitor.com/

socket system, this socket is created and the bind function is used to attach the socket to a specific port and sets it up to listen and accept all calls from the remote socket (Sikorski and Honig, 2012).

- `Kernel32.dll`[3] handles memory management, input/output operations, and interrupts this link library file is loaded into the protected memory sections.
- `AdvAPI32.dll`[4] provides security calls and functions for manipulating the registry.
- `GDI32.dll`[5] :functions that perform primitive drawing functions for output to video displays and printers.
- `Secur32.dll`[6] is a dynamic link library file which is used by programs or web browsers.
- `User.32dll`[7] is the Windows API responsible for user interface namely anything associated with the Windows GUI.
- `ws2_32.dll`[8] is responsible for communication from the Windows Sockets API that handles network connectivity for applications.
- `WINMM.dll`[9] is a module which is used by the Multimedia API which is responsible for Audio and joystick processes.
- `ole32.dll`[10] is a dynamic link library file which contains the core OLE functionality of the operating system.

5.2.4 Entropy

Entropy analysis is an important factor in malware analysis (Osaghae, 2015). This can give the researcher an indication on where to start the analysis. Signs of low entropy can indicate that the malware sample is not compressed nor have obfuscation sections. In the case where the entropy level is high within specific sections can indicate that the sample is packed or that obfuscation methods have been used. To understand the meaning of entropy one needs to understand the definition thereof. The simple definition for entropy is the randomness in which bytes display within a section within the PE file. The green sections of the visualisations is the high entropy sections and the varied black sections are less entropy. The binary representation in Figure 5.2 the areas of entropy is pixelated and the other sections are shaped. This means the examination of a file to determine the measurement of entropy is simply put the measurement of disorder of the bytes within a file (Osaghae, 2015). An interesting area of malware file analysis is the measurement of

[3]https://www.webopedia.com/TERM/K/Kernel32_dll.html
[4]https://en.wikipedia.org/wiki/Microsoft_Windows_library_files
[5]https://en.wikipedia.org/wiki/Microsoft_Windows_library_files
[6]https://www.file.net/process/secur32dll.html
[7]http://www.processlibrary.com/en/directory/files/user32/19597/
[8]http://www.processlibrary.com/en/directory/files/ws2_32/24187/
[9]http://www.processlibrary.com/en/directory/files/winmm/23783/
[10]http://www.processlibrary.com/en/directory/files/ole32/23128/

file entropy. The first two section within Figure 5.1 from left to right is the sections with the highest entropy. This binary however visually it can be observed to contain multiple layers of entropy.

MAKTUB was examined using the tool *Detect-It-Easy*. The tool displayed high levels of entropy and indicated that the binary was packed. There were three sections which contained high entropy levels these were the `.text` section, the `.rdata` section and the `.gaxe` section. These sections contain the actual code portion which is contained in the `.text` section. The `.rdata` section contains data which has been initialised and has read access this section does not contain actual code. The `.gaxe` section of the MAKTUB binary contained data which is initialised which has writes access. This makes the static analysis portion harder as the binary is packed and has high entropy levels. This is specifically to prevent reverse engineering of the code. The entropy levels can be observed in Figure 5.2 and Figure 5.1. The areas within Figure 5.2which have more randomness in pattern are the areas which can be considered to be packed. The areas which are not random in pattern are the areas which have low entropy of have not been packed.

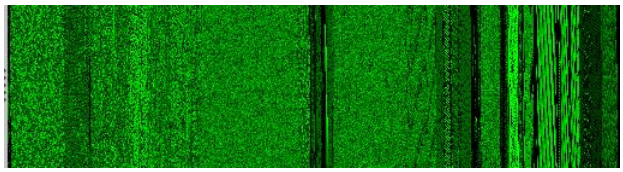


Figure 5.1: Binary visualisation using Veles

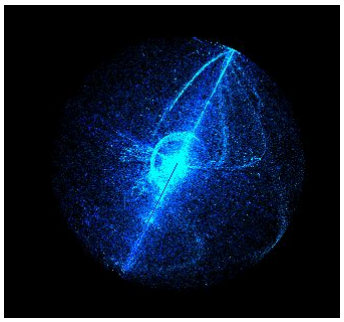


Figure 5.2: Entropy visualisation using Veles

5.2.5 Hash Fingerprinting

The hashing of malware has been a practice used by malware analysts to uniquely identify these strains of malware. The sample which was examined was run through varies methods of hashing to determine the unique has fingerprint for this particular MAKTUB strain. This allows for signature based detection of the fingerprinted strain. The traditional method of hashing a piece of malware for signature based detection is MD5 and SHA256, this is however cryptographically a black and white approach, because if one bit of data has morphed or changed the signature will fundamentally change. The suggestion

as described by Akira *et al.* (2016) is to add into the existing fingerprinting techniques SSDeep hashing which is referred to as fuzzy hashing.

5.2.6 Strings

The MAKTUB binary was examined using various static analysis tools to examine the embedded strings one of the tools which was used is PE Explorer. A string was observed which is used to inject a process into Explorer. The string was identified using PE Explorer. MalwareTech (2013) details that malware has the ability to inject string code within remote processes, this enables the malware to remotely execute code whilst avoiding creating threads within memory. This methods can be used to trick the Windows Explorer tray window procedure process to execute malicious code. When malware opens the `Shell_TrayWnd` and calls upon the `SetWindowLong` the malware is able to set the variable used by the window procedure to point to a specific portion of shell code.

MAKTUB has the ability to be able to register a top-level exception. This means that the Ransomware is able to supersede the top-level handling of a thread within a process. This also enables MAKTUB to run in Kernel mode with escalated privileges, instead of that of User-Mode which initially is used by `word.exe`. This information was found within the PE header, the second figure was found when reversing the executable MAKTUB binary using *PE Explorer*. The idea behind the use of exception usage as an anti-debugging is to make the static analysis of the malware sample harder that those who do not make use of it. The method used by the MAKTUB which was examined is to set the setting for debugging to execute as a top-level execution handler name Kernel32. This can be seen in Figure 5.3.

```
@40a097: push 0040A055h
@40a09c: call dword ptr [0041116Ch] ;SetUnhandledExceptionFilter@KERNEL32.dll
@40a0a2: xor eax, eax
@40a0a4: ret
```

Figure 5.3: Exception Handler

The `Kernel32.dll` library forms part of the default exception handlers within Windows. When an executable is run on Windows and there has been no exception handler assigned to run the debugging process, this function is taken over by the `Kernel32.dll`. The Top-level exception filters are used because that has the top-level/administrator rights on the operating system. This behaviour is done to detect the presence of a debugger on the system. Once this has been found an exception is passed and an application error is forced on the system. If this process is not being debugged it will continue on. This is a technique often used as an anti-debugging process. The basic idea behind the exception anti-debugging techniques is to use side effects of a debugging tool (or of decisions made by the person using the tool) to alter the execution flow. There are several ways to do that, the classic being setting an exception an exception handler (either for the local thread or at the process level) and implementing important parts of the logic in the exception handler before returning execution to the original EIP or changing it completely (Abdulla *et al.*, 2010).

5.2.7 Code Obfuscation Techniques

The sample examined makes use of a data obfuscation technique which is used to protect code from static analysis. In assembly code when a "call" is issues it point to a return address which is the address where the execution should take place. The MAKTUB sample used three functions together which is the `call`, `ret` and `jmp` functions. (Dalei and Li, 2016) explains that malware authors obfuscate their code with the intention to make it complicated to gain information from the code itself. Malware operators rearrange their code to make the detection process harder. The most popular method used by malware authors according to (Kumar, 2004) is that they obfuscate the call addr instruction in Table 5.2. These MAKTUB code obfuscation technique can be seen in Table 5.2. These techniques are detailed Section 4.2.7.and MAKTUJB makes use of the Instruction reordering.

Table 5.2: `call,push,ret` MAKTUB

Functions	Description
Call Address	This function will make a push to the updated program counter. This is the pointer to the instructions after the call which should be executed.
Ret	This specific function internally pops and addresses off the stack and then jumps to that address. This matched with call would allow for the return to the instruction after the prior call.
Jmp	Simply put this function is responsible to jump to a specific address .

5.2.8 Summary

The examination of the MAKTUB strain identified that this family is one that included sophisticated and advanced social engineering methods in the way that the document is crafted. The document immerses the reader in the contents whilst the compression and encryption runs in the background. The MAKTUB Ransomware employs a sophisticated and well crafted Crypter/FUD which makes the static examination of this particular strain harder.to do. This particular version of MAKTUB contains a CRC checking error and did not match the software it claimed to be according to the actual calculated value. The sample further more targets the Microsoft Windows operating system by exploiting the Windows Native API's to exploit, manipulate and and extort the system.

5.3 Dynamic Analysis

The following section details the results from the dynamic execution phase of the Hybridised-Malware Analysis portion. The system was monitored using the methodology detailed in Section 3.5.

5.3.1 Sand boxed Run-time Report

The purpose of running the sample through the sandboxed environment is to identify evasion techniques, and to confirm findings made on the physical machine examination. The MAKTUB sample was firstly run through VirusTotal[11] which checks various antivirus software to identify whether the file is flagged as malicious. The sample was found to be identified by 75% of all anti-virus software. This means that there is still 25% of antivirus tools which do no flag this sample of MAKTUB as malware. The sample was run for a second time through an online sandbox SecondWrite[12] the use of SecondWrite was to use another sandbox to fill any shortages of the other sandboxes. This however presented the same results as the use of VirusTotal.

The main objective for malware besides infecting the victim's machine is to evade detection. Malware researchers make use of a virtualised environment to analyse the behavioural traits of malware. Hassan (2015) explains that malware has become environmentally aware, malware operators are able to use key characteristics of the target system which they wish to infect to identify whether they are sandboxed. This is successful because even though the virtual environments are becoming more sophisticated they still lack some characteristics of a genuine environment. The MAKTUB malware strain makes use of what is known as a sleep call. This means that the malware is set to execute the malicious code after effectively a sleep period. Even though sandboxing solution have come up with an answer to the sleep calls made by malware, the malware has adapted to check the system time against various internet time sources such as NTP servers and websites with configures HTTPS.

MAKTUB has a sleep period which exceeds three min before the malicious portion of the code will be executed and unpacked. This is the same method used by Cerber which is detailed in Section 4.3. The TROJAN NAP is linked to the KELIHOS BOTNET (Abhishek and Bhu, 2017).The theory is that code which is executed in a sandbox should run the same way it would on the victim's machine. However as stated by Abhishek and Bhu (2017) this is not the reality. Sandboxes no matter how sophisticated have characteristics that the environment is indeed a sandboxed one rather than a physical machine. Malware operators have taken these and incorporated them into their built in defences.

Sandboxes which monitor the behaviour of an application come with a time delay, this can be measured and identified my malware and be an indication that it is in a sandboxed environment. Even though the sandboxed environments attempt to fake the time delay, malware is able to bypass the local time and compare that with an external source. MAKTUB continues the evading process by identifying the time or tick counter for the process by using the "`GetProcessHeap`" and identifying the time. It takes it a step further than the conventional evading techniques by attempting to generate a random string. This process can be seen in Table 5.3.

[11] https://bit.ly/2F4mzj8
[12] https://www.secondwrite.com/

MAKTUB similarly to findings in Chapter 4 and Chapter 6 which were examined in this research exploit the Native Cryptography API of Windows. Dell Secureworks (2015) explains that malware uses the machine-GUID value to identify when it is in a Virtual Environment. This is done by calculating the machine-GUID of the machine upon first execution to identify the value. Once the infection continues the value is calculated again. When this is done on a virtual environment the values will be different and the inference is made that the malware is run in a virtual environment. Even though there are other uses this one shows that the MAKTUB malware strain is environmentally aware and fingerprints the system before the infection continues. The difference with this strain is that even though it detects the virtual machine it will still infect the machine; however, it will not communicate out on the network. As MAKTUB has the ability to fall back on the hard-coded keys it will still encrypt a system even when there is no network. This strain works equally well online and offline. It also does not discriminate against virtualised networks and will still encrypt it but not reveal the secrets of its command and control servers.

Table 5.3: Local System time query

Function	Description
GetProcessHeap	The GetProcessHeap function obtains a handle to the default heap for the calling process.
RtlAllocateHeap	The RtlAllocate heap routine allocates memory to a heap.
GetLocalTime	This function retrieves the local date and time.
GetTickCountCrypt	This function returns a cryptographically strong random string.
GenRandom	This function generates random numbers.
Interlockedincrement	Increments (increases by one) the value of the specified 32-bit variable as an atomic operation.

MAKTUB makes use of different techniques to identify whether it is in a sandboxed environment. Thus far this Ransomware strain has gone into a long sleep before executing the malicious code to bypass the countdown timer in a sandbox, it has accessed the local machine time and done a tick count to identify whether it is in a virtualised environment, the next step that this strain takes is to identify the network adapter information. The process it follows is to identify the path of the system directory; it also obtains the information for the Volume and the network adapter. This is done to detect and gather intelligence on the environment it is run on. This is very sophisticated and shows the attention to detail the malware operators have taken to ensure that there source code is not run on sandboxed and virtualised environments. This shows a level of growth from other variants that does not detect these environments.

5.3.2 YARA Rules

There were 2 samples collected for MAKTUB, however for the focus of the research only one sample was run for the purpose of the thesis. The MAKTUB malware binary which was

chosen, is named, the sample is named using the SHA 256 hash value. The sample was run through the online *Cuckoo Online Sandbox*[13] to be in a position to generate the *YARA* rules results. The results which were identified by the online *Cuckoo Online Sandbox* was that the based on the signatures contained within *Cuckoo Online Sandbox*; this allows the researcher to anticipate the behaviour of the malware. This is however only a summary of the overall examination. MAKTUB has the ability to do the following:

- Anti-Debugging
- Take Screenshots
- Manipulate the Winsock 32 API
- Affect the Private profile within Windows
- Manipulate the Windows token process.

Table 5.4: Network Adapter Query

Function	Description
GetSystemDirectoryA	Retrieves the path of the system directory. The system directory contains system files such as dynamic-link libraries and drivers.
GetVolumeInformationA	Retrieves information about the file system and volume associated with the specified root directory.
GetAdaptersInfo	The GetAdaptersInfo function retrieves adapter information for the local computer.
GetProcessHeap	Retrieves a handle to the default heap of the calling process.
RtlAllocateHeap	Allocates a block of memory from a heap. The allocated memory is not movable.
HeapFree	Frees a memory block allocated from a heap by the HeapAlloc or HeapReAlloc function.

5.3.3 ProcMon

Yonts (2014), states that malware authors will rather avoid writing to memory areas which will leave a residual trace evidence. Not only is it sloppy to have processes committed to memory it also allows for easy detection as processes are monitored by antivirus. These are the locations which are outside the known-good locations. The researcher further states that malicious executable install themselves in writeable areas of the Windows file system it is infecting. This is generally done by installing under the user's home directory in either the Application Data, Local, Roaming or Temp directories. The temp directory is the preferred writeable folder for malware installations. The MAKTUB binary when executed will create temporary files in the %TEMP% directory, the first initial file is the malicious

[13] http://sandbox.pikker.ee/

document contained in a `.CAB file`. A CAB file is a single file that stores compressed file in a file library. The cabinet format is an efficient way to compress multiple files . MAKTUB creates a file *hecova.cab* which contains a copy of the .RTF document which was initially executed. The next step for MAKTUB is to recreate itself in the %TEMP% directory. The process followed after this is to hook onto a legitimate process the process used by the `winword.exe`.

MAKTUB uses the Service Control Manager within Windows, to identify the list of active service running on the local machine. This list contains the active services which have been recorded into the array named "Active List". The interface which manages the services on the Windows environment is the Service Control Manager. Shaaban and Sapronov (2016) explains that within Windows the Service section alternatively known as the Service Control Manager is often used to ensure that a module is run every time a computer boots. This is done to register the executable as a service. This allows for the executable to run in the background as long as Windows is running. The process is to invoke the module which is the Windows API to register the executable as a service; once this is completed it will be registered and run as a service. The MAKTUB binary requests access to the Service Control manager to enumerate a service, and to connect the enumerated service this is done to ensure that once the machine is restarted persistence is kept of the infected machine.

There are various way that malware ensures that they remain undetected, and have persistence on the local workstation. The one way that this can be done is by utilising stand-alone services. These are services that run as an executable or with a specific command-line which is the value contained in the "`ImagePath`". Langendorf (2016) states that this is not the preferred way to create persistence and hide. This technique is however effective to hide malware from "at-a-glance" analysis. MAKTUB installs itself as two services once it has gained persistence on the system.

The two stand alone services which are used by this Ransomware is the `LanManWorkstation` service which is responsible for storing configuration details for the workstation services, these include network connections and communication. This indicates that the MAKTUB exploits the standard message block within Windows. The vulnerability which has been identified as `CVE-2012-1850`[14] is a known to be vulnerable due to the implantation of the Remote Administration protocol.

This implemented in the `LanManWorkstation` service. Windows 7 contains vulnerability in that it does not handle the protocol successfully. This can cause vulnerabilities with messages and network connections. MAKTUB malware installs itself as the `LanManWorkstation` taking control of the service.

The hypothesis here is that it is used to identify Network Shares and utilised to create remote connection to the command and control server. Abrahams (2016) explains that all Windows services are set by default to be loaded at start-up either using the `svchost.exe` or by Windows directly launching the application. Once a service has been loaded by Windows

[14]https://nvd.nist.gov/vuln/detail/CVE-2012-1850

the associate file name or dynamic link file is launched this can be found in the "`ImagePath`" value. Carvey (2016) details the elements within Windows services that are targeted by malware as they are powerful control measures.

The reason for this is that services are generally done with escalated privileges or privileges that are higher than an average user. Furthermore, to the escalated privileges the services start automatically on boot with no interaction needed from the user. MAKTUB makes use of these services to ensure persistence on the system it installs itself into the `LanManWorkstation` service to exploit the vulnerabilities on the network. It furthermore installs itself as the `audiosrv` service which starts automatically by Windows to ensure it is persistent on the system. MAKTUB makes use of the `vssasmin.exe` which is responsible for managing the Volume Shadow Copy process on Windows. MAKTUB was found to create a process which identifies whether the Volume Shadow copy is enable, if this process is found the Ransomware will proceed to disable this process to ensure that the user cannot roll back or restore from a previous copy the two processes which were identified can be seen in Figure 5.11. MAKTUB will proceed to identify previous copies of the Volume Shadow and delete these. The research which has been done previously on MAKTUB did not identify this process as the focus was more placed on decompiling the code.

MAKTUB makes use of the new `MSVCR.dll` file which forms part of the Microsoft Visual C++. This dynamic link file is needed to run programs which are developed by Visual C++. This leads to a clue that the code platform used to create the MAKTUB is reliant of the `MSVCR.dll`. MAKTUB copies the dynamic link file from the Windows System folder to the Microsoft Office folder.

Deletes shadow drive data (may be related to ransomware)	
Source: unknown	Process created: C:\Windows\System32\vssadmin.exe vssadmin.exe delete shadows /all /quiet

Figure 5.4: Volume Shadow Copies - `vssadmin.exe`

Uses new MSVCR Dlls	
Source: C:\Program Files\Microsoft Office\Office14\WINWORD.EXE	File opened: C:\Windows\WinSxS\x86_microsoft.vc90.crt_1fc8b3b9a1e18e3b_9.0.30729.4940_none_50916076bcb9a742\MSVCR90.dll

Figure 5.5: `MSVCR.dll` ProcMon entry

5.4 Network Analysis

The examination focuses on communication during the encryption and post encryption for a period of 24 hours, from initial execution of the malware.

5.4.1 Protocol Breakdown

There were 24,802, packets in total and the breakdown of the different protocols. The MAKTUB sample communicates mostly using TCP which took up tha majority of the malware

traffic, closely followed by the use of HTTP and the UDP traffic which was observed. These are detailed in Table 5.5. The remainder of the traffic was made up of Windows related communication.

Table 5.5: MAKTUB Protocol Breakdown and network traffic

Protocol	Total Packets	Total Percentage
HTTP	521	0,02%
TCP	15,851	63%
UDP	560	0,03%

5.4.2 Malicious Traffic

MAKTUB attempts to download a Windows Cabinet file. This .CAB file when reviewed contained the original MAKTUB malicious file. This is not picked up during the download process. The malware makes use of the HTTP HEAD command to identify whether this cabinet file is still contained on the web server, once the web server returns the answer of HTTP 200 OK . MAKTUB proceeds to send a HTTP GET request to the same web server to download the cabinet file. The cabinet file was examined and found to contain the RTF document which was executed when launching MAKTUB for the first time. This domain was checked and it was determined to belong to Microsoft. This was checked by doing a WHOIS lookup which can be seen in Figure 5.6. The domain `download.windowsupdate.com` was checked on Threat miner and the returned to be a known download site used by various malware including Ransomware.

URI	Domain	IP	Last seen	Sources
http://download.windowsupdate.com/d/msdownload/update/software/defu/2015/09/am_delta_patch_1.207.809.0_82d800528e2102760a4d55c7bad304f6bcbbe8b9.exe	download.windowsupdate.com	N/A	2015-09-25 02:06:45	VirusTotal
http://download.windowsupdate.com/d/msdownload/update/others/2015/09/18740487_553f6eab54ae95ff21a33c994bb71b6eb28b2371.cab	download.windowsupdate.com	N/A	2015-09-20 16:09:10	VirusTotal
http://download.windowsupdate.com/d/msdownload/update/others/2015/09/18738756_56bbc57b9d18296d099523d3ee4d7f2b2da842fe.cab	download.windowsupdate.com	N/A	2015-09-20 16:08:10	VirusTotal

Figure 5.6: WHOIS lookup of `download.windowsupdate.com`

```
HEAD /v11/2/windowsupdate/redir/v6-win7sp1-wuredir.cab?1703201421 HTTP/1.1
Connection: Keep-Alive
Accept: */*
User-Agent: Windows-Update-Agent
Host: ds.download.windowsupdate.com

HTTP/1.1 200 OK
Cache-Control: public,max-age=172800
Content-Type: application/octet-stream
Last-Modified: Mon, 12 May 2014 21:56:34 GMT
Accept-Ranges: bytes
ETag: "115695a2d6ecf1:0"
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Content-Length: 23603
Date: Mon, 20 Mar 2017 14:21:38 GMT
Connection: keep-alive
X-CCC: ZA
X-CID: 2

HEAD /v11/2/windowsupdate/redir/v6-win7sp1-wuredir.cab?1703201421 HTTP/1.1
Connection: Keep-Alive
Accept: */*
User-Agent: Windows-Update-Agent
Host: ds.download.windowsupdate.com
```

Figure 5.7: Malicious MAKTUB Cabinet File

5.4.3 Universal Plug and Play Traffic

The Universal Plug and Play protocol also known as the UPnP protocol is specifically designed to allow for simplified discovery and control of network-enabled devices. These capabilities vary from devices however the control of incoming traffic on specific ports is mapped using routers. Applications which also use the UPnP protocol are BitTorrents and other Peer-to-Peer network software. This communication suite is made up of two distinct services the Simple Service Discovery Protocol (SSDP) which is a service which listens on the UDP port 1900. This service is responsible for the advertising of the available services and the service which responds to the discovery requests. The Second component of the UPnP forms part of the HTTP service. When the SSDP protocol has been used to identify the location if the UPnP device the HTTP service portion of this process assigns the service description file for a system, in addition to this the HTTP platform of the UPnP protocol hosts the Simple Object Access Protocol interface. This interface is responsible in providing a way for the other systems on a specific network to call the specified defined set of functions (Stark Win, 2016). The system which wishes to utilize an UPnP-enabled device, there must be a SSDP M-Search request sent to the local network. This means that a local system which runs this SSDP query can query the existing device cache and the response made would indicate the HTTP locations as well as the description of the device.

Effectively once the device has provided its details the request can be sent to obtain the service description as well as the defined allowed actions. The final process would be for the application to send a SOAP request for the specified URL for the target service. (Hoffmann, 2016). The implementation of the UPnP functionality is done trough a set of TCP/IP services. This protocol is very susceptible to compliance and vulnerability attacks. The first vulnerability which was identified is described as the SYSTEM remote exploit. This is done by sending malformed UPnP advertisements, which target a specific machine and this machine would eventually crash due to the access violations. The exploit which allow the attacker to execute code that needs high level of privilege on the target machine. The second vulnerability which was identified is a denial of service attack which is done by

sending forged/spoofed UPnP advertisements which triggers the host machine to connect to an HTTP URL which is specified within the packet itself. This can spiral the machine into an infinite loop denying network communication to be made by the machine. This depletes resources. The UPnP service also is a protocol which listens on broadcasts addresses, one carefully crafted packet can trigger the target machine to connect to a specified HTTP location.

The UPnP protocol is easily exploited by various malware strains, due to the protocol being plagued by basic security flaws. Hoffmann (2016) states that malware can infect a computer which forms part of a network. Network-based devices make use of the UPnP protocol, in the same way this protocol is used for legitimate by legitimate programs it can be exploited by a malicious program. This allows the malware to bypass the firewall entirely an example of this is that a Trojan horse can install software which allows for remote control on the computer it is infecting and as such creating a tunnel through your router's firewall which would allow for unlimited access to the computer. If UPnP was not enabled on the computer this would not be possible as the port would not be allowed to be opened. The reason according to Hoffmann (2016) that this is so easily exploited is due to the nature of how UPnP works.

The UPnP protocol assumes all local software, services which are run are trustworthy and would allow any software run locally to be able to forward ports and allow traffic in (Gates, 2014). Gates (2014) states that by using the SSDP protocol and with the vulnerabilities within the UPnP protocol malware operators and hackers can exploit your computer. The new kid on the block is known as the SSDP Reflective/Amplified DDOS attack. The SSDP protocol is a network-based protocol which allows the advertisement and discovery of network services. The mechanism is provided by this protocol which allows network clients to configure network services with little to no interaction needed by the user. SSDP gets the job done by providing multicast discovery support along with server-based notification and discovery routing. The SSDP attacks are categorised within the same space as DNS and NTP amplified DDOS attacks. These are done by attackers making use of smaller botnets that spoof the victims address. The attackers then proceed to query the routers or other network-enabled devices which are open to the internet using UPnP. These devices would proceed to respond with packets that are larger than which were requested. This is the amplified portion of the exploit; the amplified traffic does not affect the botnet instead the traffic is sent to the victim machine. This falls within the classic reflection/amplification attract which incorporates the slightly newer approach. Gates (2014) explains that for this specific attack the port 1900 is used with UDP traffic, this forms part of the suite which makes up the UPnP protocol. These attacks can be broken down into 6 phases:

1. The attacker firstly scans looking for any plug-and-play devices which can be utilised, and which are available.

2. The attacker then discovers the network devices; a list is created of the devices which respond to the initial packets.

3. The attacker crafts UDP packets which has the spoofed IP address of the targeted victim.

4. The attacker proceeds to use a botnet to send the spoofed discovery packet to each plug-and-play device request for as much data as possible by setting two flags `ssdp:rootdevice` or `ssdp:all`.

5. The result in the end is that each device which received the request will send a reply to the victim machine with an increased amount of data by 30 which are larger than the attacker requested.

6. The target machine receives large amounts of traffic from these devices and overwhelms the system, potentially denial-of-service for other legitimate services.

5.5 Forensic Artefacts

When conducting a **Digital Forensic** examination it is important to note, that each interaction that is made with the system will leave a residual trail. This allows for the piecing together of information from artefacts contained on the system to combine into a timeline. The chapter which follows details the forensic artefacts which were found on the system.

5.5.1 Memory Forensics

During the examination of the MAKTUB memory image, the URL which were found are detailed in Table 6.6. These have been extracted which was used for the examination of the memory image. Within memory there were calls made to various TOR related websites these are detailed in Table 6.6.

MAKTUB creates various mutex objects which have been listed in the list below. These mutex objects are used to ensure that the same host is not reinfected. This is done when a specimen attempts to create a specific handle but finds that the handle is already open thus showing that the host is already infected. Abhishek and Bhu (2017) propose that the component of creating mutex objects is for the malware to be able to synchronise the injections of multiple copies of itself into various processes. This is further supported by information detailed by Elisan (2015) which details that the use of mutex files by malware is to be able to track the infection spread.The mutex listing can be found in Section B.2.

MAKTUB employs hooking of dll in running processes. The process of hooking is detailed in Table 5.6**.** For this to be done the Ransomware needs to have administrative privileges on the host machine. This process is only followed if access and persistence has been gained on the victim's machine. This process is done to evade detection by hooking onto legitimate processes. As can be seen in the table below the MAKTUB sample uses the `winword.exe` to hook legitimate processes which it needs to effectively complete its execution. This is done by the following the following steps:

- Inject the dll into the address space of the process, this means that for the dll injection to work there needs to be a process running already such as the `winword.exe`.
- Modification need to be made to the Import Address Table to reflect the new address where the dll was injected to.
- Uses the proxy dll and manifest files to change the information.
- Loading the necessary drivers into the kernel address space.

There is however another option for dll hooking in malware and that is to inject a custom dll into the address space of a running process. This means that the injected dll leaves a smaller footprint on the system. This force the system to load the dll into the address space rather than the original one. The Registry location below contains a registry value AppInit[15] which contains the dll which are loaded in the library. The function `SetWindowsHookEx` is accessed to facilitate the hooking of dll which are contained in the registry key AppInit, this would prompt the `CreateRemoteThread` function to be invoked which would load the malicious code which was hooked onto in Table 5.6.

HKLM\SOFTWARE\Microsoft\Windows NT\Current Version\Windows

Table 5.6: Process hooking

Process Hooking		
Executable	Message	Written to
WINWORD.EXE	e9c53233f1	OLE32.DLL
WINWORD.EXE	3b00e894	MSPTLS.DLL
WINWORD.EXE	4ca8e7780bb8e77aa6e8f779fbb8e77 08bb8e7746ce8e7761388f77de2f8f7 7d0d98e77000000001779a2764f91a 2767f6fa276f4f7a27611f7a276f283a 276857ea27600000000	MSIMG32.DLL
WINWORD.EXE	5a342953	CSS7DATA0009.DLL
WINWORD.EXE	79061a7c	MSPROOF7.DLL
WINWORD.EXE	e92399baf0	OLEAUT32.DLL
WINWORD.EXE	e80e1153	MSCSS7EN.DLL
WINWORD.EXE	e99e4872ef	SetUnhandledExceptionFilter @KERNEL32.DLL
WINWORD.EXE	ad42b294	OART.DLL
WINWORD.EXE	"aa90f094	RICHED20.DLL
WINWORD.EXE	e99a54b7f0	SysFreeString @OLEAUT32.DLL
WINWORD.EXE	025f9794	MSO.DLL
WINWORD.EXE	e93655b8f0	VariantClear @OLEAUT32.DLL
WINWORD.EXE	1e6f6597	WINWORD.EXE
WINWORD.EXE	f3a5b394	WWLIB.DLL

5.5.2 Registry Analysis

MAKTUB is very active in reading, opening and changing registry keys within the victim's machine. The Windows operating system is made up of registry hives where settings and

[15]https://support.microsoft.com/en-us/help/197571/working-with-the-appinit-dlls-registry-value

controls are stored which indicate how the operating system should behave. There were in excess of 590 registry keys open, the majority of these keys were done to index or profile the host machine setup that it was attempting to infect. Some of the keys which were open and again the full 590 were not listed as it would be exhaustive and no more insight would be gained from it.

The registry key value which was changed belongs to the key responsible for storing information and values relating to software which was installed using the Windows Installer. The following key was changed:

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\UserData\S-1-5-18\\Products\00004109D30000000000000000F01FEC\Usage

The Windows Installer is responsible for the maintenance of the list of application source code in the database. MAKTUB to remove evidence from this single database; however the malware operators may not have been aware that this detail is also recorded in various registry hives. When an application is installed using the Windows Installer the application attempts to return to the original path it was installed from, when modifications need to be made or when repairs to the software need to be done. The Windows Installer is responsible for the following:

- Installation process and deployment of software
- Modification and changes
- Upgrades and patches
- Removal from the system
- Customisable installations and configurations
- Manage shared resources
- Enforce consistent file version rules
- Diagnose and repair applications at run-time

Monika *et al.* (2016) conducted research on Ransomware characteristics of malware and Ransomware and found that the key `HwOrder` is monitored by various malware strains. This registry value according to the Windows Dev Centre is that key stores values relating to devices which are installed as a network provider. This is also utilised when a dynamic link library file is installed as a network provider. The hypothesis is that MAKTUB uses this to monitor whether there is any changes made to network providers to ensure that it is not in a spoofed or intercepted network environment. The registry key which was monitored was:

HKLM\SYSTEM\ControlSet001\Control\NetworkProvider\HwOrder

MAKTUB enumerated the keys detailed in the list below. The first enumerated registry key is the registry key which is responsible for the setting or values relating to the Desktop settings of a specific user. This key is also responsible for the wallpaper details. The fact that these key are enumerated with replaced values is the process in which MAKTUB places the Ransomware note on the Desktop of the victim and uses this enumerated key to do it.

The second key which is enumerated by the MAKTUB is the SASLProfile. The SASL is also known as the simple authentication and security layer within Windows. This protocol is used by application such as SMTP for when emails are sent. enumerates this key to be enabled to send out Spam emails and communications to the address book found in the email configurations of the victim's machine. The third enumerated key which is created by MAKTUB is in relation to the CLSID this key is responsible for the identification of COM class objects by assigning them a globally unique identifier known as the CLSID. The key which was enumerated contains a globally unique identifier which upon research was found to be used by the `Boskouk.exe`, this is classified as a TROJ.FAKEAV.SM49. Tiamzon (2013) identified this as a fake anti-virus Trojan. This Trojan is dropped by other malware on the victim machine, it also has the ability to delete itself after execution, and this is also a process used by the MAKTUB Ransomware. Code reuse between malware operators is well known to occur within the industry, and it seems that the operators who designed the MAKTUB malware strain utilised code from other malware such as this Trojan to maximise its success.These registry key are as follows:

HKLMU\Control Panel\Desktop\LanguageConfiguration
HKLM\SYSTEM\ControlSet001\Control\SecurityProviders\SaslProfiles
HKLMSOFTWARE\Classes\CLSID\{FAE3D380–FEA4–4623–8C75–C6B61110B681}\Namespaces

MAKTUB accesses the Windows Registry key to identify what software policies have been put in place. This is done to identify whether there is policies in place which would hamper the executable from successfully infecting the victim machine. This process can be seen in Figure 5.8.

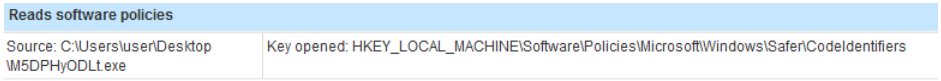


Figure 5.8: Software Policy Query

MAKTUB exhibits signs of accessing the cryptographic machine-GUID. This is done by accessing the following keys within the Windows registry of the host machine. This version acts differently than the previous versions as the extensions are no longer. MAKTUB but rather randomised extensions. This can be observed in the creation of the mutex creation in Section 5.5.1. The MAKTUB malware like other variants which were examined in other sections exploit the Native Cryptography API. Dell Secureworks (2015) states that malware uses the machine-GUID value to identify when it is in a Virtual Environment. This is done by calculating the machine-GUID of the machine upon first execution to identify the value.

Once the infection continues the value is calculated again. When this is done on a virtual environment the values will be different and the inference is made that the malware is run in a virtual environment. Even though there are other uses this one shows that MAKTUB is environmentally aware and fingerprints the system before the infection continues. The difference with this strain is that even though it detects the virtual machine it will still infect the machine; however, it will not communicate out on the network. As MAKTUB has the ability to fall back on the hard-coded keys it will still encrypt a system even when there is no network. This strain works equally well online and offline. It also does not discriminate against virtualised networks and will still encrypt it but not reveal the secrets of its command and control servers. This can be seen in Figure 5.9 where it uses the Windows registry to obtain the machine-GUID value, this is used to uniquely identify the victim's machine.

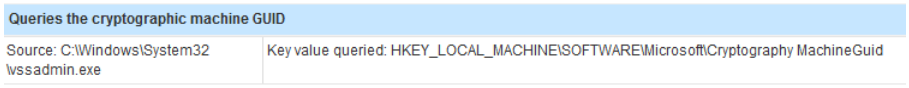


Figure 5.9: Query the Cryptographic Machine-GUID

5.5.3 Digital Forensic Artefacts

Yonts (2014) states that malware authors will rather avoid writing to areas which will raise suspicion. These are the locations which are outside the known-good locations. The researcher further states that malicious executable install themselves in writeable areas of the Windows file system it is infecting. This is generally done by installing under the user's home directory in either the AppData\Local, AppData\Roaming or Temp directories. The temp directory is the preferred writeable folder for malware installations. MAKTUB when executed will create temporary files in the %TEMP% directory, the first initial file is the malicious document contained in a `.CAB` file. The cabinet format is an efficient way to compress multiple files. MAKTUB creates a file *hecova.cab* which contains a copy of the `.RTF` document which was initially executed. This file is downloaded using a HTTP downloader which can be observed in Figure 5.10. The next step for the MAKTUB binary is to recreate itself in the %TEMP% directory. The process followed after this is to hook onto a legitimate process the processing used by the Ransomware is `winword.exe`. This process is done as a sanity check by the variant to ensure that the infection has not been altered in any way.

```
HEAD /v11/2/windowsupdate/redir/v6-win7sp1-wuredir.cab?1703201421 HTTP/1.1
Connection: Keep-Alive
Accept: */*
User-Agent: Windows-Update-Agent
Host: ds.download.windowsupdate.com

HTTP/1.1 200 OK
Cache-Control: public,max-age=172800
Content-Type: application/octet-stream
Last-Modified: Mon, 12 May 2014 21:56:34 GMT
Accept-Ranges: bytes
ETag: "115695a2d6ecf1:0"
Server: Microsoft-IIS/7.5
X-Powered-By: ASP.NET
Content-Length: 23603
Date: Mon, 20 Mar 2017 14:21:38 GMT
Connection: keep-alive
X-CCC: ZA
X-CID: 2

HEAD /v11/2/windowsupdate/redir/v6-win7sp1-wuredir.cab?1703201421 HTTP/1.1
Connection: Keep-Alive
Accept: */*
User-Agent: Windows-Update-Agent
Host: ds.download.windowsupdate.com
```

Figure 5.10: CAB file download

MAKTUB makes use of the `vssadmin.exe` which is responsible for managing the Volume Shadow Copy process on Windows. MAKTUB was found to create a process which identifies whether the Volume Shadow copy is enable, if this process is found the Ransomware will proceed to disable this process to ensure that the user cannot roll back or restore from a previous copy the two processes which were identified can be seen in Figure 5.11. MAKTUB will proceed to identify previous copies of the Volume Shadow and delete these. The research which has been done previously on MAKTUB did not identify this process as the focus was more placed on decompiling the code. MAKTUB makes use of the command line utility of Windows to perform the functions of disabling and deleting the Volume Shadow Copies. The Ransomware uses the console to preform a search to identify whether there was Volume Shadow Copy enable and the value returned was *"No items found that satisfy the query"* this was done using Power Shell, there was no screen which indicated that this was taking place. The user would not be aware of this process as it is done in the background

Source: unknown	Process created: C:\Windows\System32\vssadmin.exe vssadmin.exe delete shadows /all /quiet
Source: vssadmin.exe, 00000003.00000002.1864118576.00468000.00000004.sdmp	Binary or memory string: 0oFJoFXoFhoFroFvssadmin.exedeleteshadows/all/quiet
Source: vssadmin.exe, 00000003.00000002.1863807437.000B0000.00000004.sdmp	Binary or memory string: vssadmin.exe delete shadows /all /quiet
Source: vssadmin.exe, 00000003.00000002.1863807437.000B0000.00000004.sdmp	Binary or memory string: C:\Users\user\Desktop\C:\Windows\system32;;C:\Windows\system32;C:\Windows\system;C:\Windows;.;C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\C:\Windows\system32\vssadmin.exevssadmin.exe delete shadows /all /quietC:\Windows\system32\vssadmin.exeWinSta0\Default
Source: vssadmin.exe, 00000003.00000002.1863745034.00060000.00000008.sdmp	Binary or memory string: Example Usage: vssadmin Delete ShadowStorage
Source: vssadmin.exe, 00000003.00000002.1863745034.00060000.00000008.sdmp	Binary or memory string: Example Usage: vssadmin Delete Shadows /Type=ClientAccessible /For=C:
Source: vssadmin.exe, 00000003.00000002.1863745034.00060000.00000008.sdmp	Binary or memory string: vssadmin Delete Shadows
Source: vssadmin.exe, 00000003.00000002.1863745034.00060000.00000008.sdmp	Binary or memory string: Example Usage: vssadmin Delete Shadows /For=C: /Oldest
Source: vssadmin.exe, 00000003.00000002.1863745034.00060000.00000008.sdmp	Binary or memory string: Example Usage: vssadmin Delete ShadowStorage /For=C: /On=D:

Figure 5.11: Volume Shadow `vssadmin.exe`

Source: unknown	Process created: C:\Windows\System32\vssadmin.exe vssadmin.exe delete shadows /all /quiet
Source: vssadmin.exe, 00000003.00000002.1864118576.00466000.00000004.sdmp	Binary or memory string: 0oFJoFXoFhoFroFvssadmin.exedeleteshadows/all/quiet
Source: vssadmin.exe, 00000003.00000002.1863807437.000B0000.00000004.sdmp	Binary or memory string: vssadmin.exe delete shadows /all /quiet
Source: vssadmin.exe, 00000003.00000002.1863807437.000B0000.00000004.sdmp	Binary or memory string: C:\Users\user\Desktop\C:\Windows\system32;;C:\Windows\system32;C:\Windows\system;C:\Windows;;C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\W indows\System32\WindowsPowerShell\v1.0\C:\Windows\system32\vssadmin.exevssadmin.exe delete shadows /all /quietC:\Windows\system32\vssadmin.exeWinSta0\Default
Source: vssadmin.exe, 00000003.00000002.1863745034.00060000.00000008.sdmp	Binary or memory string: Example Usage: vssadmin Delete ShadowStorage
Source: vssadmin.exe, 00000003.00000002.1863745034.00060000.00000008.sdmp	Binary or memory string: Example Usage: vssadmin Delete Shadows /Type=ClientAccessible /For=C:
Source: vssadmin.exe, 00000003.00000002.1863745034.00060000.00000008.sdmp	Binary or memory string: vssadmin Delete Shadows
Source: vssadmin.exe, 00000003.00000002.1863745034.00060000.00000008.sdmp	Binary or memory string: Example Usage: vssadmin Delete Shadows /For=C: /Oldest
Source: vssadmin.exe, 00000003.00000002.1863745034.00060000.00000008.sdmp	Binary or memory string: Example Usage: vssadmin Delete ShadowStorage /For=C: /On=D:

Figure 5.12: `vssadmin.exe` console search

MAKTUB makes use of the `BCEDIT.exe`. This executable is responsible for maintaining and controlling the boot settings for the Windows Operating system. This controls how the Operating System boots up. This process is done by the malware to ensure that the user is not able to boot into recovery mode to effectively clean the system of the infection. This takes away from the ability to boot into another environment. An interesting observation for further examination is to determine whether this process would be successful in a machine which has an UEFI bios environment. The use of the `BCEDIT.exe` was found in during the examination of the memory of the infected machine. This points to the ability of MAKTUB to unpack resources within memory, this is done to evade detection and in relation to the obfuscation techniques used by this malware strain. The Elder Geek (2017) explains that the System Volume drive forms an important role in the restore point of the Windows operating system. The restore point for the Windows operating system is stored in the System Volume folder. MAKTUB creates a snapshot of it and embeds itself in this folder. The infection of the restore point would mean that the victim cannot restore to a previous version and should they attempt they would be restoring an infected snapshot. This is surely unique to MAKTUB and has not been observed in the other variants. This malware does not only render volume shadow copies deleted it ensures that it creates and infected restoration point. This can be seen as a sophisticated persistence mechanism or fail safe. This can be observed in the Figure 5.13.

Stores files to the Windows start menu directory	
Source: C:\Users\user\Desktop \M5DPHyODLt.exe	File created: C:\Users\All Users\Start Menu\Programs\Java\Visit Java.com.url.vjpuyr
Source: C:\Users\user\Desktop \M5DPHyODLt.exe	File created: C:\Users\All Users\Start Menu\Programs\Java\Get Help.url.vjpuyr
Source: C:\Users\user\Desktop \M5DPHyODLt.exe	File created: C:\Users\All Users\Start Menu\Programs\Java_DECRYPT_INFO_vjpuyr.html

Figure 5.13: System Volume Drive

MAKTUB uses various native Windows function to create a new security identifier and obtain the required level privileges. The Ransomware obtains the `Kernel32.dll` process which is a system level process that has administrator level privileges. The functions used by MAKTUB is detailed in Table 5.7. It then proceeds to open the process token assigned to this process. Once the information is obtained it reassigns this to a new security identifier and checks once the process is complete whether this user is indeed part of the Administrator group. Windows makes use of security descriptors which aids in the control access of these objects. The security descriptors include the information about which process owns

an object and what level of access this object has. These contain the access control list of the various objects within the Windows operating system. A Windows object can contain two types of access control lists [16]:

- **Discretionary Access Control List:** discretionary access control list which is responsible for identifying the users and the groups allowed or denied access.
- **System Access Control list:** System access control list which controls how the access is audited.

Table 5.7: Functions used by MAKTUB to escalate its privileges

Process	Description
GetCurrentProcess	Function used to create a new process instance and associate it with a process resource on the local machine
OpenProcessToken	Function opens the access token associated with a process
GetTokenInformation	Retrieves the token information for Group memberships and access tokens
CloseHandle	Function is used to close a handle associated with an object.
AllocateAndInitializeSid	Function allocated and initializes a new security identifier with up to eight sub authorities
CheckTokenMembership	Function determines whether the specified security identifier is enabling with an access token.
FreeSid	Function is used to free a security identifier which has previously been allocated and initialized
IsUserAnAdmin	Functions tests whether a user is a current member of the Administrator workgroup.

5.6 MITRE ATT&CK Map

The MAKTUB sample was examined using the online sandbox environment Hybrid-Analysis[17]. This online sandbox maps the behaviour of the samples which were submitted to the MITRE ATT&CK adversarial technique map. The results can be seen in Figure 5.14. This indicated that the initial point of access for MAKTUB is access by use spam emails. MAKTUB makes use of what is called an execution through a Module Load, this means that the Ransomware makes use of a Windows module loader which is instructed to load a specific DLL this is a functionality which resides within `ntdll.dll` using the native Windows API which makes use of the functions like `CreateProcess()` and `LoadLibrary()`. MAKTUB maintains persistence on the victim machine is done by installing itself in services and executables which run at start up. The execution indicators of compromise is that the Services within Windows is exploited for custom code to be executed which is consistent with the physical dynamic analysis which was done on MAKTUB.

The method according to this framework indicates that MAKTUB maintains persistence by including itself in the start-up items, further more it modifies existing services within Windows to exploit the environment, this is also done by creating new services which run the malicious scripts. MAKTUB makes use of process injection to execute code within an address space which has is assigned to system level processes. This also allows for evasion techniques to be used as the process looks legitimate until it is executed within memory.

[16] https://docs.microsoft.com/en-us/windows/desktop/secauthz/access-control-lists
[17] https://www.hybrid-analysis.com/

MAKTUB evades detection by using a software packer which is a method of compressing and encrypting the information within and executable file. This alters the file signature and evades detection by anti virus. Often packers used are `MPRESS` and `UPX`. Malware operators can also employ and design custom packers for their malware. The Ransomware also makes use of injecting the code within a legitimate process to evade signature code based detection. This is done by masking the execution under a legitimate process. The malware operators who designed MAKTUB also attempted to obfuscate the contents of the executable by using a packer. This means the payloads are not viewable or examinable unless it is dynamically unpacked [18].

The discovery techniques used by MAKTUB is that it attempts to identify the security software which is installed on the system. This allows the Ransomware to employ techniques which circumvent these security rules and software. MAKTUB makes use of enumerating the information of the system by identifying the operating system installed, the hardware installed and versioning information of the system. This allows for the malware to execute specific exploits which the system may be vulnerable too.Command and control tactics which are used by MAKTUB is for it to make use of the native Windows cryptographic protocol to mask communications. This is generally a weak approach to masking communication as this can be reversed engineered.

Mitre Att&ck Matrix

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command and Control
Valid Accounts	Execution through Module Load 1	Startup Items 1	Startup Items 1	Software Packing 1	Credential Dumping	Security Software Discovery 1 2 1	Application Deployment Software	Data from Local System	Data Compressed	Standard Cryptographic Protocol 1
Replication Through Removable Media	Service Execution	Port Monitors	Process Injection 1	Process Injection 1	Network Sniffing	System Information Discovery 2 2	Remote Services	Data from Removable Media	Exfiltration Over Other Network Medium	Fallback Channels
Drive-by Compromise	Windows Management Instrumentation	Accessibility Features	Path Interception	Obfuscated Files or Information 3	Input Capture	Query Registry	Windows Remote Management	Data from Network Shared Drive	Automated Exfiltration	Custom Cryptographic Protocol

Figure 5.14: MAKTUB Behavioural Matrix
Source: `https://www.hybrid-analysis.com/`

5.7 Summary

MAKTUB is unique in terms of the fact that between the tree families examined, it is the only one that makes use of a packer. The static binary code is encrypted and only unpacked during the execution. The MAKTUB Ransomware is unique in the fact that it does not need to be connected to the internet to encrypt the information on the victim machine. In this instance killing the internet connection it will have no effect on the encryption process. This strain makes use of the same instruction reordering obfuscation technique as was explained in the section on CERBER. This Ransomware exploits the Universal plug and play protocol to facilitate exploitation on the victim machine. MAKTUB has the ability to

[18] https://attack.mitre.org/techniques/T1027/

fingerprint the victim machine in that it is environmentally aware. The MITRE ATT&CK framework identifies that MAKTUB has the ability to execute itself using the native Windows loader, and maintain the persistence on the system by ensuring that it is executed at start up. The strain also escalates it privileges by injecting its custom code into processes which run an system level. The Ransomware employs various evasion techniques such as packed code and obfuscation techniques. The Ransomware enumerates the environment it is infecting and uses Windows against itself.

6

Case Study: LOCKY

6.1 Introduction

The LOCKY sample used in this analysis was obtained from Carbon Black. The LOCKY Ransomware has seen an exponential growth since 2016 which equates to a 752% increase in attacks. It has been estimated that LOCKY raked in approximately $1 Billion since 2016 (Hasherezade, 2016b). LOCKY was first seen in the wild in February 2016 and soon became a dangerous threat to organisation and individuals. LOCKY is categorised as crypto-Ransomware which means that the main aim of this malicious code to infect and encrypt the machine holding the data ransom. There also appears to be many actors who utilises this and it can be said to form part of Ransomware-as-a-service even though there has been no active marketing done by the LOCKY creators. The LOCKY Ransomware is spread by two methods which is either email or by an exploit kit which has been masked as legitimate software. The NECURS BOTNET has been found to be the main culprit behind the distribution of LOCKY as email spam. The exploit kits which have been used in the past has been the NEUTRINO EXPLOIT KIT, RIG and Nuclear Exploit kits. The LOCKY malware has been going strong since the first sighting and has shown now signs of slowing down. This strain has been chosen as it is adaptable but unlike the others have remained strong and relevant (Griffin, 2016a).

This Chapter presents the finding in relation to the framework which has been set out in Chapter 3. Section 6.2 reports the findings relating to the static analysis. The dynamic analysis which is the execution of the malware sample and is set out in Section 6.3, Section 6.4contains all the communication which was identified on the network that has been sent and received. The section which follows onto this one is Section 6.5 which contain all the

artefacts which were identified using Digital Forensic processes. Section 6.6 details the examination of the MITRE ATT&CK framework. This chapter ends with a summary of the results detailed in Section 6.7.

6.2 Static Analysis

This section forms part of the Hybrid-malware analysis framework which is introduced in Chapter 3. This process entails the analysis of the static malware file to determine what intelligence can be ascertained without dynamically running the malware. The focus is to extract as much information from the malware in its non run state "dead" state.

6.2.1 PE Structure

LOCKY according to the PE file header information imports some libraries/functions from the link library file. The following functions allows for the creation of new directories, interrogate the registry values, open keys, make changes to keys, enumerates new keys, obtain information on modules open within the memory, identify whether debuggers are run in the Windows environment. LOCKY malware also has the ability to terminate services, create shells and many more as can be seen in Appendix D the malware functionality within the PE file structure. This shows that the infection manages to gain full control of the system with maximum access.

LOCKY has the ability to register as ability to register a top-level exception handler. This means that the LOCKY which was examined is able to supersede the top-level handling thread within a process. This also enables LOCKY to run in Kernel mode with escalated privileges instead of that of User-Mode. This information was found within the PE file examination, the second figure was found when reversing the executable LOCKY strain using *PE Explorer*. The idea behind the use of exception usage as an anti-debugging is to make the static analysis of the malware sample harder than those who do not make use of it. The method used by the LOCKY malware sample, is to set the exception handler to execute at top-level using the `Kernel32.dll`. This can be seen in Figure 6.1 .

The `Kernel32.dll` forms part of the default exception handlers within Windows. When an executable runs on Windows, and there is no exception handler specified, this process is taken over by the `Kernel32.dll`. The top-level exception filters are used because that has top-level administrative rights on the operating system. This behaviour is done to detect the presence of a debugger on the system. Once this has been found an exception is passed and an application error is forced on the system. If the process is not being debugged it will continue on. This is a technique used often used as an anti-debugging process. The basic idea behind the exception anti-debugging techniques are to use side effects of a debugging tool (or decisions made by the person using the tool) to alter the execution flow. There are several ways to do that, the classics being setting an exception handlers (either for the local thread or at the process level) and implementing important parts of the logic in the

exception handler before returning execution to the original EIP or changing it completely (Branco *et al.*, 2012).

Figure 6.1: Privilege Escalation

6.2.2 Version Information

LOCKY which was examined using the methodology and tool set which was detailed in the same chapter. The aim of this portion of the Hybrid-Malware analysis framework is to collect information regarding the file itself and what the attack surface was for distribution. The malware sample was examined using *PEStudio* and it was the following was ascertained. The file description for this software upon further research is for a tool which claims to be able to a task scheduler. This shows that this specific LOCKY sample masquerades as legitimate software in drive-by downloads. The file name according to the the PE header information was #Advanced Task Scheduler 32-bit Edition. This uses the art of subterfuge to deceive the victim's into thinking it is legitimate software.

Table 6.1: File Version

Version Number	4.1.0.612
Internal File Name	#dvanced Task Scheduler 32-bit Edition
Company Name	Doubtsoftware.com
Legal Trademark	Copyright Southsoftware.com, 2002-2015
File Description	Advanced Task Scheduler 32-bit Edition
Purpose of Software	Task Scheduler
Attack Vector	Infected Software

6.2.3 Windows API Class

LOCKY was examined and according to the PE file header information imports some libraries/functions from the Microsoft Window32 Link Library. The following functions allow the malware to interrogate the registry values, open keys, make changes to keys, enumerate new keys, obtain information on modules open within the memory, identify whether debuggers are run on the Microsoft Windows environments, terminate processes, create shells and many more as can be seen in the below list the malware functionality within

the PE32 structure would indicate that the infection allows for maximum access to the target machine, this enables the malware to have the keys to the kingdom effectively (Arntz, 2017).

LOCKY accesses the `vssadmin.exe` which is responsible for managing the Volume Shadow Copy process on Windows. The sample proceeds to identify whether the Volume Shadow process is enabled and where the copies of these volume shadow copies are saved. Once this process is completed the Ransomware will exploit the `vssadmin.exe` and delete these copies to ensure that the user cannot revert to a previous copy in Figure 6.2.

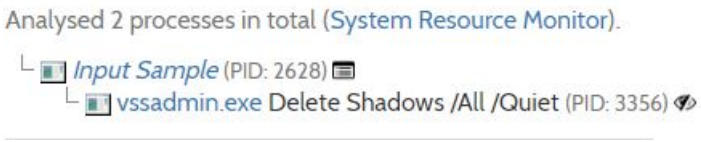


Figure 6.2: `vssadmin.exe` exploitation

LOCKY makes use of the Microsoft Windows Native API's and lives of the land. Each of these API's will be briefly discussed but will not be covered in depth as this falls outside the scope of the research. The Windows Native API's which were exploited by LOCKY is:

- `Kernel32.dll`[1] handles memory management, input/output operations, and interrupts this link library file is loaded into the protected memory sections.
- `COMCTL.dll`[2] implements a wide variety of standard Windows controls, such as File Open, Save, and Save As dialogues, progress bars, and list views.
- `GDI32.dll`[3] :functions that perform primitive drawing functions for output to video displays and printers.
- `Shell32.dll`[4] is a dynamic link library that controls certain API functions of the Windows Shell.
- `IPHLAPI.dll`[5] this module contains the functions used by the Windows IP Helper API.
- `MSIMG32.dll`[6] this is an extension component to the GDI32 API.
- `OLEAUT3.dll`[7] this library contains the core OLE functions used by the Windows Operating System.
- `PSAPI.dll`[8] this file is responsible for support to processes running and their statuses.

[1]https://www.webopedia.com/TERM/K/Kernel32_dll.html
[2]https://en.wikipedia.org/wiki/Microsoft_Windows_library_files
[3]https://en.wikipedia.org/wiki/Microsoft_Windows_library_files
[4]https://msdn.microsoft.com/en-us/library/windows/desktop/bb776779(v=vs.85).aspx
[5]http://www.processlibrary.com/en/directory/files/iphlpAPI/24220/
[6]http://www.processlibrary.com/en/directory/files/msimg32/21809/
[7]http://www.processlibrary.com/en/directory/files/oleaut32/23291/
[8]http://www.processlibrary.com/en/directory/files/psAPI/19106/

- `SHLWAPI.dll`[9] this contains the functions for the UNC and URL paths, registry entries and color settings.

6.2.4 Entropy

Entropy analysis is an important factor in malware analysis (Osaghae, 2015). This can give the researcher an indication on where to start the analysis. Signs of low entropy can indicate that the malware sample is not compressed nor have obfuscation sections. In the case where the entropy level is high within specific sections can indicate that the sample is packed or that obfuscation methods have been used. To understand the meaning of entropy one needs to understand the definition thereof. The simple definition for entropy is the randomness in which bytes display within a section within the PE file. The green sections of the visualisations is the high entropy sections and the varied black sections are less entropy. The binary representation in Figure 6.3 the areas of entropy is pixelated and the other sections are shaped. This means the examination of a file to determine the measurement of entropy is simply put the measurement of disorder of the bytes within a file (Osaghae, 2015). An interesting area of malware file analysis is the measurement of file entropy.

LOCKY malware strain was examined using the tool *Detect-It-Easy*. The tool displayed high levels of entropy in the `.data` section; this section had an unusually high unusually high levels of entropy compared to the other sections. The .data section of the LOCKY binary which was examined contained the following information in this section consists of initialized data which reads and writes to the memory. This means that there is a high level of binary randomness to protect itself this can be seen in Figure 6.4.

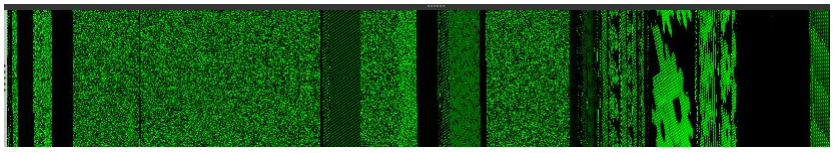


Figure 6.3: Entropy visualisation using Veles

[9] http://www.processlibrary.com/en/directory/files/shlwAPI/20075/

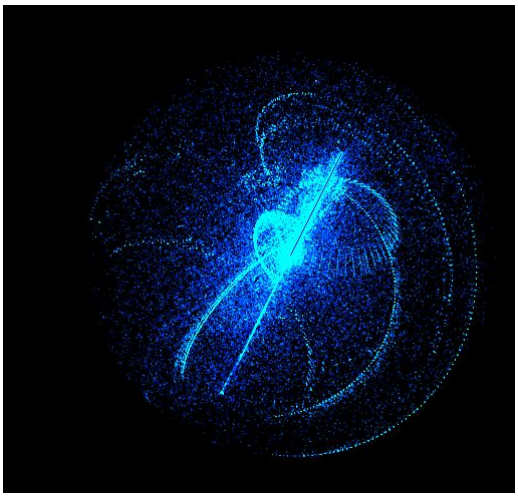


Figure 6.4: Binary visualisation using Veles

6.2.5 Hash Fingerprinting

The hashing of malware has been a practice used by malware analysts to uniquely identify these strains of malware. The sample which was examined was run through varies methods of hashing to determine the unique has fingerprint for this particular LOCKY strain. This allows for signature based detection of the fingerprinted strain. The traditional method of hashing a piece of malware for signature based detection is MD5 and SHA-256, this is however cryptographically a black and white approach, because if one bit of data has morphed or changed the signature will fundamentally change. The decision was made to add into the existing fingerprinting techniques SSDeep hashing which is referred to as fuzzy hashing. This identifies similarities within a document that are similar but not the same it is also referred to a homologous files.

6.2.6 Strings

Zeltser (2017) states that a closer look should be taken at any malware file to ascertain what static properties can be extracted from it. This includes but not limited to embedded string. By looking at the static properties the researcher can be able to ascertain suspicious activities which will take place as the binary is executed. This forms part of the triage efforts. Rocha (2015) states that embedded strings are strings which are in plain-text or Unicode which contain functions or information which has not been obfuscated. The strings are saved in ASCII and Unicode in a printable sequence which is embedded in the file. Strings which can be extracted generally contain information such as references to filenames, domain names, IP address, attack commands and much more. This does not often give a clear indication but it can hint towards what the malware is possibly capable of. When the compiling of the executable binary is done the symbol files and debug information is stored within a database. This forms part of the information produced by the compiler and linker.

6.2.7 Code Obfuscation Techniques

The LOCKY sample used three functions together which is the `call`, `ret` and `jmp` functions. Dalei and Li (2016) explains that malware authors obfuscate their code with the intention to make it complicated to gain information from the code itself. Malware operators rearrange their code to make the detection process harder. The most popular method used by malware authors according to Kumar (2004) is that they obfuscate the call addr instruction in Table 6.2. These LOCKY code obfuscation technique can be seen in Table 6.2. This technique used is part of the sixth code obfuscation techniques, described by Iliev (2017) and LOCKY makes use of the Instruction reordering.

Table 6.2: `Call,Push,Ret` LOCKY

Functions	Description
Call Address	This function will make a push to the updated program counter. This is the pointer to the instructions after the call which should be executed.
Ret	This specific function internally pops and addresses off the stack and then jumps to that address. This matched with call would allow for the return to the instruction after the prior call.
Jmp	Simply put this function is responsible to jump to a specific address .

6.2.8 Summary

The examination of the LOCKY strain identified that this family is adaptable and that their malware is not specifically aimed at one location or country. This version of the LOCKY malware was masked as a piece of legitimate software, which contain the malicious code.The sample further more target the Windows API responsible for booting the machine into a safe environment, it also proceeds to access and delete the API's responsible for the Volume Shadow Copies ensuring that the user cannot revert to a previous version. LOCKY has the ability obfuscate its code due to using. Instruction reordering.

6.3 Dynamic Analysis

The following section details the results from the dynamic execution phase of the Hybridised-Malware-Analysis portion. The system was monitored by using the tool-set from Chapter 3 following the detail processes.

6.3.1 Sand boxed Run-time Report

The purpose of running the sample through the sandboxed environment is to identify evasion techniques, and to confirm findings made on the physical machine examination. The

LOCKY sample was firstly run trough VirusTotal[10] which checks various antivirus software to identify whether the file is flagged as malicious. The sample was found to be identified by 86% of all anti-virus software. This means that there is still 14% of antivirus tools which do no flag this sample of LOCKY as malware. The sample was run for a second time through an online sandbox SecondWrite[11] the use of SecondWrite was to use another sandbox to fill any shortages of the other sandboxes. This however presented the same results as the use of VirusTotal.

LOCKY has a sleep period which exceeds three min before the malicious portion of the code will be executed and unpacked. Abhishek and Bhu (2017) explains, that with extended calls the malware refrains from being detected as malicious by waiting out the period used my sandboxed environments for analysis. This is the same approach used by CERBER detailed in Section 4.3.1and MAKTUB detailed in Section 5.3.1. The sandbox examination indicated that LOCKY has the ability to identify that is is within a sandbox environment and delay the execution of the malicious portion of code, it also has the ability to be environmentally aware.

02917AB6	8B3504B09202	mov esi, dword ptr [0292B004h]	0x00002392
02917ABC	57	push edi	
02917ABD	8D45D8	lea eax, dword ptr [ebp-28h]	
02917AC0	33FF	xor edi, edi	
02917AC2	50	push eax	
02917AC3	897DE8	mov dword ptr [ebp-18h], edi	
02917AC6	FF159C209202	call dword ptr [0292209Ch]	GetSystemTime@KERNEL32.DLL (Import, Unknown Params)

Figure 6.5: Function to get local time

There were five samples collected for LOCKY, however for the focus of the research only one sample was run for the purpose of the thesis. The LOCKY malware binary which was chosen, is named, the sample is named using the SHA 256 hash value. The sample was run through the online *Cuckoo Online Sandbox*[12] to be in a position to generate the *YARA* rules results. The results which were identified by the online *Cuckoo Online Sandbox* was that the based on the signatures contained within *Cuckoo Online Sandbox*; this allows the researcher to anticipate the behaviour of the malware. This is however only a summary of the overall examination. LOCKY has the ability to do the following:

- Anti-Debugging
- Packer protected sections
- Ability to create Windows Services
- Can set Debugger Exceptions

[10] `https://bit.ly/2TlXHHu`
[11] https://www.secondwrite.com/
[12] http://sandbox.pikker.ee/

- Ability to escalated privileges
- Can manipulate and mimic the Windows 32 HTTP API Calls
- Ability to manipulate the private profiles within Windows
- Creates mutexes within Windows
- Ability to manipulate and read the Windows Registry
- Manipulates the Windows System token process

6.3.2 ProcMon

Babic *et al.* (2011) state that the Windows operating system includes the easy to use API which supply the cryptographic services to the operating system. The Cryptographic primitives which are embedded in the dynamic link library within Windows which are divided into Cryptographic Service Providers. LOCKY was found to make use of the Microsoft Cryptographic service, to Import the cryptographic key and destroy them once the process is done this can be seen in Figure 2.17. LOCKY makes use this to be able to encrypt and move the files within one function. This makes the Cryptographic Service providers one of the Windows API's which is vulnerable to be exploited by malware especially Ransomware. This is sophisticated in that there is one section that does multiple functions in one.

Northcutt (2017) describes an enumeration attack, as an attack in which the malware operator is able to make the target machine it is infecting enumerate their file system. This enables the malware to identify and fingerprint all the files contained on an operating system. This process also allows the malware to identify the file system structure and in this instance the MAKTUB Ransomware has the ability to map the file system to ensure that no user files are missed during the encryption process. This process can also be used to identify virtualized environments and sandboxes as these have a specific structure to those of a physical machine.

LOCKY as can be seen in Figure 6.6 a call function is executed to invoke the `FindFirstFileW`[13] using the `Kernel32.dll`, according to the Windows Dev Centre this function searches for a specific file within a directory. The location of the file is then returned to the LOCKY executable. This is done to enumerate the files within a directory a map to what can be found within a specific directory. This shows that LOCKY has the ability to create a register of the files on the victim machine it is infecting.

[13] `https://msdn.microsoft.com/`

02918A74	7202	jc 02918A78h	target: 02918A78
02918A76	8B00	mov eax, dword ptr [eax]	
02918A78	8D8D14FDFFFF	lea ecx, dword ptr [ebp-000002ECh]	xref: 02918A74
02918A7E	51	push ecx	
02918A7F	50	push eax	
02918A80	FF15EC209202	call dword ptr [029220ECh]	FindFirstFileW@KERNEL32.DLL (Import, Unknown Params)

Figure 6.6: Enumeration Attack

6.4 Network Analysis

The examination focuses on communication during the encryption and post encryption for a period of 24 hours, from initial execution of the malware.

6.4.1 Protocol Breakdown

There were 10,101 packets in total and the breakdown of the different protocols were explained in Table 6.3. The LOCKY sample communicates mostly using TCP traffic as this noted 74.60% of the traffic captured on the network,the remainder of the communication was not related to the malware as determined by the DNS associated with the packet. These packets were Windows related.

Table 6.3: LOCKY Network Protocol

Protocol	Total Packets	Total Percentage per count
TCP	2,232	76.10%

6.4.2 Malicious Traffic

LOCKY attempt to contact the IP address 209.236.112.102 on port 80/tcp which is the HTTP port. The GET command is used to retrieve a file which does not exist on various hosts. The files which the LOCKY malware attempted to download are named *7gyjgg5r6* and *9YG65*. These two files have been associated with malware which downloads the LOCKY payload which is used to encrypt your system this was determine by previous work done my the website Malware Traffic[14]. In this instance the LOCKY payload could not be downloaded and the system was not encrypted. However even though there was no encryption there was system changes made along with network communication.

LOCKY makes use of the HTTP protocol using both GET and POST commands. This Ransomware strain is unable to encrypt the victim machine, without the payload which is downloaded from various domains. In this research the LOCKY strain was unable to download the payload and all the HTTP/GET commands returned a value of *'HTTP:/1.1 404 NOT FOUND'* and would indicate that these domains have been sanitised. The return of

[14]https://www.malware-traffic-analysis.net/2017/06/22/index.html

'*HTTP:/1.1 404 NOT FOUND*' is normal to see on a network and can often be legitimate communication. The LOCKY sample which was examined made three requests to the domains as listed in Table 6.4 these domains were associated with the LOCKY malspam campaign and with payload *7gyjgg5r6*. There was three requests made to domains as listed in the Table 6.4 these are domains associated with the DRIDREX STYLE MALWARE CAMPAIGN and the payload *9YG65* (Griffin, 2016a).

HTTP POST to Webserver

Locky makes use of a specific crafted HTTP Post packet which is used as a beacon to communicate with the command and control server. The packet is crafted to communicate out to specific web server. The data which it sends out is encoded and non readable. To decrypt the data was not part of the scope of the research and was not explored further. There were four unique IP addresses contacted which were: 109.234.35.128 was identified to form part of the C&C servers that is used by Locky Ransomware. The malware analysis was done by Check Point Threat Intelligence and Research (2016), whereby they identified that the strain which they were examined was sending HTTP POST packets to this IP address. None of the other matched with our strain. As we are aware Locky makes use of DGA algorithms which make the identification of malicious traffic much harder if not impossible. The IP address was checked making use of Threat Miner[15]. The IP address belongs to a hosting company `mchost.ru` which is situated in Russia.

This ISP has been has previously been associated with LOCKY campaigns and with hosting malicious software as detailed by Vlad (2016), this is common occurrence and does not make he hosting company malicious most of these servers are hijacked and used without the knowledge of the owner. The IP 5.135.76.18 was identified to form part of the EK MALWARE CAMPAIGN run by LOCKY Ransomware in 2016. This IP address has been associated with sending HTTP POST packets to this service. The IP address belongs to the hosting company OVH SAS which is situated in France, and is a huge hosting company. According to the Threat Miner report on this IP address there has been reports made of malicious LOCKY traffic to this IP address it is also on the Threat Crowd IPSet Blocklist for forming part of LOCKY malware campaigns.

The IP address 83.217.25.239 was identified to form part of the LOCKY Q RANSOMWARE campaign this is still an ongoing campaign. This IP address has been associated with sending HTTP POST packets of service. The IP address was checked using Threat Miner. This IP address belongs to a telecommunications company LTD Iptelecom[16] which is situated in Russia.

84.19.170.249 was identified to form part of a previous LOCKY campaign. This IP address has been associated with C&C traffic relating to Ransomware specifically LOCKY. The IP address was checked using Threat Miner. The IP address belongs to a hosting company

[15]https://www.threatminer.org/
[16]http://as39931.net/

Keyweb AG which is located in Germany. According to Threat Miner report there has been reports made of malicious emails being sent from this IP address.

LOCKY invokes the functions in Table 6.4, to be in a position to manipulate the network packets. This is done by means of exploiting the HTTP packets. Sarokaari and Dominicus (2012) explains that there are methods and functions which allow for a web server to provide and process requests. The GET command is one of the most popularly used commands. This command will send the parameters using the URL query string. The POST method command is used to perform actions and this allows the data to be sent also in the body of the message. These are very attractive for attackers in that it allows them to invoke function to inject malicious content into the content.

There are two methods for mapping, there is passing mapping is used to sniff the network and is very difficult to detect. The active mapping of a network is more aggressive and generates traffic and easy detectable. There are also other methods used to manipulate and exploit the HTTP packet. The mapping phase of an attack like this consists of different components, the first step which is taken in this form of attack is to scan for ports which are available, the second step is to fingerprint the operating system the final stage of mapping is spidering. The term spidering is used to describe malware which visits specific web pages and logs to gather information which is used for further spam campaigns (Sarokaari and Dominicus, 2012).

6.4.3 HTTP Discovery

In this phase the LOCKY malware attempt to enumerate the user-agent which is currently used on the victim system. A user agent is defined as the client requesting a webpage, which is sending information about itself in header-information named the “user-agent”. LOCKY as can be seen in Table 36 uses Windows functions “`ObtainUserAgentString`” , this functionality within Windows returns the value of the user-agent which is currently being used in HTTP-headers. The LOCKY malware exploits the Windows Internet (WinINet) application programme interface. This interface according the Windows Development Centre which enables the application to interact with the FTP and HTTP protocols to access Internet resources. This is done that LOCKY has the ability to enumerate HTTP packets which is crafted to contain malicious code (Sarokaari and Dominicus, 2012).

Table 6.4: LOCKY Network Communication Functions

FUNCTION	DESCRIPTION
ObtainUserAgentString	Retrieves the User-Agent HTTP header string that is currently being used.
InternetOpenA	This function initializes the application used by the WinINet.
GetLastError	This function is executed by the calling thread set and sets the value to SetLastError.
InternetCloseHandle	Closes the single Internet handle.
InternetSetOptionA	This function sets an Internet option.
HttpOpenRequestA	This function is used to create a HTTP request handle.
HttpSendRequestA	This function sends the specified request to the HTTP server allowing for callers to send extra data beyond what is normally sent.
InternetCloseHandle	Closes the single internet handle.

Backdoor Downloads and Check-Ins

The ZEUS-TROJAN which was responsible for steals passwords made use of the same user-agent string "`.Net CLR 2.0.20727`". According to Bollinger *et al.* (2015) in the initial days if the ZEUS BOT DOWNLOADER was hosted from an exploit which hosted on a compromised website. Once the system has been compromised the host would make an attempt via HTTP Post which checks-in on the script hosted by the attackers. There is however some characteristics on whether traffic is generated via a user or via the computer. This can be established by evaluating the whether the packet contained a HTTP referrer which makes mention of the script which is used in the POST.

6.4.4 Domains associated with LOCKY

LOCKY has hard-coded domains which it contacts to download the payload which is needed to encrypt the victim's machine. This specific strain of Ransomware works in two phases in that the initial infection is that of a Trojan downloader which accesses websites, which contain the LOCKY payload to download it. This specific experiment did not encrypt the machine however it was not for lack of trying as multiple domains where contacted the websites however were cleaned or the file did not download. The domains which were contacted are known to form part of two attack processes the one is known as the Malspam campaign and the other the DRIDREX-STYLE MALSPAM. This shows that the LOCKY malware has adapted to include multiple domains to be contacted in an attempt to ensure that it executes. Post/ domains contacted by LOCKY. LOCKY contacts 4 unique domains using the HTTP protocol with a POST packet. These packets contain encoded information which could not be decoded. These four IP addresses have been associated with previous campaigns of LOCKY is being the command and control server which communication is kept alive very much the same as a beaconing by traditional malware these can be seen in the the Table 6.5.

Table 6.5: Reputational check done on Threat Miner

IP ADDRESS	DOMAIN	COUNTRY	CAMPAIGN
209.236.112.102	itbouquet.com	United States	Malspam
81.169.145.73	micolon.com	Germany	Malspam
103.50.162.56	partyangel.in	India	Malspam
107.180.28.116	clayhero.com	United States	Dridrex-Style Malspam
216.87.186.88	bhmech.com	United States	Dridrex-Style Malspam
216.117.141.38	sherwoodbusiness.com	United States	Dridrex-Style Malspam

6.5 Forensic Artefacts

When conducting a **Digital Forensic** examination it is important to note, that each interaction that is made with the system will leave a residual trail. This allows for the piecing together of information from artefacts contained on the system to combine into a timeline. The chapter which follows details the forensic artefacts which were found on the system.

6.5.1 Memory Analysis

When process injection takes place on a machine within memory there is often artefacts of memory protection. LOCKY malware strain effects `PAGE_EXECUTE_READWRITE` at the base address 0x00263000. This is an indication that within memory there is dynamic unpacking of executable. LOCKY malware strain makes use of this to hollow out the 0xffffffff process handle which is associated with the `ntdll.dll` handle of `TerminateProcess`. This means that the `ntdll.dll` was not loaded normally but rather injected Monnappa (2017).

When process injection takes place on a machine within memory there is often artefacts of memory protection. LOCKY effects `PAGE_EXECUTE_READWRITE` at the base address 0x00263000. This is an indication that within memory there is dynamic unpacking of executable. LOCKY makes use of this to hollow out the 0xffffffff process handle which is associated with the `ntdll.dll` handle of `TerminateProcess`. This means that the `ntdll.dll` was not loaded normally but rather injected Monnappa (2017) . During the examination of the LOCKY memory image, the following URLS were found to be called by the LOCKY. The URL which were extracted is detailed in Table 6.6 the URL which were extracted point to websites on the which supports the findings of Locky making use of DGA Algorithms to generate domains as detailed in Section 3.9.3.

Table 6.6: URL extracted from memory

Type of Match	URL
Heuristic match	Doubtsoftware.com
Heuristic match	krhrsalptiu.pw
Heuristic match	aheertjnmsalypyvi.pw
Heuristic match	ware.com
Heuristic match	dgadeurenypknu.ru
Heuristic match	xlegxhpovqttrnk.info
Heuristic match	kybjarouj.org
Heuristic match	qlboauoteqplnmt.su

6.5.2 Registry Analysis

LOCKY exhibits signs of accessing the cryptographic machine-guid. This is done by accessing the following keys within the Windows registry of the host machine. The machine-GUID is then utilised to uniquely identify and track the infection. This can be observed in the creation of the mutex creation in Section 6.5.1. LOCKY tokenizes the machine-GUID value using a hyphen the calculations are done, this can be seen in Figure 6.7.

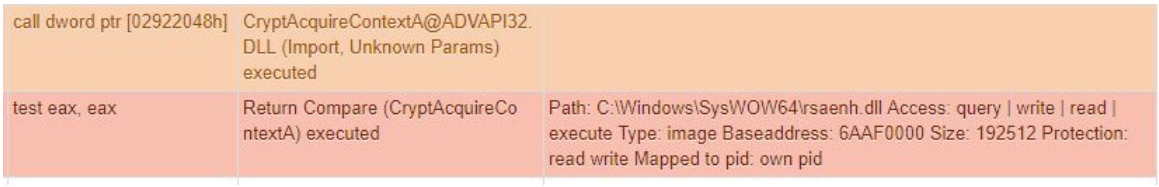


Figure 6.7: Reads the Cryptographic Machine-GUID

The Windows registry contains a cache, which is where the operating system stores information temporarily. The Internet Cache is used to store web pages that a user accesses, this information is used to enable the system to access it again. The Windows operating system caches most web pages these include SSL (secure-socket-layer) secured pages, these contain sensitive data. LOCKY malware accesses the Internet settings within the Windows Registry. This can be seen in Figure 6.8, the subkey which is accessed by LOCKY is the "`disablecachingofsslpages`" this is done to ensure that the operating system does not cache this information during the malware accessing SSL secure pages.

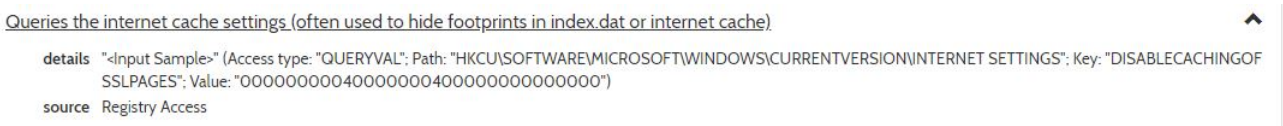


Figure 6.8: Internet Cache Registry Query

LOCKY accesses the Windows Registry key to identify what software policies have been put in place. This is done to identify whether there is policies in place which would hamper the

executable from successfully infecting the victim machine. This key was not enumerated and was only read by the malware.

HKLM\SOFTWARE\Policies\Microsoft\Windows\Safer\CodeIdentifiers.

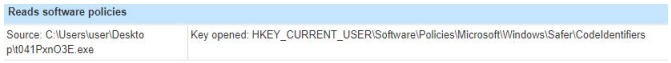


Figure 6.9: Software Policy Query

LOCKY creates a new registry key : `HKLMU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce`. This key which was created is named `csrstub` which is not an officiallydocumented registry key. LOCKY makes use of an undocumented registry key to maintain persistence on the the victim machine. This ensures that the program is executed from the terminal within the current directory. This allows an application to use the `LoadLibrary` function which will follow the dll search order. This means that when the executable is run the hooked dynamic library link file will be executed. LOCKY further more creates the a Windows lnk file of the registry value which it assigns for persistence, this creation of the executable in the Start Up directory will ensure that the executable file will execute at start up and run the malicious hooked dynamic link library file this can be seen.

LOCKY creates various mutex objects which have been listed in the list below. These mutex objects are used to ensure that the same host is not reinfected. This is done when a specimen attempts to create a specific handle but finds that the handle is already open thus showing that the host is already infected. There is also a theory by Abhishek and Bhu (2017) that the component of creating mutex objects is for the malware to be able to synchronise the injections of multiple copies of itself into various processes. The listing for the mutexes is detailed in Section B.3.

6.5.3 Digital Forensic Artefacts

LOCKY makes use of the specific functions to make changes to the victim's wallpaper. This is done by invoking the portion of code detailed in Table 6.7 launches a shell which is executed which changes the wallpaper of the user. This allows LOCKY to execute a shell script which changes the wallpaper to `_HELP_instructions.bmp` which contains the instructions on how to pay the ransom. This however was only partially observed as the full encryption did not happen and the malware went into a loop. This change in wallpaper can be seen in Figure 6.10.

Table 6.7: Sequence within assembly code which changes the user's desktop wallpaper

Function	Description
EH_Prolog	Mechanism used by Windows to handle exception.
RegOpenKeyExA	Opens the specified registry key.
SystemParametersInfoW	Retrieves the value of one of the system-wide parameters.
ShellExecuteW	Performs Shell-Code on a specific operation on a specified file.
RegCloseKey	Closes a handle to the specified registry key.

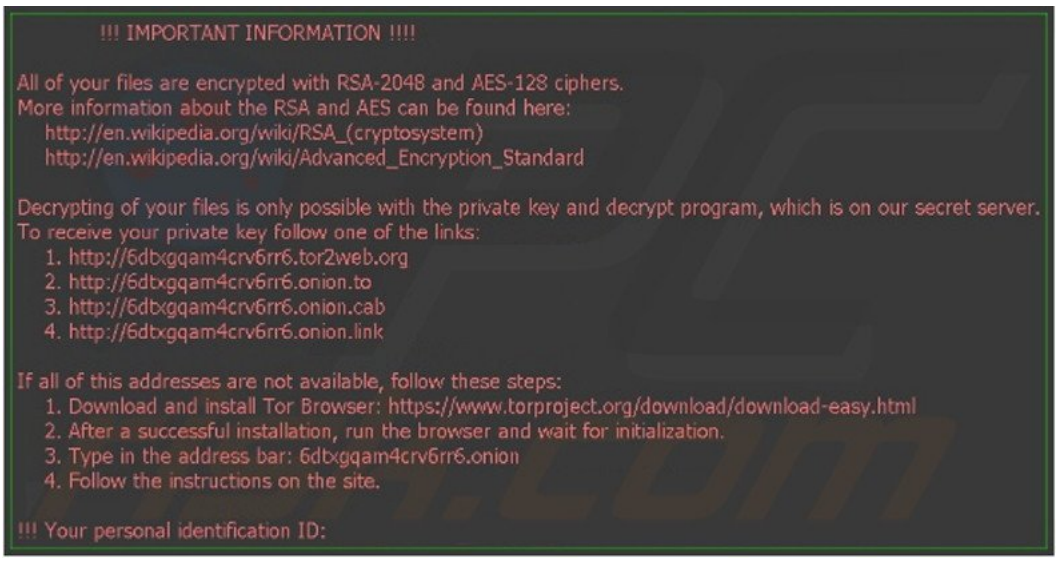


Figure 6.10: Wallpaper image _Help_instructions.bmp_

LOCKY makes use of the vssadmin.exe which is responsible for managing the Volume Shadow Copy process on Windows. LOCKY was found to create a process which identifies whether the Volume Shadow Copy is enabled, if this process is found to be enabled Ransomware proceeds to disable this and ensure that the user cannot roll back to a previous snapshot. These strings identify the versions of the Volume Shadow copy and proceed to delete them. This is done using the command line utility within Windows to perform the function of disabling and deleting the snapshots. The command which was used can be seen in Figure 6.11.

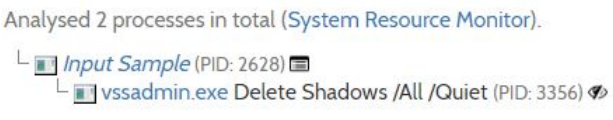


Figure 6.11: Deletion of the vssadmin.exe to delete Volume Shadow Copies

LOCKY executes the code below which calls upon the Kernel32.dll to perform a GetLogicalDrives function which checks the free disk space available. This is done by using the following process of firstly identifying which logical drives there are this is done using the function

`GetLogicalDrives` . The second step that LOCKY follows is to identify the type of drive which is contained within the machine, whether they are fixed, removable or a network drive. Thirdly LOCKY identifies how much of the drive is used and how much is still free and not used by the system. The final step in this process which is shown in Figure 6.12 is to use the `GetVolumeInformation` which identifies the drive which is associated with the file system and the root directory. This is done in attempt to map the drives associated with the victim's machine to ensure that the malware encrypts all the data.

029190C5	C644247C01	mov byte ptr [esp+7Ch], 00000001h	
029190CA	FF15C0209202	call dword ptr [029220C0h]	GetLogicalDrives@KERNEL32.DLL (Import, 0 Params)

Figure 6.12: Flow process of LOCKY accessing information on the system volumes

Arntz (2016b) states that the host file is likened to the speed dial directory to the internet. The Windows host file is located in the `%systemroot%\SYSTEM32\DRIVERS\ETC`. The Windows operating system before trying to resolve and internet request to resolve the IP to the domain, Windows would read the host file to identify whether this domain is predefined entry in the host file for the specific domain. Legitimate communication initiated from the host file would allow for communication with security vendors and mapped intranet and mapped drives. The aim of malware is to ensure that the victim machine cannot connect to security vendor servers and would block these communications and redirect the traffic to a server of their choice. In Figure 6.13 the LOCKY malware reads the host file contained on the test machine. This is done to hijack the file and identify whether there is any blocked domains which would affect the malware from downloading the payloads. It also ensures that the security software cannot communicate with their servers online.

C:\Windows\System32\drivers\etc\hosts
C:\Windows\System32\drivers\etc\hosts

Figure 6.13: LOCKY malware reading the local host file

LOCKY uses various native Windows function to create a new security identifier and obtain the required level privileges .LOCKY obtains the `Kernel32.dll` process which is a system level process that has administrator level privileges. It then proceeds to open the process token assigned to this process. Once the information is obtained it reassigns this to a new security identifier and checks once the process is complete whether this user is indeed part of the Administrator group.

The sample makes use of the COM classes is exploited to enable the LOCKY sample to hook onto the legitimate module is the System32 folder `ole32.dll` to be replaced by the malicious process which is loaded by the malware. This is done by replacing code within the class, when the class and objects associated with it is loaded the malicious code would be run instead of the legitimate code also referred to as hollowing. This is done as a method to evade detection by the malware as it seems to be a legitimate process taking place on

the system (Dell Secureworks, 2015)

6.6 MITRE ATT&CK Map

The LOCKY sample was examined using the online sandbox environment Hybrid Analysis[17]. This online sandbox maps the behaviour of the samples which were submitted to the MITRE ATT&CK adversarial technique map. The results can be seen in Figure 6.14. The initial infection vector which is followed by LOCKY did not match any tactics within the framework and thus can be included to be unique in that the initial infection is a remote access trojan. The LOCKY Ransomware uses the process of injecting malicious code into a process which runs at system level. This means that the process which is identified to be injected into is one that runs with administrator of system access. This means that the code which is injected would run at system level. The Ransomware also makes use of injecting the code within a legitimate process to evade signature code based detection. This is done by masking the execution under a legitimate process.

LOCKY makes use of an evasion technique where is attempts to make the executable file difficult to identify and discover by obfuscating the executable. This is done by using entropy within the executable. The obfuscating technique use by LOCKY is that it reorders the instructions within the code which makes signature based detection harder. The Ransomware makes use of a technique which is noted underneath the credential access portion of the matrix in which the malware attempts to use methods to capture the user credentials. This is done by collecting information from user input by using a keylogger. This is done by intercepting the keystrokes made by the user using a User Account Control prompt or wrapping the Windows default credential provider. This would indicate that LOCKY attempts post exploitation by exfiltrating user credentials. LOCKY is environmentally aware in that is queries registry keys to obtain key values to system information such as the operating system, hardware and versioning information of software. The Ransomware scans the system to identify the security software which is installed and the configuration of defensive tools used to ensure that the system is not compromised. This includes specific sensors put in place. LOCKY makes attempts to get a listing of the IP addresses, hostnames and other logical identifiers on the system. This is done to make lateral movement easier on the current system. This could be exploited using remote access tools. The fact that LOCKY makes use of a remote access trojan to download and facilitate the infection fits with the findings of the Hybrid analysis examination.

The exfiltration of data by LOCKY is done, however the data which is exfiltrated is encrypted which makes the detection harder. The encryption protocol which is used is the standard Windows cryptographic protocol. The communication by LOCKY with the C&C is done by using the known native Windows cryptographic protocol to communicate via HTTP\POST. LOCKY uses a non-standard application layer protocol for communication between the host and the C&C. Specific examples of this include but are not limited to

[17] https://www.hybrid-analysis.com/

ICMP, TCP and UDP protocols. The use of standard application layer protocols can be used by malware to communicate to the C&C server. In this instance LOCKY makes use of HTTP\POST for posting information to a web-server.

Mitre Att&ck Matrix

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command and Control
Valid Accounts	Windows Remote Management	Winlogon Helper DLL	Process Injection 1	Software Packing 2	Input Capture 1	Security Software Discovery 2 1	Application Deployment Software	Input Capture 1	Data Encrypted 1	Standard Cryptographic Protocol 2
Replication Through Removable Media	Service Execution	Port Monitors	Accessibility Features	Process Injection 1	Network Sniffing	System Information Discovery 1 2	Remote Services	Data from Removable Media	Exfiltration Over Other Network Medium	Standard Non-Application Layer Protocol 1
Drive-by Compromise	Windows Management Instrumentation	Accessibility Features	Path Interception	Obfuscated Files or Information 4	Input Capture	Remote System Discovery 1	Windows Remote Management	Data from Network Shared Drive	Automated Exfiltration	Standard Application Layer Protocol 1

Figure 6.14: LOCKY Behavioural map
Source: `https://www.hybrid-analysis.com/`

6.7 Summary

In this instance the LOCKY strain which was used was unable to complete the encryption process, this was due to the payload being removed from the internet. This means that LOCKY is unable to encrypt a system without the additional payload. The first level of exploitation is the installation of a remote access trojan which prepares the system for download and installation of the payload. LOCKY is generally spread via spam emails and dedicated campaigns. This LOCKY remote access trojan has the ability to identify and map the system which it is infecting. This is also the one strain that has keylogging characteristics and exfiltrates information to a web-server. Chapter 7 take the finding form the examination detailed in Chapter 4, Chapter 5 and Chapter 6 and compares them accross all three strains to answer the research questions as presented in Chapter 1 in Section 1.3.

7

Similarities between Ransomware

7.1 Introduction

Three major families of Ransomware were examined in the preceding Chapters 4, 5 and 6. using the new Hybridised-Malware Analysis framework as set out in Chapter 3. This enabled the researcher to gain an in depth comparative understanding on how Ransomware works. This can be equated to a ready bake cake mixture which contains the basic ingredients needed for a cake, the differences between the cakes are flavouring and topping. This research has shown that there are indeed more similarities between these families than differences. By further understand what elements of Windows is being exploited in further research methods can be developed for early detection and prevention. The three families were examined using various techniques and in this chapter the focus will be on what fundamentally makes these strains successful and what similarities there are between the three strains.

Section 7.2 details the similarities between the three families in terms of the examination done using the MITRE ATT&CK framework. This framework aided in successfully completing the comparison. The section to follow on this is Section 7.3 this is detailed similarities between the three families in relation to methodology as explained in Chapter 3. Section 7.4 is detailed information on the similarities in the methods used by the malware to query the Cryptographic machine-GUID. Section 7.5 details the use of security descriptor manipulation by the malware. Section 7.6 details the methods used by the malware to be environmentally aware and query the local system time. Section 7.7 is the processes used by the malware to query the Windows Registry to obtain software policy information. Section 7.8 details the exploitation used by the malware of processes. Section 7.9 details

the methodology or approach taken by the malware examined to obfuscate their code. Section 7.10 is the comparison between the three strains examined as to their effect on the hardware of the test machine. The chapter concludes with a summary of the comparison in Section 7.11.

7.2 MITRE ATT&CK Framework Comparison

The MITRE ATT&CK framework was used to map the behaviour of the three Ransomware strain to identify the similarities relating to their behaviour. This was done by obtaining the information during the research using the proposed Hybridised-Malware Analysis framework. The comparison was sectioned by phases. The purpose of this framework is to enable the user or organisation to map current threats against Advanced Persistent Threats.

7.2.1 Initial Access

This tactic represents the approach used by the adversary to gain initial access and also the method used to gain the initial foothold within a target network. The three strains which were examined all are spread via malspam campaigns identified in Chapter 2.1. Section 2.6 which have been uniquely crafted to entice the victim to open the document which is attached to an email. The three strains which were also examined had the ability to mimic legitimate software, which is downloaded by the user. The information relating to the file versioning information is detailed in Sections 4.2.2, 5.2.2 and 6.2.2. This shows that the main access point across all three samples of Ransomware is the targeting the human factor. Often malware operators will focus on the human factor as explained by (Metalidou *et al.*, 2014). Unlike other malware types such as Ransomware is reliant on the human interaction with the document which contains the malicious code.

7.2.2 Execution

This tactic represents the execution phase that results in the execution of the adversary remote code in the local or remote system. This tactic invokes techniques in conjunction with the initial access as the way to execute the code once the access has been secured. The three strains which were examined all exhibited different execution methods. The MAKTUB family exploits the Windows Loader module to execute the malicious which is detailed in Chapter 5. The CERBER strain exploits it's ability to install itself as a Windows Service with escalated privileges this is detailed in Chapter 4. The LOCKY strain on the other hand install a remote access trojan which facilitates the download and installation of the payload this is detailed in Chapter 6. This shows that Ransomware exploits the native abilities within Windows which are vulnerable to exploit the Literatyr.

7.2.3 Persistence

The persistence which is created when access has been obtained. This is the changes which is made by making changes to configuration settings to obtain longer access and maintaining the presence on the system. One of the prime directives for malware is to maintain the access to the system when interruptions happen and connection is lost with the victim machine. The three samples which were examined all exhibited similar methods of ensuring persistence. The key similarity across the three strains was that they all ensured persistence in the `Start-Up` items registry keys which is detailed in Sections 4.5.2, 5.5.2 and 6.5.2. The three strains all ensure that they query and change the values to ensure that when the system is rebooted the infection and persistence is maintained.

7.2.4 Privilege Escalation

This tactic is surround the techniques used to escalate the privileges of the malware to that of `administrator` or `SYSTEM`. The one element which all successful malware has in common is that they obtain higher privileges on the system that they are exploited. An user account with administrator privileges is always more favourable however through various techniques and exploits the privileges can be escalated on the system. The three strains which were examined had one clear similarity in that they escalate the processes of the malicious code by means of process injection. This is done by identifying a process which executes at `SYSTEM` level and injecting the malicious code into these processes. This can also be observed by the fact that the malware executes and injects itself into a process which is running at `SYSTEM` level.

7.2.5 Defence Evasion

These techniques are used by the adversary to evade detection on the target network. These actions can be variation of multiple techniques from one category (Johnson, 2018). The evading of detection can be seen as an attribution which can apply to all malware. The three Ransomware strains which were examined had multiple similarities in this important phase of exploitation. The strains examined all made use of some sort of software packing discussed in Section 2.7 which obfuscates the code portions of the static binary. They make use of exploiting valid accounts. The use of process injection which is the use of a legitimate process with injected malicious code ensures that detection is evaded from detection by anti-virus.

7.2.6 Credential Access

This section of tactics describes the methods used by malware or adversaries which result in them obtaining access to control the system. Malware will attempt to obtain legitimate credentials from users or administrator accounts on a network (Strom *et al.*, 2018). This

allows them to assume the identity of the account to manipulate and control the access to the system. The examination of the three strains yielded no similarities and only MAKTUB matched the criteria or techniques in this section. MAKTUB attempts to gain access to privilege accounts by intercepting the keystrokes made by the user detailed in Section 5.3.

7.2.7 Discovery

The discovery phase further describes the gathering of information which allows the adversary or malware to exploit the system. This is done once access has been gained on the network, and host. This can include mapping of the outlay and internal network. The three families of Ransomware all were environmentally aware which means that various registry keys and values are queried to determine the information of the system these are detailed in the separate examination Sections 4.3.1, 5.3.1 and 6.3.1. This includes interrogating the system to obtain the local time, operating system of the victim machine, the policies in place and which security software is run on the local machine. This enables the malware to circumvent or alter settings to ensure full exploitation.

7.2.8 Lateral Movement

The Lateral movement tactic is the techniques which enable the adversary or malware to access and control systems remotely whilst pivoting through the network. This technique allows the adversary or malware to gather information and living off the land. With this tactic there were was no similarities within the approach and only CERBER has the ability to spread through the use of removable devices as can be observed in Figure 4.17.

7.2.9 Collection

These are the collection tactics used by malware or adversaries to obtain information that can identify sensitive files which can be exfiltrated. This category covers the locations on the local machine and the network shares which contain valuable information. The MAKTUB Ransomware enumerates files which are contained on network shares by exploiting the SMB 1 protocol (Assor, 2016), it is also the only strain which makes use of keystroke interception to obtain credentials needed to access information from locked accounts. This shows that it exploits known vulnerabilities within Widows. MAKTUB utilises the vulnerabilities within Windows to laterally move within the network. None of the other variants were flagged to contain collections methods (Hasherezade, 2016c).

7.2.10 Exfiltration

This refers to the methods used to exfiltrate the information which have been accessed and collected on the system. This is the techniques which are used to remove information and files from the system and the network. The three strains which were examined did not

have any similarities in this section, the LOCKY Ransomware posts encrypted information to a web-server making use of the HTTP protocol detailed in Section 6.4.2.

7.2.11 Command and Control

The command and control tactic is the techniques used to communicate with the system which are being controlled within the targeted network. There are various ways that this can be done to establish a clear communication pipeline from the controlling server to the victim machines. The three strains all exhibited and exploited different protocol the main similarity was that they all made use of HTTP protocol. However the main difference was the method used to communicate by these three strains. The main similarity is that they make use of a standard cryptographic protocol.

7.3 Similarities from overall examination

The three strain which were examined to determine whether there are basic things that Ransomware does to exploit a system and whether there is evidence of an approach for exploiting Windows. The similarities which were highlighted below are the similarities which exploit a specific portion of Windows which are imperative to the success of the infection. Due to space and scope limitation all the similarities could not be highlighted and the researcher only focused on the most important similarities.

7.3.1 Windows Native API's

The following Windows API were invoked by all three Ransomware strains and exploited. These are the API's which are vulnerable to exploitation. They are also the ones which are needed to maintain persistence and infection. These were discussed in more detail in Sections 4.2.3, 5.2.3 and 6.2.3.

- `Kernel32.dll`
- `Shell32.dll`
- `User32.dll`
- `GDI32.dll`

7.3.2 Entropy

The use of entropy of encryption of the binary file in malware is done to protect the binary file from detection or analysis. The word entropy in terms of malware analysis can be described explained as the randomness which is collected by the operating system or application to create obfuscation or cryptography. The entropy examination is to basically

determine what the level of discord is with the placement of the bytes are. This enables the researcher to adapt the examination approach to determine whether a binary file is encrypted or not. The MAKTUB binary file contained seemingly arbitrary code, however examination using *Detect-It-Easy* identified that the binary file was packed and as such the static analysis was limited in what could be observed, both the samples from LOCKY and CERBER contained high entropy levels on sections which contained writeable data and the executable code. This is done by the malware operators to protect that portion from antivirus and analysis techniques. The important note is that all three families used some form of entropy to protect the binary from static analysis however only MAKTUB made use of a packer. The information is detailed in Sections 4.2.4, 5.2.4 and 6.2.4.

7.3.3 Anti-Debugging

The ability for malware to register as top-level exception allows the malware to execute with the same privileges as the operating system, which is kernel mode. The methods used by the three Ransomware families were examined to determine whether the same approach is used. The `Kernel32.dll` forms part of the default exception handlers within Windows. When an executable is run on Windows and there has been no exception handler assigned to run the debugging process, this function is taken over by the `Kernel32.dll`. The Top-level exception filters are used because that has the top-level/administrator rights on the operating system. This behaviour is done to detect the presence of a debugger on the system. Once this has been found an exception is passed and an application error is forced on the system. If this process is not being debugged it will continue on. This is a technique often used as an anti debugging process. All three of the Ransomware families made use of the same method for anti-debugging which is the "`SetUnhandledExceptionFilter@KERNEL32.dll`".

7.3.4 Volume Shadow Copies

The three families of Ransomware all makes use of the `vssadmin.exe` which is responsible for managing the Volume Shadow Copy process on Windows. The three Ransomware families were found to create a process which identifies whether the Volume Shadow copy is enable, if these are indeed enabled they will proceed to use the '`vssadmin.exe Delete Shadows /All /Quiet`' to disable and deleted the copies of it. This keep the user from rolling back to a previous version to recover a working process of infected machine. MAKTUB however takes it one step further once it has disabled and deleted the previous volume shadow copies, it will make a snapshot of the machine in it's current infected state, this would mean that when the user boots into a previous version thought to be stable the infection would reoccur. The similarities in this step shows that Ransomware targets this portion of Windows to ensure that the infection is not rolled back. The method used by MAKTUB us brilliant in that it guarantees reinfection.

7.3.5 BCEDIT.exe manipulation of Boot system configuration

The `BCEDIT.exe` within Windows is responsible for editing the boot configuration data. The malware is known to exploit this on two parts to ensure that the user cannot boot into safe mode to debug or fix the operating system. This is often also employed according to Sikorski and Honig (2012). Malware also uses this approach to ensure that the malware is executed in kernel-mode. Kernel-Mode within the Windows operating system is the highest level of execution which is only allocated to the operating system and drivers. The three Ransomware families in the beginning of the infection all exploit the internal native API for Windows which controls the method in which the machine boots to ensure that the victim no longer can boot into safe mode to discard the infection.

7.4 Queries the Cryptographic machine-GUID

Ransomware much like other malware seem to exploit the Native Cryptography API of Windows. Dell Secureworks (2015) states that malware uses the machine-GUID value to identify when it is in a Virtual Environment. This is done by calculating the machine-GUID of the machine upon first execution to identify the value. Once the infection continues the value is calculated again. When this is done on a virtual environment the values will be different and the inference is made that the malware is run in a virtual environment. The examination showed that only two of the three families which were examined queried the Cryptographic machine-GUID, these were CERBER and MAKTUB; LOCKY however did not query this registry key.

7.5 Security Descriptor Manipulation

Windows makes use of security descriptors which aids in the control access of these objects. The security descriptors include the information about which process owns an object and what level of access this object has. The three Ransomware families were examined, of the three families only two manipulated the security descriptors within Windows. Both the CERBER and MAKTUB family of Ransomware exploited this was also done by identifying a `FreeSID` and creating a new security descriptor with administrative privileges detailed in Sections 4.5.3 and 5.5.3. The LOCKY Ransomware did not exploit this functionality within Windows.

7.6 Get Local Time

The Ransomware families displayed the potential ability to be environmentally aware, in they have the ability to query the time of the host machine it is infecting. This was found during the assembly code examination to determine which static information can be

drawing from the malicious executable. This API contains top-level privileges; this means that the privileges of the malware have been escalated to Administrator. This technique is used by malware to evade or detect sandboxed environments. Sikorski and Honig (2012) explains that malware use this approach to detect whether a virtual machine has been run as there is a time penalty when running a virtual machine. This can be bypassed by potentially using and external time source such as NTP. All three the Ransomware families have the ability to be environmentally aware in that they can query the local machine time. The CERBER and MAKTUB family uses the same function within Windows which is "`GetLocalTime`", the LOCKY malware makes use of another function within Windows to do the same thing which is the "`GetSystemTimeAsFile`". There are similarities in what they would like to establish however it seems that LOCKY takes a different approach.

7.7 Queries Software Policies

The Ransomware malware accesses the Windows Registry key to identify what software policies have been put in place. This is done to identify whether there is policies in place which would hamper the executable from successfully infecting the victim machine. All three of the Ransomware families which were examined all displayed the functionality to read and augment this registry key to ensure that execution is done. The registry location which is opened or enumerated in all three is the following key:

HKLM\SOFTWARE\Policies\Microsoft\Windows\Safer\CodeIdentifiers

7.8 Uses an in-Process (OLE) Automation Server

The COM objects is the platform which independent, distributed, object orientated system that a binary or software component can interact with. This was designed by Microsoft to provide an interface that allowed for developers to control and manipulate other objects of other applications. This process is often used by the malware operators to create persistence in a stealthier manner. This is done in the case of the MAKTUB malware strain by altering the following registry key:

HKLM\SOFTWARE\Classes\CLSID\{1f486a52-3cb1-48fd-8f50-b8dc300d9f9d}\InProcS

According to Rascagneres (2014) this new way of persistence mechanism has many advantages over the other methods of persistence. As soon as the malware has infected the system it is able to bypass the anti-virus monitoring due to Windows natively executing the library which contain the process which has been infected. This also can make it harder for the victim to identify the process which is malicious. This attack has been seen to be used in conjunction with a Remote Access Trojan (Rascagneres, 2014). The three Ransomware families were examined and only two of the three families uses an in-process automation server. The LOCKY family does not use it however the CERBER and MAKTUB families both query the same registry key.

7.9 Code Obfuscation Technique

Malware makes use of a data obfuscation technique which is used to protect code from static analysis. In assembly code when a "call" is issues it point to a return address which is the address where the execution should take place. The malware generally use three functions together which is the `call`, `ret` and `jmp` functions. Kumar (2004) explains that malware authors obfuscate their code with the intention to make it complicated to gain information from the code itself. Malware operators rearrange their code to make the detection process harder. The most popular method used by malware authors according to Kumar (2004) is that they obfuscate the `call addr` instruction. The three Ransomware families which were examined all made use of the instruction reordering obfuscation technique showing that this was a similarity in how the code was designed these are detailed in Sections 4.2.7, 5.2.7 and 6.2.7.

7.10 Hardware Usage Comparison

The three Ransomware families were examined to determine in terms of hardware, whether there is similarities between CERBER, LOCKY and MAKTUB. This was done by monitoring the behaviour using *ProcMon* to determine the statistical usage from execution to encryption of the physical test machine RAM/Memory and the CPU. This for the researcher was relevant due to the reason that these form part of the more encompassing analysis approach to determine whether changes can be noted on the physical hardware of the machine. The methodology followed by the authors Ferrante *et al.* (2016) consists of three steps which is the Hybrid-Malware analysis approach, the monitoring of the CPU usage to identify key events and Memory monitoring events to identify discrepancies.

Malware makes use of the CPU to pull resources towards the infection and control of the system. Ransomware is now different in that it allocates services to the infection of the machine, installing itself onto the system and encrypting the data contained on the machine. The researcher examined the CPU usage by the malicious Ransomware strains which were examined to identify which processes made use of the CPU most intensively and which Ransomware strain was heaviest on the victim machine.

This would allow for the better understanding of the usage of the hardware on the victim machine. The CERBER malware which was examined had 20 processes which intensively used the CPU of the test machine. The malicious binary itself was the most intensive in that it utilised approximately 45% of the CPU during the infection and encryption phase. The MAKTUB malware which was examined had 3 processes which intensively used the CPU of the test machine. The malicious binary itself was the most intensive in that it utilised approximately 82% of the CPU during the infection and encryption phase. The LOCKY malware which was examined only had 1 process which used the CPU of the test machine. The malicious binary itself was the most intensive in that it utilised approximately 18% of the CPU of the test machine. The results of these would indicate that

MAKTUB which is the fully encrypted binary file utilises more CPU than the other two variants which contain less encryption on the binary itself.

Teller and Hayon (2014)indicates in their research that there has been an increase in non-persistant malicious payloads which only operate within memory. This includes the malware strains which are encrypted or packed and only unpack or decrypt themselves within the memory. These are called file-less malware. This would mean that static analysis would not be in a position to gather all the information regarding the execution of the malware as the code is dynamically unpacked within memory. The approach which was taken by the researcher was to identify which processes associated with the Ransomware which was examined to determine whether there is a pattern to the approach by each strain which would allow the researcher to better understand the working of the malware. The CERBER malware strain which was examined spawns various processes, however the malicious binary which is introduced to the system has an initial spike of 60 MB + usage of the memory of the test machine and tapers off .

The hypothesis in this is that when the installation and encryption process takes place the memory preference is allocated to that process. The CERBER binary file was not packed or encrypted but did contain high levels of entropy. The MAKTUB malware strain which was examined shows a constant usage by the malicious binary which was introduced to the system using 30 MB and 60 MB of memory constantly, this binary file was encrypted and the hypothesis is that the constant use of the memory is due to the dynamic unpacking of the malware within memory. The LOCKY malware strain interesting enough shows minimal usage of the memory with the malicious binary only using approximately 5 MB to 15 MB of memory this malware strain could also not encrypt due to being unable to download the malicious payload. The information collected with regards to the CPU usage and Memory usage by these three strains would indicate that the encrypted binary file of MAKTUB utilises the most CPU and memory due to the fact that the file is dynamically unpacked within memory. The two other variants which only contain levels of entropy has highest usage with initial infection and encryption which is two processes which are CPU intensive.

7.11 Summary

The three families which were examined all made use of similar code obfuscation techniques in that in made use of the function reordering. The Ransomware all had the ability to query the time on the local machine and this indicated that they are environmentally aware. The main similarity is that these strains all had the ability to be able to manipulate the method in which the machine would boot by manipulating and using the local native API responsible for this function. They also have the ability to discard of the volume shadow copies which is a contingency put in place for when there is a system failure for the user to be able to roll back to a previous working version. The Ransomware all showed the ability to circumvent debugging using the same method and escalating their

privileges to that of system. The research has shown that the exploitation of Windows is done by living of the land and using what is natively accessible from within Windows. This both shows attention to detail but also detailed knowledge of the functionality and built of the Windows operating system. The research has shown that there exists more similarities in the method used to exploit Windows than initially was thought however the network communication methods and post exploitation remains vastly different.

8 Conclusion

8.1 Introduction

The research aimed to further and broaden the methodology used to examine Windows Platform malware, specifically the identification of similarities between Ransomware families. This research was done with the aim to identify the base line exploits or tasks preformed by different Ransomware families. This led to the expansion of the Hybrid-Malware analysis framework[1] w (Roundy and Miller, 2010) which was discussed in Section 2.6. The expansion of the framework is discussed in Chapter 3.

This chapter begins with a summary of the research done in this thesis in Section 8.2. This is followed by the evaluation of the research goals which was presented in Section 8.2. The impact of the research as well as the relevance is discussed in Section 8.4. The final section of this chapter is the proposal for future work which build on this research in Section 8.5.

8.2 Research Summary

This thesis document starts with an introduction chapter (Chapter 1) which identifies the impact which Ransomware has had in the field of Information Security. This chapter also defines the problem statement, research goals, the scope of the work and the document conventions as well as the structure of the document. This research sets out to examine the methodology which is used to examine malware and to identify whether there is an alternate or improved way to examine malware to gain the maximum information from the

[1]The traditional Hybrid-Malware analysis framework consists of Static and Dynamic Analysis only

malware. The other aim of the research was to identify the similarities between the Ransomware strains to identify whether there are similar approaches to exploiting a Windows 7 operating system, The hypothesis proposed by the researcher was that there was basic similarities to the approach Ransomware takes to exploit Window 7.

Chapter 2 presented a literature review, where concepts related to malware analysis, Ransomware, network communication, and malware was examined and explained. An observation which was made that there was not academic resources discussing strain specific research into Ransomware or Ransomware holistically. This identified a gap within the academic field for more research to be done which is strain specific. The research which is presented here would address the gap within the academic field in relation to Ransomware.

Chapter 3 presents the new Hybridised-Malware analysis framework which was adapted from the previous standard approach. The chapter also presents the data collections and selection process. An extension of standard malware analysis which only included static analysis and dynamic analysis to include additional disciplines was proposed. This approach allowed for a comprehensive collection of data from the examination which addresses multiple facets of malware and the layers of exploits. The researcher also identified that additional exploits may also be run post encryption and this was specifically examined for each strain.

Chapters 4, **5** and **6** present the examination of the CERBER, MAKTUB and LOCKY strains of Ransomware which was selected. This chapter uses the methodology as set out in Chapter 3. This presents the findings for each step within the framework and identified whether there was additional exploits taking place post encryption.

presented the similarities between the Ransomware families which was examined using the methodology in Chapter 3. The similarities which were identified and discussed was those most relevant to the exploitation of a Windows 7 Operating System, due to constraints within the scope of the thesis they needed to be limited to only those most significant.

8.3 Evaluation of Research Goals and Questions

In order to be in a position to evaluate the effectiveness of this research, the outcomes discussed within the thesis are compared with the research goals which were presented in Chapter 1are presented below and discussed to the degree which they were met. The research questions were

8.4 Impact of Research

This research is valuable and of significance due to the fact that it has fully achieved the research goals set out at the onset of this research. These research goals were aimed at

producing a new approach to malware analysis with the focus on Ransomware. The research also identified significant similarities between Ransomware and the methodology they use. The research also will be the first seminal work done on examination using the new proposed methodology which focused on specific strains. There are multiple academic papers discussing Ransomware on a high level however this research is done with focusing on specific strains however doing a cross comparison between the strains. This work highlights a new method of malware analysis which add to the data collection and extends on a working framework which has been utilised in the industry.

8.5 Future Work

1. The expansion of the Ransomware behavioural study to include more families to enable a database to be drafted containing the similarities of a more substantive quantity of Ransomware strains.
2. Further testing using the extended Hybrid-Malware analysis framework as detailed in Chapter 3 but testing it against other types of malware.
3. The development of a methodology to identify, protect and mitigate against Ransomware attacks using the commonalities between them.

References

Abdulla, K., Jones, A., and Anthony Martin, T. Forensic Analysis of the Windows 7 Registry. *Research Online*, 5, 12 2010.

Abhishek, S. and Bhu, Z. Hot knives through butter, 2017. Accessed: 05 February 2018.
URL https://www.fireeye.com/content/dam/fireeye-www/current-threats/pdfs/pf/file/fireeye-hot-knives-through-butter.pdf

Abrahams, L. The Art of the Maktub Locker Ransomware, 2016. Accessed: 05 February 2018.
URL https://www.bleepingcomputer.com/news/security/theartofthemaktublockerransomware/

Akira, Y., Kou, I., Rui, T., and Yinmin, P. SandPrint: Fingerprinting Malware Sandboxes to Provide Intelligence for Sandbox Evasion. In *International Symposium on Research in Attacks, Intrusions and Defenses*. 2016.

Akkas, A., Chachamis, C. N., and Fetahu, L. Malware analysis of wanacry ransomware. Technical report, MIT Computer Science and Artificial Intelligence Laboratory, 2017. Accessed: 05 February 2018.
URL https://courses.csail.mit.edu/6.857/2017/project/20.pdf

Antonopoulos, A. Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media, 2014. ISBN 9781491921982.

Anubhav, A. and Ellur, R. Cerber: Analyzing a Ransomware Attack Methodology To Enable Protection. Technical report, FireEye, 2016. Accessed: 25 January 2017.
URL https://www.fireeye.com/blog/threat-research/2016/07/cerber-ransomware-attack.html

Arghire, I. Samas Ransomware uses Active Directory infect entire networks, 2017. Accessed: 05 February 2018.
URL https://www.securityweek.com/samas-ransomware-uses-active-directory-infect-entire-networks

Arntz, P. Explained: Domain Generating Algorithm, 2016a. Accessed: 05 February 2018.
URL https://blog.malwarebytes.com/security-world/2016/12/explained-domain-generating-algorithm/

Arntz, P. Hosts file hijacks, 2016b. Accessed: 05 February 2018.
URL https://blog.malwarebytes.com/cybercrime/2016/09/hosts-file-hijacks/

Arntz, P. Analyzing malware by API calls, 2017. Accessed: 05 February 2018.
URL https://blog.malwarebytes.com/threat-analysis/2017/10/analyzing-malware-by-api-calls/

Assor, Y. Anti-VM and Anti-Sandbox Explained, 2016. Accessed: 05 February 2018.
URL https://www.cyberbit.com/anti-vm-and-anti-sandbox-explained/

Babic, D., Reynaud, D., and Song, D. Malware analysis with tree automata inference. In **Gopalakrishnan, G. and Qadeer, S.**, editors, *Computer Aided Verification*, pages 116–131. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 9783642221101.

Baier, D. Caching and SSL Pages, May 2017. Accessed: 25 June 2017.
URL https://leastprivilege.com/2006/08/21/caching-and-ssl-pages/

Balaban, D. Locky Ransomware Statistics: Geos Targeted, Amounts Paid, Spread Volumes and Much More. Technical report, 2016. Accessed: 15 January 2017.
URL https://www.smartdatacollective.com/locky-ransomware-statistics-geos-targeted-amounts-paid-spread-volumes-and-much/

Barkly Research. Ransomware by the Numbers: Must-Know Ransomware Statistics 2016. Technical report, Barkly Research, 2016. Accessed: 15 January 2017.
URL https://blog.barkly.com/ransomware-statistics-2016

Barkly Research. 2017 Ransomware Trends and Forecasts. Technical report, Barkly Research, 2017a. Accessed: 25 January 2017.
URL https://blog.barkly.com/new-ransomware-trends-2017

Barkly Research. Cerber Ransomware Everything You Need to Know. Technical report, Barkly Research, 2017b. Accessed: 02 January 2017.
URL https://blog.barkly.com/cerber-ransomware-statistics-2017

Barkly Research. Must-Know Ransomware Statistics Must-Know Ransomware Statistics 2017. Technical report, Barkly Research, 2017c. Accessed: 05 February 2018.
URL https://blog.barkly.com/ransomware-statistics-2017

Barkly Research. Must know ransomware statistics 2018. Technical report, Barkly Research, 2018. Date Accessed: 11 December 2018.
URL https://blog.barkly.com/ransomware-statistics-2018

Bethencourt, J., Song, D., and Waters, B. New techniques for private stream searching. *ACM Transaction on Information System Security*, 12(3):1–32, January 2009. ISSN 1094-9224. doi:10.1145/1455526.1455529.

Blunden, B. The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System. Jones & Bartlett Learning, 2009. ISBN 1598220616.

Bollinger, J., Enright, B., and Valites, M. Crafting the InfoSec Playbook: Security Monitoring and Incident Response Master Plan. O'Reilly Media, 2015. ISBN 9781491913604.

Booth, W. C., Colomb, G. G., and Williams, J. M. The Craft of Research. University of Chicago Press, 1995. ISBN 9780226065687.

Branco, R. R., Barbosa, G. N., and Neto, P. D. Scientific but not academical overview of malware anti-debugging, anti-disassembly and anti-vm technologies, May 2012. Accessed Date: 15 May 2017.
URL https://media.blackhat.com/bh-us-12/Briefings/Branco/BH_US_12_Branco_Scientific_Academic_Slides.pdf

Bullguard. Definition of Malware, 2017. Accessed: 05 January 2018.
URL https://www.bullguard.com/bullguard-security-center/pc-security/computer-threats/malware-definition,-history-and-classification.aspx

Campbell, R. Researchers Uncover Cerber as Largest Ransomware-As-A-Service Ring, August 2016. Accessed: 22 February 2018.
URL https://www.ccn.com/90571-2/

Carvey, H. Windows Registry Forensics: Advanced Digital Forensic Analysis of the Windows Registry. Elsevier Science, 2016. ISBN 9780128033357.

Check Point Software Technologies. VB2017 paper: Nine circles of Cerber. Technical report, 2017. Accessed: 25 May 2017.
URL https://www.virusbulletin.com/virusbulletin/2017/12/vb2017-paper-nine-circles-cerber/

Check Point Threat Intelligence and Research. Jigsaw Ransomware Decryption. Technical report, 2016. Accessed: 25 January 2017.
URL https://blog.checkpoint.com/2016/07/08/jigsaw-ransomware-decryption/

Cho, I. K., Kim, T. G., Shim, Y. J., Ryu, M., and Im, E. G. Malware Analysis and Classification Using Sequence Alignments. *Intelligent Automation & Soft Computing*, 22(3):371–377, 2016. doi:10.1080/10798587.2015.1118916.

Cobalt Strike. User Account Control : What Penetration Testers Should Know, 2014. Accessed: 05 February 2018.
URL https://blog.cobaltstrike.com/2014/03/20/

Costlow, K. The year of ransomware: 2017 recap and 2018 predictions, 2018. Accessed: 05 February 2018.
URL https://blog.stealthbits.com/The-year-of-ransomware-market-trends

Dalei, W. and Li, Y. Metamorphic malware detection using opcode frequency rate and decision tree. *International Journal of Information Security and Privacy (IJISP)*, Volume 10:67–86, 2016.

Dalziel, H. and Crosby, S. How to Defeat Advanced Malware. Syngress, United States, 2014. ISBN 0128027533.

Davis, J. S. Dark web forums found offering Cerber Ransomware as a Service, 2016. Accessed: 05 February 2018.
URL https://www.scmagazine.com/darkwebforumsfoundofferingcerberransomwareasaservice/article/528953/

Deepti, V., Choudhary, Subrata, R., and Kumar, C. Malware Detection by Static Checking and Dynamic Analysis of Executables. *International Journal of Information Security and Privacy*, 11:29–41, 07 2017.

Dell Secureworks. 5 Ways Advanced Malware Evades the Sandbox. Technical report, Dell Secure Works, 2015. Accessed: 15 January 2017.
URL https://www.secureworks.com/blog/bg-5-ways-malware-evades-the-sandbox

Elisan, C. Malware, Rootkits & Botnets A Beginner's Guide. Beginner's Guide. McGraw-Hill Education, United States, 2012. ISBN 9780071792059.

Elisan, C. Advanced Malware Analysis. McGraw-Hill, United States, 2015. ISBN 9780071819749.

Ferrante, A., Medvet, E., Mercaldo, F., Milosevic, J., and Visaggio, C. A. Spotting the malicious moment: Characterizing malware behavior using dynamic features. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pages 372–381. Aug 2016. doi:10.1109/ARES.2016.70.

Gates, S. SSDP Amplified Attacks, a Sitting Duck against Sophisticated DDoS Analytics. 2014.
URL https://www.corero.com/blog/597-ssdp-amplified-attacks-a-sitting-duck-against-sophisticated-ddos-analytics-.html

Gibson, D. Microsoft Windows Security Essentials. EBL-Schweitzer. Wiley, 2011. ISBN 9781118114575.

Goodchild, J. and Hulme, G. What is social engineering How criminals take advantage of human behavior. Technical report, 2017. Accessed: 05 January 2018.
URL https://www.csoonline.com/article/2124681/social-engineering/what-is-social-engineering.html

Griffin, B. Locky New Malware Borrowing Ideas From Dridex and Other Ransomware. Technical report, 2016a. Accessed: 27 February 2017.
URL https://phishme.com/lockyanewencryptionransomwareborrowingideasfromthebest/

Griffin, N. Locky Ransomware - Encrypts Documents, Databases, Code, BitCoin Wallets and More. Technical report, 2016b. Accessed: 25 January 2017.
URL https://www.forcepoint.com/blog/security-labs/locky-ransomware-encrypts-documents-databases-code-bitcoin-wallets-and-more

Gross, G. Botnet CNC servers issue commands in many ways, 2015. Accessed: 05 January 2018.
URL https://www.alienvault.com/blogs/security-essentials/command-and-control-server-detection-methods-best-practices

Hasherezade. Malware Crypters the Deceptive First Layer. Technical report, 2015. Accessed: 15 January 2017.
URL https://blog.malwarebytes.com/threat-analysis/2015/12/malware-crypters-the-deceptive-first-layer/

Hasherezade. Cerber Ransomware New But Mature. Technical report, 2016a. Accessed: 15 January 2017.
URL https://blog.malwarebytes.com/threat-analysis/2016/03/cerber-ransomware-new-but-mature/

Hasherezade. Look Into Locky Ransomware. Technical report, 2016b. Accessed: 25 January 2017.
URL https://blog.malwarebytes.com/threat-analysis/2016/03/look-into-locky/

Hasherezade. Maktub Locker Beautiful And Dangerous. Technical report, MalwareBytes, 2016c. Accessed Date: 25 May 2017.
URL https://blog.malwarebytes.com/threat-analysis/2016/03/maktub-locker-beautiful-and-dangerous/

Hassan, M. Sleeping your way out of a sandbox. Technical report, SANS Security Report, United States, 2015. Accessed: 05 February 2018.
URL https://uk.sans.org/reading-room/whitepapers/malicious/sleeping-sandbox-35797

Hoffmann, C. Is UPnP a Security Risk?, 2016. Accessed: 05 February 2018.
URL https://www.howtogeek.com/122487/htg-explains-is-upnp-a-security-risk/

IBM Security. IBM X-Force Threat Intelligence Index 2018. Technical report, IBM Security, 2018.
URL https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=77014377USEN

Iliev, K. Top 6 Advanced Obfuscation Techniques Hiding Malware on Your Device. Technical report, SENSOR Tech Forum, 2017. Accessed: 05 February 2018.
URL https://sensorstechforum.com/advanced-obfuscation-techniques-malware/

Jacob, G., Comparetti, P. M., Neugschwandtner, M., Kruegel, C., and Vigna, G. A static, packer-agnostic filter to detect similar malware samples. In *Proceedings of the 9th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA'12, pages 102–122. Springer-Verlag, Berlin, Heidelberg, 2013. ISBN 978-3-642-37299-5. doi:10.1007/978-3-642-37300-8_6.

Jacobson, R. Malicious Ransomware Can Hold Computer Files Hostage, 2013. Accessed: 05 February 2018.
URL https://www.scientificamerican.com/article/ransom-malware/

Jennings, R. Just say NO to Adobe Flash Player - emergency patch vs. Cerber ransomware. 2016.
URL https://www.computerworld.com/article/3053959/malware-vulnerabilities/adobe-flash-player-cerber-ransomware-itbwcw.html

Johnson, W. How to submit a threat profile to mitre att&ck. Technical report, SANS, 2018. Accessed: 12 December 2018.
URL https://www.sans.org/cyber-security-summit/archives/file/summit-archive-1536332932.pdf

Khouzani, M. and Karyotis, V. Malware Diffusion Models for Modern Complex Networks. Morgan Kaufmann, 2016. ISBN 9780128027165.

KnowBe4. Maktub locker ransomware, 2018. Date Accessed: 31 May 2018.
URL https://www.knowbe4.com/maktub-locker-ransomware

Kumar, E. U. Abstract Stack Graph as a Representation to Detect Obfuscated Calls in Binaries. Ph.D. thesis, University of Louisiana at Lafayette, 2004.

Langendorf, S. Back to the Basics: Detecting Malicious Windows Services with Tanium, 2016. Accessed: 05 February 2018.
URL https://blog.tanium.com/Wback-basics-detecting-malicious-windows-services-tanium/index.html

Laves, J. Ransomware Proofing Your Business. *Business NH Magazine*, 35(4):14, 2018. ISSN 10469575.

Ligh, M. H., Case, A., Levy, J., and Walters, A. The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory. John Wiley & Sons, 2014. ISBN 9781118825099.

Liska, A. and Gallo, T. Ransomware: Defending Against Digital Extortion. O'Reilly Media, 2016. ISBN 9781491967850.

Malin, C., Casey, E., and Aquilina, J. Malware Forensics: Investigating and Analyzing Malicious Code. Elsevier Science, 2008. ISBN 9780080560199.

MalwareBytes. Satana ransomware Threat coming soon? Technical report, 2016. Accessed: 10 January 2017.
URL https://blog.malwarebytes.com/threat-analysis/2016/06/satana-ransomware/

MalwareTech. PowerLoader Injection - Something truly amazing. 2013. Accessed: 25 January 2017.
URL https://www.malwaretech.com/2013/08/powerloader-injection-something-truly.html

Marak, V. Windows Malware Analysis Essentials. Packt Publishing, 2015. ISBN 178528763X.

McAfee Labs. Mcafee labs threat report 2018. Technical report, McAfee Labs, 2018. Date Accessed: 11 December 2018.
URL https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-sep-2018.pdf

McNicholas, E. R. and Cunningham, T. D. A Cyberisk Grows Up : Once considered an innocuous threat, ransomware has morphed into a new, formidable risk. *Best's Review*, 118(8):59, 2017.

Meskauskas, T. Cerber Ransomware [Updated], 2017. Accessed: 05 February 2018.
URL https://www.pcrisk.com/removal-guides/9842-cerber-ransomware

Metalidou, E., Marinagi, C., Trivellas, P., Eberhagen, N., Skourlas, C., and Giannakopoulos, G. The human factor of information security: Unintentional damage perspective. *Procedia - Social and Behavioral Sciences*, 147, 08 2014. doi:10.1016/j.sbspro.2014.07.133.

Monika, Zavarsky, P., and Lindskog, D. Experimental Analysis of Ransomware on Windows and Android Platforms: Evolution and Characterization. *Procedia Science Direct*, 94:465–472, 2016.

Monnappa, K. What Malware Authors Don't Want You to Know - Evasive Hollow Process Injection, 2017. Accessed: 05 February 2018.
URL https://www.blackhat.com/docs/asia-17/materials/asia-17-KA-What-Malware-Authors-Don27t-Want-You-To-Know-Evasive-Hollow-ProcessInjection-wp.pdf

Morgan, S. 2018 Cybersecurity Market Report, 2018. Accessed: 05 February 2018.
URL https://cybersecurityventures.com/cybersecurity-market-report/

National Cybersecurity and Communications Integration Center. Using yara for malware detection. Technical report, National Cybersecurity and Communications Integration Center, 2018. Date Accessed: 12 November 2018.
URL https://ics-cert.us-cert.gov/sites/default/files/FactSheets/NCCIC%20ICS_FactSheet_YARA_S508C.pdf

Northcutt, S. Security Laboratory: Methods of Attack Series, 2017. Accessed: 05 February 2018.
URL https://www.sans.edu/cyber-research/security-laboratory/article/attacks-browsing

O'Brien, D. Ransomware 2017, 2017. Accessed: 05 February 2018.
URL https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-ransomware-2017-en.pdf

Osaghae, E. O. Packed Malware Detection using Entropy Related Analysis: A Survey. *IOSR Journal of Engineering (IOSRJEN)*, 5(11):44–45, 2015. ISSN 2250-3021.

Panda Security. Tales from Ransomwhere: Shadow Copies. Technical report, Panda Security, 2015. Accessed: 05 February 2018.
URL https://www.pandasecurity.com/mediacenter/news/tales-ransomwhere-shadow-copies/

Patterson, D. Why ransomware attacks were so powerful in 2017, 2018. Accessed: 05 February 2018.
URL https://www.techrepublic.com/article/why-ransomware-attacks-were-so-powerful-in-2017/

Proofpoint. Ransomware Survival Guide. Technical report, Proofpoint, 2017. Accessed: 10 January 2017.
URL https://www.proofpoint.com/us/resources/white-papers/ransomware-survival-guide

RadWare. WannaCry Ransomware, 2017. Accessed: 05 January 2018.
URL https://security.radware.com/ddos-threats-attacks/wannacry-ransomware/

Rascagneres, P. COM Object hijacking: the discreet way of persistence. Technical report, 2014. Accessed: 15 January 2017.
URL https://www.gdatasoftware.com/blog/2014/10/23941-com-object-hijacking-the-discreet-way-of-persistence

Riley, D. Ransomware-as-a-service to drive malware growth in 2018, 2017. Accessed: 05 February 2018.
URL https://siliconangle.com/blog/2017/11/02/ransomware-service-drive-malware-growth-2018

Rocha, L. Static Malware Analysis with malicious intent. Technical report, 2015. Accessed Date: 23 March 2017.
URL https://countuponsecurity.com/2015/02/16/static-malware-analysis-find-malicious-intent/

Roundy, K. A. and Miller, B. P. Hybrid analysis and control of malware. In *International Workshop on Recent Advances in Intrusion Detection*, doi:10.1007/978-3-642-23644-0_1., pages 317–338. Springer, 2010.

Sarokaari, N. and Dominicus, H. How to identify malicious HTTP Requests, 2012. Accessed: 05 February 2018.
URL https://www.sans.org/reading-room/whitepapers/detection/identify-malicious-http-requests-34067

Savage, K., Coogan, P., and Lau, H. The evolution of ransomware. Technical report, Symantec Corporation, 2015. Accessed: 25 January 2017.

URL http://www.symantec.com/content/en/us/enterprise/media/securityresponse/whitepapers/theevolution-of-ransomware.pdf

Segura, J. Cerber ransomware delivered in format of a different order of magnitude, 2017. Accessed: 05 February 2018.
URL https://blog.malwarebytes.com/threat-analysis/2017/08/cerber-ransomware-delivered-format-different-order-magnitude/

Shaaban, A. and Sapronov, K. Practical Windows Forensics. Packt Publishing, 2016. ISBN 9781783554102.

Sikorski, M. and Honig, A. Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. NoStarch Press, 2012. ISBN 978-1-59327-290-6.

Sison, G. Cerber Version 6 Shows How Far the Ransomware Has Come (and How Far it'll Go). Technical report, The Tech Republic, 2017. Accessed: 25 January 2017.
URL http://blog.trendmicro.com/trendlabs-security-intelligence/cerber-ransomware-evolution/

Stark Win. UPnP (Universal Plug and Play) vulnerabilities, 2016. Accessed: 05 February 2018.
URL https://hydrasky.com/network-security/upnp-universal-plug-and-play-vulnerabilities/

StatCounter . Desktop windows version market share, 2018. Accessed: 24 July 2018.
URL http://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide

Statista. Statistics and facts about ransomware. Technical report, Statista, 2018. Accessed: 11 December 2018.
URL https://www.statista.com/topics/4136/ransomware/

Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., and Thomas, C. B. Finding cyber threats with att and ck based analytics. Technical report, MITRE, 2017. Accessed: 12 December 2018.
URL https://www.mitre.org/sites/default/files/publications/16-3713-finding-cyber-threats%20with%20att%26ck-based-analytics.pdf

Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., and Thomas, C. B. Mitre att and ck design and philosophy. Technical report, MITRE, 2018. Accessed: 12 December 2018.
URL https://www.mitre.org/sites/default/files/publications/pr-18-0944-11-mitre-attack-design-and-philosophy.pdf

Symantec Corporation. 2018 Internet Security Threat Report. Technical report, Symantec Corporation, 2018. Accessed Date: 30 May 2018.
URL https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-executive-summary-en.pdf

Szmigielski, A. Bitcoin Essentials. Community experience distilled. Packt Publishing, 2016. ISBN 9781785284670.

Teller, T. and Hayon, A. Enhancing Automated Malware Analysis Machines with Memory Analysis. Technical report, 2014. Accessed Date: 18 December 2017.
URL https://www.blackhat.com/docs/us-14/materials/arsenal/us-14-Teller-Automated-Memory-Analysis-WP.pdf

The Elder Geek. System Volume Information Folder, 2017. Accessed: 05 February 2018.
URL http://www.theeldergeek.com/system1.htm

Tiamzon, M. TROJFAKEAV.SM49. Technical report, 2013. Accessed: 15 January 2017.
URL https://www.trendmicro.com/vinfo/au/threat-encyclopedia/malware/troj/_fakeav.sm49

Townsend Security. AES Encryption, 2016. Accessed: 05 February 2018.
URL https://townsendsecurity.com/sites/default/files/AES.pdf

Trend Micro. Command and Control [C&C] Server. Technical report, Trend Micro, Accessed: 10 January 2017, 2017.
URL https://www.trendmicro.com/vinfo/us/security/definition/command-and-control-server

Upchurch, J. and Zhou, X. Malware provenance: code reuse detection in malicious software at scale. In *2016 11th International Conference on Malicious and Unwanted Software (MALWARE)*, pages 1–9. Oct 2016. doi:10.1109/MALWARE.2016.7888735.

Vincent, R. Data Held Hostage: The damage inflicted in this year's ransomware attacks will force corporations to review their IT resilience., 2017. Accessed Date: 02 February 2018.
URL http://ww2.cfo.com/risk-management/2017/09/data-held-hostage/

Vlad. Understanding Locky. Technical report, 2016. Accessed Date: 27 January 2017.
URL https://sensepost.com/blog/2016/understanding-locky/

Yonts, J. Digging for Malware: Suspicious Filesystem Geography, 2014. Accessed: 05 February 2018.
URL http://www.malicious-streams.com/resources/articles/DGMW1.html

Yusirwan, S., Prayudi, Y., and Riadi, I. Implementation of Malware Analysis using Static and Dynamic Analysis Method. *International Journal of Computer Application*, 117(6):0975–8887, 2015.

Zaharia, A. Ransomware's New Turning Point: Pay up or we'll breach your data & you'll have to pay a HUGE fine! Technical report, 2016. Accessed: 10 January 2017.
URL https://heimdalsecurity.com/blog/ransomwares-new-turning-point-extortion/

Zeltser, L. Mastering the four stages of Malware analysis. Technical report, 2017. Accessed Date: 12 December 2016.
URL https://zeltser.com/mastering-4-stages-of-malware-analysis/

A

Virus Total Report

In this appendix the link to the full Virus Total reports of the three Ransomware samples that were used during the research as discussed in Chapter 3. The information from these reports were detailed in the chapter which discuss the examination finding of the research in Chapters 4, 5 and 6.

A.1 CERBER Virus Total Report

The report can be accessed on the following URL, due to the constraints on printing of the reports to maintain the integrity the link to the web page was included.

`https://bit.ly/2SvkYa2`

A.2 MAKTUB Virus Total Report

The report can be accessed on the following URL, due to the constraints on printing of the reports to maintain the integrity the link to the web page was included.

`https://bit.ly/2s9Wqba`

A.3 LOCKY Virus Total Report

The report can be accessed on the following URL, due to the constraints on printing of the reports to maintain the integrity the link to the web page was included.

`https://bit.ly/2R40DvM`

B

Mutex Created by the Ransomware

This appendix included the listings of all mutexes created by the Ransomware samples which were examined. These mutexes are discussed in the Chapters 4, 5 and 6.

B.1 CERBER Mutex Listing

```
\Sessions\1\BaseNamedObjects\Local\_!MSFTHISTORY!_
\Sessions\1\BaseNamedObjects\Local\c:!users!rgzmuyf!appdata!roaming!microsoft!windows!cookies!
\Sessions\1\BaseNamedObjects\Local\c:!users!rgzmuyf!appdata!local!microsoft!windows!history!history.ie5!
\Sessions\1\BaseNamedObjects\Local\WininetStartupMutex
\Sessions\1\BaseNamedObjects\Local\WininetConnectionMutex
\Sessions\1\BaseNamedObjects\Local\WininetProxyRegistryMutex
\Sessions\1\BaseNamedObjects\RasPbFile
\Sessions\1\BaseNamedObjects\Local\c:!users!rgzmuyf!appdata!local!microsoft!windows!temporary internet files!content.ie5!
\Sessions\1\BaseNamedObjects\Local\ZonesCounterMutex
\Sessions\1\BaseNamedObjects\Local\ZoneAttributeCacheCounterMutex
\Sessions\1\BaseNamedObjects\Local\ZonesCacheCounterMutex
\Sessions\1\BaseNamedObjects\Local\ZonesLockedCacheCounterMutex
\Sessions\1\BaseNamedObjects\Local\!IETld!Mutex
\Sessions\1\BaseNamedObjects\Local\c:!users!rgzmuyf!appdata!roaming!microsoft!windows!ietldcache!
\Sessions\1\BaseNamedObjects\shell.{1D92E976-C282-FCC7-9033-A7F49749543B}
Local\!IETld!Mutex
Local\c:!users!rgzmuyf!appdata!local!microsoft!windows!history!history.ie5!
Local\ZonesCacheCounterMutex
shell.{1D92E976-C282-FCC7-9033-A7F49749543B}
Local\ZonesLockedCacheCounterMutex
```

B.2 MAKTUB Mutex Listing

```
Local\ZoneAttributeCacheCounterMutex
Local\MidiMapper_modLongMessage_RefCnt
Local\ZonesCacheCounterMutex
\Sessions\1\BaseNamedObjects\425a6a21" "425a6a21
\Sessions\1\BaseNamedObjects\Local\MidiMapper_modLongMessage_RefCnt
\Sessions\1\BaseNamedObjects\Local\10MU_ACBPIDS_S-1-5-5-0-58053
\Sessions\1\BaseNamedObjects\Global\552FFA80-3393-423d-8671-7BA046BB5906
\Sessions\1\BaseNamedObjects\Local\ZonesCounterMutex
\Sessions\1\BaseNamedObjects\Local\ZoneAttributeCacheCounterMutex
```

\Sessions\1\BaseNamedObjects\Local\ZonesCacheCounterMutex
\Sessions\1\BaseNamedObjects\Local\ZonesLockedCacheCounterMutex
\Sessions\1\BaseNamedObjects\Global\MTX_MSO_Formal1_S-1-5-21-4162757579-3804539371-4239455898-1000
\Sessions\1\BaseNamedObjects\Global\MTX_MSO_AdHoc1_S-1-5-21-4162757579-3804539371-4239455898-1000
\Sessions\1\BaseNamedObjects\Local\10MU_ACB10_S-1-5-0-58053
Global\MTX_MSO_AdHoc1_S-1-5-21-4162757579-3804539371-4239455898-1000
Global\552FFA80-3393-423d-8671-7BA046BB5906
Global\MTX_MSO_Formal1_S-1-5-21-4162757579-3804539371-4239455898-1000

B.3 LOCKY Mutex Listing

\Sessions\1\BaseNamedObjects\MSCTF.Shared.MUTEX.b64b27e5
\Sessions\1\BaseNamedObjects\MSCTF.Shared.MUTEX.2fadcfe0
\Sessions\1\BaseNamedObjects\MSCTF.Shared.MUTEX.00fdf789
\Sessions\1\BaseNamedObjects\MSCTF.Shared.MUTEX.3e94160f
\Sessions\1\Locky Mutex ListingBaseNamedObjects\shell.{6E8CD1E8-3AA4-8152-A1AC-9DF81B4CF52F}
\Sessions\1\BaseNamedObjects\MSCTF.Shared.MUTEX.a8bdbd11
\Sessions\1\BaseNamedObjects\MSCTF.Shared.MUTEX.0e44d0c4
\Sessions\1\BaseNamedObjects\MSCTF.Shared.MUTEX.710dbf32
\Sessions\1\BaseNamedObjects\shell.{C1BB8BB5-D12C-7249-81F4-7F24325F8B14}

C

CERBER Configuration file

C.1 Targeted Folders

```
\\$recycle.bin\\",
\\$windows.~bt\\",
\\boot\\",
\\drivers\\",
\\program files\\",
\\program files (x86)\\
\\programdata\\
\\users\\all users\\
\\windows\\
\\windows.old\\
\\appdata\\local\\
\\appdata\\locallow\\
\\appdata\roaming\\adobe\flash player\\
\\appdata\roaming\\ati\\
\\appdata\roaming\\google\\
\\appdata\roaming\\identities\\
\\appdata\roaming\\installshield\\
\\appdata\roaming\\intel\\
\\appdata\roaming\\macromedia\flash player\\
\\appdata\roaming\\media center programs\\
\\appdata\roaming\\microsoft\\
\\appdata\roaming\\mozilla\\
```

```
\\appdata\roaming\nvidia\\
\\appdata\roaming\\opera\\
\\public\\music\\sample music\\
\\public\\pictures\\sample pictures\\
\\public\u000bideos\\sample videos\\
\tor browser\\
```

C.2 Geolocation on Victim Machine

```
"ip_geo": [
{"property_name": "country",
"url": "http://ipinfo.io/json"},
{"property_name": "country_code",
"url": "http://freegeoip.net/json/"},
{"property_name": "countryCode",
"url": "http://ip-api.com/json"}
```

C.3 Server Statistics

```
"servers":
{"statistics":
{"knock": "hi{PARTNER_ID}",
"data_finish": "{MD5_KEY}"
"data_start": "{MD5_KEY}{PARTNER_ID}{OS}{IS_X64}{IS_ADMIN}{COUNT_FILES}{STOP_REASON
    }",
"ip": "31.184.234.0/23",
"port": 6892,
"send_stat": 1,
"timeout": 255
```

C.4 Hardcoded Global Public Key

```
},   "global_public_key": "LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0KTUlJQklqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCQ2dLQ0
FRRUF2a3R5NXFocUV5ZFI5MDc2RmV2cAowdU1QN0laTm1zMUFBN0dQUVVUaE1XYllpRVlJaEJLY1QwL253WXJCcTBPZ3Y3OUsxdHRhMDRFSFRyWGdjQXA
vCk9KZ0JoejlONThhZXdkNHlaQm0yY29lYURHdmNHUkFjOWU3Mk9iRlEvVE1FL0lvN0xaNXFYRFd6RGFmSThMQTgKSlFtU3owTCsvRytMUFRXZzdrUE9wS
lQ3V1NrUmI5VDh3NVFnWlJKdXZ2aEVySE04M2tPM0VMVEgrU29FSTUzcAo0RU5Wd2ZOTkVwT3BucE9PU0tRb2J0SXc1NkNzUUZyaGFjMHNRbE9qZWsvbXV
WbHV4amlFbWMwZnN6azJXTFNuCnFyeWlNeXphSTVEV0JEallLWEExdHAyaC95Z2JrWWRGWVJiQUVxd3RMeFQyd01mV1BRSTVPa2hUYTl0WnFEMEgKblFJRE
FRQUIKLS0tLS1FTkQgUFVCTElDIEtFWS0tLS0tCg==",   "help_files": {     "files": [
```

C.5 Blacklisted Countries List

am Armenia
az Azerbaidjan
by Belarus
de Germany
kg Kirgistan
kz Kazachstan

md Moldavia
ru Russian Federation
tm Turmenistan
tj Tadjikistan
ua Ukraine
uz Uzbekistan

D

Import Functions for Locky

Function	Description
CreateDirectoryW	Creates a new directory. If the underlying file system supports security on files and directories, the function applies a specified security descriptor to the new directory.
CreateFileA	Creates or opens a file or I/O device. The most commonly used I/O devices are as follows: file, file stream, directory, physical disk, volume, console buffer, tape drive, communications resource, mailslot, and pipe. The function returns a handle that can be used to access the file or device for various types of I/O depending on the file or device and the flags and attributes specified.
CreateFileMappingW	Creates or opens a named or unnamed file mapping object for a specified file.
CreateFileW	Creates or opens a file or I/O device. The most commonly used I/O devices are as follows: file, file stream, directory, physical disk, volume, console buffer, tape drive, communications resource, mailslot, and pipe. The function returns a handle that can be used to access the file or device for various types of I/O depending on the file or device and the flags and attributes specified.
CreateProcessAsUserW	Creates a new process and its primary thread. The new process runs in the security context of the user represented by the specified token.
CreateProcessW	Creates a new process and its primary thread. The new process runs in the security context of the user represented by the specified token.
CreateServiceW	Creates a service object and adds it to the specified service control manager database.
CreateThread	Creates a thread to execute within the virtual address space of the calling process.
CreateToolhelp32Snapshot	Takes a snapshot of the specified processes, as well as the heaps, modules, and threads used by these processes.
DeleteFileW	Deletes an existing file.
DeviceIoControl	Sends a control code directly to a specified device driver, causing the corresponding device to perform the corresponding operation.
ExitThread	Ends the calling thread.
FindFirstFileW	Searches a directory for a file or subdirectory with a name that matches a specific name (or partial name if wildcards are used).
FindNextFileW	Determines the location of the resource with the specified type, name, and language in the specified module.
FindResourceExW	Determines the location of the resource with the specified type, name, and language in the specified module
FindResourceW	FindResourceW Determines the location of a resource with the specified type and name in the specified module.
GetCommandLineW	Retrieves the command-line string for the current process.
GetDriveTypeW	Determines whether a disk drive is a removable, fixed, CD-ROM, RAM disk, or network drive.
GetFileAttributesW	Retrieves file system attributes for a specified file or directo
GetFileSize	Retrieves the size of the specified file, in bytes.
GetFileSizeEx	Retrieves the size of the specified file.
GetModuleFileNameA	Retrieves the fully qualified path for the file that contains the specified module. The module must have been loaded by the current process.
GetModuleFileNameExW	Retrieves the fully qualified path for the file containing the specified module.
GetModuleFileNameW	Retrieves the fully qualified path for the file that contains the specified module. The module must have been loaded by the current process.

Function	Description
GetModuleHandleA	Retrieves a module handle for the specified module. The module must have been loaded by the calling process.
GetModuleHandleW	Retrieves a module handle for the specified module. The module must have been loaded by the calling process.
GetProcAddress	Retrieves the address of an exported function or variable from the specified dynamic-link library (DLL).
GetStartupInfoA	Retrieves the contents of the STARTUPINFO structure that was specified when the calling process was created.
GetStartupInfoW	Retrieves the contents of the STARTUPINFO structure that was specified when the calling process was created.
GetTempFileNameW	Creates a name for a temporary file. If a unique file name is generated, an empty file is created and the handle to it is released; otherwise, only a file name is generated.
GetTempPathW	Retrieves the path of the directory designated for temporary files.
GetTickCount	Retrieves the number of milliseconds that have elapsed since the system was started, up to 49.7 days
GetVersionExW	GetVersionEx may be altered or unavailable for releases after Windows 8.1. Instead, use the Version Helper functions
GetWindowThreadProcessId	Retrieves the identifier of the thread that created the specified window and, optionally, the identifier of the process that created the window.
IsDebuggerPresent	Determines whether the calling process is being debugged by a user-mode debugger.
LoadLibraryA	Loads the specified module into the address space of the calling process. The specified module may cause other modules to be loaded.
LoadLibraryExA	Loads the specified module into the address space of the calling process. The specified module may cause other modules to be loaded.
LoadLibraryExW	Loads the specified module into the address space of the calling process. The specified module may cause other modules to be loaded.
LoadLibraryW	Loads the specified module into the address space of the calling process. The specified module may cause other modules to be loaded.
LockResource	Retrieves a pointer to the specified resource in memory.
MapViewOfFile	Maps a view of a file mapping into the address space of a calling process.
OpenFileMappingW	Opens a named file mapping object.
OpenProcess	Opens an existing local process object.
OpenProcessToken	The OpenProcessToken function opens the access token associated with a process.
Process32FirstW	Retrieves information about the first process encountered in a system snapshot.
Process32NextW	Retrieves information about the next process recorded in a system snapshot.
ReadProcessMemory	Reads data from an area of memory in a specified process. The entire area to be read must be accessible or the operation fails.
RegCloseKey	Closes a handle to the specified registry key.
RegCreateKeyExW	Creates the specified registry key. If the key already exists, the function opens it. Note that key names are not case sensitive.
RegDeleteKeyW	Deletes a subkey and its values. Note that key names are not case sensitive.
RegEnumKeyExW	Enumerates the subkeys of the specified open registry key. The function retrieves information about one subkey each time it is called.
RegOpenKeyExW	Opens the specified registry key. Note that key names are not case sensitive.

Function	Description
`RegOpenKeyW`	Opens the specified registry key.
`Sleep`	Suspends the execution of the current thread until the time-out interval elapses
`StartServiceW`	Starts a service.
`TerminateProcess`	Terminates the specified process and all of its threads.
`UnhandledExceptionFilter`	An application-defined function that passes unhandled exceptions to the debugger, if the process is being debugged. Otherwise, it optionally displays an Application Error message box and causes the exception handler to be executed. This function can be called only from within the filter expression of an exception handler.
`VirtualAlloc`	Reserves, commits, or changes the state of a region of pages in the virtual address space of the calling process. Memory allocated by this function is automatically initialized to zero.
`VirtualProtectEx`	Changes the protection on a region of committed pages in the virtual address space of a specified process.
`WriteFile`	Writes data to the specified file or input/output (I/O) device.