Evaluation of Live Forensic Techniques in Ransomware Attack Mitigation

Simon Rhys Davies

Submitted in partial fulfilment of
the requirements of Edinburgh Napier University
for the degree of Master of Science in
Advanced Security and Digital Forensics

Edinburgh Napier University

School of Computing

December 2019

Submission Checklist

Milestones	Date of Completion	Target Deadline
Proposal	Week 7	Week 7
Initial Report	Week 13	Week 13
Full Draft	week 10	Week 10
Final Submission	Week 12	Week 14

Learning Outcome	**The Markers will assess**	**Pages**	**Hours Spent**
Learning Outcome 1 Conduct a literature search using an appropriate range of information sources and produce a critical review of the findings.	•Range of materials and quality and identification/understanding of the quality of sources; list of references •The critical literature review/exposition/background information chapter •Critical analysis and discussion of the related work, and highlighting of key findings •Does the work/key findings inform the method (LO2)	6	400[1]
Learning Outcome 2 Demonstrate professional competence by sound project management and (a) by applying appropriate theoretical and practical computing concepts and techniques to a non-trivial problem, or (b) by undertaking an approved project of equivalent standard.	•Evidence of project management (Gantt chart, diary, etc.) •Depending on the topic: chapters on design, implementation, methods, experiments, results, etc. •Method justification from literature/applying appropriate method, and typically solid scientific method	90 25,41, 55 25	200

[1] Includes time spent on research area that was later rejected

Learning Outcome 3 Show a capacity for self-appraisal by analysing the strengths and weakness of the project outcomes with reference to the initial objectives, and to the work of others.	•Chapter on evaluation •Assessing your outcomes against the project aims and 'objectives'. A good things to do is align your objectives against the main chapters in your dissertation – lit review, design, implement/experiment/results, evaluation and then you can copy and rewrite your conclusions from these chapters! •Discussion of your project's output compared to the work of others. With reference to other sources found in the literature review/-comparison against other work/how this has built on previous work •Future work – how might we take forward. If student carried out work again how might they have been done things differently	55 75 75 78	125
Learning Outcome 4 Provide evidence of the meeting learning outcomes 1-3 in the form of a dissertation which complies with the requirements of the School of Computing both in style and content.	• Is the dissertation well-written (academic writing style, grammatical), spell-checked, free of typos, neatly formatted. • Does the dissertation contain all relevant chapters, appendices, title and contents pages, etc. • Style and content of the dissertation.		200

Learning Outcome 5 Defend the work orally at a viva voice examination.	• Performance • Confirm authorship	1 Hour

Have you previously uploaded your dissertation to Turnitin?	Yes/No
Has your supervisor seen a full draft of the dissertation before submission?	Yes/No
Has your supervisor said that you are ready to submit the dissertation?	Yes/No

Authorship Declaration

I, Simon Rhys Davies, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

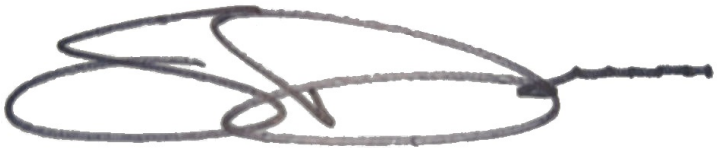
I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained **informed consent** from all people I have involved in the work in this dissertation following the School's ethical guidelines.

Signed: Simon Rhys Davies



Date: 1^{st} December 2019

Matriculation no: 40290841

General Data Protection Regulation Declaration

Under the General Data Protection Regulation (GDPR) (EU) 2016/679, the University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please write your name below one of the options below to state your preference.

☑ The University may make this dissertation, with indicative grade, available to others.

☐ The University may make this dissertation available to others, but the grade may not be disclosed.

☐ The University may not make this dissertation available to others.

Abstract

Ransomware continues to grow in both scale, cost, complexity and impact since its initial discovery nearly 30 years ago. Security practitioners are engaged in a continual "arms race" with the ransomware developers attempting to defend their digital infrastructure against such attacks. Recent manifestations of ransomware have started to employ a hybrid combination of symmetric and asymmetric encryption to encode user's files.

This report describes an investigation to determine if the techniques currently employed in the field of digital forensics could be leveraged to discover the encryption keys used by these types of malicious software.

A safe, isolated virtual environment was created and ransomware samples were executed within it. Memory was captured from the infected system and its contents was examined using three different live forensic tools in an attempt to identify the symmetric encryption keys being used by the ransomware. NotPetya, BadRabbit and Phobos ransomware samples were were tested during the investigation on two different operating systems. The samples were chosen as they were recent, high profile attacks generating significant ransom payments and causing serious disruption to many organisations.

If keys were discovered, the following two steps were also performed. Firstly, a timeline was manually created to show when the keys were present in memory and how long they remained there. Secondly, an attempt was made to decrypt the files encrypted by the ransomware using the found keys. In all cases the investigation was able to confirm that it was possible to discover the encryption keys used and these found keys successfully decrypted files that had been encrypted by the ransomware samples.

No research was found that conducted cryptographic key examination specifically on ransomware using live forensic techniques, however research was found that investigated other types of cryptographic programs. The results of this investigation matched similar findings from these related research fields, as the keys used by the cryptographic programs were successfully recoved and used to decrypt the files.

The ransomware time lining also highlighted different key manage-

ment processes used by these ransomware programs, where some tended to leave the key in memory for the whole execution while others practiced more dynamic key management.

Keywords: Ransomware, Live Forensics

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my thesis advisor Rich MacFarlane of the School of Computing at Edinburgh Napier University. Both the physical and virtual door to Rich's office was always open whenever I ran into a trouble 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. Throughout the project he continually guided me while challenging my assumptions, proving to be an invaluable interlocutor allowing this project to become far deeper and more complete than I could have ever hoped for. For this I am extremely grateful.

Chapter 1

Introduction

The aim of this project was to investigate if the techniques commonly used in live forensics can be applied to the analysis of ransomware and in so doing allow the investigator to discover useful cryptographic fragments from the malware that could later be used to reverse the effects of the ransomware's execution.

The techniques described in this report are not designed to specifically identify the presence of ransomware on a machine, but rather to provide remedial techniques that could be employed to mitigate an on-going attack.

1.1 Background

While ransomware first appeared more than 30 years ago (Salvi, 2015), its initial impact on the computing community was small with only a few people being affected and recovery from the attack being trivial.

The threat landscape changed in 2013 with the release of the CryptoLocker ransomware (Bradley, 2016) where attackers adopted the three new technologies of crypto currency, TOR onion routing and cryptography. Combining them to produce a new breed of ransomware programs that have become more effective and aggressive than anything previously experienced. These new sophisticated attacks have generated large amounts of money for the perpetrators. Culminating in two of the biggest ransomware attacks in recent times, WannaCry which is estimated to have cost $8 billion and NotPetya which is estimated to have cost $10 billion(Mekynyk, Speier-Pero & Connors, 2019).

The number of malware attacks is generally considered to be growing year on year (Europol, 2016; Intelligence & Analysis, 2019). It was observed that the majority of research into ransomware could be broken down in to three distinct areas. One concentrating on the detection of ransomware attacks using techniques such as machine learning or neural networks. A summary of the most recent work performed in this area is presented by Al-rimy (Al-rimy, Maarof & Shaid, 2018) and describes the current techniques used and anticipated future direction of this research area. The second area of ransomware research

was found to concern itself with dissecting the mechanics of an attack and the techniques used by the ransomware program for encryption, persistence, infection and propagation. Prime examples of this type of research being the work performed by Akkas (Akkas, Chachamis & Fetahu, 2017) into the analysis of the WannaCry or Sai (Sai & Kumar, 2019) for the NotPetya ransomware families. The final area of research being prevention techniques such as education, phishing identification (Gupta, Arachchilage & Psannis, 2018), social engineering and other techniques found by the researchers to be effective in preventing ransomware from entering the system (F-Secure Labs, 2016).

There exists a separate research field concerning itself with the study and development of live forensics techniques and more specifically with live memory analysis. Some research performed in this field has been into the recovery of cryptographic fragments, specifically encryption keys, from the contents of the systems memory where cryptographic processes are active. A lot of this research has proven very successful (Balogh & Pondelik, 2011; Maartmann-Moe et al., 2009) allowing the researchers to determine the encryption keys used by the cryptographic programs and then using them to decrypt files. However no specific research has been found where these techniques have been applied to machines that have ransomware active on them.

The current advice to users who discover that they are under a ransomware attack is that they should switch off their machine immediately (Sittig & Singh, 2019). However if the user follows this advice then the critical data in memory, such as encryption keys, would be unrecoverable. The findings of this investigation may suggest that another course of action could possibly be considered. These alternative solutions may provide techniques that could be used to recover affected files, without the user having to pay the requested ransom.

1.2 Aims and objectives

The aim of this project is to determine if the techniques currently employed in live forensics could be applied to the mitigation of the affects of a ransomware attack. More specifically if the inspection of the live memory, of a system where ransomware is currently running, could provide information that could be used to reverse the impact of the ransomware program.

To achieve this the following sub aims have also been identified:

1. Conduct a critical literature review on the subject of ransomware and live forensics. Included in this would be an analysis of the current status of ransomware attacks, trends, techniques for delivery, general steps in-

volved in such an attack as well as live forensic techniques relating to memory analysis with regards to cryptanalysis.

2. Develop and validate a robust, isolated, realistic test environment that can be used to execute the designed experiments.
3. Building on the findings from the literature review, design and develop specific experiments to test the hypothesis that live forensic techniques may be used in mitigating the affects of a ransomware attack. Using three different detection tools on three examples of ransomware. These experiments being run on two different operating systems.
4. Discuss and evaluate the results and findings from the experiments and compare them to similar research on the subject. Draw conclusions from the experiments, results and findings and critically evaluate the project.

1.3 Ethical compliance

Dealing with malware, and ransomware in particular should be performed with the utmost care. Adequate safeguards were incorporated in to all experiments in order to prevent the accidental dissemination of the code from the test environment. By its very nature, these types of programs are designed to try and bypass security and network controls and attempt to propagate themselves to other machines. So particular care was taken in designing and implementing a safe test environment ensuring that no accidental leak of the code could occur.

Criminal laws and professional codes of conduct also stipulate how a computer scientist should work. The British Computer Society's code of conduct (BCS, 2015) states that the computer researcher should:

> *"have due regard for public health, privacy, security and well being of others and the environment"*

And

> *"avoid injuring others, their property, reputation, or employment by false or malicious or negligent action or inaction"*

In the UK, the researcher is also under the jurisdiction for the Computer Misuse act (Spiritual, 2017) which stipulates in section 3 that the researcher should not perform:

> *"Unauthorised acts with intent to impair, or with recklessness as to impairing, operation of computer, etc."*

Bearing these laws and guidelines in mind, the management and execution of the ransomware samples was performed with great care an attention. Steps taken during the project to comply with these laws and guidelines were:

1. The execution of the sample was only performed in a virtual environment that contained no external connections.
2. The virtual environment used during the experiments was hosted on a machine that itself had no external connections. This provided a double isolation and an "air gap" to the test machine.
3. The test environment and its host machine were deleted and the hard disk reformatted after completion of the tests.
4. Files were transferred to the test machine via a removable USB stick, which was reformatted after each transfer.
5. Only encrypted document files were transferred from the test machine.
6. No personal or private data was present on the test machines.
7. The test machines, snapshots, infected files and ransomware samples were deleted on completion of the project.
8. An up-to-date anti-virus program was running on the host machine.
9. Documented proof that these safe guards were performed.

It is believed that the implementation of these safe guards provide sufficient protection from the intentional execution of these ransomware programs and demonstrate that no malicious intent existed by the researcher in executing them.

1.4 Structure

The remainder of this document is structured as follows:

Chapter 2 - A literature review in to ransomware. Specifically focusing on the techniques used for delivery, infection process, general steps in ransomware execution, techniques for prevention, current status and future trends. A secondary literature review in to live forensic techniques specifically focusing on the acquisition of memory fragments relating to cryptanalysis and the recovery of symmetric keys from memory.

Chapter 3 - Describes the philosophy guiding the design of the test methodology developed, including the environment and the experiments that will be

performed within it. This section also covers the criteria used during the selection of hardware and software packages.

Chapter 4 - Presents the reasoning used in the ransomware sample selection. It goes on to describe the implementation of the experiments performed together with the results achieved when these experiments were executed.

Chapter 5 - Critical analysis of the experimental results and comparison to similar work in the field. Also contains critical analysis of the techniques and methods employed.

Chapter 6 –Discussion on the findings from the literature review and how they relate to the experiments performed in this work. A critical review of the overall project, design and implementation highlighting key finding and identifying areas of strength and weakness. The section concludes with suggestions of future work that could be performed.

Chapter 2

Literature Review

2.1 Introduction

The aim of this project was the evaluation of live forensic techniques in ransomware attack mitigation. The following literature review is structured as follows. It begins with a definition of ransomware and the main classifications used when describing it. The review then continues to describe the history of ransomware attacks and the current status, demonstrating that they remain a real and serious threat. The first section of the literature review ends with a description of the main phases of a ransomware attack.

The second part of the literature review focuses on live forensic techniques and how other researchers have used these techniques to gather cryptographic information. This is followed by a discussion of how the experiments were designed and the methods used by the researchers. The review concludes with a critical discussion of the findings and what aspects of previous research can be leveraged in this project.

2.2 Ransomware definition

Ransomware is one of the most widespread and damaging threats that internet users face today (Sophos, 2019). In its simplest form the definition of ransomware is a "malicious program that after its execution prevents a user from accessing their data". These programs are classified under a broader family of programs known as malware which also include viruses, worms, Trojans, botnets, spyware and rootkits (Grégio, Afonso, Filho, Geus & Jino, 2014).

More specific definitions are also provided by (Salvi, 2015) where they expand and broaden the definition to be "a program that prevents you from using your computer or your documents whether stored locally or in the cloud, locking them with unbreakable encryption until a ransom in bitcoin is paid". While this definition is more specific, not all ransomware programs use bitcoin as the payment method, or are able to reverse the encryption used. Also some ransomware programs attack other elements of the computer system such at

the Master Boot Record (MBR) which is used in the management of the file system.

Morato (Morato, Berrueta, Magaña & Izal, 2018) try to improve the definition by categorising the different behaviours of ransomware in to different classifications. For example if the program just prevents access to the machine it is referred to as lockscreen ransomware, where as if access to the files are prevented by the use of encryption then these programs are referred to as encryption ransomware, crypto ransomware or cryptoware. However they do not expand these definitions to include ransomware that contain a combination of both behaviours and programs that masquerade as ransomware, but in fact are not. For example programs that encrypt the data with no intention of being able to decrypt it even if the ransom is paid are sometimes referred to as wipers (O'Brien, 2017) as they effectively permanently remove access to the user's files.

The types of encryption used by the ransomware is also used to classify ransomware types (Al-rimy et al., 2018). The main classifications are show below:

Symmetric Crypto-Ransomware (SCR) uses, as the name suggests one key for both encryption and decryption which allows the attack to complete in a shorter time, reducing the chances of it being discovered. SCR can be implemented by several symmetric key algorithms, such as Advanced Encryption Standard (AES), Data Encryption Standard (DES), and Rivest Cipher 4 (RC4) (Kong, Ang & Seng, 2015), however this type of cryptography has some limitations when it comes to keeping the keys secret (Sihim, 2016).

Asymmetric Crypto-Ransomware (ACR) uses different keys for encryption and decryption, using techniques such as RSA and is more robust but slower than symmetric encryption. ACR-based crypto-ransomware is more likely to be able to survive decryption attempts (Al-rimy et al., 2018). Having said that, public key cryptography is not an ideal solution as it is possible that after one victim has paid the ransom and received the private key for decryption, they could then theoretically distribute it to all other affected users, who could then use it to recover their own files. To avoid such a situation, crypto-ransomware authors generate lists of private keys, one for each victim, which makes it quite difficult (but not impossible) for these unique private keys to be shared with all victims. However this turn raises a second problem of key management for the attacker.

Hybrid Key Crypto-Ransomware (HCR) combines the best of both the previous classes of ransomware into a hybrid solution. Firstly it uses symmetric encryption to encrypt the user's files as fast as possible. After which the symmetric key is encrypted using asymmetric encryption. The AES key is encrypted using

the attackers public key that is delivered with the ransomware software. The ransom message will ask the victim to send the encrypted key, together with payment, back to the attacker, whereby the AES key is extracted by the attacker using their own private key and the AES key is sent back to the victim allowing them to decrypt their files.

There is currently no definition for ransomware which also takes in to account the psychological aspect of ransomware relating to the extortion message. This extortion is imposed by exploiting victim's fear of losing valuable data, revealing sensitive information or locking key resources (Al-rimy et al., 2018) .

2.2.1 History of ransomware

It is generally agreed (Furnell & Security, 2017; Salvi, 2015) that the first recognised ransomware was the AIDS (Aids Info Desk) Trojan, released in 1989. This program was propagated using floppy disks and once a machine was infected, it would encrypt files and render the machine unusable. The user then needed to pay $189 to recover their files. The payment being processes via a cheque sent to a bank in Panama.

Although proposed by Young (Young & Yung, 1996) in 1996, the use of strong encryption in attacks didn't gain popularity until the mid 2000's when GPCode was released (Furnell & Security, 2017). GPCode represented the first real world implementation of the schemes proposed by Young and Yung; encrypting disk content and demanding ransom payment. Many variants of GPCode contained flaws including poorly implemented encryption routines, insecure encryption keys, or poor file deletion strategies, which allowed recovery of deleted content; significantly however, GPCode continued to evolve, its deletion strategies became stronger and the encryption schemes and key lengths improved over time (Hampton & Baig, 2015).

2013 was the breakthrough year for ransomware attacks (Bradley, 2016). CryptoLocker streamlined the ransomware process by adding the ability to pay the ransom using an electronic payment method called Bitcoin. This efficiency in collection meant that CryptoLocker was able to collect an estimated $5 million dollars during four months in 2013. This ransomware combined elliptic curve cryptography, anonymous onion routing using the Tor network and bitcoins and gave rise to the CTB acronym (Curve,Tor,Bitcoin) which is used to describe many forms of ransomware today. The use of these technologies further obfuscate the perpetrator of the attack making them even more difficult to identify or trace (Savage, Coogan & Lau, 2015).

Technically capable cyber criminals then developed and enhanced ransom-

ware techniques over the coming years moving from SCR and ACR to the more robust HCR encryption solutions.

Occasionally, the implementation of a ransomware strain has contained a programming or design error which has allowed researchers to reverse engineer the code and develop techniques to recover systems. For example in the CryptoDefense attack (Symantec, 2014), the developers left copies of the encryption keys used on the file system. These mistakes are however uncommon and the general consensus for mitigating such attacks remains to use up to date firewalls and take regular backups (Europol, 2018). There have been no recorded incidents where the actual encryption algorithm used has been broken, so currently as long as it is implemented correctly it remains a robust approach.

The explosion of ransomware attacks can easily seen in a diagram produced by F-Secure (F-Secure, 2017) and shown below in Figure 2.1. This explosion has been attributed to several factors including the fact that Ransomware as a Service (RaaS) became available, ransomware development kits and guides could be purchased and code from previous strains were also available for purchase (Sultan, Khalique, Alam & Tanweer, 2018).

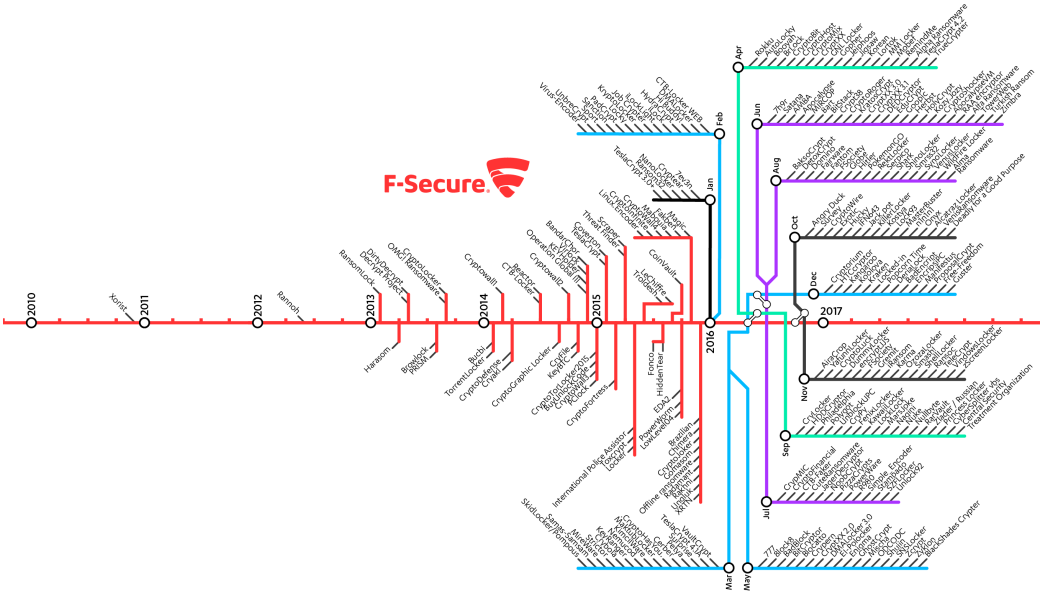


Figure 2.1: F-Secure explosion of Ransomware. (F-Secure, 2017)

Notable examples of attacks being Cryptowall in 2013 which used a fake campaign on an advertising network for propagation, CTB locker using bitcoin for payments in 2014, Chimera in 2015 which threatened to leak the files if the

ransom was not paid, Petya which was non recoverable, jigsaw and Zcrypt in 2016 which used horror film references to further distress the victim, WannaCry and NotPetya in 2017 which have netted the largest payouts and Ryuk in 2018 designed to specifically target enterprise environments.

A time line of the most financially successful ransomware attacks is provided by Vanderburg (Vanderburg, 2019) and shown in the Figure 2.2 below, with just the last two strains alone netting in excess of $18 billion (Kapersky, 2018).

Figure 2.2: Notable Attacks. (Vanderburg, 2019)

The most recent ransomware variants are using HCR for each infection and supposedly wiping the session key from memory after usage. As discussed earlier, the advantages of using HCR becoming apparent as the attack completes in a significantly shorter time period increasing the likelihood that it is undetected. They also use privacy-enabling services, such as Tor, and favour bitcoins for payment. Making it virtually impossible to trace the attackers or recover the affected files without paying the ransom (Savage et al., 2015).

2.2.2 Current status and future trends

While historically the incidents of ransomware have been increasing year on year prompting Interpol to declare in 2016 that ransomware had become “the most prominent malware threat [...] for citizens and enterprises alike” (Europol, 2016) a later report from 2019 (2019 Malwarebytes LABS, 2019) has indicated

that there was a in fact a decrease of 26% in detected cases during 2018. This is however contradicted by a report by Sonicwall(Intelligence & Analysis, 2019) who claim that they have witnessed a 11% increase during this time frame. Reasons for these difference in recorded incidents could be attributed to a couple of factors:

1. Attackers have shifted focus from private individuals towards enterprises (Malwarebytes, 2019; Symantec, 2019) and the authors of these reports may be gathering their statistics from different areas of the industry (Davies, 2019).

2. Reduction in available online ransomware exploit kits available for ransomware development (Europol, 2018).

However, this decline is predicted to reverse dramatically in 2019 (Malwarebytes, 2019) with the data from the first quarter showing a 500% year on year increase in attacks and the trend upwards is set to continue (Levy & Cto, 2019). Europol confirming in 2018 that they believe that ransomware will retain its dominance for several years to come (Europol, 2018).

The 'Cyber Security Breaches Survey 2019' indicated that 12% of businesses had experienced a ransomware attack in the past 12 months making it the fourth most frequent encountered category of identified breaches. (Klahr, Amili, Shah, Button & Wang, 2019) and Microsoft confirming that the threat is an evolving menace (Microsoft, 2017).

Even at the time of writing American government agencies are struggling with the affects of a recent ransomware attack (O'Donnall, 2019) and US authorities are preparing for similar attacks during the voter registration for the 2020 elections (Hautala, 2019).

These reports of increased activity are corroborated by the latest malware statistics for 2019 (Purplesec, 2019) shown in Figure 2.3 below:

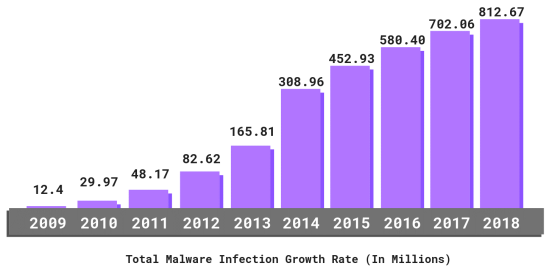


Figure 2.3: Malware Infection Growth Rate (Purplesec, 2019)

The damage from the NotPetya cyberattack is estimated at $10 billion, whereas WannaCry, according to various estimates, lies in the $4–$8 billion range. NotPetya is considered the costliest global cyberattack in history (Kapersky, 2018).

2.2.3 Ransomware infection path

According to Wang (Wang & Wang, 2015), ransomware follows the same approach as traditional malware when exploiting a system's vulnerabilities. For example email attachments or compromised websites are leveraged as attack vectors in order to infect a victims computer. Several different organisations have proposed models that attempt to describe the behaviour of ransomware. Many of which suggest that the classification of a ransomware attack can be divided up in to six distinct phases (Al-rimy et al., 2018; Sultan et al., 2018; Eric Vanderberg, 2018). An example of one such proposal is the model published by McAfee (McAfee Labs, 2016) shown in Figure 2.4.

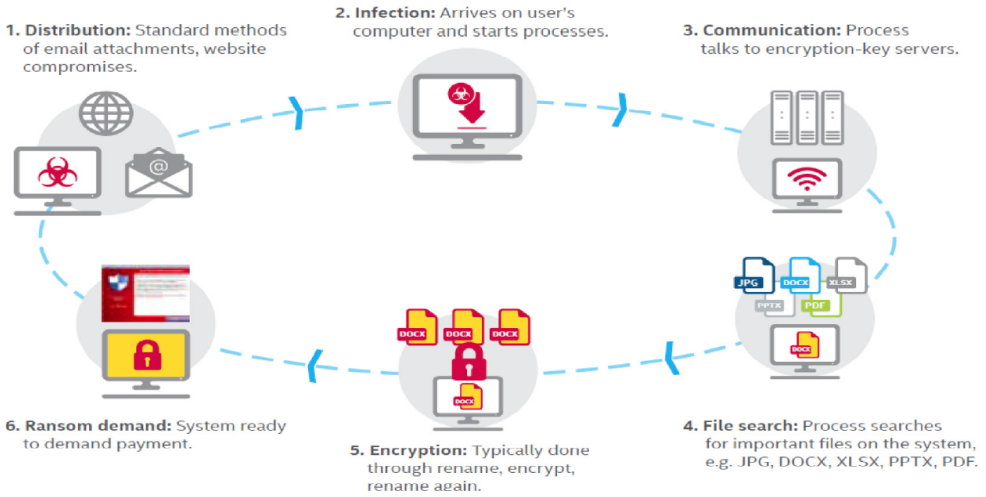


Figure 2.4: McAfee 6 Phase Ransomware Model.(McAfee Labs, 2016)

Distribution - This phase deals with the packing of the ransomware and delivery of the malicious code to the victims system. Different exploitation techniques are employed to facilitate the delivery such as phishing attacks, spam mail campaigns, social engineering, infiltration and drive by download.

Infection - This phase deals with the execution of the ransomware and its initial behaviour. During this phase the software explores the running environment and collects information about the victim's device, such as platform type, OS version, and installed programs. (Prakash, Nafis & Sankar Biswas, 2017)

Communication – During this phase the ransomware retrieves the encryption keys from the command and control (C&C) server. This is an external machine controlled by the attacker and is used to supply the encryption keys and execution modules required at a later stage of the infection. Not all ransomware perform this task as some variations have the keys already attached to its payload (Sgandurra, Muñoz-González, Mohsen & Lupu, 2016). Some ransomware will also try and pivot to other machines on the network during this phase.(Sophos, 2019)

File Search – During this phase the ransomware starts looking for targeted resources such as user's files, resources, and accessibility functions. Files are normally selected based on their file extensions (Sultan et al., 2018)

Encryption - Based on its family type, the ransomware starts hijacking the targeted resources found in the previous phase and locks and/or encrypts these resources (Mbol, Robert & Sadighian, 2016; Paik, Shin & Cho, 2016). Depending on its family type, if the attack is a wiper (e.g. Petya), then the encryption key is not handled in any way and just thrown away. For other families (e.g. Wannacry, NotPetya), then the key is communicated back to the attacker via the C&C server or via the ransom message. As highlighted earlier, the majority of modern ransomware programs are now a hybrid key crypto-ransomware (HCR) type using a combination of encryption techniques to improve performance. Files maybe moved or renamed and backups may also be spoiled during this phase (Klein, 2017). The machine may reboot and also encrypt the MBR. If the ransomware has been successfully covert during its execution, the reboot of the machine maybe be the first time a user is aware of its presence.

Ransom demand - Once the encryption process completes, a message is shown to the victim demanding a ransom accompanied by payment instructions. Paying the ransom does not guarantee that a decryption key would be sent, but it does increase the chances of data recovery (Richardson & North, 2017).

This model was used during the evaluation phase of this project as the fine granularity of each steps was useful in being able to capture specific information concerning a ransomware's execution. However not everyone is in agreement regarding the number of phases that should be included in an infection model. Ahmadian(Ahmadian, Shahriari & Ghaffarian, 2016), Kumar(Kumar & Kumar, 2013) and Gazet (Gazet, 2010) prefer a simpler, less granular three phase model that is more generalised and easier to adapt to specific instances of ransomware behaviour. It is worth noting that no model was found which includes a phase for the user paying the ransom and hopefully recovering their files.

An example of a ransomware attack is provided by Microsoft (Microsoft De-

fender ATP Research Team, 2017) is shown in Figure 2.5.

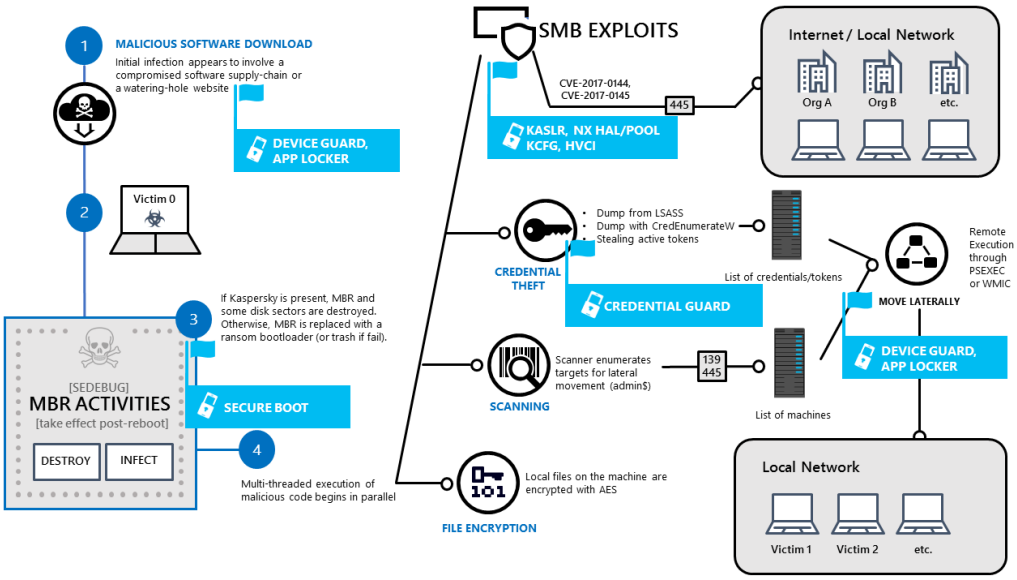


Figure 2.5: NotPetya Attack Phases.(Microsoft Defender ATP Research Team, 2017)

The diagram describes the attack process of the NotPetya ransomware family and highlights the difference phases that the ransomware attack goes through including infecting the MBR, lateral movement to other machines on the network by exploiting SMB connections or credential harvesting, before finally encrypting the victims files and displaying the ransom note.

2.3 Live forensics

Static forensic analysis methods are used in analysing evidence from a computer system that has been turned off. The problem with this approach is that significant information stored in the computers volatile memory is lost when the machine is switched off. Examples of information that could be present in memory are encryption keys, open connection details, running processes, logged in users etc.(Bashir & Khan, 2013)

To address this issue a complimentary forensics approach has also been developed know as live forensics. Live forensic analysis primarily targets the computers volatile data which can only be collected from a running system; hence, the term "live". Live forensics is a methodology to extract memory, system processes and network related information from a running system helping to

understand the overall state of the system under investigation.(Bashir & Khan, 2013). When applied to ransomware analysis, the live forensics techniques can be complimented by combining them with malware analysis techniques. These techniques can be generally divided in to two main categories static analysis and dynamic analysis.

2.3.1 Static analysis

This is a passive technique where the code of a suspicious piece of software is examined, without it being executed. The structure of the code is analysed and attempts are made to determine if its purpose is malicious.(Wang & Wang, 2015; Zhang & Tan, 2015). This form of analysis is safe to perform and will not negatively affect the system where it is done. A lot of information may be gathered using this technique for example execution paths and file access(Galal, Mahdy & Atiea, 2016). However this technique is unable to work with programs that have been packed or obfuscated (Choudhary & Vidyarthi, 2015) and can be ineffective against sophisticated programs(Sai & Kumar, 2019) or programs that have their modules encrypted until just before they are required for execution (Malwarebytes, 2018). Advanced forms of this technique require deep knowledge and understanding of OS concepts, machine code and assembly languages (Sai & Kumar, 2019)

2.3.2 Dynamic analysis

This approach actually executes the malicious code in an isolated controlled environment such as a sandboxed or virtual environment and the programs activities and interaction with the system are analysed (Kaur & Singh, 2014). The dynamic approach is more effective in identifying the actual intent of the program under scrutiny, as it observes what the malicious code does rather than what it looks like. Likewise, employing dynamic analysis contributes to detection of previously unknown malicious code variants, based on the general behavioural signature of the ransomware family(Al-rimy et al., 2018). Using this technique it is possible to obtain information that is difficult to gather using other methods(Sai & Kumar, 2019).

2.3.3 Memory acquisition

One important aspect of live forensics is the examination of the systems memory where the malicious code is running. This examination is normally performed off-line so that the contents of the memory are not affected by the examination

or the memory capture tools used. To achieve this, the memory of the system to be examined needs to be captured and saved.

According to (Ruff, 2008) there are three main memory capturing techniques:

1. Software-based, which typically involves executing extraction programs. An issue with this approach being that the execution of software would impact the contents of the captured systems memory.
2. Hardware-based, which typically involves connecting devices, such as PCMCIA cards or USB sticks and are not always practical in live scenarios as physical access to the machine is required (McLaren, Russell, Buchanan & Tan, 2019).
3. Virtualization technology-based techniques.

The memory acquisition process is especially unstandardised, and different researchers have used different methods (Maartmann-Moe et al., 2009). A detailed compilation of the techniques available is provided in (Carvey & Casey, 2009) along with advantages and disadvantages of each approach.

Using the virtualization approach, a snapshot of the analysed system's volatile memory is extracted using tools provided by the virtualization software. This snapshot is then inspected by an analyst using a variety of specialised forensic tools(Nissim, Lahav, Cohen, Elovici & Rokach, 2019). Obviously to use this technique the system must be running in a virtualised environment. The advantages of this approach being that no trace of any extraction program exists in the captured memory and any running malicious programs are unaware that they are being analysed or that the memory dump was taken (Ligh, Case, Levy & Walters, 2014; Dinaburg, Royal, Sharif & Lee, 2008; McLaren, Buchanan, Russell & Tan, 2019).

The challenge of memory acquisition in this context is to discover cryptographic artefacts, such as the encryption keys, in a manner that allows the target device to continue to operate normally, while the memory is being acquired (McLaren, Russell et al., 2019).

2.4 Cryptanalysis live forensics

The idea of extracting cryptographic material from a large body of plain text was first discussed in 1998 (Shamir & Van Someren, 1998) and was framed in the concept of a "lunchtime attack". The scenario discussed in this type of side channel attack focuses on the concept of a time restricted attack window, such

as a user leaving their machine unlocked while they are away from their desk, for instance while going for lunch.

The paper discusses the ability to quickly and efficiently identify cryptographic key material in large amounts of data. The researchers did not inspect volatile memory at this time rather they focused on locating asymmetric RSA keys on a plain text disk. Although such keys are often encrypted when stored on disk, the idea of exploiting the cryptographic properties of the keys to efficiently find them was rather novel. They used simple statistical and visual methods to locate regions that are likely to contain encryption keys (Maartmann-Moe et al., 2009). In a more recent article it was proposed to use the structural properties of the code to identify the cryptographic keys (Pettersson, 2007), however this technique is not realistic as it would require access to the source code of the cryptographic program for it to be applicable.

Some work has been previously performed into the possibility of using dynamic analysis techniques to discover encryption keys that may be present in a computers memory. It was not possible to find any literature that focused specifically on ransomware in particular, however similar work on key determination in volatile memory has been performed for SSH tunnels, encrypted volumes, WinRAR, WinZip and Skype (Maartmann-Moe et al., 2009; Balogh & Pondelik, 2011; McLaren, Russell et al., 2019).

2.4.1 Keys present in memory

Several research papers confirm the assumption that for a system to be able to encrypt/decrypt data, then the cryptographic algorithm needs to have access the encryption keys and these are normally held in volatile memory. Balogh (Balogh & Pondelik, 2011) state that encryption in real-time is only performed in memory which means that the encryption keys must also be present there. So in the case of symmetric encryption it means that the keys also needed for decryption will also be recoverable from memory.

When discussing the TrueCrypt and BitLocker software Hargreaves (Hargreaves & Chivers, 2008) states that these packages only decrypt content as it is required. Therefore the keys needed for decryption have to be continuously accessible and should always be recoverable from memory.

With regards to key management Maartmann-Moe (Maartmann-Moe et al., 2009) state that it is clear that cryptographic keys need to be present in memory during encryption when using standard computer hardware.(Maartmann-Moe et al., 2009). It is recommended by these researchers that keys that reside in memory while the application is running should be purged the moment it terminates.

When examining encryption keys for ssh tunnels in virtual environments McLaren (McLaren, Russell et al., 2019) suggest that if timely acquisition of target device memory is obtained, the stream or artefacts may be discovered, enabling the compromise of secure communications. Similarly, in extensive tests conducted on 10 different cryptographic systems the researchers (Maartmann-Moe et al., 2009) were always able to retrieve all the cryptographic keys from memory for every application tested using their specifically developed tool called 'interrogate'. While neither of these researchers have investigated ransomware in particular, their findings strongly indicate that it would be possible to extract ransomware cryptographic keys using similar techniques.

2.4.2 Examination of memory methods

The usual process for locating something is to try to identify some characteristic of what is being located and then to look for that characteristic. One characteristic of cryptographic keys is that they are usually chosen at random. Most code and data is not chosen at random and it turns out that this differentiation is significant (Shamir & Van Someren, 1998). When data is random it has higher entropy than patterned information that is not random. This means that it should be possible to locate cryptographic keys among other data by locating sections with unusually high entropy (Balogh & Pondelik, 2011). The authors found that the block where the main and the auxiliary ASE keys are located has a recognizable structure and high entropy.

One technique investigated by Hargreaves(Hargreaves & Chivers, 2008) was to treat the memory as a large blob of bytes and sequentially traverse through it looking for candidate keys, which seemed to be an overly time intensive approach. Memory is in fact quite structured containing known patterns and hierarchies (Maartmann-Moe et al., 2009). In reality, symmetric cryptographic keys are just short sequences of random looking data, often 16–32 bytes long residing amongst other pieces of data with a much lower entropy.

Another technique investigated, when looking for encryption keys for the BitLocker program was to search for the keyword "External Key" or a hidden read only file with the extension ".BEK" in the memory image (Saravanan & Mukesh, 2014). While this technique does work for this product, it is too specific for the application to ransomware as their keys do not have any such identifying titles or labels. This, like any other method that relies on access to the source code to determine a keys features or structures would not be useful when dealing with ransomware (Balogh & Pondelik, 2011) as the structure of this code would be unknown. Techniques that focus on the cryptographic keys properties and attributes have proven to be a more successful approach for identifying cryp-

tographic keys in memory for ransomware.

2.4.3 Identifying keys in memory

In order to extract encryption keys from memory, they must first be identified (Halderman et al., 2009). To do this, techniques needs to be developed that can identify the keys. Several researchers have discovered (Halderman et al., 2009; Ptacek, 2008; Shamir & Van Someren, 1998), that encryption keys in memory are far more structured than previously believed, several strategies to locate the keys have been proposed and are discussed below.

A brute-force approach using the memory image as a dictionary (Kaplan, 2007; Hargreaves & Chivers, 2008). In this approach sequential segments of memory are taken and then used as the key to determine if it decrypts the data. If not the next segment of memory is read and the process repeats until the end of the memory is reached. This method does not require any understanding of the memory structure and does not use any logic to improve the performance. This type of approach for ransomware is obviously unrealistic due to the large amount of work that would be required to check all the data as candidate keys.

Using a high entropy searching approach as suggested by Shamir (Shamir & Van Someren, 1998) and tested by (Maartmann-Moe et al., 2009). Key entropy or randomness, means that the sequence of bits cannot be easily predicted and can be evaluated using Shannon's entropy measure for discrete variables (Shannon, 1948). Since it is known that key data has more entropy than non-key data, one way to locate a key is to divide the data into small sections, measure the entropy of each section and display the locations where there is particularly high entropy (Shamir & Van Someren, 1998). There is no real need to calculate the exact entropy value of a section as the entropy of most program code is extremely low and it is a relatively simple task to identify encryption keys of high entropy amongst this data. This technique is also program agnostic with no knowledge of the programs structure being required. Having the large concentration of entropy in the key data makes it easy to identify amongst many other types of memory regions(McLaren, Russell et al., 2019).This approach should have a better performance than the brute force approach as only certain specific memory segments are tested for valid keys.

Try and search for identifiable structural properties of the encryption program (Pettersson, 2007; Walters & Petroni, 2007; Klein, 2017). As mentioned above this is also not realistic as access to the source code or significant reverse engineering of the program would be required.

Search for certain known patterns in the memory such as key schedule which

are specific to certain types of encryption (Halderman et al., 2009; Ptacek, 2008). These patterns could also consist of known memory offsets, specific lengths of high entropy memory locations, or known patterns of entropy, as discussed below. This approach could in theory provide the best performance as it would result in the lowest number of candidate keys and these would have the highest probability of being correctly identified.

2.4.4 Identifying AES keys

From the literature review it has been found that the majority of modern crypto ransomware are now hybrid in nature (HCR) (Al-rimy et al., 2018) using both symmetric and asymmetric encryption. The public key of the asymmetric encryption being delivered with the ransomware, while the private key is retained by the attacker. As the private key of the asymmetric encryption is never present on the machine, this investigation will concentrate on the identification of the key used during the symmetric encryption phase of the ransomware's execution which in the majority of cases is the Advanced Encryption Standard (AES) key.

The Rijndael cipher was selected as the Advanced Encryption Standard (AES) in 2001 (NIST, 2001) formed from a proposal by Joan Daemen and Vincent Rijmen. It is a Substitution-Permutation (SP)-network based cipher that works on 128-bit blocks, and can use either 128, 198 or 256 bit keys. AES is widely used, fast and is regarded as the de-facto standard in most new cryptographic applications. It is considered by some of the researchers that AES is virtually unbreakable (Saravanan & Mukesh, 2014) and impossible to decrypt without the correct key (Maartmann-Moe et al., 2009). AES encryption is present in a vast range of applications, among others TrueCrypt, Vista BitLocker, OS X FileVault, BestCrypt, PGP, Protect- Drive and Pointsec. (Maartmann-Moe et al., 2009).

Modern symmetric key cryptosystems are constructed by repeatedly applying a simpler function where several iterations or "rounds," are done. From the master key, a derivation function, derives different sub-keys in each round. This is known as the key schedule algorithm and has been stated by many researchers that this information needs to be present in memory (Balogh & Pondelik, 2011; Hargreaves & Chivers, 2008; Maartmann-Moe et al., 2009). This knowledge of the cryptosystem can be used to search for this key pattern within the memory (Balogh & Pondelik, 2011). The search criteria being that there is a mathematical relationship between the master key and sub-keys, and the key schedule is often computed ahead of time, in what appears to be a security-performance trade-off, and kept in the memory while encryption/decryption is performed

(Maartmann-Moe et al., 2009). An example of an 128-bit empty AES key (all zeros) and its associated key schedule (Maartmann-Moe et al., 2009), extracted from memory is shown below in Figure 2.6.

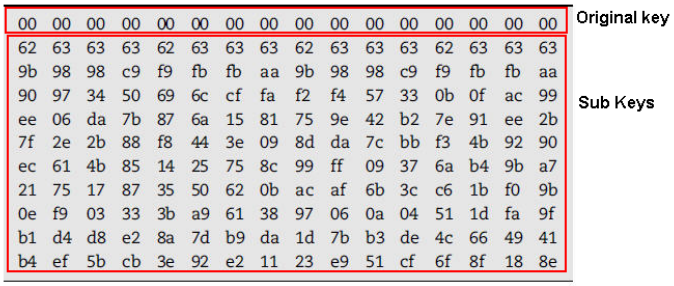


Figure 2.6: AES Key and Key Schedule (Maartmann-Moe et al., 2009)

Notably, the key schedule for a 128 bit AES key is represented as a flat array of bytes in memory, where the first 16 bytes (or 128 bits) constitutes the original key. The remaining 160 bytes are the round keys derived from this key. For larger AES keys, the corresponding key schedule is also larger.

Balogh et al. propose that there are two major approaches as to how the encryption keys can be located in memory or memory image (Balogh & Pondelik, 2011):

- By looking for the keys at a fixed address in memory - it uses the fact that all installations of an application usually contain the password/key in the same fixed location in the memory. However for each kind of encryption package investigated this address would have to be determined somehow.
- By locating a particular pattern in the memory placed always in the constant distance from the encryption keys and then searching for this pattern.

Several tools have been developed that use these criteria to identify AES keys in memory for example AESFinder has been developed (Halderman et al., 2009; Heninger & Feldman, 2008) , Volatools (Walters & Petroni, 2007), interrogate (Maartmann-Moe et al., 2009) and Findaes (Kornblum, 2019). The AESFinder and interrogate tools were even able to identify and recreate AES keys that had some errors, by using the information held in the key schedule.

2.4.5 Method

Several of the papers reviewed have used the same basic experimental method during their evaluation of their hypotheses,(Walters & Petroni, 2007; Hargreaves & Chivers, 2008; Maartmann-Moe et al., 2009; Balogh & Pondelik, 2011; Nissim et al., 2019).

The fundamentals of their methods being firstly running the program under investigation in a virtual environment, then using tools from the virtual environment to capture memory dumps of the systems memory in a secured and trusted manner. Once the required number of memory captures has been completed then they are analysed.

All the papers reviewed, relating to the discovery of cryptographic information in volatile memory were successful in extracting the encryption keys.

Halderman states that the keys used to encrypt the disk were found to reside in RAM, in scheduled form, for as long as the disk is mounted (Halderman et al., 2009). McLaren was able to confirm that in each experiment, encryption keys were discovered and valid plain text produced (McLaren, Russell et al., 2019) and memory analysis identifies cryptographic artefacts with 100% success (McLaren, Buchanan et al., 2019). Balogh successfully recovered encryption keys from a memory dump of a live system (Balogh & Pondelik, 2011). Maartmann-Moe(Maartmann-Moe et al., 2009) confirm that cryptographic systems that pre-compute key schedule have all been found to be vulnerable to key schedule searches also stating that the chances of locating encryption keys are surprisingly high.

One thing to remember when applying this method to the ransomware samples analysed in this report is that these samples perform several fundamental steps before they actually begin encrypting data on the victims system (Nissim et al., 2019). So unlike the programs investigated by other researchers the ransomware encryption keys will not be immediately present in the memory when the programs starts executing and determining when to capture the memory becomes critical to the success of the experiments. Examples of steps performed prior to encryption could be a scanning phase, in which it scrutinizes which files it should encrypt (e.g., documents) and which files it should avoid encrypting (e.g., system files). When considering the phases of a ransomware attack (Al-rimy et al., 2018), ideally the memory should be captured after phase 4 has completed and when a ransomware starts encrypting files in phase 5. The encryption process can last from several minutes to a few hours, and the program may perform good key management on the completion of encryption by removing the keys from memory, so determining the point when the memory

capture should be performed is crucial.

2.4.6 Issues

Some of the techniques discussed have some limitations with regards to their application to ransomware. For example some tools required that the key schedule keys being stored in consistent patterns in memory (Balogh & Pondelik, 2011) or were ineffective if the memory contained even a small amount of errors. Some techniques required access to the source code to identify key structures (Hargreaves & Chivers, 2008) but on the whole the tools used and the results achieved proved encouraging.

2.5 Conclusion

This chapter presents the findings from the literature review conducted firstly on ransomware and then on live forensic techniques. It begins with a brief history of ransomware after which it moves on to discuss the current threat landscape and expected future trends of this type of attack. The review confirms from multiple sources that ransomware attacks remains a real and current threat to computer systems and that their impact, complexity and relevance is projected to grow in the foreseeable future (Europol, 2018). While there is some discussion regarding the trajectory of growth during 2018 (2019 Malwarebytes LABS, 2019), the difference of opinion can be explained by the different sample sets used by the researchers. Some researchers proposing that the focus of the attacker is moving more from private individuals towards enterprise targets(Symantec, 2019). There is little doubt for 2019 that the number of incidents will be the highest so far recorded (Malwarebytes, 2019; Levy & Cto, 2019).

The methods of ransomware infection is then discussed together with a description of two different models that can be used to describe ransomware behaviour. The six phase model (McAfee Labs, 2016) with quite high granularity was preferred by the author due to the ability for this type of model to capture more precise details of the programs execution as apposed to a more generalised three stage model (Kumar & Kumar, 2013). The six phase model is used throughout the report as an aid in describing the behaviour of a ransomware sample. Techniques, approaches and terminology from this model were also used during the work relating to timelining of ransomware execution. Analysis of ransomware samples using this six step model also aided in the work performed during ransomware sample selection. The model provided a techniques that allowed different samples behaviour to be compared to each other

and aided in the identification of samples that exhibited the preferred behaviour. Guidance when selecting the ransomware samples was also derived from this literature review as the researcher wished to use recent, high profile samples that had a wide spread well documented impact and be familiar to a wider audience.

The chapter then moves on to cover the subject of live forensics, starting with a discussion of the different terminology and techniques used in this field. Classification of ransomware based on the encryption techniques were presented and it was identified that most of the modern crypo ransomware families were hybrid in nature (HCR) using both symmetric and asymmetric encryption during an attack. Due to this and the fact that the private key for the asymmetric encryption is never present on the target machine, it was proposed to focus on the identification of the symmetric key and more specifically the AES key. Approaches that can be used to identify AES keys in memory were described, two of which were "entropy" and "key schedule" searching. These would be used at later stages in the project.

While no specific research in to using dynamic analysis techniques to combat ransomware were found, several key papers describing work using these techniques to recover artifacts from other cryptographic programs were presented (Maartmann-Moe et al., 2009; Balogh & Pondelik, 2011; Shamir & Van Someren, 1998). Information gathered from these papers featured heavily in the design and implementation of this reports environment and experiments. Some of the tools identified in these papers were also tested during the experimentation phase, achieving similar success rates as the original papers.

The literature review presented in this chapter containing the background to the subject area, why this research remains relevant, examples of comparable research in related fields and how this can be applied to investigate the hypothesis of determining if live forensic techniques can by used to mitigate a ransomware attack, aligns perfectly with the learning outcome 1 in the marking scheme.

Chapter 3

Design

3.1 Introduction

The chapter starts with a determination of what the most applicable research methodology to use would be, to fulfil the projects objectives, ensuring that the conducted experiments have a methodical, rigorous and well based approach (Edgar & Manz, 2017a). The chapter then moves on to apply the findings made in the literature review to the design and development of specific experiments to validate the hypothesis on the usefulness of live forensic techniques in the mitigation of the affects of a ransomware attack. The chapter also describes how the findings from the literature review were used to design and develop an appropriate test environment that can be used to execute these experiments safely.

3.2 Research methodology

The process of science has evolved with the goal of instilling confidence in what can be learnt from observation (Edgar & Manz, 2017a). With the majority of research methods having the following characteristics:

- Providing a rigorous and methodical approach to study.
- Providing a process to empirically ground theories and conceptual models.
- Ensuring that the evidence is driven by logical and reasoned thinking.
- Continually challenging the approach and results found.

There are three main forms of research methods:

Observational - The phenomenon of interest is embedded in a larger system that is dynamic. The investigator can seek instances where the dynamics are less noisy, but it's not possible to conduct an experiment free of influences from uncontrolled or uncontrollable variables. The vast majority of science research

engages in some form of observational experiments, using simplifications to gain understanding.

Mathematical - is based upon logic and formal proofs.

Experimental - The investigator has full control of the phenomenon being observed and the mechanisms for data collection. All of the variables are known and can be either held constant or made to change in order to assess the consequences of those changes on the phenomenon of interest.

The research carried out in this project falls in to the experimental classification and more specifically when following the guidelines of Edgar (Edgar & Manz, 2017a) this turns out to be a **Hypothetico-deductive** research project. This type of research covers what is largely considered the traditional scientific method; experimentation. Experimentation is one of the strongest methods available to understand the behaviour and response of a system under varied conditions. In general, hypothetico-deductive experimentation is good for eliminating or reinforcing prior knowledge about a system.

This project employed the generally accepted six step scientific method (Khan Academy, 2017; Lin, 2019) which is an abstract simplification of the process that a researcher goes through when performing this type of research. The six steps and how they were applied to this project is shown below:

1. Purpose.
2. Research. Conduct background research and literature reviews in to the field to determine what is already known about the subject. The outcome of this being presented in Chapter 2.
3. Hypothesis. Propose a hypothesis which predicts the outcome of the experiment. Outlined in Chapter 1.
4. Experiment. Design and perform experiments that test the hypothesis. The outcome of this step appearing in Chapters 3 and 4.
5. Data analysis. Record the observations and analyse the meaning of the data. A full analysis of the results is given in Chapter 5.
6. Conclusions. Conclude whether to accept or reject the proposed hypothesis.This being done in Chapter 6. It is possible that no conclusive findings are made and further investigation maybe required.(Khan Academy, 2017).

3.3 Environment design

3.3.1 Safe environment proposal

For both safety and ethical reasons (BCS, 2015), it is imperative that any research performed on malware analysis be conducted in a controlled and responsible manner and that the systems used have sufficient safeguards in place to reasonably prevent any damage , infection or propagation of the malicious code. A safe environment will allow analysis of the program without exposing the machine or networks to unnecessary risk (Sikorski & Hong, 2012).

When designing a test environment the following points have been suggested by Sanabria (Sanabria, 2007) regarding its content:

- Simplicity. The environment should be as complex as necessary but no more. The more complex an environment is the harder it is to maintain.
- Containment is paramount when designing an environment for testing malware and can be thought of as the safety net when control is lost.
- A flexible, stable environment that is easy to start and revert.

3.3.2 Technology options available

In order to conduct valid, realistic malware experiments the test victim's machine needs to be as close to a real machine as possible with any private or confidential information removed(Hoopes, 2009) and ideally isolated from the internet(Ahmad, Woodhead & Gan, 2016).

The four main methods used to construct realistic test environments are discussed below:

Dedicated physical systems with no wifi or cable connection and being isolated from the network. This approach is sometimes referred to as an 'air gapped' system (Sikorski & Hong, 2012). However this configuration can be fairly laborious to revert back to its original state after testing.

Simulation systems use tools and processes to imitate and model a real network environment. Simulation is an accepted and widely used technology however it does do not scale well and is resource intensive (Ahmad et al., 2016).

Emulation systems are generally better than simulation systems and scale better. They are often used by AnitVirus programs when testing samples (Egele, Scholte, Kirda & Kruegel, 2012).

Virtualisation systems use the technique of separating resources and services from the underlying physical platform to form an environment with virtual machines and other network infrastructure (Ahmad et al., 2016). They exhibit better performance than emulation systems, provide strong isolation of resources and can be reverted quickly (Hoopes, 2009)

3.3.3 Software selection

When dealing with malware samples that do not actively interrogate for the presence of virtualisation, a popular and flexible way to create a malware analysis lab involves using virtualization software. This approach uses a single physical computer for hosting multiple virtual systems, each running a potentially different operating system (Zeltser, 2015). This approach has long been recognised as an efficient, isolated substitute for real physical machines (Goldberg, 1974). Increased flexibility is also provided by the snapshot functionality which can be used to quickly revert virtual machines back to their original pristine state. It has been shown that this approach provides benefits on cost, flexibility and network isolation (Sanabria, 2007) when compared to the alternatives.

Free virtualization software options considered for this project include

- VirtualBox
- VMWare vSphere Hypervisor
- Microsoft Virtual Server

Based on the research performed by Bose (Bose, 2018), the experience of the researcher and the cost it was decided to implement the test environment for this project using the VirtualBox virtualization software provided by Oracle (Virtualbox, 2019).

Testing in a virtual machine (VM) that is isolated from the host device, as well as isolated from the production network, ensures that a security analyst can execute malware safely and repeatably in a manner that yields the most accurate test results (Carvey, 2018)

3.3.4 Environment design

When designing a test environment, one of the key recommendations from Rossow (Rossow et al., 2012) was that it should be as realistic as possible. Using these guidelines as well as the three points raised by Sanabria (Sanabria, 2007), the test environment was designed to contain three virtual machines.

Two of these virtual machines were victim test machines. While previous researchers (Maartmann-Moe et al., 2009; Shamir & Van Someren, 1998) have used test machines running the windows operating system, the versions that they used were outdated. It was decided that the test machines in this investigation would use the Windows 7 and Windows 10 operating systems as these have the highest market penetration (Statista, 2019). Only one of these victim machines was ever active at any given time and it is on these where the ransomware was executed.

In all but the most basic experiments at least one other system is required, providing network support services (Sanabria, 2007) for the victim, as denying all-network access to the sample under analysis will most likely result in incomplete observations of the malware's behaviour (Egele et al., 2012). Therefore a common technique (Sikorski & Hong, 2012) in malware analysis was used where a third virtual machine was present on the virtual network to provide network services to the victim machines such as DNS, IRC, HTTP as well as handling possible requests made by the malware back to its command and control (C&C) server (Sanabria, 2007).

These machines were connected via a 'host-only' virtual network connection, and they were the only machines on this virtual network. This configuration provides complete isolation of these guest machines from the host machine and thus the host's network connections. The physical host machine consisted of a laptop which itself was "air gapped," thus providing a second layer of isolation.

To prevent the victim virtual machine from appearing too sterile, it contained an assortment of content spread across its file system. This pseudo content was sourced from GovDocs (Open Preservation Soceity, 2019) which is a large collection of approximately 1 million documents that are freely available for research. Having both network access and content on the victim virtual machine contributed significantly to it appearing to the malware as a real machine, encouraging the malware to behave normally. Appropriate containment policies such as firewall and anti-virus were also deployed on the host machine (Rossow et al., 2012).

3.4 Experiment design

In its most abstract level the designed experiments could be considered as follows. A ransomware sample is executed within a virtual environment. During this execution, copies of the machines volatile memory are taken. Forensic

tools are then used to analyse these captured memory files in an attempt to discover the encryption key. The found keys are then used to decrypt the files and confirm that the correct key was identified.

Using the terminology proposed by Edgar (Edgar & Manz, 2017b) when describing a hypothetical-deductive research experiment, the memory dump files and decrypted files are all "dependent variables". These are variables that are expected to show if there is an effect from the intervention applied in an experiment. Analysis of these variables will allow the researcher to confirm or disprove the hypothesis. The virtual machines, ransomware samples and forensic tools are all under the control of the researcher and remain fixed during the experiment and are all examples of "independent variables". These types of variables are inputs to the system and have the potential to cause an effect to the dependent variables. The found keys are dependent variables in experiment-1 and independent variables in experiment-3.

A high level graphic representation of the combined experiments is shown in Figure 3.1.

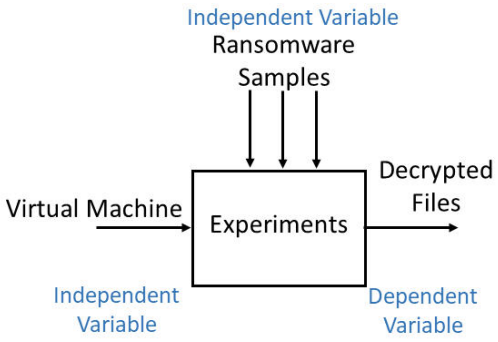


Figure 3.1: Experiment Overview

This investigation can be broken down in to three separate sub experiments. The results of which, when combined were used to validate or disprove the hypothesis that live forensic techniques could be used to mitigate a ransomware attack. Each round of experiments began with a fresh version of a virtual machine that reflected a realistic workstation being started and an example of the ransomware under investigation being executed. The following three experiments were then performed.

3.4.1 Experiment 1 - Is the key in memory

It was found during the literature review that most modern crypto ransomware programs are hybrid (HCR) in nature using both symmetric and asymmetric encryption. The public key being delivered with the ransomware, while the private key is retained by the attacker (Al-rimy et al., 2018). As the private key of the asymmetric encryption is never present on the machine, these experiments concentrated on the recovery of the key used during the symmetric encryption phase of the ransomware's execution. A memory dump from the machine where the ransomware was being executed was captured. It was known during selection of the ransomware samples for these experiments that AES was being used for the symmetric part of the encryption, so once the dump had been completed it was analysed using live forensics tools to determine if the AES key can be discovered. A graphic representation of this and the following experiment is shown in Figure 3.2.

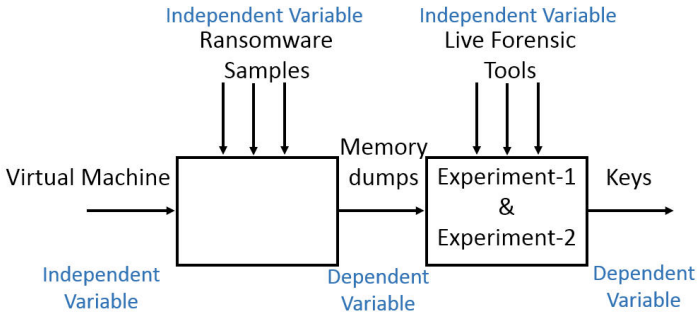


Figure 3.2: Overview of Experiment-1 & Experiment-2

3.4.2 Experiment 2 - How long is the key present

A memory dump was taken at regular intervals during the complete execution life cycle of the ransomware to determine at what point the key is loaded into memory and for how long it remains there. This aids the execution of experiment-1 by determining when to execute point A in Figure 3.4 and can also be used to determine when the execution of the ransomware enters phase 5 of the 6 phase ransomware model discussed in the literature review(McAfee Labs, 2016). An outcome of this experiment was an approximate timeline for the execution of the ransomware.

3.4.3 Experiment 3 - Does the key decrypt files

If any keys were discovered during experiment-1, they were tested to determine if they could decrypt any of the control files encrypted by the ransomware.

As indicated by points B and C in Figure 3.4. A tool developed by the author using the Python programming language was used to perform the decryption attempt on both the Windows 7 and Windows 10 memory dumps. An example of the code for this decryption program can be found in Appendix B.2.

A graphic representation of this experiment is shown in Figure 3.3.

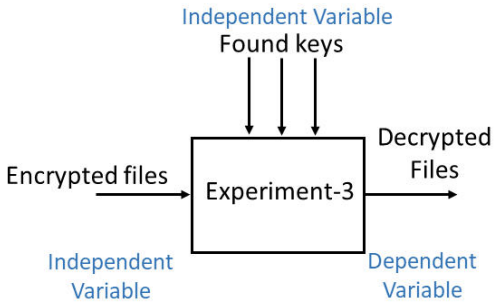


Figure 3.3: Overview of Experiment-3

3.4.4 Combined experiment

The overall experimental suite was based on similar methods identified in the literature review (Walters & Petroni, 2007; Maartmann-Moe et al., 2009; Hargreaves & Chivers, 2008; Balogh & Pondelik, 2011; Nissim et al., 2019). An overall representation of the steps involved in these experiments is shown below in Figure 3.4. While the designed experiments in this investigation have significant similarities with previous work such as using windows operating systems on virtual machines and using similar tools for key extraction. These experiments differ in that multiple key extraction tools on multiple operating systems were tested and the discovered keys were checked to confirm that they decrypted the files. Also multiple memory dumps were taken during the experiment to allow for the creation of a timeline of the ransomware's execution.

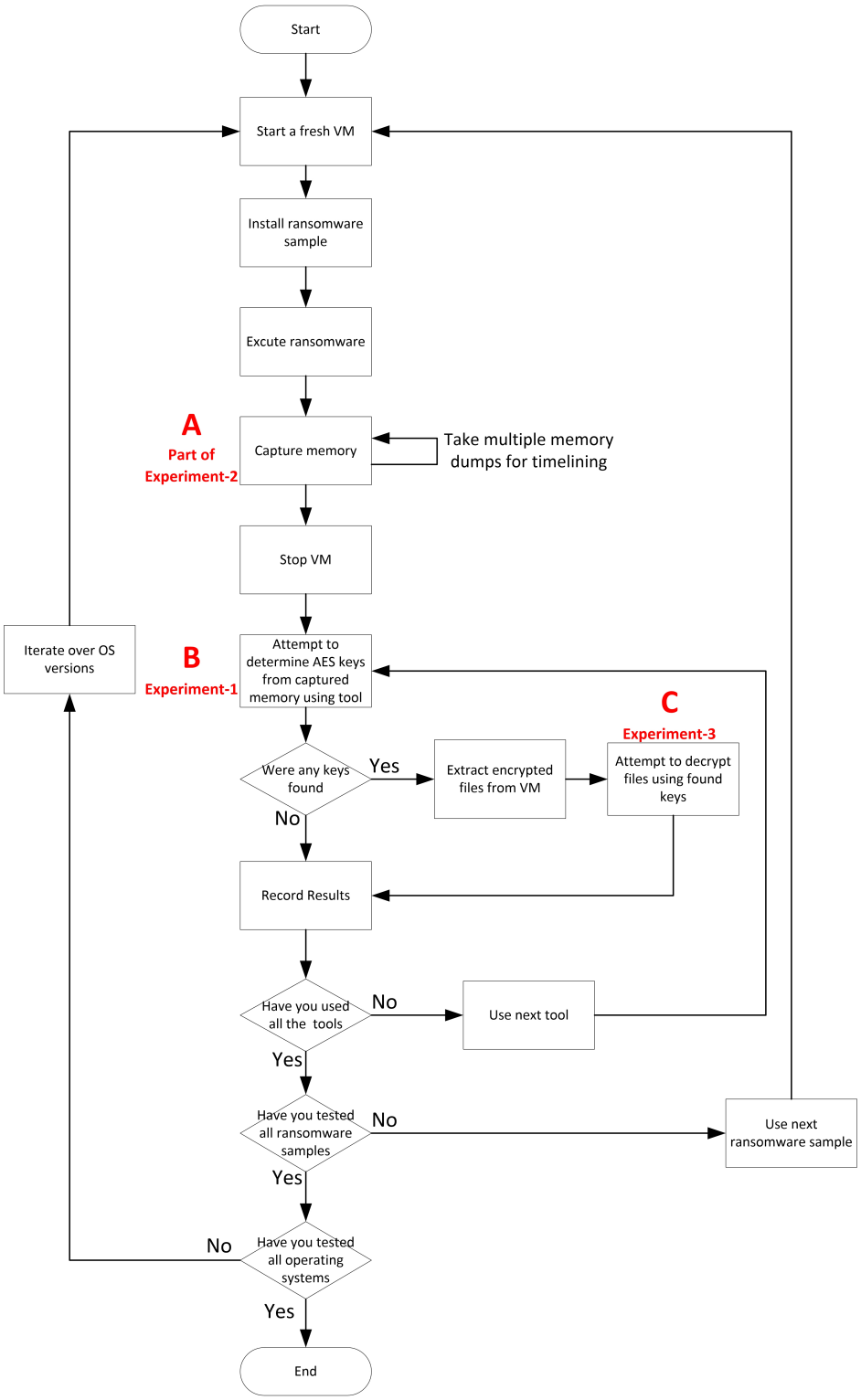


Figure 3.4: Experiment Flow

The main steps of the experiments are discussed below:

Iterate over Operating Systems versions A paper by Rossow(Rossow et al., 2012) has been extensively consulted during the design of these experiments. When discussing realism in experimentation Rossow cautioned against performing experiments against just one operating system and then drawing general conclusions. To guard against this potential criticism, the experiments performed in this research were conducted against the top two most commonly used versions of the windows operating systems currently in use, these being Windows 10 and Windows 7 as confirmed by Statista (Statista, 2019) and shown below in Figure 3.5.

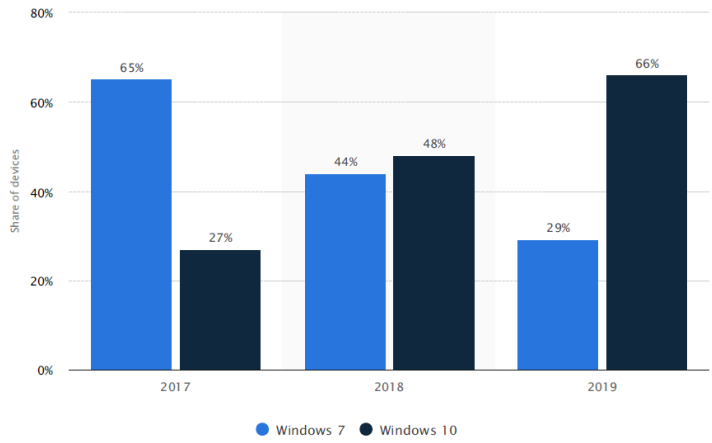


Figure 3.5: Windows 7 and 10 adoption rate in N. America and Western Europe from 2017 to 2019

Start a fresh VM Prior to each execution of a ransomware sample the virtual test machine was reverted back to its original state using the virtualbox snapshot functionality. This is done to ensure that each execution of a ransomware sample begins with the victim's machine being in the same state containing the same amount of information, allowing for direct comparisons to be made between each execution cycle. Results are only comparable if each sample is executed in an identical environment (Hoopes, 2009). This approach is know as baselining and has been used in many similar experiments (Egele et al., 2012; Ahmad et al., 2016).

Install Ransomware The three ransomware samples are already present in an archive on the victims virtual machine. The sample to be tested was extracted from the archive and prepared for execution. The decision to use three malware samples was a result of a trade-off between limited time available for testing

against performing the tests on a representative sample of ransomware families. (Rossow et al., 2012).

Execute Ransomware The chosen ransomware sample was executed from the command line.

Capture Memory The timings of when to take a copy of the working memory of the virtual machine was determined by the outcome of experiment-2. If the keys became available in memory, then a copy of the machines memory was taken.

There are several options available to capture a memory sample, for example by running a tool from within the test virtual machine to dump the memory to a file using a tool like RamCapturer (Belkasoft, 2019), however it was found that this approach caused contamination of the captured memory by the executing program. The technique employed during these experiments was to take a snapshot, from the virtualbox host machine, of the test victims virtual volatile memory using tools provided by the virtualization software in similar techniques used by other researchers (McLaren, Buchanan et al., 2019; Nissim et al., 2019).

Stop VM Once the required number of memory captures has been performed, or if the ransomware had completed, then the victim virtual machine is halted.

Attempt to determine AES keys from captured memory Analysis was performed on the captured memory samples with the aim of identifying candidate AES keys, using some of the dynamic analysis forensic tools identified in the literature review. The first two tools shown below were selected due to the success that the researchers had achieved with them during their own investigations. The tools used in this investigation being:

1. **findaes** - This was a tool developed by Kornblum (Kornblum, 2019) based on the work by Trenholme (Trenholme, 2014; Halderman et al., 2009) and tries to find the keys using the AES key schedule.

2. **interrogate** - This was a tool developed by Maartmann-Moe (Maartmann-Moe et al., 2009) also based on the work by Trendholme (Trenholme, 2014), and was used during their research to investigate both RSA and AES keys in cryptographic applications such as disk encryption and PGP.

3. **RansomAES** - A hybrid tool developed by the author which incorporates logic from the Volatility Framework (Volatility, 2019) together with the logic from findaes in an attempt to improve the accuracy and performance of the key detection. A listing of this code is shown in Appendix B.3. This tool firstly extracts a copy of the malware processes memory from the

dump file using the 'volatility' tool and then checks this extract for AES keys using the 'findaes' program.

Extract Encrypted Files The control files described in Table 3.1 were placed in the documents directory of the target virtual machine prior to the execution of the ransomware sample.

File Name	Comment
word.doc	Basic MS Word document containing one line of text in old format
word.docx	Basic MS Word document containing one line of text in new format
excel.xls	Basic MS Excel document containing one line of text in old format
excel.xlsx	Basic MS Excel document containing one line of text in new format
pdf.pdf	Basic PDF document containing one line of text
pdf-large.pdf	A large PDF document containing more that 100 pages
text.txt	A simple text file containing one line of text
jpeg.jpg	A small jpeg picture

Table 3.1: Control file details

The original contents of these files is known to the researcher and they represent typical formats targeted by ransomware (Sittig & Singh, 2019; Sultan et al., 2018; CERT-EU, 2017). Once the ransomware had executed for a specified period, these files were copied from the test victims virtual machine to the host machine to analyse if they had been encrypted. If they had not, then the ransomware was allowed to continue and the process repeated after a specified period. The execution of the ransomware occasionally changed the name or attribute of the control files, however in all cases it was still possible to identify these encrypted control files and from which original control file they had been generated from.

Attempt to decrypt files If any candidate keys were discovered during the analysis stage of the experiment, they were then used to try and decrypt the encrypted control files extracted from the victim virtual machine. The AES Initialisation Vector (IV) required for decryption, were contained within the encrypted file and used in combination with the candidate keys to decrypt the file. The results of the decryption attempts were recorded and presented in sections 5.2, 5.3 and 5.4.

3.4.5 Control file example

As mentioned previously a set of standard files were placed on the machine prior to the execution of the ransomware. These files being referred to as control files. The purpose of these files is to provide a simple method for the researcher to test the decryption of a file and confirm that the found key was the one used by the ransomware. For example one of the control files was called `pdf.pdf` and looked like the document shown in Figure 3.6

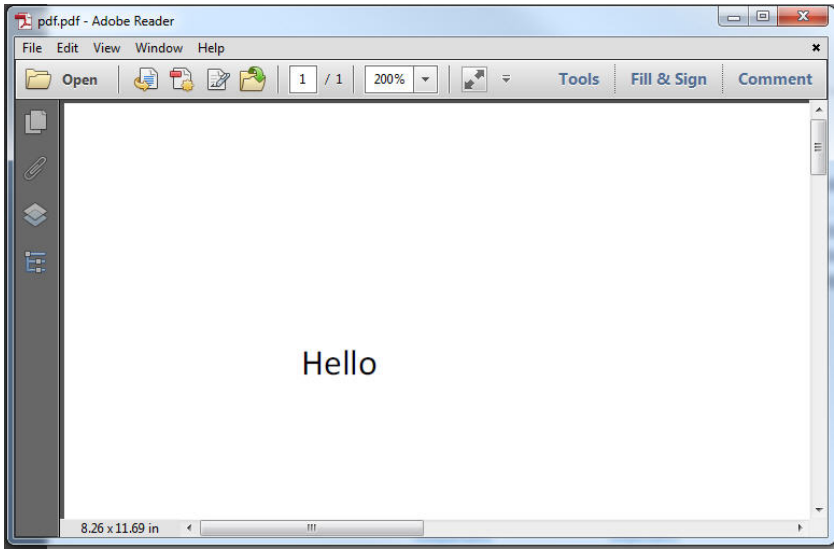


Figure 3.6: Control pdf.pdf file

Its contents is shown Figure 3.7

```
Address  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f Dump
00000000 25 50 44 46 2d 31 2e 35 0a 25 c7 ec 8f a2 0a 35 %PDF-1.5.%Çì.¢.5
00000010 20 30 20 6f 62 6a 0a 3c 3c 2f 4c 65 6e 67 74 68  0 obj.<</Length
00000020 20 36 20 30 20 52 2f 46 69 6c 74 65 72 20 2f 46  6 0 R/Filter /F
00000030 6c 61 74 65 44 65 63 6f 64 65 3e 3e 0a 73 74 72 lateDecode>>.str
00000040 65 61 6d 0a 78 9c 4d 8c bd 0a c3 30 0c 84 e9 df eam.xœMŒ½.Ã0.„éß
00000050 a2 a7 d0 68 0f 55 65 c5 b6 ec b5 d0 a5 5b 8b b7 ¢§Ðh.UeÅ¶ìµÐ¥[‹·
00000060 d2 29 d0 4c 1e d2 bc 3f c4 49 29 e4 6e f9 b8 3b Ò)ÐL.Ò¼?ÄI)änù¸;
00000070 6e 44 26 27 c8 8b ff d0 57 b8 3c 15 87 09 d6 18 nD&'È‹ÿÐW¸<.‡.Ö.
00000080 bf 03 8c 90 a8 5b b4 06 5b ee 2b 5e 4b db 27 74 ¿.Œ.¨[´.[î+^KÛ't
```

Figure 3.7: Control pdf.pdf file contents

3.5 Results and analysis

3.5.1 Analysis method

The results obtained from the execution of the experiments presented in this chapter can then be used to draw conclusions on the validity of the hypothesis into the usefulness of forensic techniques in mitigating the affects of a ransomware attack. The expected results will fall in to the following categories:

1. Were any possible keys extracted from memory.
2. How long were the keys present in memory.
3. Could the found keys be used to successfully decrypt the control files, thus confirming that they were the symmetric keys used by the ransomware program.
4. How quickly did each of the symmetric key analysis tools take to identify the keys.

While no experiments were found during the literature review that could be used for direct comparison of results, using a well defined and understood research method allowed the researchers to be able to draw techniques taken from other research that performed analysis in a similar area using the same model. This gave the researcher an opportunity to be able to make comparisons to the results with these projects.

3.5.2 Data capture

In accordance with a scientific approach the results for the execution of the experiments described above will be recorded in a table to aid presentation. An example of such a results table is shown in Table 3.2.

	findaes			**interrogate**			**RansomAES**		
	Key Found	Time Taken (s)	Files Decrypted	Key Found	Time Taken(s)	Files Decrypted	Key Found	Time Taken(s)	Files Decrypted
NotPetya									
Bad Rabbit									
Phobos									

Table 3.2: Example of Results Matrix

When the 'findaes' and 'interrogate' tools were used in other research projects, just their success in key identification was recorded and not their total execution time. This extra measurement was included during the experiments per-

formed in this research to allow for a more robust comparison of the tools performance compared to the hybrid tool, 'ransomaes' developed by the author.

3.6 Conclusion

This chapter described the design process used to identify the fundamental concepts applied in the development of this projects experimental method. After thorough analysis of the proposed hypothesis and how the results would be validated, it was decided that the most suitable approach would be to use the hypothetico-deductive research project (Edgar & Manz, 2017a) and this was then used to guide the design choices. Once the research method had been defined, it was then easier for the researcher to find a generally accepted model that could be used (Khan Academy, 2017) to base the designs on.

When deciding on the environment design two key factors were considered; Firstly the suitability of the environment, as it was key that the environment used was as realistic as possible. This is critical to the quality of the results as if the environment was not realistic, then no relevant conclusions could be drawn from the experiments, making the whole project of limited use. To achieve the required realism, operating systems with a high market penetration were used on machines that appeared to be connected to a functioning network. The second main design criteria being the safety and isolation of the environment as it would be totally unacceptable, as well as unethical, for any deliberate execution of a malware sample to have an impact on assets outside the test environment.

By definition, when selecting specific software for an environment, the generality of the experiments and their results is impacted. To counteract this, the software that was selected were some of the most common software currently in use and being based on recommendations taken from the literature review. This should allow for the findings to remain relevant. The chosen software proved to offer the best compromise between a real physical environment and a safe and flexible environment that could be recreated quickly and shield external assets.

The only disadvantages of this configuration being that the host laptop can not and should not be used for any other tasks during testing and will have to be reinstalled at the conclusion of the experiments.

The resulting experiments and environment design relied considerably on the findings from the literature review, most significantly the recommendations from Rossow (Rossow et al., 2012) for the overall structure of the experiments

and Sanabria (Sanabria, 2007) for configuring the environment to be more realistic and robust. The information provided by these authors contributing to a final design which is considered to be contained and realistic. The designed environment is able to provide a safe and stable platform allowing for the execution of a comprehensive set of experiments in a controlled manner. Being in control of the environment is critical when working with malware due to its nature and the possible catastrophic consequences of unrestricted execution. While no specific implementation details are provided by other researchers conducting similar malware and live forensics testing (Maartmann-Moe et al., 2009; Balogh & Pondelik, 2011; Walters & Petroni, 2007), it is considered that this projects environment is of at least a similar quality and would only require slight modifications for it to be used for other experiments, such as the ones conducted by the other researchers in this area.

There were several options available for memory acquisition and during this project, after review, it was decided to utilize the hypervisor tools executed on the VM host machine to capture the volatile memory of the victims machine. The benefits of non contamination of the memory dumps by the acquisition method while relying on product specific hypervisor tools was considered to be better alternative than using memory capture tools from within the operating system of the victims machine which while being more generic would contaminate the captured memory.

The chapter concludes with an overview of the types of results expected from the execution of the experiments and how they will be presented.

It is felt that the work performed for this chapter and the resulting description contributes significantly to the second learning outcome as it clearly demonstrates a robust environment and methodical experiment design, based on well known design methodologies sourced from a strong literature review.

Chapter 4

Implementation and results

4.1 Introduction

This chapter discusses in detail the configuration of the test environment used to perform the experiments to determine if live forensic techniques can be used to mitigate the affects of a ransomware attack. The chapter provides specific information on which ransomware samples were used, how they were selected, how the test environment was created and configured and what tools were used for the analysis. The chapter then goes on to define in detail how the experiments were performed and ends with a presentation of the raw results achieved.

There exists sufficient information in this chapter to allow for third party researchers to replicate the experiments performed and validate the findings.

4.2 Ransomware sample selection

Recent ransomware attacks were researched in depth as part of the literature review (Comodo, 2018; Kapersky, 2018; Vanderburg, 2019; O'Donnall, 2019; Hautala, 2019) and based on the findings, combined with initial validation, it was decided to select the following three ransomware examples for analysis. All of which were examples of the HCR type of ransomware strains using AES for the symmetric portion of the encryption. They have occurred in the last two years and have received significant coverage in the mainstream press (Fruhlinger, 2019).

NotPetya. This ransomware attack is considered to be the most damaging attack ever (Kapersky, 2018) and has been estimated to have cost more than $10 Billion (Mekynyk et al., 2019). Unlike the WannaCry ransomware, NotPetya used multiple propagation techniques to infect other computers in the network (Sai & Kumar, 2019).

Bad Rabbit. This is an even more recent adaptation of the NotPetya ransomware family that emerged in 2018(SonicWall, 2019). It is reported to use the re-

cently discovered NSA EternalBlue exploit (Perekalin, 2018; Mamedov, Sinitsyn & Ivanov, 2018) and is propagated via a fake flash plugin update.

Phobos. This is one of the most recent ransomware samples found where detailed information is available (Issa, 2019). This ransomware appeared in the spring of 2019. It is believed to be a new strain of ransomware strongly based on the previously known ransomware family: Dharma (a.k.a. CrySis), and probably distributed as a Ransomware as a Service (Raas) by the same group as Dharma.

Specific details of the ransomware samples are given in Table 4.1.

Name	SHA256 Checksum	Date
NotPetya	027cc450ef5f8c5f653329641ec1fed91f694e0d229928963b30f6b0d7d3a745	2019-08-20
Bad Rabbit	630325cac09ac3fab908f903e3b00d0dadd5fdaa0875ed8496fcbb97a558d0da	2019-08-20
Phobos	a91491f45b851a07f91ba5a200967921bf796d38677786de51a4a8fe5ddeafd2	2019-08-20

Table 4.1: Ransomware Samples

Once the samples were retrieved, they were loaded to VirusTotal (wwww.virustotal.com) to confirm that they were valid ransomware programs, the results of which are shown below in Figures 4.1, 4.2 and 4.3

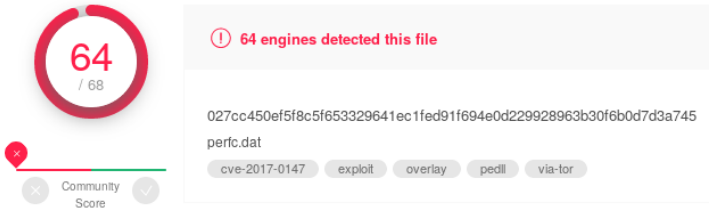


Figure 4.1: VirusTotal Results for NotPetya

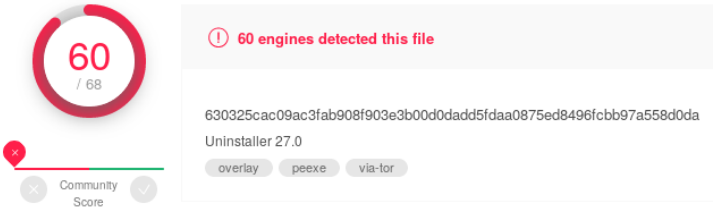


Figure 4.2: VirusTotal Results for Bad Rabbit

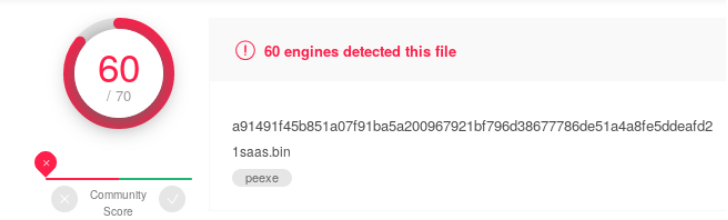


Figure 4.3: VirusTotal Results for Phobos

4.2.1 Other ransomware

Several alternative ransomware samples were considered before deciding to use the three examples mentioned above. The following were initially considered before being rejected.

Wannacry - Testing confirmed what was detailed in the literature that this strain used unique AES keys for each encrypted file(Vipre Security, 2017; Berry, Homan & Eitzman, 2017). After conducting multiple tests of the memory acquired during the execution of this malware using all the live forensics tools, no recoverable AES keys were found.

Cerber - This ransomware appears not to use AES encryption.

Lucky/nmare - The sample of this ransomware required access to the internet to be able to download the file encryption modules as they are not delivered with the initial sample. The services provided by the Debian virtual machine were not able to trick the ransomware into executing normally. It was deemed too risky to allow this external network access.

Satan, SamSam & GrandCab - It was not possible to trigger these samples of ransomware to encrypt any of the control the files or display the ransom message.

4.3 Laptop configuration

A laptop with the following specifications was used as the main hardware platform for the investigation, details of which are given in Table 4.2.

Hardware	Details
CPU	Intel Core i7-3667U
Memory	8GB
Software	
Operating System	Windows 10 Pro Version 1809
Virtual Box	5.2.8

Table 4.2: Laptop Configuration

This laptop has no external network connections and as an added precaution was set to airplane mode and had the wifi card switched off. When discussing the virtual environments this physical machine is referred to as the host machine as it hosts the virtual environment within VirtualBox.

4.4 Virtual hardware configuration

Using the design concepts outlined in section 3.3.4 , three guest machines were defined in the virtual environment. Two victim machines with different operating systems on each, used to test the behaviour of the ransomware code and one network services machine that was used to provide any network services that maybe required by the software running on the victim machines. Details of the virtual guest machines are given in Table 4.3.

Guest Machine Name	Operating System	CPU	Memory	Purpose
Windows 7	Windows 7 Ultimate	Core i7-3667	4GB	Ransomware Victim Machine
Windows 10	Windows 10 Pro	Core i7-3667	4GB	Ransomware Victim Machine
Debian	Debian 5.2.9	Core i7-3667	2GB	Network Services

Table 4.3: Virtual Hardware Configurations

VirtualBox technology is also referred to as hypervisor type-2 (Simic, 2019) and a conceptual representation of how this technology is utilized in our test envir-

onment is shown in Figure 4.4 below. During an experiment only one victim machine was active at any one time.

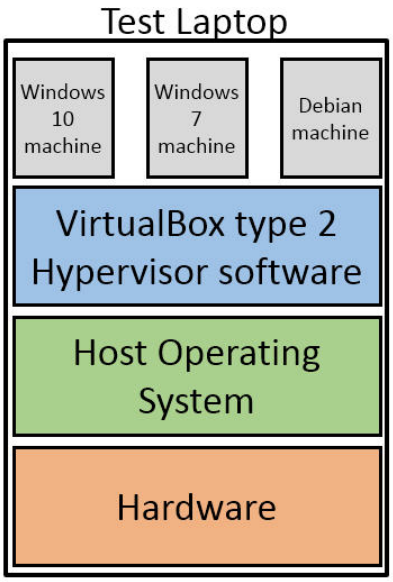


Figure 4.4: Conceptual Laptop Software Stack

4.5 Virtual network topology

The guest machines were connected together using the recommended 'host-only' connection technique (Bellardo & Savage, 2003; Hoopes, 2009). This creates a separate virtual LAN and provided isolation of the guest machines from the host machine as well as providing containment for the malware execution. The network details of this virtual LAN is shown in Table 4.4.

Machine	IP Address	Purpose
Windows 7	10.1.1.7	Victim machine
Windows 10	10.1.1.10	Victim machine
Debian	10.1.1.30	Default gateway, DNS server

Table 4.4: Virtual Network Configuration

The virtual LAN and the guest machines were run in two separate configurations depending on what operating system was being tested. If the behaviour of the ransomware sample was being tested on the Windows 7 operating system,

then the windows 10 machine was switched off. An example of this configuration is shown in Figure 4.5.

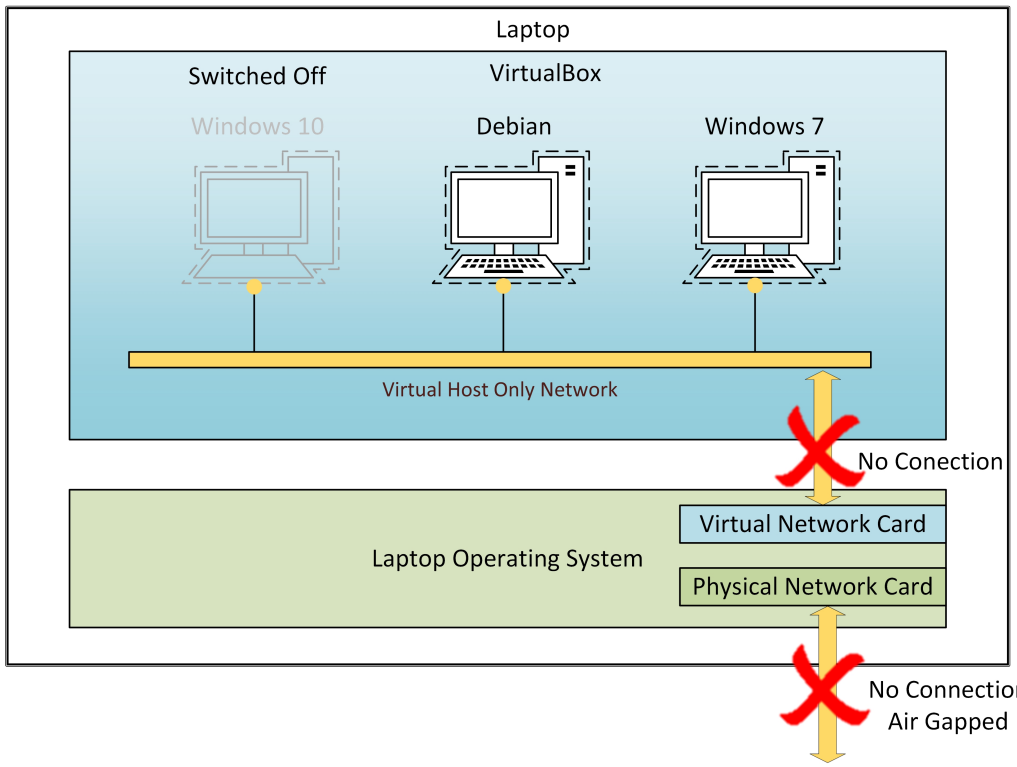


Figure 4.5: Windows 7 Test Environment

Likewise, if the behaviour of the ransomware sample was being tested on the Windows 10 operating system, then the windows 7 machine was switched off.

Switching unused machines off was done to reduce the possible attack surface for the ransomware and also to conserve the host machine resources. The operating system of the machine providing network services was chosen to be Linux which is known to be more resilient to ransomware attacks thus further limiting the available attack surface.

4.6 Tools

The following tools were used during the configuration of the test environment and execution of the experiments.

fakedns.py – This is a python script running on the Debian machine that provides DNS services to the network. The other guest machines are configured to use

this as their DNS server. This server will always return the IP address of the Debian machine for whatever domain it is queried for. This was configured to prevent any attempted external network connections.

INetSim - Is a free, Linux-based software suite for simulating common internet services and was installed on the Debian machine. INetSim is considered to be the best free tool (Sikorski & Hong, 2012) for providing fake services, emulating services such as HTTP, HTTPS, FTP, IRC, SMTP, and others. This combined with the services provided by fakedns.py will create the illusion of a realistic pseudo network that the malware can interact with if required.

Interrogate – This was a tool developed by Maartmann-Moe (Maartmann-Moe et al., 2009) and was used during their research to investigate AES keys in cryptographic applications such as disk encryption and PGP. This is one of the three tools that will be used to examine the captured memory try and discover the ransomware's AES keys.

findaes – This was a tool developed by Kornblum (Kornblum, 2019) and tries to find the AES key using the AES key schedule. This is one of the three tools that will be used to try and discover the ransomware's AES key from the captured memory.

RansomAES – This is a program that was developed by the author. It combines the logic from findaes program together with some logic from the volatility framework (Volatility, 2019) that targets the specific ransomware memory portion of a memory dump. This was developed with the aim of improving the performance and accuracy of key identification. This will be the third tool used to examine the captured memory and a listing of the code for this tool can be found in Appendix B.3.

4.7 Experiments

Copies of the ransomware code was already present on the virtual guest machines in the directory:

```
C:\dissertation\ransomware
```

Details of the experiments performed are given below:

4.7.1 Experiment 1 - Is the key in memory

A fresh VM was started and the researcher connected to the machine (Hoopes, 2009). A copy of the machines memory was taken prior to the execution of the

ransomware, so that any AES keys that are present in the machines memory prior to the execution of the ransomware could be identified and excluded from the experiments results. The researcher then opened a command prompt window and one of the test ransomware programs was started, using one of the commands below:

For NotPetya

```
C:\windows\system32\rundll32.exe
    c:\dissertation\ransomware\netpetya.dll, #1 30
```

For BadRabbit

```
c:\dissertation\ransomware\sample.badrabbit1.bin.exe
```

For Phobos

```
c:\dissertation\ransomware\1saas.bin.exe
```

After waiting for a short period, a copy of the guest's machines memory was taken. The length of waiting time varied between ten seconds and two minutes depending on the ransomware strain. The time required to wait was determined through trial and error. To capture the memory, the following command was executed on the host machine:

```
VBoxManage.exe debugvm <VBox Machine Name> dumpvmcore --filename
    <filename>.elf
```

The format of the memory dump file was elf, which is the default dump format for VirtualBox memory capture. The memory dump file was then analysed by each of the selected live forensics memory tools, using the following commands:

```
findaes <filename>.elf
or
interrogate -a aes -k 128 <filename>.elf
or
ransomaes -p <ransomware pid> -t Win7SP0x86 <filename>.elf
```

Any keys found resulting from the execution of these tools were recorded and used as input for experiment-3. If no keys were found then the experiment was extended in one minute intervals and new memory dumps taken. The experiment terminated when keys were found or the execution of the ransomware completed.

4.7.2 Experiment 2 - How long is the key present

If keys were discovered in experiment-1, then this experiment was also performed. It is similar to experiment-1, except the memory dumps are taken regularly throughout the execution time frame of the ransomware. The time interval used between memory dumps varied depending on the ransomware sample and was determined via trial and error over multiple executions. The dumps were analysed using one of the selected tools to confirm that they keys were still present. The times when the keys were present was recorded and a basic timeline for the ransomware execution was created. The timeline creation process was predominantly a manual task, combining the results recorded from the experiments, observations of system behaviour, descriptions gained from the literature review and utilisation of the six step ransomware model also described in the literature review (McAfee Labs, 2016)

4.7.3 Experiment 3 - Does the key decrypt files

If candidate AES keys were discovered in experiment-1, then the found keys were used in an attempt to decrypt the control files extracted from the guest machine after the ransomware had completed. This was partially an automated task with some manual steps. A tool `decrypt.py` was created by the author to perform the basic AES decryption using the `'AES'` cipher object from the `'Crypto.Cipher'` python library. A copy of the decryption program appears in Appendix B.2 and an example of the command used to decrypt a file is shown below:

```
python decrypt.py --file <encrypted file> --key <key-file>
```

The tool will attempt to use all the keys present in the `<key-file>` to try and decrypt the the `<encrypted-file>`. The contents of the key-file are the keys found by the live forensics tools. The program was able to determine the required IV from the supplied encrypted file and then use this together with the candidate keys to try and decrypt the file. Determination if the file was correctly decrypted

remained a manual task. The resulting decrypted file normally required extra modifications such as adding a header or removing a trailer.

4.8 Experimental process overview

As an example of how the experiments were performed, the order of the executed commands and their purpose is described here. This section describes a theoretical walk through of an experiment chain. Figure 4.6 below gives a graphic example of how the different steps are performed , a description of the individual steps, and where appropriate, an example of the commands that are performed to complete each step.

This example focuses on recovering the control text file called "text.txt" which is initially encrypted by the ransomware, then extracted from the victims machine, moved to a Linux machine, decrypted and finally modified back to its original state. The reason why the files are moved to a Linux machine is purely due to the fact that the live forensic and AES decryption tools were installed on this machine. These steps could in theory have been performed on the host laptop, if the correct tools had existed there.

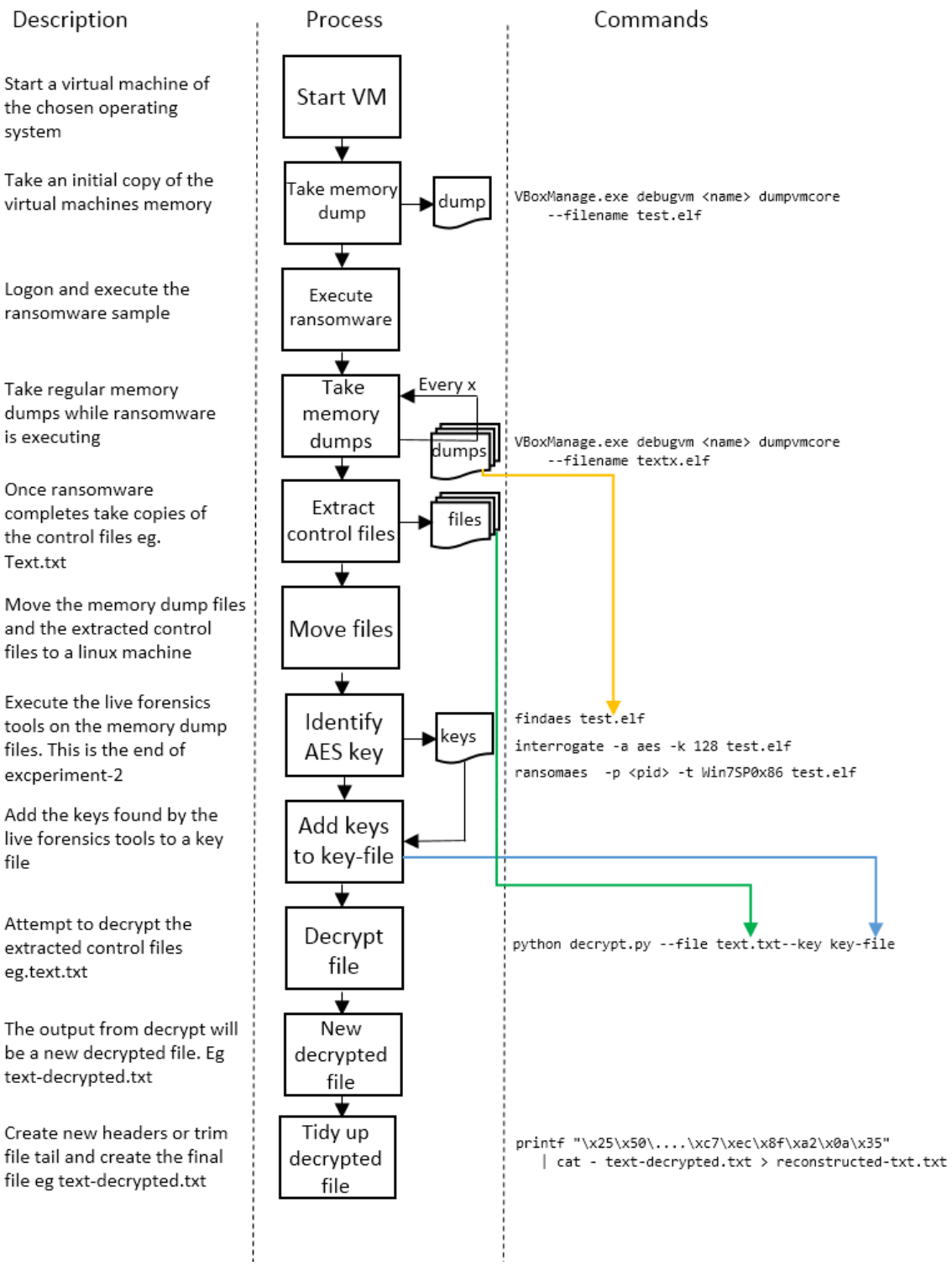


Figure 4.6: Experiment Process Overview

4.9 Experimental results

The following sections present the results achieved from executing the described experiments. Tables 4.5 and 4.6 show the outcome of executing the three different live forensics programs on memory dump files taken during the execution of the three ransomware samples. The results show if each of the live forensics programs were able to identify the encryption key used by the ransomware, how long the tool took to complete execution and if the found key did in fact decrypt the control files encrypted by the ransomware.

4.9.1 Ransomware execution on Windows 7

It can be clearly seen from the results in Table 4.5 that all three live forensics tools were able to correctly identify an AES encryption key in the memory dump file, and the found key was successful in decrypting the encrypted control files. The performance of the interrogate tool being significantly longer than the other tools used.

	findaes			**interrogate**			**RansomAES**		
	Key Found	Time Taken (s)	Files Decrypted	Key Found	Time Taken(s)	Files Decrypted	Key Found	Time Taken(s)	Files Decrypted
NotPetya	yes	198	yes	yes	17,813	yes	yes	191	yes
Bad Rabbit	yes	151	yes	yes	17,659	yes	yes	127	yes
Phobos	yes	113	yes	yes	17,887	yes	yes	119	yes

Table 4.5: Windows 7 Results

4.9.2 Ransomware execution on Windows 10

As with the results from the tests performed on the Windows 7 operating system, again it can be clearly seen from the results in Table 4.6 that all three live forensics tools were able to correctly identify an AES encryption key in the memory dump file, and the found key was successful in decrypting the encrypted control files.

	findaes			**interrogate**			**RansomAES**		
	Key Found	Time Taken(s)	Files Decrypted	Key Found	Time Taken(s)	Files Decrypted	Key Found	Time Taken(s)	Files Decrypted
NotPetya	yes	125	yes	yes	17,812	yes	yes	168	yes
Bad Rabbit	yes	163	yes	yes	17,692	yes	yes	179	yes
Phobos	yes	170	yes	yes	17,779	yes	yes	185	yes

Table 4.6: Windows 10 Results

4.10 Conclusion

This chapter documents the implementation of the environment and experiments, taken from the previous design chapter. The chapter describes in detail the experimental processes followed and the commands executed in trying to prove the hypothesis that live forensic techniques can be used to mitigate a ransomware attack. The chapter concludes with a presentation of the results collected from the experiments.

During the execution of the ransomware samples, no external effects of their execution were observed and no assets outside the experimental environment were affected. This is considered to be a good endorsement of quality of the implemented experimental environment.

The experiments were conducted on two different operating systems and as suspected by the researcher, these systems behaved in a similar manner with similar times and results. It was felt that the extra effort required to perform these parallel experiments was worthwhile as it allowed the researcher to draw realistic conclusions from the experiments (Rossow et al., 2012) and also defend the findings and drawn conclusions. It also provided a defence against possible criticisms that could otherwise be raised about the experiments being unrepresentative or too specific.

The presence of the Debian machine on the virtual network, supplying network services to the victims machine, was implemented to allow the network connected to the victims machine to appear to be as realistic as possible. However during the execution of the three ransomware samples selected for investigation, no network traffic was observed leading the researchers to conclude that the ransomware samples used were self contained and did not require network access to function correctly. However network traffic was generated when executing some of the ransomware samples that were rejected, for example the 'Lucky/nmare' ransomware, which attempted to contact it's C&C server to download the encryption modules required for its execution. When testing ransomware samples whose behaviour is unknown, it is recommended that this Debian machine remains active on the network. Monitoring the logs of this machine could provide some useful information regarding the behaviour of the ransomware sample under investigation.

Some implementation issues were encountered while trying to determine which ransomware samples would be included in the investigation. For example, the researchers were not able to capture any keys when executing the 'wannacry' ransomware sample, while this could have been used as a good example for a

null hypothesis (Edgar & Manz, 2017a), it was decided to reject this sample and focus the research on samples where the capture was successful. Difficulties were also encountered in having some of the other ransomware samples execute correctly, or in a way that was described in the literature and for these reasons, they were also rejected.

It is felt that the work performed for this chapter and the resulting description contributes significantly to learning outcome two, as the decisions and processes involved in the implementation of the environment and experiment design are presented in depth and discussed in detail here. Discussion towards the end of the chapter relating to the provisional results and issues experienced during execution should also contribute in part to the learning outcome three.

Chapter 5

Evaluation

5.1 Introduction

The aim of this project was to determine if live forensic techniques could be used in the mitigation of a ransomware attack. This chapter evaluates how well the objectives have been met and whether the aim has been achieved.

This chapter is divided into separate sections one for each of the ransomware samples tested. Each of these sections discusses the results of the three experiments conducted.

One thing to note is that no significant differences were observed in the behaviour or results of the experiments when performing them on the Windows 7 and Windows 10 operating systems apart from differing key values which is to be expected as these will change with every execution of the ransomware program. Bearing this in mind, for the remainder of this section the report will concentrate on the results of the experiments performed on the Windows 7 operating system.

5.2 NotPetya

The sample described in Table 5.1 was used during these experiments.

Name	NotPetya
SHA256	027cc450ef5f8c5f653329641ec1fed91f694e0d229928963b30f6b0 d7d3a745
URL	https://github.com/fabrimagic72/malware-samples/tree/master/Ransomware/NotPetya

Table 5.1: NotPetya Sample Details

During an experiment only one victim machine was active at any one time.

The execution of the ransomware appeared to follow the descriptions provided by (Berry et al., 2017; Vipre Security, 2017). The main steps being:

1. Adding persistence and gathering user's credentials.
2. Scanning the machine for files to encrypt. This sample seems to ignore the control files with the 'txt' extension.
3. Encrypting the identified files using AES encryption with what appears to be the same AES key being used for all the files. Interestingly neither the filename or the file meta information changes when the file is encrypted.
4. Attempting lateral movement to other machines, however no actual evidence of this was discovered.
5. After 1 hour, rebooting the machine automatically.
6. Displaying the screen shown in Figure 5.1, while encrypting the Mater Boot Record.

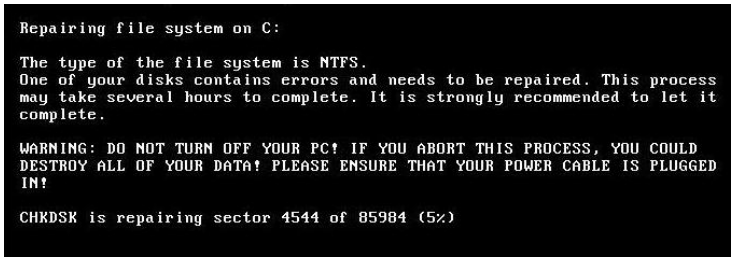


Figure 5.1: Fake Chkdsk

7. Once the fake chkdsk completes, the machine reboots automatically again.
8. After reboot, the ransomware message shown in Figure 5.2 is displayed

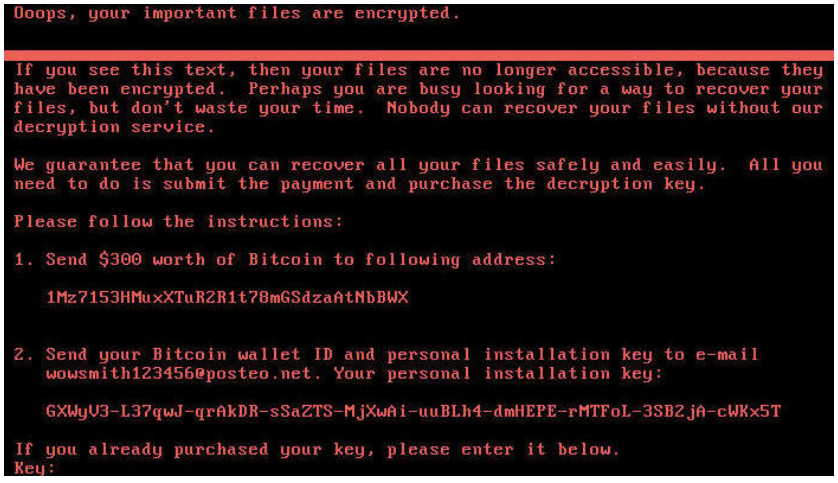


Figure 5.2: NotPetya Ransom Message

5.2.1 Experiment 1 – Is the key in memory

While the ransomware was executing, memory dumps were taken at regular intervals and used as input to the live forensic tools. All three live forensics tools used to examine the memory dumps were able to identify AES keys in memory, some of the found keys were ignored as they were present prior to the execution of the ransomware. However all the tools also successfully identified the key used by the ransomware to encrypt the files. The key discovered in this experiment is shown below in Figure 5.3.

```
f3 84 37 98 7e 62 d9 18 86 d2 d8 30 67 3c 61 88
```

Figure 5.3: NotPetya AES Encryption Key

The tool interrogate gave the output shown in Figure 5.4 below.

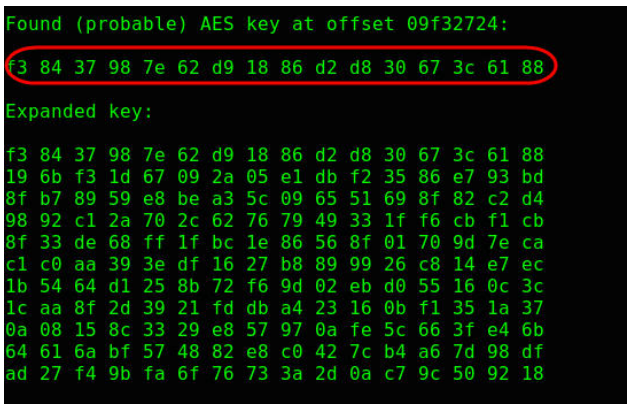


Figure 5.4: NotPetya Interrogate Output

The tool findaes gave the output shown in Figure 5.5 below:

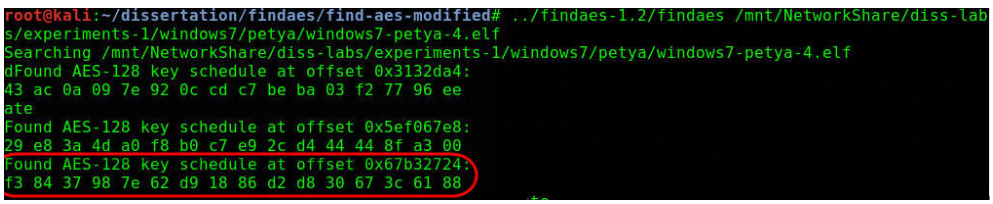


Figure 5.5: NotPetya findaes Output

The RansomAES tool gave the output shown in Figure 5.6 below:

```
root@kali:~/dissertation/findaes/ransomaes# ./ransomaes -p 2168 -t Win7SP0x86 /mnt/NetworkShar
e/diss-labs/experiments-1/windows7/NotPetya/windows7-petya-2.elf
Looking for process id: 2168
Using profile type: Win7SP0x86
Command is: /usr/bin/volatility --profile Win7SP0x86 -f /mnt/NetworkShare/diss-labs/experiment
s-1/windows7/NotPetya/windows7-petya-2.elf memdump -p 2168 --dump-dir .
Volatility Foundation Volatility Framework 2.6
************************************************************************
Writing rundll32.exe [  2168] to 2168.dmp
Command is: 2168.dmp
Searching 2168.dmp
Found AES-128 key schedule at offset 0x11a18c:
f3 84 37 98 7e 62 d9 18 86 d2 d8 30 67 3c 61 88
Found AES-128 key schedule at offset 0x9bd980c:
43 ac 0a 09 7e 92 0c cd c7 be ba 03 f2 77 96 ee
```

Figure 5.6: NotPetya RansomAES Output

5.2.2 Experiment 2 – How long is the key present

For each operating system a total of fifteen memory dumps were taken at regular intervals during the execution of the ransomware. These dumps were then analysed to determine in which of them, the AES keys were present. When reviewing the memory dumps for this execution it was identified that the key became available within 2 minutes of the start of the ransomware, and remained in memory until the machine was rebooted by the ransomware program after 60 minutes. The key did not survive the reboot and was not recoverable from memory after this. A graphical representation of some of the ransomware's behaviour and key availability is shown below in Figure 5.7 below.

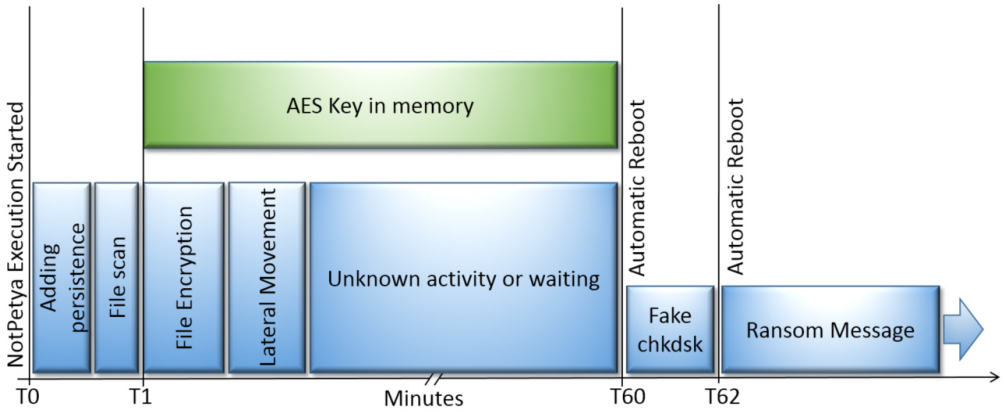


Figure 5.7: NotPetya Timeline

This timeline correlates well with the findings of other researchers who have analysed the general behaviour of this ransomware from a malware perspective (Berry et al., 2017; Vipre Security, 2017). However no work was found that analysed when and for how long the actual key remains in memory. Using this

diagram it is clear to see that it is present for a total 59 minutes, which is the majority of the ransomware's execution time.

Also no similar graphical representation of the ransomware time line was found in the literature, the one shown in Figure 5.7 being generated by the author. It is felt that this type of representation aids understanding of the overall behaviour of the ransomware and could provide an area for future research.

5.2.3 Experiment 3 – Does the key decrypt files

When the NotPetya ransomware encrypts a file, the first 16 bytes of the file are overwritten with the AES Initialisation Vector (IV) value (Sood & Hurley, 2017). A program was developed that firstly reads the IV value from the encrypted file, then used this together with the key found in experiment-1 to decrypt the files contents. The process used to recover a file is shown below. In this example a PDF file was being recovered.

The content of a PDF file encrypted by the NotPetya ransomware is show in Figure 5.8.

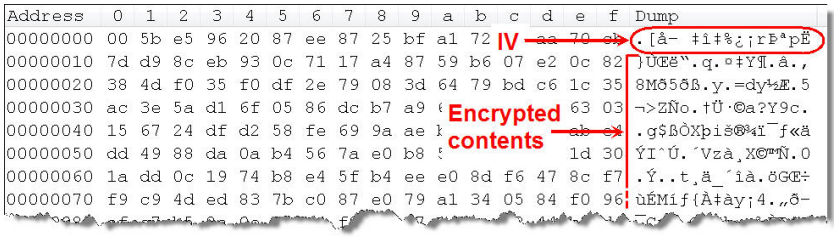


Figure 5.8: File Encrypted by NotPetya

After execution of the decryption program, using the AES key extracted from the memory dump, the contents of the file can be seen in Figure 5.9.

Figure 5.9: Partially Decrypted NotPetya File

As the file header information was overwritten by the IV during encryption, the

final stage involves recreation of the file header. In this example the file is a PDF, so the following command was used to recreate the header information and insert it back in to the decrypted file.

```
printf
    "\x25\x50\x44\x46\x2d\x31\x2e\x35\x0a\x25\xc7\xec\x8f\xa2\x0a\x35"
    | cat - pdf-decrypted.pdf > reconstructed-pdf.pdf
```

The resulting decrypted file is shown in Figure 5.10. This decrypted file can be read by any PDF reader without any issue and is exactly the same as the example control file shown in Figure 3.7.

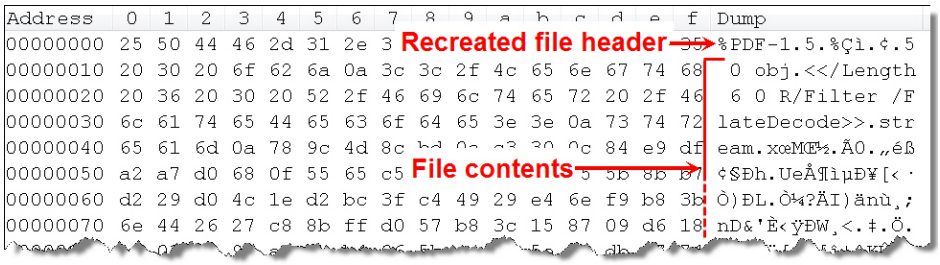


Figure 5.10: Decrypted NotPetya File

Using this technique pdf, doc, docx, xls and xlsx extracted control files were successfully recovered using the same AES key. Each of these file types required different headers to be inserted and some files required that some bytes be removed from the end of the file. All the commands used to do this are provided in Appendix B.1. In the future this logic could be incorporated in to the `decrypt.py` tool directly.

5.3 Bad Rabbit

The sample described in Table 5.2 was used during these experiments.

Name	Bad Rabbit
SHA256	630325cac09ac3fab908f903e3b00d0dadd5fdaa0875ed8496fcbb97a558d0da
URL	https://www.hybrid-analysis.com/sample/630325cac09ac3fab908f903e3b00d0dadd5fdaa0875ed8496fcbb97a558d0da?environmentId=100

Table 5.2: Bad Rabbit Sample Details

The execution of this ransomware followed the description provided by (Perekalin, 2018; Mamedov et al., 2018; 2017 Malwarebytes LABS, 2017) and is similar to the steps used by the NotPetya ransomware.The main steps being:

1. Adding persistence.
2. Scanning the machine for files to encrypt. This sample seems to ignore control files with the 'txt' and 'jpg' file extensions.
3. Encrypting the identified files using AES encryption with what appears to be the same AES key being used for all the files.
4. After 14 minutes a ransom note file is created on the C drive.
5. One minute later the machine is automatically rebooted.
6. The machine restarts with what appears to be a normal windows desktop. However in the background the MBR is being encrypted.
7. 22 minutes after the initial execution of the ransomware, the machine automatically reboots again.
8. After reboot, the ransomware message shown in Figure 5.11 is displayed and the user is prevented from accessing their windows installation.

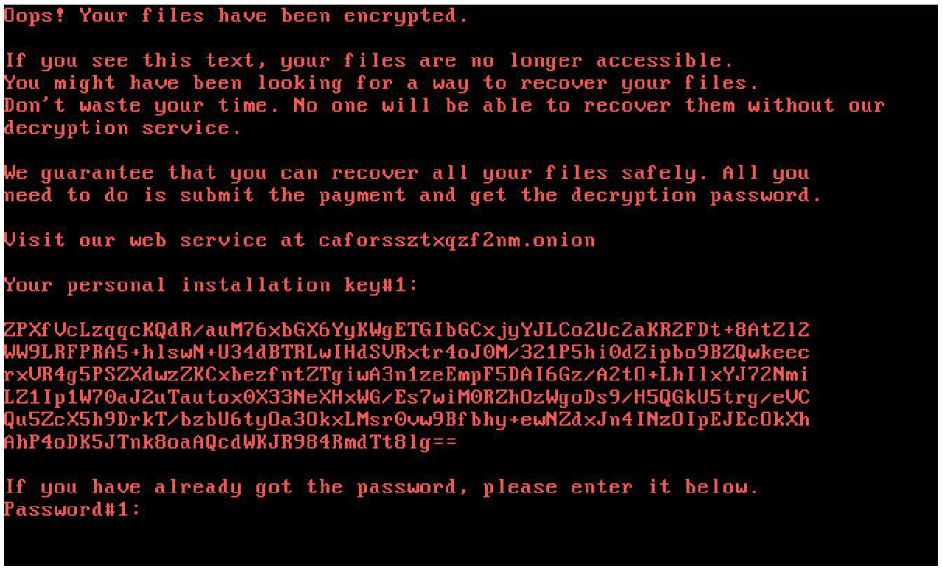


Figure 5.11: Bad Rabbit Ransom Note

5.3.1 Experiment 1 – Is the key in memory

All three live forensics tools used to examine the memory dumps were able to identify AES keys in memory, some of the found keys were ignored as they were present prior to the execution of the ransomware. However all the tools also successfully identified the key used by the ransomware to encrypt the files. The key used in this experiment is shown below in Figure 5.12.

```
cb 84 e1 eb 11 b2 03 85 fe d4 af a4 10 21 c8 f8
```

Figure 5.12: Bad Rabbit AES Encryption Key

The tool interrogate gave the output shown in Figure 5.13 below.

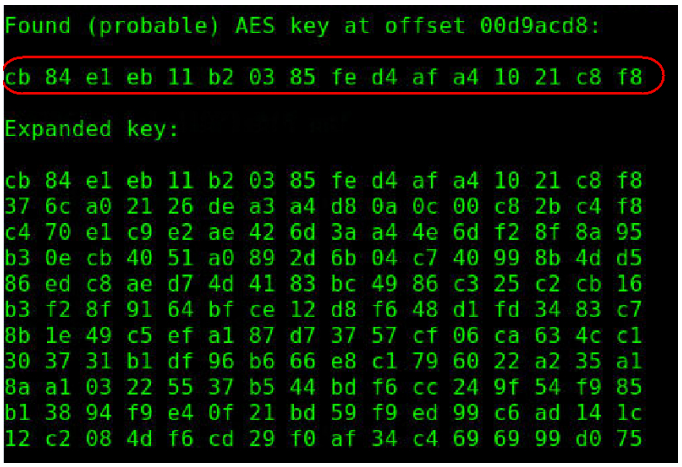


Figure 5.13: Bad Rabbit Interrogate Output

The tool findaes gave the output shown in Figure 5.14 below.

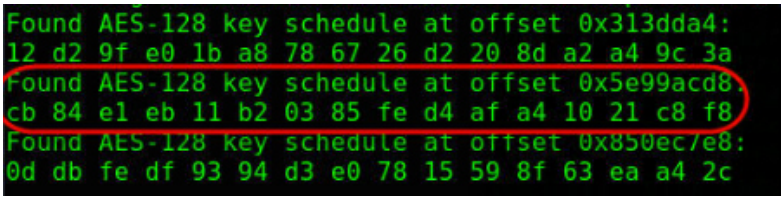


Figure 5.14: Bad Rabbit findaes Output

The RansomAES tool gave the output shown in Figure 5.15 below:

```
root@kali:~/dissertation/findaes/ransomaes# ./ransomaes -p 2256 -t Win7SP0x86 /mnt/NetworkShar
e/diss-labs/experiments-1/windows7/BadRabbit/windows7-badrabbit-2.elf
Looking for process id: 2256
Using profile type: Win7SP0x86
Command is: /usr/bin/volatility --profile Win7SP0x86 -f /mnt/NetworkShare/diss-labs/experiment
s-1/windows7/BadRabbit/windows7-badrabbit-2.elf memdump -p 2256 --dump-dir .
Volatility Foundation Volatility Framework 2.6
************************************************************************
Writing rundll32.exe [  2256] to 2256.dmp
Command is: 2256.dmp
Searching 2256.dmp
Found AES-128 key schedule at offset 0x214740:
cb 84 e1 eb 11 b2 03 85 fe d4 af a4 10 21 c8 f8
Found AES-128 key schedule at offset 0x9ff380c:
12 d2 9f e0 1b a8 78 67 26 d2 20 8d a2 a4 9c 3a
```

Figure 5.15: Bad Rabbit RansomAES Output

5.3.2 Experiment 2 – How long is the key present

For each operating system a total of fifteen memory dumps were taken at regular intervals during the execution of the ransomware. These dumps were then analysed to determine in which of them, the AES keys were present. When reviewed it was identified that the key became available within 1 minute of the start of the ransomware execution, and only remained in memory while the encryption was being done. An approximation of this being 30 seconds. The key did not appear again even after the machine reboot. A graphical representation of some of the ransomware's behaviour and key availability is shown below in Figure 5.16 below.

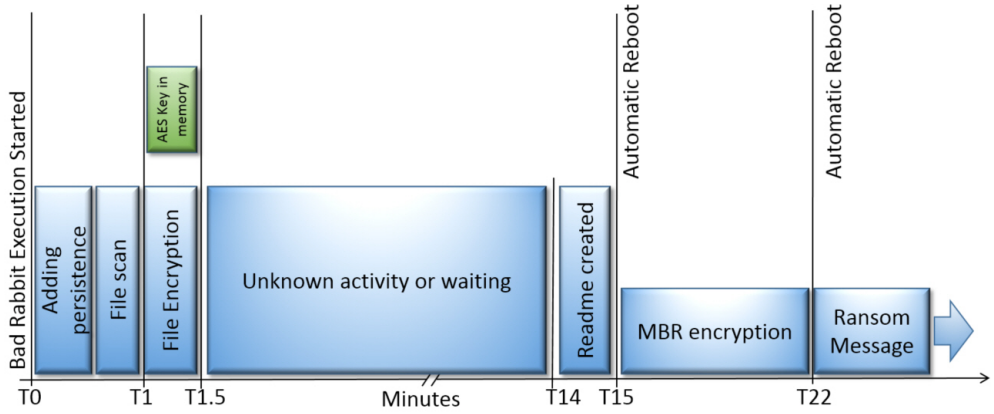


Figure 5.16: Bad Rabbit Timeline

This timeline correlates well with the findings of other researchers who have analysed the general behaviour of this ransomware from a malware perspective (2017 Malwarebytes LABS, 2017). However no work was found that analysed when and for how long the actual key remains in memory. Using this diagram

it is clear to see that it is only present for 30 seconds which is a fraction of the overall execution time and much shorter than the NotPetya ransomware. The key could possibly be present at other times, but missed due to the sampling period.

Also no similar graphical representation of the ransomware time line was found in the literature, the one shown in Figure 5.16 being generated by the author. It is felt that this type of representation aids understanding of the overall behaviour of the ransomware and could provide an area for future research.

5.3.3 Experiment 3 – Does the key decrypt files

Files are encrypted using the same format as the NotPetya ransomware and the steps described in section 5.2.3 can be used to decrypt the files encrypted with the Bad Rabbit ransomware. Using this technique pdf, doc, docx, xls and xlsx files were successfully recovered using the same AES key.

5.4 Phobos

The sample described in Table 5.3 was used during these experiments.

Name	Phobos
SHA256	a91491f45b851a07f91ba5a200967921bf796d38677786de51a4a8fe5ddeafd2
URL	https://any.run/report/a91491f45b851a07f91ba5a200967921bf796d38677786de51a4a8fe5ddeafd2/deebe359-037a-405d-a863-3c72cc3d8442

Table 5.3: Phobos Sample Details

The execution of the ransomware followed the description provided by (Issa, 2019). The main steps being:

1. Adding persistence and gathering credentials.
2. Scanning the machine for files to encrypt. This sample encrypted all the control files.
3. Encrypting the identified files using AES encryption with what appears to be the same AES key being used for all the files initially encrypted. The file names were also changed.
4. After approximately 2 minutes the ransom note shown in Figure 5.17 is displayed.

Figure 5.17: Phobos Ransom Message

5. Interestingly the machine still operates to some extent and any new files created are encrypted using a different AES key. The researcher was not able to discover this secondary AES key. This could be an area of further research as these keys must be accessible somehow.

6. The machine does not reboot automatically. If the user reboots the machine, then the same ransomware message is displayed.

5.4.1 Experiment 1 – Is the key in memory

All three live forensics tools used to examine the memory dumps were able to identify AES keys in memory, some of the found keys were ignored as they were present prior to the execution of the ransomware. However all the tools also successfully identified the 256 bit key used by the ransomware to encrypt the files. The key used in this experiment is shown below in Figure 5.18.

```
96 87 41 3b a3 80 de a3 11 a4 d5 56 13 2d 42 bf 44 31 34 1f 49 67 45 8b 83 c5 77 d4 33 35 bb 74
```

Figure 5.18: Phobos AES Encryption Key

The tool interrogate gave the output shown in Figure 5.19 below.

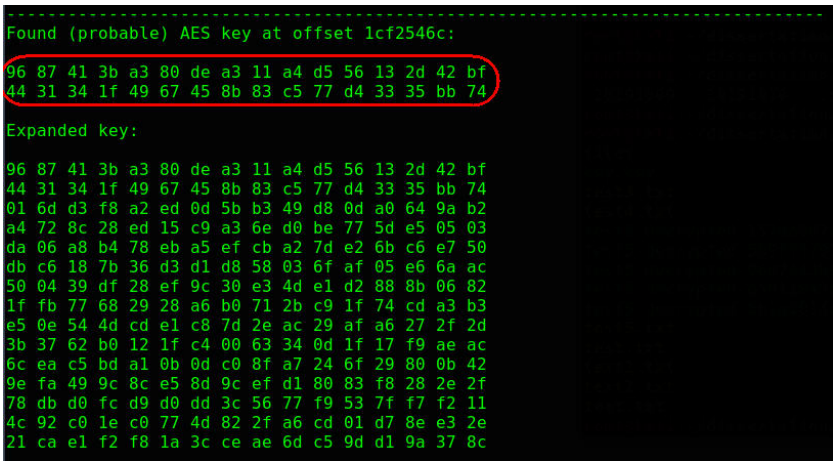


Figure 5.19: Phobos Interrogate Output

The tool findaes gave the output shown in Figure 5.20 below:

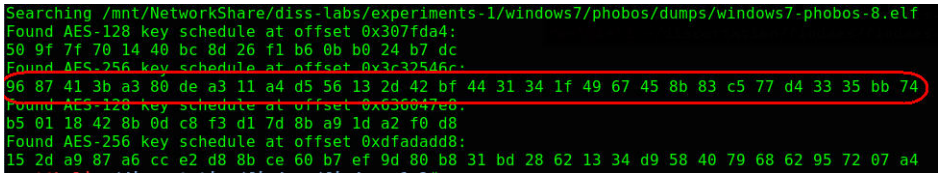


Figure 5.20: Phobos findaes Output

The RansomAES tool gave the output shown in Figure 5.21 below:

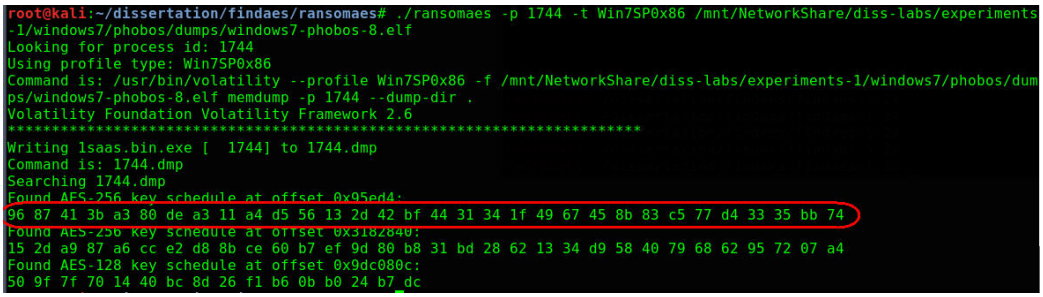


Figure 5.21: Phobos RansomAES Output

5.4.2 Experiment 2 – How long is the key present

For each operating system a total of fifteen memory dumps were taken at regular intervals during the execution of the ransomware. These dumps were then

analysed to determine in which of them, the AES keys were present. When reviewed it was identified that the key initially became available within 1 minute of the start of the ransomware execution. The same AES key was loaded in to memory and removed several times during this initial encryption of the machine. The key was erased when the ransom message was displayed. The ransomware continued to encrypt any new files created, using a different AES key. Several unsuccessful attempts were made to try and capture this secondary key from memory. A graphical representation of some of the ransomware's behaviour and key availability is shown in 5.22 below.

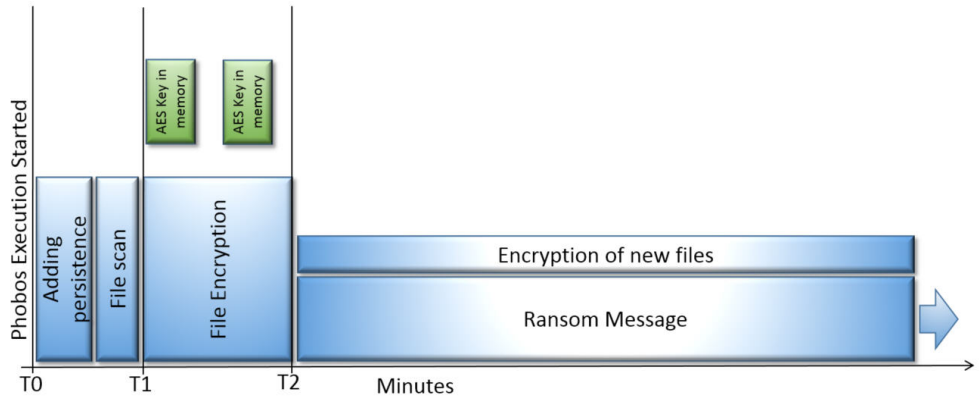


Figure 5.22: Phobos Timeline

This timeline correlates well with the findings of other researchers who have analysed the general behaviour of this ransomware from a malware perspective (Issa, 2019; Panda Security, 2017). However no work was found that analysed when and for how long the actual key remains in memory. Using this diagram it is clear to see that the same AES key is present in memory on several different occasions. It is also worth bearing in mind that there could be occasions where the keys presence was missed due to the sampling period. No similar graphical representation of the ransomware time line was found in the literature, the one shown in Figure 5.22 being generated by the author. It is felt that this type of representation aids understanding of the overall behaviour of the ransomware and could provide an area for future research.

5.4.3 Experiment 3 – Does the key decrypt files

Additional information is added to the end of a file that has been encrypted by the Phobos ransomware. This extra information includes some padding, followed by what is believed to be the AES IV value, then followed by 128 bytes

that is the same for all the encrypted files. A detailed description of the encrypted file appears in the work by Issa (Issa, 2019) who hypothesis that this 128 byte block could be the encrypted asymmetric key. There is also a fixed string at the end of the block, in this case it is 'LOCK96' but other versions of Phobos have been observed with different keywords such as 'DAT260'.

A program was developed that firstly reads the IV value from the encrypted file, then uses this together with the key found in experiment-1 to decrypt the file. The process used to recover a text file is shown below.

The content of a text file encrypted by the Phobos ransomware is show in Figure 5.23.

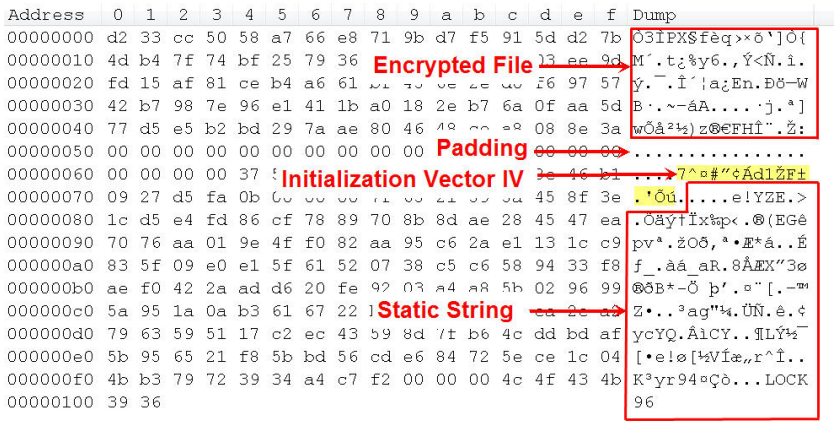


Figure 5.23: File Encrypted by Phobos

After execution of the decryption program using the AES key extracted from the memory dump, the contents of the file looks like Figure 5.24.

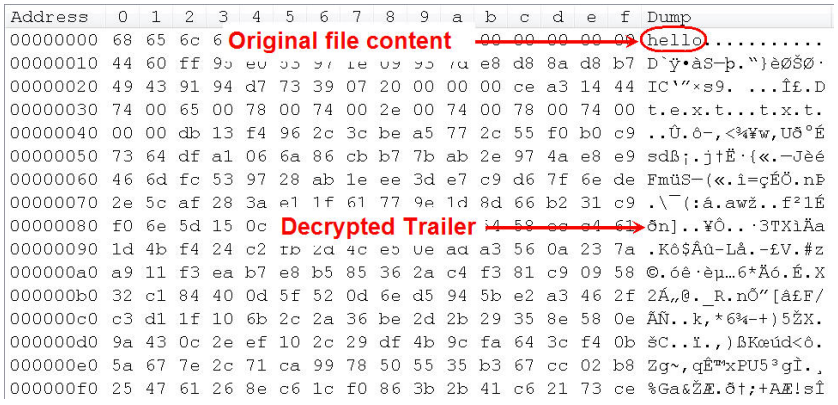


Figure 5.24: Partially Decrypted Phobos File

The final step is to remove the trailer added by Phobos. This trailer is a constant length so the following command can be used.

```
dd if=<decrypted file> of=<final file> bs=$((‘cat text.txt| wc -c‘
    - 225)) count=1
```

The resulting decrypted file without the trailer is exactly the same as the file, prior to encryption. Using this technique pdf, jpg, txt, doc, docx, xls and xlsx files were successfully recovered using the same AES key.

5.5 Conclusions

All three live forensic tools successfully identified the AES key used by all three ransomware samples tested in these experiments, supporting the hypotheses that live forensic techniques could be used to mitigate a ransomware attack. These results correlate well with the findings of other researchers who have worked on AES key recovery from volatile memory (Maartmann-Moe et al., 2009; Balogh & Pondelik, 2011; Hargreaves & Chivers, 2008), though none of them directly with ransomware in particular.

It can be seen from the generated time lines that for two of the ransomware samples the found keys were only available in memory for a limited time and once the encryption completed they were removed from the memory, did not return and were not recoverable. Any encryption performed after the original AES keys used for the initial encryption have been removed, is achieved using a secondary AES key which could not be recovered using the same technique. In all cases the key does not survive a machine reboot. The graphical timelining of the execution proved a valuable tool for understanding the behaviour of the ransomware and this together with more analysis of the the secondary AES keys used for encryption after the initial run could be good candidates for further research work.

Similar execution times and results were recorded for the findaes and RansomAES live forensic tools, while the time taken for the interrogate tool to complete was almost 100 times longer. Apart from this extended execution time the final results from all three tools were identical. The author believes that the increased execution time of the interrogate tool could be caused by the implementation of the logic this tool used to identify the AES key as fundamentally it is based on the same research as findaes (Trenholme, 2014). Also, this tool is more generic than the findaes tool as it is also able to find other key types such as RSA and this extra logic could possibly be the reason for the degradation in perform-

ance of the tool. This extreme computation time means that the interrogate tool does not lend itself for use in this type of analysis. A separate test was performed to determine if this tools logic would help in identifying AES keys for the 'wannacry' ransomware, but it was not able to extract any useful information, despite its enhanced functionality. The RansomAES tool created by the author, which combined the logic of the volatility framework (Volatility, 2019) together with the logic from the findaes (Kornblum, 2019), had similar results as the pure findaes tool, indicating that the added extra functionality for ransomware did not result in an improvement in performance. However the time spent on developing this tool was limited and it may prove beneficial to spend some more time in trying to improved its design and hence performance.

The three chosen ransomware samples were similar to each other in that they all used AES symmetric encryption and also all used the same key for all files encrypted. Other ransomware samples that used different keys for each file, different encryption algorithms such as DES or purely asymmetric encryption were rejected.

It has been proposed (LogRhythm Labs, 2017) that the encryption techniques employed by the Bad Rabbit ransomware are different from the ones used by NotPetya. However, this is not supported by the results from our experiments as it seems that both use a single AES key for all the encrypted files and the format of the encrypted files are the same. The only recorded difference between the two programs with regards to encryption appears to be that the Bad Rabbit ransomware removes the keys from memory immediately after the encryption has completed, thus reducing the window of opportunity to capture the key from memory.

A significant amount of time was spent researching the formats of the files encrypted by the ransomware samples (Sood & Hurley, 2017; Mamedov et al., 2018; Issa, 2019). It seems that each ransomware uses a different file format and it was necessary to discover the format used so that the IV could be determined. Without this value it would be much harder to decrypt the files even when the encryption key was known. Although the papers produced by Issa (Issa, 2019) on the Phobos sample, Sood (Sood & Hurley, 2017) on the Petya sample and Mamedov (Mamedov et al., 2018) on the BadRabbit sample, were not peer reviewed, the fact the information provided by them, gave the author enough information to be able to decrypt the file, strongly suggests that the information contained within these papers is accurate and reliable.

The information presented in this chapter together with the discussions, evaluation and comparison aligns well with the learning outcome three mainly, however some parts should also contribute to learning outcome two.

Chapter 6

Conclusion

All three live forensic tools successfully identified the AES key used by all three ransomware samples tested in these experiments, supporting the hypotheses that live forensic techniques could be used to mitigate a ransomware attack. While there is no known similar research in this specific area, these results do correlate well with the findings from other researchers who have worked on AES key recovery from volatile memory (Maartmann-Moe et al., 2009; Balogh & Pondelik, 2011; Hargreaves & Chivers, 2008) in similar areas such as key recovery from TrueCrypt or Skype applications.

However these results support the hypothesis with some caveats. These being that the machine has not been rebooted since the commencement of the attack, and also that the memory capture is performed near the start of the ransomware's execution, to ensure that the required keys are still present in memory. Also this technique cannot be applied to all ransomware families, as demonstrated by the analysis of the 'wannacry' sample, where no AES keys were able to be identified.

It was also observed that in cases of a ransomware attacks where the user was still able to interact with the operating system after the initial encryption has completed, a different AES key was used to perform any subsequent encryption. It is still believed by the author and corroborated by others (Hargreaves & Chivers, 2008; Balogh & Pondelik, 2011), that the AES keys must be in memory for this secondary encryption to take place. A further area of research could be to analyse the ransomware and memory dumps further to determine if capture of these secondary keys is also possible.

6.1 Aims and Objectives

To critically assess the success of this project, it should be evaluated against its original aims and objectives bearing in mind that the overall objective was the evaluation of live forensics techniques in ransomware attack mitigation. The stated aims of the project being:

1. Conduct a critical literature review on the subject of ransomware and

live forensics. Included in this would be an analysis of the current status of ransomware attacks, trends, techniques for delivery, general steps involved in such an attack as well as live forensic techniques relating memory analysis with regards to cryptanalysis.

2. Develop and validate a robust, isolated, realistic test environment that can be used to execute the designed experiments.

3. Building on the findings from the literature review, design and develop specific experiments to test the hypothesis that live forensic techniques may be used in mitigating the affects of a ransomware attack. Using three different live forensic tools on three examples of ransomware. These experiments being run on two different operating systems.

4. Discuss and evaluate the results and findings from the experiments and compare them to similar research on the subject. Draw conclusions from the experiments, results and findings and critically evaluate the project.

6.2 Objective One – Literature Review

The outcome from the literature review is presented in Chapter 2. It is divided in to two main parts; firstly on ransomware and secondly on live forensic techniques focusing on AES key recovery.

The review begins with a brief history of ransomware, documenting recent well know attacks before moving on to discuss the current threat landscape, confirming that the risk of a ransomware attack remains a current and real one and a valid field that warrants further research. A description using the authors preferred six phase model (McAfee Labs, 2016) to describe the behaviour of the ransomware is given. This model was selected over an alternative three phase models proposed (Kumar & Kumar, 2013; Gazet, 2010), as it provides a finer granularity for the description of the program's actions, allowing the model to capture more precise details of the execution process and timeline. However it is harder to implement than the three phase model.

The preferred model also provided techniques for comparing different ransomware sample behaviour and aided in the identification of samples that exhibited the preferred behaviour for this project. Guidance when selecting the ransomware samples was also derived from this literature review as the author wished to use recent, high profile samples that had a wide spread well documented impact and would be familiar to a wider audience.

The chapter then moves on to cover the subject of live forensics, starting with

descriptions of the different terminology and techniques used in this field. While no specific research papers in to using live forensic techniques to combat ransomware were found, several other papers describing similar techniques to recover artifacts from other cryptographic program were presented, the key papers that most closely resembled the work performed on this project being Maartmann-Moe (Maartmann-Moe et al., 2009), Balogh (Balogh & Pondelik, 2011) and Shamir (Shamir & Van Someren, 1998). Information and recommendations gathered from these papers contributed significantly to the design of the environment and experiments shown in Chapter 3 and their implementation shown in Chapter 4.

Some of the tools identified in these papers were also tested during the experimentation phase of this project. In all experiments, with all the used forensic analysis tools, the researcher was able to recover the AES keys, mirroring the success that authors of these reference papers had also achieved.

A part of this review also describes the techniques these tools employed to identify the keys used by the ransomware when it encrypts the files. How these tools use entropy and the existence of the pre-calculated key schedule or in some cases both to be able to identify the AES keys amongst the other data structures in the memory dump.

All of the proposed goals for this aim have been met and fulfilled the requirements for objective one.

6.3 Objective Two – Experiment Design

The design chapter took the findings from the literature review and using a good scientific approach applied them to designing and developing a system suitable for testing the hypothesis.

The design process was divided in to two distinct parts; the first concerned with the underlying environment design and the second relating to the actual experimental process design.

The environmental design relied heavily on the descriptions found in literature on how ideal test environments should be configured when working with malware samples (Rossow et al., 2012; Sanabria, 2007; Bose, 2018). These recommendations were combined with examples found in the literature review of environments used by researchers working in similar areas (Maartmann-Moe et al., 2009; Balogh & Pondelik, 2011; Shamir & Van Someren, 1998) to produce the final design. It is felt that the resulting design used in this project

provided a realistic, flexible and safe environment allowing the researchers to define and develop the experiments in a robust and isolated environment. Satisfying the criteria for a good environment design as stipulated in the literature (Bose, 2018; Sanabria, 2007).

While no experiments were found in the literature that could be directly utilized to test the hypothesis, good experiment design techniques (Edgar & Manz, 2017a) were used to develop the three hypothetico-deductive experiments where the dependant and independent variables were clearly identified. Basing the experiment design on these principles provided a high degree of certainty over the quality of the results achieved and allowed the findings from these experiments to be compared to the findings from experiments performed in related fields.

There were several options available for memory acquisition and during this project it was decided to utilize the hypervisor tools to capture the memory from outside the system. The benefits of non contamination of the memory dumps by the acquisition method while relying on product specific hypervisor tools was considered to be a better alternative than using memory capture tools from within the operating system of the victims machine which while being more generic, would contaminate the captured memory.

On review the work performed mainly in Chapter 3 and partially in Chapter 4 fully met the requirements for objective two.

6.4 Objective Three – Design and Implementation

The design of the environment and experiments is covered in Chapter 3 and their implementation in Chapter 4. During the execution of the ransomware samples as part of the experiments, no external effects of their execution were observed and no assets outside the experimental environment were affected. This is considered to be a good endorsement of quality of the implemented experimental environment.

Despite the fact that there were no identifiable differences between the execution times or results recorded when investigating the ransomware samples on Windows 7 or Windows 10 operating system, it is felt that the extra effort involved in performing these parallel experiments was worthwhile as it allowed the researcher to draw realistic conclusions from the experiments (Rossow et al., 2012) as well as providing a platform for defending the findings and drawn conclusions. It also provided a defence against possible criticisms that could otherwise be raised about the experiments being unrealistic or too specific.

The experimental environment was designed with an additional third machine present on the virtual network, identified as the Debian machine. The purpose of this machine was to supply network services to the victims machine (Sanabria, 2007). However during the execution of the three ransomware samples selected for investigation, no network traffic was observed leading the author to conclude that the ransomware samples used were self contained and did not require network access to function correctly. Even though no network traffic was recorded during these experiments, it was still considered worthwhile to have this Debian machine and its associated services available on the network. The reason for this being that prior to execution of the ransomware, the researcher could not know what network services the sample would require, these would only become apparent after examination of the Debian machine logs on completion of the ransomware's execution. Also the presence of this Debian machine on the network may have misled the ransomware to behave normally as it believed that it was executing on a typical network and not a simulated one (Egele et al., 2012). However this benefit was not investigated as part of the experiments.

Some issues were encountered during the selection of the ransomware samples that would be included in the investigation. For example, the researchers were not able to capture any keys when executing the 'wannacry' ransomware sample, while this could have been used as a good example for a null hypothesis (Edgar & Manz, 2017a), it was decided to reject this sample and focus the research on samples where the capture was successful. Difficulties were also encountered in having some of the other ransomware samples execute correctly, or in a way that was described in the literature and for these reasons, they were also rejected.

Apart from these minor issues, the environment performed in a safe and flexible manner (Sikorski & Hong, 2012) allowing the experiments to be executed and the results collected without encountering any issues or having any 3^{rd} party computer assets affected.

On review the work performed mainly in Chapter 4 and partially in Chapter 3 fully met the requirements for objective three.

6.5 Objective Four – Evaluation

All three live forensic tools successfully identified the AES key used by all three ransomware samples tested in these experiments, supporting the hypotheses that live forensic techniques could be used to mitigate a ransomware attack.

These results correlate well with the findings of other researchers who have worked on AES key recovery from volatile memory (Maartmann-Moe et al., 2009; Balogh & Pondelik, 2011; Hargreaves & Chivers, 2008), though none of them directly with ransomware in particular.

It can be seen from the generated time lines that for two of the ransomware samples the found keys were only available in memory for a limited time and once the encryption completed they were removed from the memory, did not return and were not recoverable. Any encryption performed after the keys used for the initial encryption have been removed is achieved using a secondary AES key which could not be recovered using the same technique. In all cases the key does not survive a machine reboot. The graphical timelining of the execution proved a valuable tool in understanding the behaviour of the ransomware and this together with more analysis of the the secondary AES keys used for encryption after the initial run could be good candidates for further research work.

The performance of the findaes and RansomAES live forensic tools were similar and were both successful in identifying the correct AES keys. Either of these can be recommended as a useful tool when performing this type of research. In contrast, it is felt that the performance of the interrogate tool prohibits is use in this type of investigation. The RansomAES tool created by the author, which combined the logic of the volatility framework (Volatility, 2019) together with the logic from the findaes (Kornblum, 2019), had similar results as the pure findaes tool, indicating that the added extra functionality for ransomware did not result in an improvement in performance. However the time spent on developing this tool was limited and it may prove beneficial to spend some more time trying to improved its design and hence performance.

The three chosen ransomware samples were similar to each other in that they all used AES symmetric encryption and also all used the same key for all files encrypted. This sample selection may have contributed to the fact that the project was successful in identifying the AES keys for all three samples. Other ransomware samples that used different keys for each file, different encryption algorithms such as DES or purely asymmetric encryption were rejected prior to the commencement of the experiments.

A significant amount of time was spent researching the file formats of the files encrypted by the ransomware samples (Sood & Hurley, 2017; Mamedov et al., 2018; Issa, 2019). It seems that each ransomware uses a different file format and it was necessary to discover the format used so that the IV could be determined. Without this value it would be much harder to decrypt the files even when the encryption key was known.

The success of the experiments combined with the analysis of the results, the evaluation performed in Chapter 5 and the discussion in this conclusion chapter fully meet the requirements for objective four.

6.6 Self Appraisal

The fact that the project was able to confirm that some live forensic techniques could prove useful when dealing with some types of ransomware attacks is considered by the author to be a successful outcome of the investigation, especially as the main aims of the project were also met. Another positive achievement being the design and implementation of a safe, robust and flexible test environment that allowed the researcher to perform the experiments without any adverse affects on external systems.

However, on reflection, if the project would be repeated, then a reduced scope should be considered as too many research questions were attempted to be answered, resulting in the necessity of conducting a large set of experiments.

In summary the following themes were investigated. Ransomware execution timelining, decryption techniques of files encrypted by ransomware, identification and extract of AES keys from memory, multiple ransomware sample comparisons, AES extraction tool comparison and ransomware behaviour on multiple operating systems.

While some of these combinations were useful and enhanced the overall relevance of the project, such as using the found AES keys to decrypt the file. This both confirmed that the correct AES keys were found and that the decryption technique was valid. Another example being the testing on multiple operating systems. While this gave the researcher the ability to draw more realistic conclusions from the results, it also doubled the number of experiments that were required to be performed. The overall number of experiments conducted was quite high putting a significant workload on the project.

One solution would have been to run separate projects each investigating a subset of this projects scope.

Additional ideas for further research generated during this projected were recorded and are described in section 6.7

6.7 Future Work

It has been shown that ransomware is and continues to be, a significant threat to individuals and organisations so further research into some of the techniques identified and utilized during this investigation could prove to be beneficial in supporting users strengthen their systems and provide tools that would be useful in mitigating future attacks. Some specific topics covered in this research that could benefit from further investigation are:

Timelining of ransomware execution. One way to understand and possibly categorise ransomware execution is the graphical timelining of the steps involved and the consequences of each phase of the ransomware execution. This approach proved very useful during this project and it is believed that further investigation and research could result in some useful findings. Currently there does appear to be much work being performed on this subject and the little research that does appear in the literature use differing notation systems and granularity.

Specific tool development Mostly standard tools were used to perform the AES key discovery during this project. These tools were initially developed for cases of key discovery in different circumstances and for different cryptographic programs. One area of further research could be to develop a tool specifically for ransomware with the aim of using its particular behaviour to design a tool with greater performance and accuracy.

Integration in to Volatility Framework(Volatility, 2019). This is an open framework tool used for live forensics. It is designed to allow 3rd party developers to create plugins for this framework. A good future area of research would be to develop a specific plugin for ransomware that could either be used to identify the presence of ransomware within the system or key recovery for attack mitigation and recovery.

Memory capture techniques The technique used during this project relied on the fact that the victim's system was running as a virtual machine on a VirtualBox hypervisor. Further research could be performed to try and identify a more generic solution for memory acquisition that could be used both with the virtual machine, within a normal machine (non virtual) or from other types of hypervisor implementations.

Real time memory monitoring During these experiments, memory dumps were analysed to discover the AES keys. Further research could be to try and adapt these analysis techniques so that the memory is analysed directly and not via a dump file. An expected result of this approach would be an increase in per-

formance and a reduction in the AES key discovery time as currently there is a delay between initiating the memory dump and the dump becoming available for analysis.

Post infection investigation It was observed that when the user still had access to the computer after it had been infected, a different AES key than the one originally used for the initial encryption, was being used to encrypt the new files. Further research in to how to capture this secondary AES key would be helpful in mitigating this type of ransomware attack.

References

2017 Malwarebytes LABS. (2017). *Bad Rabbit: a closer look at the new version of Petya/NotPetya.* Retrieved 2019-08-23, from https://blog.malwarebytes.com/threat-analysis/2017/10/badrabbit-closer-look-new-version-petyanotpetya/

2019 Malwarebytes LABS. (2019). *State of Malware* (Tech. Rep.). Retrieved from https://resources.malwarebytes.com/resource/2019-state-malware-malwarebytes-labs-report/

Ahmad, M. A., Woodhead, S. & Gan, D. (2016). The V-network testbed for malware analysis. *Proceedings of 2016 International Conference on Advanced Communication Control and Computing Technologies, ICACCCT 2016*, 629–635. doi: 10.1109/ICACCCT.2016.7831716

Ahmadian, M. M., Shahriari, H. R. & Ghaffarian, S. M. (2016). Connection-monitor & connection-breaker: A novel approach for prevention and detection of high survivable ransomwares. *12th International ISC Conference on Information Security and Cryptology, ISCISC 2015*, 79–84. doi: 10.1109/ISCISC.2015.7387902

Akkas, A., Chachamis, C. N. & Fetahu, L. (2017). Malware Analysis of WanaCry Ransomware. *Semantics Scholar*, 1–11.

Al-rimy, B. A. S., Maarof, M. A. & Shaid, S. Z. M. (2018). Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers and Security*, *74*, 144–166. Retrieved from https://doi.org/10.1016/j.cose.2018.01.001 doi: 10.1016/j.cose.2018.01.001

Balogh, Š. & Pondelik, M. (2011). Capturing encryption keys for digital analysis. *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS'2011*, *2*(September), 759–763. doi: 10.1109/IDAACS.2011.6072872

Bashir, M. S. & Khan, M. N. A. (2013). Triage in Live Digital Forensic Analysis. *The International Journal of Forensic Science*, 35–44. doi: 10.5769/J201301005

BCS. (2015). *conduct.pdf* (No. June). The British Computer Socienty. Retrieved 2019-08-27, from https://www.bcs.org/upload/pdf/conduct.pdf

Belkasoft. (2019). *RamCapturer.* Retrieved 2019-09-01, from https://belkasoft.com/ram-capturer

Bellardo, J. & Savage, S. (2003). 802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions. *Usenix*, 1–13. Retrieved from http://

cseweb.ucsd.edu/{~}savage/papers/UsenixSec03.pdf

Berry, A., Homan, J. & Eitzman, R. (2017). *Threat Research WannaCry Malware Profile*. Retrieved 2019-08-20, from https://www.fireeye.com/blog/threat-research/2017/05/wannacry-malware-profile.html

Bose, M. (2018). *A Complete Comparison of VMware and VirtualBox*. Retrieved 2019-09-20, from https://www.nakivo.com/blog/vmware-vs-virtual-box-comprehensive-comparison/

Bradley, S. (2016). *Information Security Reading Room Ransomware* (Tech. Rep.). SANS Institute. Retrieved from https://www.sans.org/reading-room/whitepapers/awareness/paper/37317

Carvey, H. (2018). Setting Up A Testing Environment. *Investigating Windows Systems*, 97–115. doi: 10.1016/b978-0-12-811415-5.00005-6

Carvey, H. & Casey, E. (2009). *Windows Forensic analysis DVD toolkit* (2nd ed.). Burlington, MA : Syngress.

CERT-EU. (2017). *WannaCry Ransomware Campaign Exploiting SMB Vulnerability* (Tech. Rep.). Computer Emergency Response Team - EU. Retrieved from https://cert.europa.eu/static/SecurityAdvisories/2017/CERT-EU-SA2017-012.pdf

Choudhary, S. P. & Vidyarthi, M. D. (2015). A Simple Method for Detection of Metamorphic Malware using Dynamic Analysis and Text Mining. *Procedia Computer Science*, *54*, 265–270. doi: 10.1016/j.procs.2015.06.031

Comodo. (2018). *8 Ransomware Attacks that have Occurrred Recently*. Retrieved 2019-09-20, from https://enterprise.comodo.com/forensic-analysis/ransomware-attacks.php

Davies, D. B. (2019). *Ransomware Activity Declines, But Remains Dangerous Threat*. Retrieved 2019-09-05, from https://www.symantec.com/blogs/expert-perspectives/ransomware-activity-declines-remains-dangerous-threat

Dinaburg, A., Royal, P., Sharif, M. & Lee, W. (2008). Ether: Malware Analysis via Hardware Virtualization Extension. , 51–62.

Edgar, T. W. & Manz, D. (2017a). *Research Methods for Cyber Security*. Syngress.

Edgar, T. W. & Manz, D. (2017b). Research Methods for Cyber Security. In *Research methods for cyber security* (p. 213). Syngress, Boston.

Egele, M., Scholte, T., Kirda, E. & Kruegel, C. (2012). A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys*, *44*(2). doi: 10.1145/2089125.2089126

Eric Vanderberg. (2018). *Part 3: The 6 Phases of an Advanced Ransomware Threat*. Retrieved 2019-09-03, from https://www.tcdi.com/6-phases-advanced-ransomware-threat/

Europol. (2016). *INTERNET ORGANISED CRIME 2016 IOCTA* (Tech. Rep.). Retrieved from https://www.europol.europa.eu/

activities-services/main-reports/internet-organised-crime-threat-assessment-iocta-2016 doi: 10.2813/275589

Europol. (2018). *INTERNET ORGANISED CRIME 2018 IOCTA* (Tech. Rep.). Retrieved from https://www.europol.europa.eu/internet-organised-crime-threat-assessment-2018 doi: 10.2813/858843

F-Secure Labs. (2016). *Ransomware: How to prevent, predict, detect & respond* (Tech. Rep. No. November). F-Secure. Retrieved from https://www.f-secure.com/documents/996508/1030745/Ransomware{_}how{_}to{_}ppdr.pdf

Fruhlinger, J. (2019). *The 6 biggest ransomware attacks of the last 5 years*. Retrieved 2019-09-20, from https://www.csoonline.com/article/3212260/the-5-biggest-ransomware-attacks-of-the-last-5-years.html

F-Secure. (2017). *F-Secure State of Cyber Security* (Tech. Rep.). Retrieved from https://www.f-secure.com/content/dam/f-secure/en/labs/whitepapers/Cyber{_}Security{_}Report{_}2017.pdf

Furnell, S. & Security, C. (2017). The ABC of ransom- ware protection. *Computer Fraud and Security*(October), 5–11.

Galal, H. S., Mahdy, Y. B. & Atiea, M. A. (2016). Behavior-based features model for malware detection. *Journal of Computer Virology and Hacking Techniques*, *12*(2), 59–67. doi: 10.1007/s11416-015-0244-0

Gazet, A. (2010). Comparative analysis of various ransomware virii. *Journal in Computer Virology*, *6*(1), 77–90. doi: 10.1007/s11416-008-0092-2

Goldberg, R. P. (1974). Survey of Virtual Machine Research. *Computer*, *7*(6), 34–45. doi: 10.1109/MC.1974.6323581

Grégio, A. R. A., Afonso, V. M., Filho, D. S. F., Geus, P. L. D. & Jino, M. (2014). Toward a Taxonomy of Malware Behaviors. *Computer Journal*, *58*(10), 2758–2777. doi: 10.1093/comjnl/bxv047

Gupta, B. B., Arachchilage, N. A. & Psannis, K. E. (2018). Defending against Phishing Attacks : Taxonomy of Methods , Current Issues and Future Directions. *Telecommunication Systems*, *67*(2), 247–267. Retrieved from https://link.springer.com/article/10.1007/s11235-017-0334-z

Halderman, J. A., Schoen, S. D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J. A., ... Felten, E. W. (2009). Lest We Remember : Cold Boot Attacks on Encryption Keys. *Communications of the ACM,*, *52*(5), 91–98. doi: 10.1145/1506409.1506429

Hampton, N. & Baig, Z. A. (2015). Ransomware: Emergence of the cyber-extortion menace. *The Proceedings of the13th Australian Information Security Management*, *2015*, 47–56. doi: 10.4225/75/57b69aa9d938b

Hargreaves, C. & Chivers, H. (2008). Recovery of encryption keys from memory

using a linear scan. *ARES 2008 - 3rd International Conference on Availability, Security, and Reliability, Proceedings*, 1369–1376. doi: 10.1109/ARES.2008.109

Hautala, L. (2019). *States brace for ransomware assaults on voter registries*. Retrieved 2019-09-03, from https://www.cnet.com/news/wi-fi-6-is-barely-here-but-wi-fi-7-is-already-on-the-way/

Heninger, N. & Feldman, A. (2008). *AESKeyFind*. Retrieved from https://github.com/eugenekolo/sec-tools/tree/master/crypto/aeskeyfind/aeskeyfind

Hoopes, J. (2009). Chapter 6. Malware Analysis Solutions. In *Virtualization security protecting virtualized environments* (1st ed., chap. Chapter 6.). O'Reilly. Retrieved from https://learning.oreilly.com/library/view/virtualization-for-security/9781597493055/{#}toc

Intelligence, T. & Analysis, I. (2019). *2019 SonicWall Cyber Threat Report* (Tech. Rep. No. July). SonicWall. Retrieved from www.sonicwall.com

Issa, J. (2019). *A deep dive into Phobos ransomware*. Retrieved 2019-09-10, from https://blog.malwarebytes.com/threat-analysis/2019/07/a-deep-dive-into-phobos-ransomware/

Kapersky. (2018). *Top 5 most notorious cyberattacks*. Retrieved 2019-09-06, from https://www.kaspersky.com/blog/five-most-notorious-cyberattacks/24506/

Kaplan, B. (2007). *RAM is Key Extracting Disk Encryption Keys From Volatile Memory* (Doctoral dissertation, Carnegie Mellon). Retrieved from https://cryptome.org/0003/RAMisKey.pdf

Kaur, R. & Singh, M. (2014). A survey on zero-day polymorphic worm detection techniques. *IEEE Communications Surveys and Tutorials*, *16*(3), 1520–1549. doi: 10.1109/SURV.2014.022714.00160

Khan Academy. (2017). *The Scientific Method*. Retrieved 2019-09-15, from https://www.khanacademy.org/science/high-school-biology/hs-biology-foundations/hs-biology-and-the-scientific-method/a/the-science-of-biology

Klahr, R., Amili, S., Shah, J. N., Button, M. & Wang, V. (2019). Cyber Security Breaches Survey 2019. *Cyber Security Breaches Survey 2016*, 56. Retrieved from https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/521465/Cyber_Security_Breaches_Survey_2016_main_report_FINAL.pdf

Klein, T. (2017). *5 Phases of Ransomware Attacks*. Retrieved 2019-09-03, from https://www.edci.com/2017/03/5-phases-of-ransomware-attacks/

Kong, J. H., Ang, L. M. & Seng, K. P. (2015). A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environ-

ments. *Journal of Network and Computer Applications*, *49*, 15–50. Retrieved from http://dx.doi.org/10.1016/j.jnca.2014.09.006 doi: 10.1016/j.jnca.2014.09.006

Kornblum, J. (2019). *findaes*. Retrieved 2019-08-10, from http://jessekornblum.com/tools/findaes/

Kumar, S. & Kumar, M. R. (2013). Cryptoviral Extortion: A virus based approach. *International Journal of Computer Trends and Technology*, *4*(5), 1149–1153. Retrieved from http://www.ijcttjournal.org

Levy, J. & Cto, S. (2019). *Sophoslabs 2019 threat report* (Tech. Rep.). Sophos. Retrieved from https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/sophoslabs-2019-threat-report.pdf

Ligh, M. H., Case, A., Levy, J. & Walters, A. (2014). *The Art of memory Forensics*. Wiley.

Lin, H. (2019). *Six Steps of the Scientific Method*. Retrieved 2019-09-15, from https://www.thoughtco.com/steps-of-the-scientific-method-p2-606045

LogRhythm Labs. (2017). *Bad Rabbit Ransomware Technical Analysis*. Retrieved from https://logrhythm.com/blog/bad-rabbit-ransomware-technical-analysis/

Maartmann-Moe, C., Thorkildsen, S. E. & Årnes, A. (2009). The persistence of memory: Forensic identification and extraction of cryptographic keys. *DFRWS 2009 Annual Conference*, *6*, 132–140. doi: 10.1016/j.diin.2009.06.002

Malwarebytes. (2018). *What's new in TrickBot? Deobfuscating elements*. Retrieved from https://blog.malwarebytes.com/threat-analysis/malware-threat-analysis/2018/11/whats-new-trickbot-deobfuscating-elements/

Malwarebytes. (2019). *Cybercrime Tactics and Techniques Q1 2019* (Tech. Rep.). Retrieved from https://www.malwarebytes.com/pdf/labs/Cybercrime-Tactics-and-Techniques-Q1-2017.pdf

Mamedov, O., Sinitsyn, F. & Ivanov, A. (2018). *Bad Rabbit ransomware*. Retrieved 2019-10-15, from https://securelist.com/bad-rabbit-ransomware/82851/

Mbol, F., Robert, J.-M. & Sadighian, A. (2016). An Efficient Approach to Detect TorrentLocker Ransomware in Computer Systems. In *15th international conference, cans 2016* (pp. 532–41). Retrieved from http://www.worldwidewebsize.com. doi: 10.1007/978-3-319-48965-032

McAfee Labs. (2016). Understanding Ransomware and Strategies to Defeat it. *Network Security*, 1–18. Retrieved from https://www.mcafee.com/us/resources/white-papers/wp-understanding-ransomware-strategies-defeat.pdf

McLaren, P., Buchanan, W. J., Russell, G. & Tan, Z. (2019). Deriving ChaCha20 Key Streams From Targeted Memory Analysis. *Journal of Information Security and Applications*, *48*. Retrieved from http://arxiv.org/abs/1907.11941{%}0Ahttp://dx.doi.org/10.1016/j.jisa.2019.102372 doi: 10.1016/j.jisa.2019.102372

McLaren, P., Russell, G., Buchanan, W. J. & Tan, Z. (2019). Decrypting live SSH traffic in virtual environments. *Digital Investigation*, *29*, 109–117. doi: 10.1016/j.diin.2019.03.010

Mekynyk, S. A., Speier-Pero, C. & Connors, E. (2019). Blockchain is Vastly Overrated; Supply Chain Cyber Security is Vastly Underrated. *Supply Chain Management Review*, *June*. Retrieved from scmr.com

Microsoft. (2017). *Ransomware: A declining nuisance or an evolving menace?* Retrieved from https://www.microsoft.com/security/blog/2017/02/14/ransomware-2016-threat-landscape-review/

Microsoft Defender ATP Research Team. (2017). *Windows 10 platform resilience against the Petya ransomware attack*. Retrieved from https://www.microsoft.com/security/blog/2017/06/29/windows-10-platform-resilience-against-the-petya-ransomware-attack/

Morato, D., Berrueta, E., Magaña, E. & Izal, M. (2018). Ransomware early detection by the analysis of file sharing traffic. *Journal of Network and Computer Applications*, *124*(June), 14–32. Retrieved from https://doi.org/10.1016/j.jnca.2018.09.013 doi: 10.1016/j.jnca.2018.09.013

Nissim, N., Lahav, O., Cohen, A., Elovici, Y. & Rokach, L. (2019). Volatile memory analysis using the MinHash method for efficient and secured detection of malware in private cloud. *Computers & Security*, *87*.

NIST. (2001). Announcing the ADVANCED ENCRYPTION STANDARD (AES). *US Department of Commerce, National Institute of Standards and Technology*. doi: 10.6028/NIST.FIPS.197

O'Brien, D. (2017). *Internet Security Threat Report (ISTR) Ransomware 2017* (Tech. Rep.). Symantec. Retrieved from https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf

O'Donnall, L. (2019). *Coordinated Ransomware Attack Hits 23 Texas Government Agencies*. Retrieved 2019-09-03, from https://threatpost.com/coordinated-ransomware-attack-hits-23-texas-government-agencies/147457/

Open Preservation Soceity. (2019). *Open Preservation Society*. Retrieved 2019-09-20, from https://openpreservation.org/technology/corpora/govdocs/

Paik, J.-Y., Shin, K. & Cho, E.-S. (2016). Poster: Self-Defensible Storage Devices based on Flash memory against Ransomware. In *37th ieee symposium on security and privacy* (pp. 2–3).

Panda Security. (2017). Technical Analysis of Bad Rabbit. *Panda Security*, *5*, 1–5.

Perekalin, A. (2018). *Bad Rabbit: A new ransomware epidemic is on the rise*. Retrieved 2019-10-15, from https://www.kaspersky.com/blog/bad-rabbit-ransomware/19887/

Pettersson, T. (2007). Cryptographic key recovery from Linux memory dumps. *Chaos Communication Camp*, 1–14.

Prakash, K. P., Nafis, T. & Sankar Biswas, S. (2017). Preventive Measures and Incident Response for Locky Ransomware. *International Journal of Advanced Research in Computer Science*, *8*(5), 392–395. doi: 10.26483/ijarcs.v8i5.3311

Ptacek, T. (2008). *Recover a Private Key from Process Memory*. Retrieved from http://www.matasano.com/log/178/recovera-{%}0Aprivate-key-from-process-memory

Purplesec. (2019). *The Ultimate List Of Cyber Security Statistics For 2019*. Retrieved 2019-09-03, from https://purplesec.us/resources/cyber-security-statistics/

Richardson, R. & North, M. (2017). Ransomware: Evolution, Mitigation and Prevention. *International Management Review*, *13*(1), 10.

Rossow, C., Dietrich, C. J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., … Van Steen, M. (2012). Prudent practices for designing malware experiments: Status quo and outlook. *Proceedings - IEEE Symposium on Security and Privacy*, 65–79. doi: 10.1109/SP.2012.14

Ruff, N. (2008). Windows memory forensics. *Journal in Computer Virology*, *4*(2), 83–100. doi: 10.1007/s11416-007-0070-0

Sai, R. L. P. & Kumar, T. P. (2019). Reverse Engineering the Behaviour of NotPetya Ransomware. *International Journal of Recent Technology and Engineering*(6), 574–578.

Salvi, H. U. (2015). Ransomware : A Cyber Extortion. *Asian Journal of Convergence in Technology*, *II*(III 2350-1146).

Sanabria, A. (2007). *Malware Analysis : Environment Design and Architecture* (Tech. Rep.). SANS Institute. Retrieved from https://www.sans.org/reading-room/whitepapers/threats/paper/1841

Saravanan, M. & Mukesh, K. (2014). Forensic Recovery of Fully Encrypted Volume. *International Journal of Computer Applications*, *91*(7), 18–21. doi: 10.5120/15892-4896

Savage, K., Coogan, P. & Lau, H. (2015). *The evolution of ransomware* (Tech. Rep.). Symantec. Retrieved from https://www.symantec.com/content/en/us/enterprise/media/security{_}response/whitepapers/the-evolution-of-ransomware.pdf

Sgandurra, D., Muñoz-González, L., Mohsen, R. & Lupu, E. C. (2016). Auto-

mated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection. *Cornell University*(October). Retrieved from http://arxiv.org/abs/1609.03020

Shamir, A. & Van Someren, N. (1998). Playing 'hide and seek' with stored keys. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1648*, 118–124.

Shannon, C. (1948). A Mathematical Theory of Communication. *Bell System Technology*, *27*(I), 379–656.

Sihim, K.-a. (2016). A survey of public-key cryptographic primitives in wireless sensor networks. *IEEE Communications Surveys and Tutorials*, *18*(1), 577–601. doi: 10.1109/COMST.2015.2459691

Sikorski, A. & Hong, A. (2012). *Practical Malware Analysis*. San Francisco: No Starch Prtess.

Simic, S. (2019). *What is a Hypervisor? Types of Hypervisors 1 & 2*. Retrieved 2019-10-20, from https://phoenixnap.com/kb/what-is-hypervisor-type-1-2

Sittig, D. F. & Singh, H. (2019). *A Socio-Technical Approach to Preventing , Mitigating , and Recovering from Ransomware Attacks* (Tech. Rep.). PWC. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4941865

SonicWall. (2019). Unmasking the threats that target glkobal enterprises, governments abd SMBs. *Journal of Chemical Information and Modeling*, *53*.

Sood, K. & Hurley, S. (2017). *NotPetya Technical Analysis – A Triple Threat: File Encryption, MFT Encryption, Credential Theft*. Retrieved from https://www.crowdstrike.com/blog/petrwrap-ransomware-technical-analysis-triple-threat-file-encryption-mft-encryption-credential-theft/

Sophos. (2019). Ransomware: How an attack works - Sophos Community. *Sophos*, 1–2. Retrieved from https://community.sophos.com/kb/en-us/124699

Spiritual, L. (2017). *Computer Misuse Act 1990 (oct-2017)* (Vol. 18; Tech. Rep. No. October). UK Government. Retrieved from https://www.legislation.gov.uk/ukpga/1990/18/data.pdf

Statista. (2019). *Windows 7 and 10 adoption rate in North America and Western Europe from 2017 to 2019*. Retrieved 2019-09-20, from https://www.statista.com/statistics/897222/north-america-western-europe-windows-7-10-adoption/

Sultan, H., Khalique, A., Alam, S. I. & Tanweer, S. (2018). A SURVEY ON RANSOMEWARE: EVOLUTION, GROWTH, AND IMPACT. *International Journal of Advanced Research in Computer Science*, *9*(2), 187–192. Retrieved from http://dx.doi.org/10.26483/ijarcs.v9i2.5858Volu

doi: http://dx.doi.org/10.26483/ijarcs.v9i2.5858Volu

Symantec. (2014). *CryptoDefense, the CryptoLocker Imitator, Makes Over $34,000 in One Month.* Retrieved from https://www.symantec.com/connect/blogs/cryptodefense-cryptolocker-imitator-makes-over-34000-one-month

Symantec. (2019). *Internet Security Threat Report* (Vol. 24; Tech. Rep. No. February). Semantec. Retrieved from https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf

Trenholme, S. (2014). *findaes.* Retrieved 2019-09-01, from https://sourceforge.net/projects/findaes/

Vanderburg, E. (2019). *A Timeline of Ransomware Advances.* Retrieved 2019-09-02, from https://www.tcdi.com/ransomware-timeline/

Vipre Security. (2017). *WannaCry Technical Analysis : Support.* Retrieved 2019-09-24, from https://support.threattracksecurity.com/support/solutions/articles/1000250396-wannacry-technical-analysis

Virtualbox. (2019). *VirtualBox.* Retrieved 2019-09-20, from https://www.virtualbox.org/

Volatility. (2019). *Volatility Foundation.* Retrieved 2019-09-14, from https://www.volatilityfoundation.org/

Walters, A. & Petroni, N. L. (2007). Volatools: Integrating Volatile Memory Forensics into the Digital Investigation Process. *Black Hat DC*, 1–18.

Wang, P. & Wang, Y. S. (2015). Malware behavioural detection and vaccine development by using a support vector model classifier. *Journal of Computer and System Sciences*, *81*(6), 1012–1026. Retrieved from http://dx.doi.org/10.1016/j.jcss.2014.12.014 doi: 10.1016/j.jcss.2014.12.014

Young, A. & Yung, M. (1996). Cryptovirology: extortion-based security threats and countermeasures. *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, 129–140.

Zeltser, L. (2015). *5 Steps to Building a Malware Analysis Toolkit Using Free Tools.* Retrieved 2019-09-20, from https://zeltser.com/build-malware-analysis-toolkit/

Zhang, P. & Tan, Y. (2015). Hybrid concentration based feature extraction approach for malware detection. *Canadian Conference on Electrical and Computer Engineering*, *2015-June*(June), 140–145. doi: 10.1109/CCECE.2015.7129175

Appendices

Appendix A

Project management

Further evidence to support that the project has archived a high level of fulfillment of learning outcome 2. Below is presented evidence of project management including project plans, Gantt charts and project diaries.

A.1 Project proposal

This was agreed verbally during the meeting held between S.Davies and R.Macfarlane on the 20th of August 2019.

A.2 Project timeline

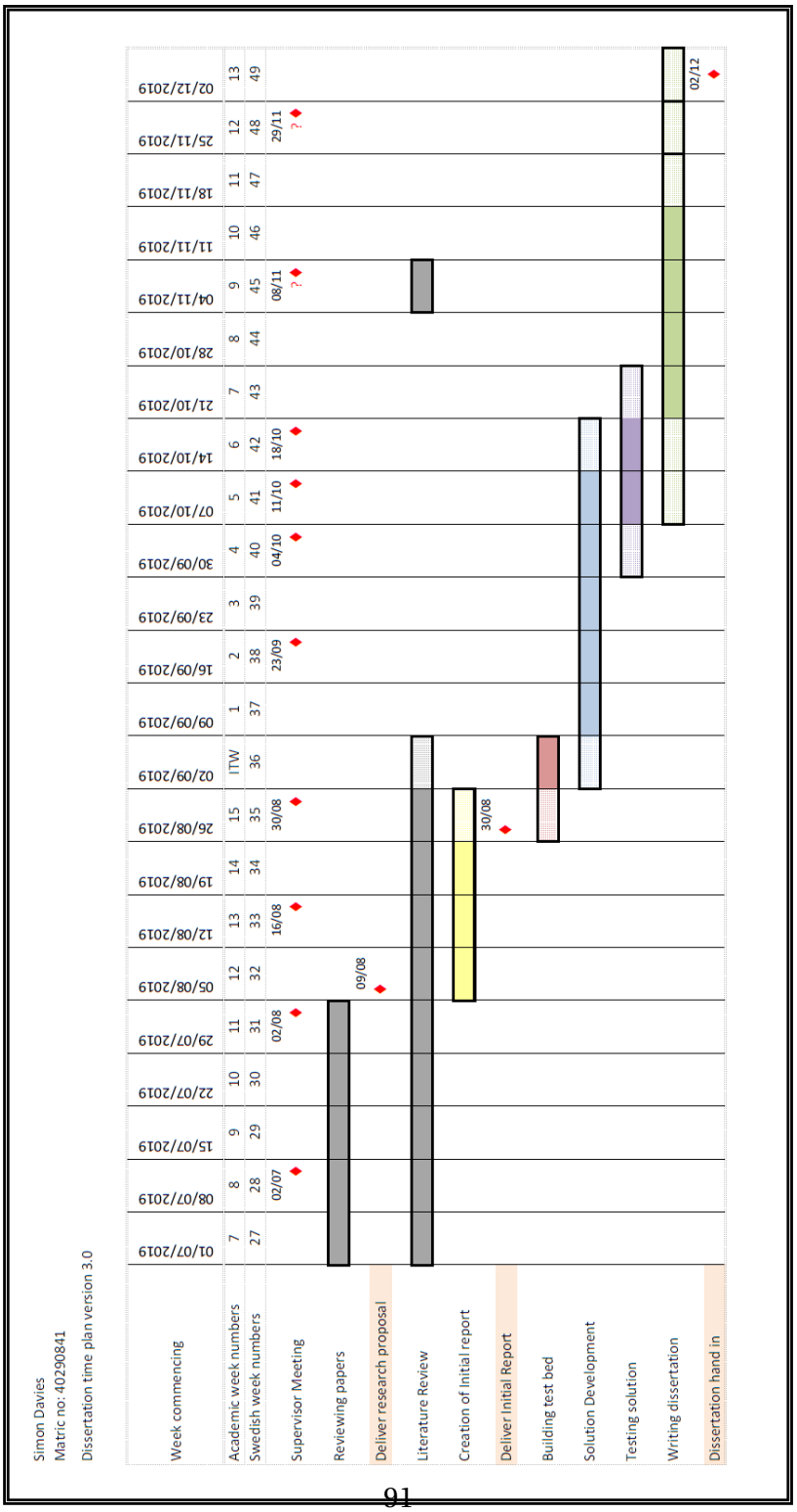


Figure A.1: Project Plan

A.3 Project diary

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Supervisor: Rich MacFarlane

Date: 2nd August 2019 Last diary date: First Entry

Objectives:

- To identify research project
- Generate initial research proposal
- Generate project diary

Progress:

12.07.2019
- Meeting with Rich

20.07.2019
- Re read the Module organiser.
- Re watched the dissertation lifecycle video https://moodle.napier.ac.uk/mod/url/view.php?id=1174834
- Created an overall project approach/workbook document
- requested slides from dissertation lifecycle video.

21.07.2019
- read writing styles an ethics https://moodle.napier.ac.uk/mod/workbook/view.php?id=1215476
- started on project plan. Initally in excel.
- read guide to writing research proposal https://moodle.napier.ac.uk/course/view.php?id=5225#section-0
- read the three example research proposal documents provided by Napier.
- critically read candidate paper 'Automatic Generation of Attack Scripts from Attack Graphs'.
- reviewed and installed Nessus.
- Reviewed Nessus automation using secubius.
- Ran a test Nessus scan on an XP machine to examine the results.

22.07.2019
- Started on creation of diagrams to support my proposal and describe to Rich what I intend to do.
- Started Gant project plan.
- Started the creation of the research proposal document, completed brief description section amongst others.

23.07.2019
- Re read paper 'Automatic Generation of Attack Scripts from Attack Graphs' and made critique.
- Researching similar papers and found two more.
- Found paper 'NetSecuritas An Integrated Attack Graph-based Security Assessment Tool for Enterprise Networks'.
- Automated Test Generation from Vulnerability Signatures.
- Enhanced proposed test network description diagram.

24.07.2019
- Wrote critique of "Automated Test Generation from Vulnerability Signatures" paper.
- Wrote critique of "Automated Generation of Attack Graphs Using NVD" paper.
- Continued literature review &nad paper Advances in Topological Vulnerability Analysis.
- Created a detailed project plan Gantt chart.
- Continued with research proposal document.
- Investigated setting up multiple virtual networks within virtualbox.
- Found paper "Advances in Topological Vulnerability Analysis".

25.07.2019
- Updated project plan.
- Created network diagram.
- Created example attack graph diagram.
- Critique "Advances in Topological Vulnerability Analysis".
- Continued on first draft of research proposal document.

26.07.2019
- Developed a program flow diagram to describe how I anticipate the developed program to work.
- Continued on first draft of research proposal document.

29.07.2019
- Reviewed the example research proposal papers.

30.07.2019
- Updated my draft proposal based on the findings I had from the example research proposal papers.
- Created further example attack graph diagrams to try and describe what I intend to do.
- Updated project plan.

31.07.2019
- Finalised first draft of research proposal document.

02.08.2019
- Read the project deliverable documents in preparation for supervisor meeting
- Added the project deliverable documents to the shared dropbox folder, and also emailed them to the project supervisor
- Meeting with supervisor

Supervisor's Comments:

Figure A.2: Project Dairy 20190802

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Simon Davies 40290841 **Supervisor: Rich MacFarlane**

Date: 16th August 2019 **Last diary date: 2nd August 2019**

Objectives:

- To identify research project
-

Progress:

02.08.2019
- Meeting with Rich
- Transcribed notes from meeting

05.08.2019
- Re read the Module organiser.

06.08.2019
- Conducted a search to find any research relating to automatic implementation of nmap/nessus/metasploit exploit scripts, without success.

08.08.2019
- Conducted a search to find any research relating to automatic validation of vulnerabilities without success.

09.08.2019
- Conducted a search to find any research relating to automatic validation of vulnerabilities without success.
- Found some papers relating to the inclusion of CVSS details into attack graphs to improve usefulness. However papers were fairly mathematical and theoretical

12.08.2019
- Started reading Research Methods for Cyber Security. Forward, intro and chapter 1
- Started to think about alternative research areas

13.08.2019
- Continued reading Research Methods for Cyber Security, chapter 2
- Read previous dissertation marked by my supervisor by Evaluating Techniques for phishing detection
- More work on alternative research areas

14.08.2019
- Continued reading Research Methods for Cyber Security, chapter 3
- Read previous dissertation marked by my supervisor by A.Gooch 40292769 'Evaluate the effectiveness of built-in features to reduce the impact of ransomware'

15.08.2019
- Sent mail to supervisor outlining the current status and lack of progress and suggestions of possible alternative research areas

16.08.2019
- Meeting with Rich
- Continued investigating other research areas, found some possibilities for live forensics with regards ransomware

Supervisor's Comments:

Figure A.3: Project Dairy 20190816

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Simon Davies 40290841 **Supervisor: Rich MacFarlane**

Date: 23th August 2019 **Last diary date: 16th August 2019**

Objectives:

- To identify a new research area

Progress:

19.08.2019
- Continued to investigate the use of live forensics with regards to ransomware mitigation.
- Sent a mail to my supervisor outlining some of my ideas for a new research area.

20.08.2019
- Two hour meeting with supervisor, where we discussed the current status of my research and my suggestions on alternative research ideas. Agreed that it would be OK to switch topics. Supervisor suggested that some quick proof of concept experiments should be performed to confirm the validity of my hypothesis.
- Installed a laptop that could be used for testing

21.08.2019
- Further research into the use of live forensics techniques in retrieving encryption keys from memory,

22.08.2019
- Further research into the use of live forensics in retrieving encryption keys from memory. Found some key papers relating to full disk encryption that could be used as a basis for research proposal.
- Started to build a test environment that can be used to validate my hypothesis

23.08.2019
- Further research into the use of live forensics in retrieving encryption keys from memory
- Continued building a test environment that can be used to validate my hypothesis
- Started to develop proof of concept experiment. Gathered examples of ransomware.
- Read in detail how NotPetya ransomware behaved when it was executed

Supervisor's Comments:

Figure A.4: Project Dairy 20190823

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Simon Davies 40290841 **Supervisor: Rich MacFarlane**

Date: 30th August 2019 **Last diary date: 23rd August 2019**

Objectives:

- To confirm the viability of new research areas

Progress:

23.08.2019
- Started the viability experiment.
- Issues with how to capture the memory. Either using VirtualBox techniques or a program running on the system

24.08.2019
- Managed to have the NotPetya ransomware execute an encrypt test machine.
- Managed to have the WannaCry ransomware execute an encrypt test machine.
- Decided to focus on NotPetya
- Manged to capture a memory dump from machine.

25.08.2019
- Took multiple memory dumps during all phases of the execution of the ransomware to determine if I was able to capture the memory at each phase.
- Copied the captured memory dumps on to a Kali machine and ran live forensics on them
- Confirmed that the live forensics was able to process the captured memory dumps

26.08.2019
- Used some of the techniques/programs mentioned in the found literature to see if we were able to identify AES keys in the memory dumps
- Some strings that could be key candidates were identified
- Started on dissertation template

27.08.2019
- Extracted some of the files encrypted by the ransomware and placed them on the kali machine
- Developed a program to use the identified AES keys found, to try and decrypt the encrypted files from the test machine.
- Continued with dissertation template
- Started writing the introduction

28.08.2019
- Started the viability experiment.
- Issues with how to capture the memory. Either using VirtualBox techniques or a program running on the system

29.08.2019
- Managed to identify possible AES key candidates
- Continued with ransomware literature research
- Continued writing introduction in dissertation

30.08.2019
- Trying to use found AES key candidates to decrypt files
- Continues literature review.

Supervisor's Comments:

Figure A.5: Project Dairy 20190830

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Simon Davies 40290841 **Supervisor: Rich MacFarlane**

Date: 6th September 2019 **Last diary date: 30th August 2019**

Supervisor's Comments:

Objectives:

- To complete the literature review

Progress:

31.08.2019
- Continued with ransomware literature review
- Started live forensics literature review.
- Completed introductions
- Started writing literature review

01.09.2019
- Continued with ransomware literature review
- Continued with live forensics literature review.
- Continued writing up the ransomware section of the literature review
- Started writing live forensics section of the literature review

02.09.2019
- Continued with ransomware literature review
- Continued with live forensics literature review
- Continued writing up the ransomware section of the literature review
- Continued writing live forensics section of the literature review

03.09.2019
- Continued with live forensics literature review.
- Continued writing live forensics section of the literature review

04.09.2019
- Continued with live forensics literature review.
- Continued writing live forensics section of the literature review

05.09.2019
- Continued with live forensics literature review.
- Continued writing live forensics section of the literature review
- Sent draft to project supervisor

06.09.2019
- Meeting with project supervisor
- Adding comments from meeting to document
- Continued writing live forensics section of the literature review
- Started wrapping up literature review

Figure A.6: Project Dairy 20190906

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Simon Davies 40290841 **Supervisor: Rich MacFarlane**

Date: 13th September 2019 **Last diary date: 6th September 2019**

Objectives:

- To complete the design

Progress:

07.09.2019
- Started experiment design literature review.
- Adding comments from project meeting with supervisor to main dissertation document.

08.09.2019
- Reviewing found documents on malware experiment design
- Researched specific design experiments for ransomware
- Found some documents on malware best practice experiment design

09.09.2019
- Reviewed best practice design documents
- Wrote section framework
- Continued searching for malware experiment design examples

10.09.2019
- Started writing section introduction
- Started to collate found references and examples
- Continued searching for malware experiment design examples

11.09.2019
- Writing design section
- Continued searching for malware experiment design examples

12.09.2019
- Writing design section

13.09.2019
- Postponed meeting with project supervisor until next week
- Writing design section
- Started to build test environment

Supervisor's Comments:

Figure A.7: Project Dairy 20190913

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Simon Davies 40290841 **Supervisor: Rich MacFarlane**

Date: 20th September 2019 **Last diary date: 13th September 2019**

Objectives:

- To complete the design and start on environment creation

Progress:

14.09.2019
- Writing design section.

15.09.2019
- Writing design section.
- Continued to build test environment

16.09.2019
- Completed first draft of design section.
- Completed first version of environment setup

17.09.2019
- Started writing implementation section
- Proof read document so far
- Searching for malware samples

18.09.2019
- Writing implementation section
- Continued searching for malware samples

19.09.2019
- Writing implementation section

20.09.2019
- Meeting with project supervisor until next week
- Testing test environment
- Writing implementation section

Supervisor's Comments:

Figure A.8: Project Dairy 20190920

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Simon Davies 40290841 **Supervisor: Rich MacFarlane**

Date: 20th September 2019 **Last diary date: 13th September 2019**

Objectives:

- To complete the design and start on environment creation

Progress:

21.09.2019
- Writing implementation section
- Added comments from yesterdays meeting

22.09.2019
- Testing test environment
- Writing implementation section
- Researching Latex and Latex editors

23.09.2019
- Researching Latex and Latex editors
- Started testing windows 7 NotPetya

24.09.2019
- Testing windows 7 NotPetya

25.09.2019
- Testing windows 7 NotPetya
- Researching other ransomware, Wannacry

26.09.2019
- Testing windows 7 NotPetya
- Researching other ransomware, Satan and Lucky

27.09.2019
- Meeting with project supervisor
- Testing test NotPetya

Supervisor's Comments:

Figure A.9: Project Dairy 20190927

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Simon Davies 40290841 **Supervisor: Rich MacFarlane**

Date: 11th October 2019 **Last diary date: 27th September 2019**

Objectives:

- To complete the experimentation and convert the thesis to Latex

Progress:

28.09.2019
- Researching other ransomware samples
- Added comments to thesis from yesterdays meeting

29.09.2019
- Testing test NotPetya
- Writing results from tests
- Researching NotPetya behaviour

30.09.2019
- Testing test NotPetya
- Writing results from tests

01.10.2019
- Testing test NotPetya
- Writing results from tests

02.10.2019
- Testing test NotPetya
- Started to test Bad Rabbit ransomware
- Writing results from tests

03.10.2019
- Completed first round of test for NotPetya
- Testing Bad Rabbit ransomware
- Writing results from tests

04.10.2019
- Testing Bad Rabbit ransomware
- Writing results from tests

05.10.2019
- Testing Bad Rabbit ransomware
- Recording results from tests

06.10.2019
- Testing Bad Rabbit ransomware
- Recording results from tests
- Researching other ransomware families

07.10.2019
- Completed first round of test for Bad Rabbit
- Started testing Phobos ransomware
- Writing results from tests

08.10.2019
- Testing Phobos ransomware
- Recording results from tests

09.10.2019
- Testing Phobos ransomware
- Recording results from tests
- Started drafting evaluation section

10.10.2019
- Testing Phobos ransomware
- Recording results from tests
- Drafting evaluation section

11.10.2019
- Meeting with project supervisor
- Testing Phobos ransomware

Supervisor's Comments:

Figure A.10: Project Dairy 20191011

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Simon Davies 40290841 **Supervisor: Rich MacFarlane**

Date: 18th October 2019 **Last diary date: 11th October 2019**

Objectives:

- To complete the experimentation and convert the thesis to Latex

Progress:

12.10.2019
- Recording results from tests
- Updating document with comments from yesterdays meeting

13.10.2019
- Started to convert document from word to latex
- Relearning latex

14.10.2019
- Continued to convert document from word to latex
- Relearning latex
- Writing up evaluation section, adding screen shots from experiments

15.10.2019
- Continued to convert document from word to latex
- Writing up evaluation section, adding screen shots from experiments
- Created a pseudo process flow diagram as suggested from the supervisor meeting

16.10.2019
- Continued to convert document from word to latex
- Writing up evaluation section, adding screen shots from experiments

17.10.2019
- Continued to convert document from word to latex
- Reviewing implementation and design sections, tidying up layout adding more text

18.10.2019
- Meeting with project supervisor

Supervisor's Comments:

Figure A.11: Project Dairy 20191018

EDINBURGH NAPIER UNIVERSITY

SCHOOL OF COMPUTING

PROJECT DIARY

Student: Simon Davies 40290841 **Supervisor: Rich MacFarlane**

Date: 25th October 2019 **Last diary date: 19th October 2019**

Objectives:

- To complete the first draft of final document

Progress:

19.10.2019
- Updating document with comments from yesterdays meeting
- Writing the evaluation section

20.10.2019
- Writing the evaluation section

21.10.2019
- Writing the evaluation section
- Started writing the conclusions

22.10.2019
- Writing the evaluation section
- Writing the conclusions

23.10.2019
- Writing the conclusions
- Writing the evaluation section

24.10.2019
- Writing the conclusions

25.10.2019
- Writing the conclusions
- Meeting with project supervisor

Supervisor's Comments:

Figure A.12: Project Dairy 20191025

Appendix B

Code and Command Samples

B.1 Decrypted file modification

commands used to recreate header and footer information in decrypted files

```
NotPetya
header reconstruction for pdf
printf
    "\x25\x50\x44\x46\x2d\x31\x2e\x35\x0a\x25\xc7\xec\x8f\xa2\x0a\x35"
    | cat - pdf-decrypted.pdf > reconstructed-pdf.pdf

header reconstruction for xlsx
printf
    "\x50\x4b\x03\x04\x14\x00\x06\x00\x08\x00\x00\x00\x21\x00\x7c\x6c"
    | cat - excel-decrypted.xlsx > reconstructed-xlsx.xlsx
we then need to cut off the last 3 bytes
dd if=reconstructed-xlsx.xlsx of=reconstructed-xlsx1.xlsx
    bs=$((‘cat reconstructed-xlsx.xlsx| wc -c‘ - 3)) count=1

header reconstruction for doc
printf
    "\xd0\xcf\x11\xe0\xa1\xb1\x1a\xe1\x00\x00\x00\x00\x00\x00\x00\x00"
    | cat - Word-decrypted.doc > reconstructed-doc.doc

header reconstruction for dox
printf
    "\x50\x4b\x03\x04\x14\x00\x06\x00\x08\x00\x00\x00\x21\x00\x09\x24"
    | cat - Word-decrypted.docx > reconstructed-docx.docx
we then need to cut off the last 7 bytes
dd if=reconstructed-docx.docx of=reconstructed-docx1.docx
    bs=$((‘cat reconstructed-docx.docx| wc -c‘ - 7)) count=1
```

B.2 decrypt.py

The following is the code used to extract the IV value from the first 16 bytes of the encrypted file and then decrypt files encrypted using NotPetya or Bad Rabbit ransomware

```
"""
Script to decrypt a file encrypted with AES
The script uses a file that can contain multiple AES keys and it will try each in turn

Simon Davies simonrdavies@yahoo.com
Version 1.0
20190822

"""

import os, random, struct, argparse, ntpath
import binascii
from Crypto.Cipher import AES

def decrypt_file(key, in_filename, out_filename=None, chunksize=24*1024):
    """ Decrypts a file using AES (CBC mode) with the
        given key. Parameters are similar to encrypt_file,
        with one difference: out_filename, if not supplied
        will be in_filename without its last extension
        (i.e. if in_filename is 'aaa.zip.enc' then
        out_filename will be 'aaa.zip')
    """
    print "key:       ", key
    print "in_filename: ", in_filename
    if not out_filename:
        out_filename = os.path.splitext(in_filename)[0]

    with open(in_filename, 'r') as infile:
        iv = infile.read(16)
        decryptor = AES.new(key, AES.MODE_CBC, iv)

        with open(out_filename, 'wb') as outfile:
            #outfile.write(iv)
            #print "here"
            #exit()
            while True:
                chunk = infile.read(chunksize)

                if len(chunk) == 0:
                    break
                if len(chunk) <> 16:
                    break
                outfile.write(decryptor.decrypt(chunk))

#generate a file name for the decrypted file based on the encrypted filename and the key
def destination_filename(source_filename,key):
    head,tail = ntpath.split(source_filename)
        if len(head) == 0:
        head = "."
        f, e = os.path.splitext(tail)
    newfilename = head+"/"+f+"-decrypted-"+key+e
    print "newfilename: " , newfilename
    return newfilename

def process_keyfile(key_file, encrypted_filename):
    if not os.path.exists(key_file):
        print "Key file doesnot exist: ",key_file
        return
    with open(key_file, 'rb') as infile:
        keyline = str.rstrip(infile.readline())
        while keyline:
            if keyline.find("Found",0, 50)>=0:
                print "found a comment row: ", keyline
            elif keyline.find("#",0, 50)>=0:
                print "found a comment row: ", keyline
            else:
                key_nospace=keyline.replace(" ","")
                #get rid of the carrage return
                fkey=binascii.unhexlify(key_nospace)
                decrypt_file(fkey,encrypted_filename,destination_filename(encrypted_filename,key_nospace),16)
                keyline = infile.readline()
    infile.close()
```

```
def main():
    parser_description = "Decrypt a file encoded with AES encryption"
    parser = argparse.ArgumentParser(description=parser_description)
    parser.add_argument("--file",
                        help="Path to the encrypted file",
                        required=True)
    parser.add_argument("--key",
                        help="Path to the file which holds the AES key(s)",
                        required=True)
    args = parser.parse_args()

   process_keyfile(args.key, args.file)


if __name__ == "__main__":
    main()
```

The above file can be modified with the following lines for it to decrypt files encrypted using the Phobos ransomware. This ransomware places the IV at a different position. Replace the code on line 39

```
        iv = infile.read(16)
```

with the lines

```
        infile.seek(-158,2)
        #read IV from near end of the file
        iv = infile.read(16)
        #reset pointer to file start
        infile.seek(0,0)
```

B.3 RansomAES.py

This code was developed by the author and combines the functionality of the live forensic tool `volatitlity` (Volatility, 2019) with the functionality of `findaes` (Trenholme, 2014). Arguments to this program are a memory dump file and a process id. This program uses the `volatility` program to extract the specific memory associated with the supplied process id from the memory dump file. Once the memory for this process has been extracted, it is searched for AES keys using the logic found in the `findaes` program.

```
// FindAES version 1.2 by Jesse Kornblum
// http://jessekornblum.com/tools/findaes/
// This code is public domain.
//
// Revision History
//  24 Sept 2019 - Modified to use volatitity
//  7 Feb 2012 - Added processing of multiple files
```

```
//  3 Feb 2012 - Added entropy check. Limited to one file
// 18 Jan 2011 - Initial version

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <inttypes.h>

#include "aes.h"

// Use a 10MB buffer
#define BUFFER_SIZE 10485760
#define WINDOW_SIZE AES256_KEY_SCHEDULE_SIZE


/// @brief Hex-dump sz bytes of data to standard output
///
/// @param key Bytes to display
/// @param sz Number of bytes to display
void display_key(const unsigned char * key, size_t sz)
{
  size_t pos = 0;
  while (pos < sz)
  {
    printf("%02x ", key[pos]);
    ++pos;
  }
  printf("\n");
}


/// @brief Returns true if any byte in the block repeats more than eight times
///
/// @param buffer Buffer to scan
/// @param size The size of the buffer
/// @param first Is this the first buffer in the file? Used to clear
/// the previous values, if any.
/// @return Returns TRUE iff. the buffer contains more than eight repititions
/// of any single byte, even if not next to each other. Otherwise, FALSE.
int entropy(const unsigned char * buffer, size_t size,int first)
{
  size_t i;
  static unsigned int count[256];
  static int first_entropy = 1;
  int result = 0;

  if (first)
    first_entropy = 1;

  // We only need to compute the full frequency count the first time
  if (first_entropy)
  {
    first_entropy = 0;

    // Set the entropy to all zeros, then count values
    for (i = 0 ; i < 256 ; ++i)
      count[i] = 0;
    for (i = 0 ; i < size ; ++i)
      count[buffer[i]]++;
  }

  // Search for repititions
  for (i = 0 ; i < 256 ; ++i)
  {
    if (count[i] > 8)
    {
      result = 1;
      break;
    }
  }

  // Shift the frequency counts
  count[buffer[0]]--;
  count[buffer[size]]++;

  return result;
}


void scan_buffer(unsigned char * buffer, size_t size, size_t offset)
{
  uint64_t pos;
  for (pos = 0 ; pos < size ; ++pos)
```

```
  {
    int first = FALSE;
    if (0 == offset +pos)
      first = TRUE;
    if (entropy(buffer + pos, AES128_KEY_SCHEDULE_SIZE,first))
      continue;

    if (valid_aes128_schedule(buffer + pos))
    {
      printf ("Found AES-128 key schedule at offset 0x%"PRIx64": \n",
         offset + pos);
      display_key(buffer + pos, AES128_KEY_SIZE);
    }
    if (valid_aes192_schedule(buffer + pos))
    {
      printf ("Found AES-192 key schedule at offset 0x%"PRIx64": \n",
         offset + pos);
      display_key(buffer + pos, AES192_KEY_SIZE);
    }
    if (valid_aes256_schedule(buffer + pos))
    {
      printf ("Found AES-256 key schedule at offset 0x%"PRIx64": \n",
         offset + pos);
      display_key(buffer + pos, AES256_KEY_SIZE);
    }
  }
}


// Use a sliding window scanner on the file to search for AES key schedules
int scan_file(char * fn)
{
  size_t offset = 0, size;
  unsigned char * buffer;
  FILE * handle;
  size_t bytes_read;

  if (NULL == fn)
    return TRUE;

  buffer = (unsigned char *)malloc(sizeof(unsigned char) * BUFFER_SIZE + WINDOW_SIZE);
  if (NULL == buffer)
    return TRUE;
  memset(buffer, 0, sizeof(unsigned char) * (BUFFER_SIZE + WINDOW_SIZE));

  handle = fopen(fn,"rb");
  if (NULL == handle)
  {
    perror(fn);
    free(buffer);
    return TRUE;
  }

  printf ("Searching %s\n", fn);

  while (!feof(handle))
  {
    // Clear out the buffer except for whatever data we have copied
    // from the end of the last buffer
    memset(buffer + WINDOW_SIZE, 0, BUFFER_SIZE);

    //   printf ("Reading from 0x%"PRIx64"\n", ftello(handle));

    // Read into the buffer without overwriting the existing data
    bytes_read = fread(buffer + WINDOW_SIZE,1,BUFFER_SIZE,handle);

    if (0 == offset)
    {
      if (bytes_read < BUFFER_SIZE)
   size = bytes_read;
      else
   size = bytes_read - WINDOW_SIZE;

      scan_buffer(buffer + WINDOW_SIZE, size , 0);
    }
    else
      scan_buffer(buffer, bytes_read, offset - WINDOW_SIZE);

    offset += bytes_read;

    // Copy the end of the buffer back to the beginning for the next window
    memcpy(buffer, buffer + BUFFER_SIZE, WINDOW_SIZE);
  }
```

```
  free(buffer);
  return FALSE;
}


int main(int argc, char **argv)
{
  if (argc < 5)
  {
    printf ("FindAES version 1.1 by Jesse Kornblum\n");
    printf ("Searches for AES-128, AES-192, and AES-256 keys\n\n");
    printf ("Modified by Simon Davies 20190924\n\n");

    printf ("Usage: findaes -p <pid> -t <profile type> [FILES]\n");
    return EXIT_FAILURE;
  }

  int i = 1;
  int pid = atoi(argv[2]);
  printf ("Looking for process id: %d\n", pid);
  //char profile_type[20] = argv[4];:wq
  //weh
  printf ("Using profile type: %s\n", argv[4]);
  char command[1024] = "/usr/bin/volatility --profile ";
  snprintf(command, sizeof(command),"/usr/bin/volatility --profile %s -f %s memdump -p %d --dump-dir .", argv[4],
        argv[5],pid);
  printf ("Command is: %s\n", command);
  system(command);
  snprintf(command, sizeof(command),"%d.dmp",pid);
  printf ("Command is: %s\n", command);
  scan_file(command);
  exit(0);
  while (i < argc-4)
  {
    scan_file(argv[5]);
    ++i;
  }

  return EXIT_SUCCESS;
}
```