Evaluation of Live Forensic Techniques in Ransomware Attack Mitigation

Simon R. Davies[a,*], Richard Macfarlane[a] and William J. Buchanan[a]

[a] *School of Computing, Edinburgh Napier University, Edinburgh, UK*

ARTICLE INFO

Keywords:
Live Forensics
Ransomware

Article history
Received 1st February 2020
Accepted 5th April 2020
Available

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 paper describes an investigation that tried 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 thus mitigating the effects of a ransomware attack.

Memory was captured from a system infected by ransomware and its contents was examined using live forensic tools, with the intent of identifying the symmetric encryption keys being used. NotPetya, Bad Rabbit and Phobos hybrid ransomware samples were tested during the investigation. If keys were discovered, the following two steps were also performed. Firstly, a timeline was manually created by combining data from multiple sources to illustrate the ransomware's behaviour as well as showing when the encryption 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 identify the encryption keys used. A description of how these found keys were then used to successfully decrypt files that had been encrypted during the execution of the ransomware is also given.

The resulting generated timelines provided a excellent way to visualise the behaviour of the ransomware and the encryption key management practices it employed, and from a forensic investigation and possible mitigation point of view, when the encryption keys are in memory.

1. Introduction

The aim of this paper 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 possibly reverse the effects of the ransomware's execution. While ransomware first appeared more than 30 years ago [41], 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 [8] where attackers adopted the three new technologies of cryptocurrency, 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 [32]. The number of malware attacks is generally considered to be growing year-on-year [14, 21].

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 system's memory where cryptographic processes are active. A lot of this research has proven very successful [4, 26] allowing the researchers to determine the encryption keys used by the cryptographic programs and subsequently 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 main contributions of this research is first to provide confirmation that forensic techniques are able to be employed to investigate the volatile memory of machines infected by ransomware and can be leveraged to extract useful cryptographic fragments that can be used to mitigate the effects of the attack. Secondly time lining the behaviour of the ransomware attack and highlighting when and for how long the encryption keys are available for extraction.

The rest of the paper is structured as follows. Section 2 - A review and discussions of related work in these areas. Section 3 - Description of the design philosophy. Section 4 - Description of the experiment implementation and the results achieved. Section 5 - Critical analysis of the experimental results and comparison to similar work in the field. Section 6 Discussion of the findings.

*Principal corresponding author
s.davies@napier.ac.uk (S.R. Davies)
0000-0003-0809-3523 (W.J. Buchanan)

2. Related Work

Ransomware is one of the most widespread and damaging threats that internet users face today [48] and is often classified by the type of encryption used [3]:

- **Symmetric Crypto-Ransomware (SCR)** uses one key for both encryption and decryption allowing the attack to complete in a shorter time, reducing the chances of it being discovered.
- **Asymmetric Crypto-Ransomware (ACR)** uses different keys for encryption and decryption.
- **Hybrid Key Crypto-Ransomware (HCR)** firstly uses symmetric encryption to encrypt the users files as fast as possible. After which the symmetric key is encrypted using asymmetric encryption.

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* [14] and Malware-Bytes [27] reporting a 500% year on year increase in attacks demonstrating that the trend upwards is set to continue. Europol confirming in 2018 that they believe that ransomware will retain its dominance for several years to come [15].

Even at the time of writing American government agencies are struggling with the effects of a recent ransomware attack [34] and US authorities are preparing for similar attacks during the voter registration for the 2020 elections [18].

The NotPetya cyber attack is considered the costliest attack in history [23] with an estimated cost of $10 billion, whereas WannaCry, according to various estimates, lies in the $4$8 billion range.

2.1. Live forensics and memory acquisition

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, and so on [5]. To address this issue a complimentary forensics approach known as live forensics has been developed. Live forensic analysis primarily targets the computers volatile data which can only be acquired from a running system. When applied to ransomware analysis, the live forensics techniques can be complimented by combining them with malware analysis techniques.

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[40] 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 [31].
3. Virtualization technology-based techniques.

A detailed compilation of the techniques available is provided in [9] along with advantages and disadvantages of each approach.

Using the virtualization approach, a snapshot of the analysed systems 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[33]. 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 [12, 25, 30].

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 [31].

2.2. Cryptanalysis live forensics

Some work has been previously performed into the possibility of using live forensic 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. [4, 26, 31].

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 [4] 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. With regards to key management, Maartmann-Moe [26] state that it is clear that cryptographic keys need to be present in memory during encryption when using standard computer hardware.

In extensive tests conducted on 10 different cryptographic systems the researchers [26] were always able to retrieve all the cryptographic keys from memory for every application tested using their specifically developed tool called *interrogate*. While these researchers have not investigated ransomware specifically, their findings strongly suggest that it would be possible to extract ransomware cryptographic keys using similar techniques.

2.2.1. *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 [44]. When data is random it has higher entropy than patterned information. This means that it should be possible to locate cryptographic keys among other data by locating sections with unusually high entropy [4]. The authors found that the memory block where the main and the auxiliary AES keys are located has a recognizable structure and high entropy.

In reality, symmetric cryptographic keys are just short sequences of random looking data, often 1632 bytes long residing amongst other pieces of data with a much lower entropy.

2.2.2. *Identifying keys in memory*

In order to extract encryption keys from memory, they must first be identified [16]. Several researchers have discovered [16, 38, 44], that encryption keys in memory are far more structured than previously believed and several strategies to locate the keys have been proposed and are discussed below.

Using a high-entropy searching approach as suggested by Shamir [44] and tested by [26]. Since it is known that key data has a higher 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 [44].

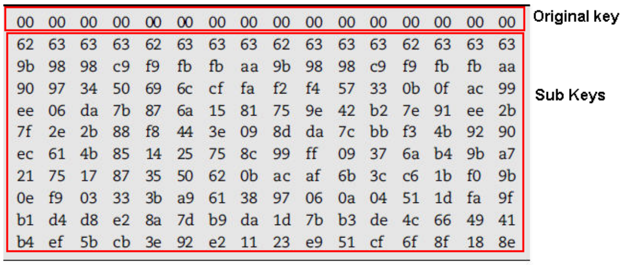
Search for certain known patterns in the memory such as key schedule which are specific to certain types of encryption [16, 38]. These patterns could also consist of known memory offsets, specific lengths of high entropy memory locations, or known patterns of entropy.

2.2.3. *Identifying AES keys*

From a review of current ransomware samples it has been determined that the majority of modern crypto ransomware are now hybrid in nature (HCR) [3]. 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 paper will concentrate on the identification of the key used during the symmetric encryption phase of the ransomwares execution which in the majority of cases is the Advanced Encryption Standard (AES) key.

AES is a Substitution-Permutation (SP)-network based cipher that works on 128-bit blocks, and can use either 128, 198 or 256 bit keys. It is considered by some researchers to be virtually unbreakable [43] and impossible to decrypt without the correct key [26].

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 used 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 [4, 17, 26]. This knowledge of the cryptosystem can be used to search for this key pattern within the memory [4]. The search criteria being that there is a mathematical relationship between the master key and sub-keys. 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 [26]. An example of an 128-bit empty AES key (all zeros) and its associated key schedule [26], extracted from memory is shown in Fig. 1.



```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
62 63 63 63 62 63 63 63 62 63 63 63 62 63 63 63
9b 98 98 c9 f9 fb fb aa 9b 98 98 c9 f9 fb fb aa
90 97 34 50 69 6c cf fa f2 f4 57 33 0b 0f ac 99
ee 06 da 7b 87 6a 15 81 75 9e 42 b2 7e 91 ee 2b
7f 2e 2b 88 f8 44 3e 09 8d da 7c bb f3 4b 92 90
ec 61 4b 85 14 25 75 8c 99 ff 09 37 6a b4 9b a7
21 75 17 87 35 50 62 0b ac af 6b 3c c6 1b f0 9b
0e f9 03 33 3b a9 61 38 97 06 0a 04 51 1d fa 9f
b1 d4 d8 e2 8a 7d b9 da 1d 7b b3 de 4c 66 49 41
b4 ef 5b cb 3e 92 e2 11 23 e9 51 cf 6f 8f 18 8e
```

Fig. 1. AES Key and Key Schedule [26].

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.

Several tools have been developed that use this phenomenon to identify AES keys in memory such as AESFinder [16, 19], Volatools [54], interrogate [26] and Findaes [24], however no research has been found to suggest that they have ever been used to analyse cryptographic ransomware in particular.

2.2.4. *Method*

Some of the key works in the area have used the same common experimental method [4, 17, 26, 33, 54]. 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 they are then analysed.

Several researchers including [4, 16, 26, 30, 31] have had success in extracting the encryption keys through the discovery of cryptographic information in volatile memory.

One thing to remember when applying this method to the ransomware samples analysed in this report is that these samples perform several steps before they actually begin encrypting data on the victims system [33]. 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. Ideally the memory should be captured while the ransomware is encrypting files. 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. The technique described uses volatile memory analysis combined with empirical observations rather than focusing on specific API calls that the ransomware may employ.

3. Design

3.1. Environment design

In order to conduct valid, realistic ransomware experiments the test victim's machine needs to mimic a real machine as much as possible with any private or confidential information removed[20] and ideally isolated from the internet [2]. Based on the research performed by Bose [7], it was decided to implement the test environment for this project using the VirtualBox virtualization software provided by Oracle.

When designing a test environment, one of the key recommendations from Rossow [39] was that it should be as realistic as possible. Using these guidelines as well as the three points raised by Sanabria [42], the test environment was designed to contain three virtual machines. Two of these virtual machines were victim test machines. Only one of which were 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 to provide network support services [42] for the victim, as denying all-network access to the sample under analysis will most likely result in incomplete observations of the malwares behaviour [13]. Therefore a common technique [45] 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 [42].

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 hosts network connections. The physical host machine consisted of a laptop which itself was air gapped, thus providing a second layer of isolation. 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 [39].

3.2. Experiment design

At 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 files encrypted during the ransomware's execution to confirm that the correct symmetric encryption key has been identified.

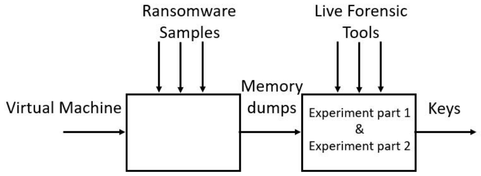


Fig. 2. Overview of Experiment part 1 & Experiment part 2.

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.

3.2.1. Experiment part 1 - Identify the key in memory

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. Special attention being paid to areas of memory that exhibited high entropy. A graphic representation of this and the following experiment is shown in Fig. 2.

3.2.2. Experiment part 2 - Key time line creation

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 part 1 by determining when to execute point A in Fig. 4. An outcome of this experiment was an approximate timeline for the execution of the ransomware.

3.2.3. Experiment part 3 - Validate found keys

If any keys were discovered during experiment part 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 Fig. 4. A tool developed by the author using the Python programming language was used to perform the decryption attempt. A graphic representation of this experiment is shown in Fig. 3.

3.2.4. Combined experiment

The overall experimental suite was based on similar methods identified during the literature review [4, 17, 26, 33, 54]. While the designed experiments in this investigation have significant similarities with previous work such as using windows operating systems on virtual machines and using simi-

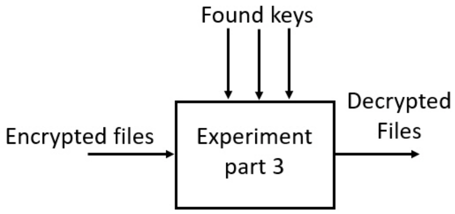


Fig. 3. Overview of Experiment part 3.

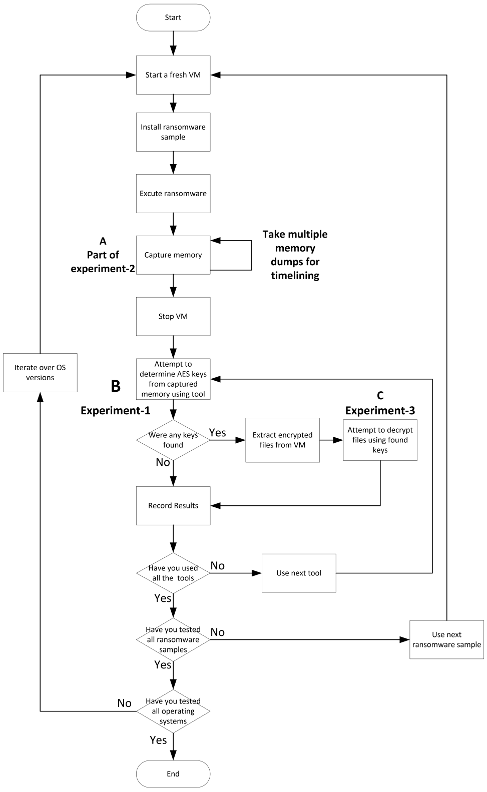

Start
Start a fresh VM
Install ransomware sample
Excute ransomware
A
Part of experiment-2
Capture memory
Take multiple memory dumps for timelining
Stop VM
Iterate over OS versions
Attempt to determine AES keys from captured memory using tool
B
Experiment-1
C
Experiment-3
Were any keys found
Yes
Extract encrypted files from VM
Attempt to decrypt files using found keys
No
Record Results
Have you used all the tools
No
Use next tool
Yes
Have you tested all ransomware samples
No
Use next ransomware sample
Yes
No
Have you tested all operating systems
Yes
End

Fig. 4. Experiment flow.

lar 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 successfully decrypted the control files. Also multiple memory dumps were taken during the experiment to allow for the creation of a timeline for the ransomware's execution. A graphical representation of the overall experiment is given in Fig. 4 and the main steps are discussed below.

Iterate over Operating Systems versions. When discussing realism in experimentation Rossow [39] cautioned against performing experiments on 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.

Start a fresh VM. Results are only comparable if each sample is executed in an identical environment [20] so a fresh VM was started at the beginning of each experiment.

Install Ransomware. The sample to be tested was extracted from the archive and prepared for execution.

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 part 2. If the keys became available in memory, then a copy of the machines memory was taken, using tools provided by the virtualization software in similar techniques used by other researchers [30, 33].

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 the three tools discussed below:

1. **findaes** - A tool developed by Kornblum [24] based on the work by Trenholme [16, 49] and tries to find the keys using the AES key schedule structure [50].
2. **interrogate** - A tool developed by Maartmann-Moe [26] also based on the work by Trendholme[49].
3. **RansomAES** - A hybrid tool developed by the author which incorporates logic from the Volatility Framework [53] together with the logic from findaes in an attempt to improve the accuracy and performance of the key detection.

Extract Encrypted Files. A set of typically targeted [10] control files with known content were placed on the victim machine prior to the ransomware's execution. Once the ransomware had executed, these files were analysed to determine if they had been encrypted.

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.

4. Implementation and results

4.1. Ransomware sample selection

Recent ransomware attacks were researched [11, 18, 23, 34, 51] and based on the findings, it was decided to select the following three recent ransomware samples for analysis. All of which used AES for the symmetric portion of the encryption.

NotPetya. This ransomware attack is considered to be the most damaging attack ever [23].

Bad Rabbit. An adaptation of the NotPetya ransomware family that emerged in 2018 [46].

Phobos. One of the most recent ransomware samples found where detailed information is available [22] and also one of the most prevalent in Q4 2019 [35].

Table 1. Ransomware Samples

Name	SHA256 Checksum
NotPetya	027cc450ef5f8c5f653329641ec1fed91f694e0d22992896 3b30f6b0d7d3a745
Bad Rabbit	630325cac09ac3fab908f903e3b00d0dadd5fdaa0875ed8 496fcbb97a558d0da
Phobos	a91491f45b851a07f91ba5a200967921bf796d38677786 de51a4a8fe5ddeafd2

Specific details of the ransomware samples used were validated by VirusTotal (www.virustotal.com) are described in Table 1. 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

4.1.1. *Other ransomware*

The following ransomware samples 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[6, 52]. 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 and it was deemed too risky to allow this external network access.
- Satan, SamSam and GrandCab. It was not possible to trigger these samples of ransomware to encrypt any of the control the files or display the ransom message.

4.2. Test machine configuration

The laptop used for testing had 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.

Three guest machines were defined in the virtual environment running on the laptop. 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. Details of the virtual guest machines are given in Table 2.

The guest machines were connected together using the recommended host-only connection technique [20],creating a separate virtual LAN providing isolation and containment of the guest machines. The virtual LAN and the guest machines were run in two separate configurations depending on which windows version was being tested.

Table 2. Virtual Hardware Configurations

Machine Name	Operating System	Purpose
Windows 7	Windows 7 Ultimate Build 7600	Ransomware Victim Machine
Windows 10	Windows 10 Pro Build 10586.494	Ransomware Victim Machine
Debian	Debian 5.2.9	Network Services

The following tools were used during the configuration of the Debian machine. *fakedns.py* used to provide DNS services to the network and *INetSim* - Considered to be the best free tool [45] for providing fake network service emulation creating the illusion of a realistic pseudo network that the malware can interact with if required.

4.3. Experiments

Details of the experiments performed are given below

4.3.1. *Experiment part 1 - Identify the key in memory*

A fresh VM was started and 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. To aid experiment reproducibility, the commands used to launch each of the ransomware samples is provided below:
For NotPetya

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

For BadRabbit

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

For Phobos

```
c:\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 memory dump file was then analysed by each of the selected live forensics memory tools. Again to aid reproducibility, the commands used are given below:

```
findaes <filename>.elf
interrogate -a aes -k 128 <filename>.elf
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 part 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.

4.3.2. *Experiment part 2 - Key time line creation*

If keys were discovered in experiment part 1, then this experiment was also performed. 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 [29].

4.3.3. *Experiment part 3 - Validate found keys*

If candidate AES keys were discovered in experiment part 1, then these were used in an attempt to decrypt the control files. 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.

The program was able to determine the required IV from the supplied encrypted file and then use this together with the discovered 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.

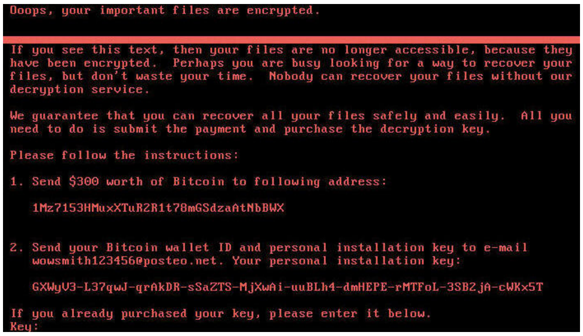
5. Results and Discussion

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.

5.1. NotPetya

The execution of the ransomware appeared to follow the descriptions provided by [6, 52]. 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.



```
Ooops, your important files are encrypted.

If you see this text, then your files are no longer accessible, because they
have been encrypted.  Perhaps you are busy looking for a way to recover your
files, but don't waste your time.  Nobody can recover your files without our
decryption service.

We guarantee that you can recover all your files safely and easily.  All you
need to do is submit the payment and purchase the decryption key.

Please follow the instructions:

1. Send $300 worth of Bitcoin to following address:

   1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWX


2. Send your Bitcoin wallet ID and personal installation key to e-mail
   wowsmith123456@posteo.net. Your personal installation key:

   GXWyV3-L37qwJ-qrAkDR-sSaZTS-MjXwAi-uuBLh4-dmHEPE-rMTFoL-3SB2jA-cWKx5T

If you already purchased your key, please enter it below.
Key:
```

Fig. 5. NotPetya Ransom Message.

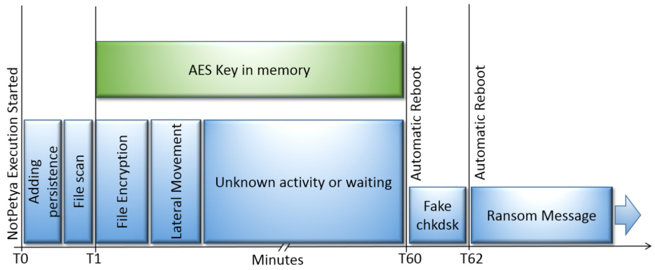


Fig. 6. NotPetya Timeline.

5. After 1 hour, rebooting the machine automatically.
6. Displaying a fake chkdsk command output, while encrypting the Mater Boot Record.
7. Once the fake chkdsk completes, the machine reboots automatically again.
8. After reboot, the ransomware message shown in Fig. 5 is displayed.

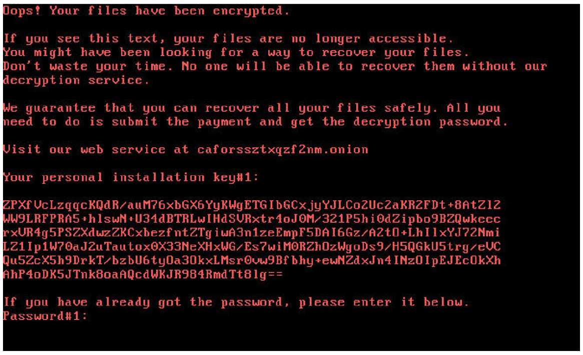
5.1.1. *Experiment part 1 - Identify the key in memory*

All three live forensics tools used to examine the memory dumps were able to successfully identify AES keys in the memory of the ransomware process.

5.1.2. *Experiment part 2 - Key time line creation*

A total of fifteen memory dumps were taken and then analysed to determine if they contained AES keys. 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 automatically rebooted by the ransomware 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 Fig 6.

This timeline correlates well with the findings of other researchers [6, 52]. However no research 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 Fig. 6 being generated by the author.



```
Oops! Your files have been encrypted.

If you see this text, your files are no longer accessible.
You might have been looking for a way to recover your files.
Don't waste your time. No one will be able to recover them without our
decryption service.

We guarantee that you can recover all your files safely. All you
need to do is submit the payment and get the decryption password.

Visit our web service at caforssztxqzf2nm.onion

Your personal installation key#1:

ZPXfVcLzqqcKQdR/auM76xbGX6YyKWgETGIbGCxjyYJLCo2UcZaKRZFDt+8AtZ1Z
WW9LRFPRA5+hlswN+U34dBTRLwIHdSVRxtr4oJ0M/321P5hi0dZipbo9BZQwkeec
rxVR4g5PSZXdwzZKCxbezfntZTgiwA3n1zeEmpF5DAI6Gz/A2tO+LhIlxYJ72Nmi
LZ1Ip1W70aJ2uTautox0X33NeXHxWG/Es7wiM0RZhOzWgoDs9/H5QGkU5trg/eVC
Qu5ZcX5h9DrkT/bzbU6tyOa3OkxLMsr0vw9Bfbhy+ewNZdxJn4INzOIpEJEcOkXh
AhP4oDK5JTnk8oaAQcdWKJR984RmdTt8lg==

If you have already got the password, please enter it below.
Password#1:
```

Fig. 7. Bad Rabbit Ransom Note.

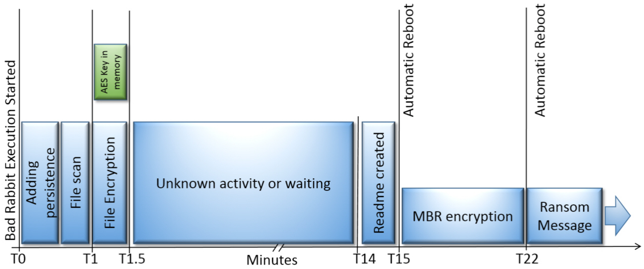


Fig. 8. Bad Rabbit Timeline.

5.1.3. *Experiment part 3 - Validate found keys*

When the NotPetya ransomware encrypts a file, the first 16 bytes of the file are overwritten with the AES Initialisation Vector (IV) value [47]. A program was developed that firstly reads the IV value from the encrypted file, then used this together with the key found in experiment part 1 to decrypt the files contents.

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.

5.2. Bad Rabbit

The execution of this ransomware followed the description provided by [1, 28, 37] 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.
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 Fig. 7 is displayed and the user is prevented from accessing their windows installation.

5.2.1. *Experiment part 1 - Identify the key in memory*

All three live forensics tools used to examine the memory dumps were able to identify AES keys in the memory of the ransomware process.

5.2.2. *Experiment part 2 - Key time line creation*

Again a total of fifteen memory dumps were taken and analysed to determine if they contained AES keys. 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 Fig. 8. This timeline matches the description given by [1]. Using the time line 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. No similar graphical representation of the ransomware time line was found in the literature, the one shown in Fig. 8 being generated by the author.

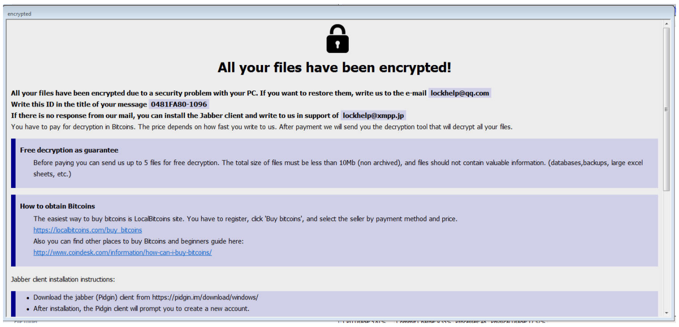
5.2.3. *Experiment part 3 - Validate found keys*

Files are encrypted using the same format as the NotPetya ransomware and the steps described in section 5.1.3 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.

5.3. Phobos

The execution of the ransomware followed the description provided by [22]. 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 Fig. 9 is displayed.
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 it is believed that these keys should also be present.
6. The machine does not reboot automatically. If the user reboots the machine, then the same ransomware message is displayed.


encrypted

All your files have been encrypted!

All your files have been encrypted due to a security problem with your PC. If you want to restore them, write us to the e-mail lockhelp@qq.com
Write this ID in the title of your message 0481FA80-1096
If there is no response from our mail, you can install the Jabber client and write to us in support of lockhelp@xmpp.jp
You have to pay for decryption in Bitcoins. The price depends on how fast you write to us. After payment we will send you the decryption tool that will decrypt all your files.

Free decryption as guarantee
Before paying you can send us up to 5 files for free decryption. The total size of files must be less than 10Mb (non archived), and files should not contain valuable information. (databases,backups, large excel sheets, etc.)

How to obtain Bitcoins
The easiest way to buy bitcoins is LocalBitcoins site. You have to register, click 'Buy bitcoins', and select the seller by payment method and price.
https://localbitcoins.com/buy_bitcoins
Also you can find other places to buy Bitcoins and beginners guide here:
http://www.coindesk.com/information/how-can-i-buy-bitcoins/

Jabber client installation instructions:

- Download the jabber (Pidgin) client from https://pidgin.im/download/windows/
- After installation, the Pidgin client will prompt you to create a new account.

Fig. 9. Phobos Ransom Note.

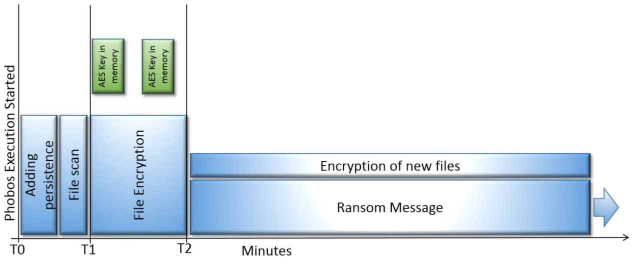


Fig. 10. Phobos Timeline.

5.3.1. *Experiment part 1 - Identify the key in memory*

All three live forensics tools used to examine the ransomware process memory were able to identify the 256 bit AES key used by the ransomware to encrypt the files.

5.3.2. *Experiment part 2 - Key time line creation*

Similarly a total of fifteen memory dumps were taken and analysed to determine if they contained AES keys. 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. As with the previous two ransomware samples no AES key survives a machine reboot. A graphical representation of some of the ransomware's behaviour and key availability is shown in Fig. 10.

This timeline correlates well with the description given by [22, 36]. Using this diagram it can be easily seen 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 Fig. 10 being generated by the author.

5.3.3. *Experiment part 3 - Validate found keys*

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 [22] 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 part 1 to decrypt the file. Using this technique pdf, doc, docx, xls and xlsx files were successfully recovered using the same AES key.

6. Conclusion

Building on other work in the field of live forensics [4, 17, 26], AES key recovery from volatile memory techniques were successfully used to identify encryption keys being used by ransomware samples. As research in this specific area was not identified, techniques and tools used for TrueCrypt and Skype encryption key recovery were used to identify keys for recent, high impact ransomware samples. Secondly the work aimed to attempt to evaluate whether consistent timelines of when these keys where stored in memory could be generated.

Similar execution times and results were recorded for the findaes and RansomAES live forensic tools, The RansomAES tool created by the author had similar results as the pure findaes tool, indicating that the added extra functionality for ransomware did not result in an improvement in performance. The time taken for the interrogate tool to complete was almost 100 times longer. Apart from this extended execution time all three tools were shown to successfully identify AES keys being used by three ransomware samples chosen for the experiments on both windows 7 and windows 10 operating systems. This supports the hypotheses that live forensic techniques could be used to mitigate a ransomware attack. The authors believe 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 [50, 49]. Interrogate seems not to take into account the entropy of the candidate keys prior to calculation of the schedule, rather it calculates the schedule for all candidate keys irrespective their entropy and this may be one reason for the impact in performance.

However these results support the hypothesis with some caveats from a forensic investigation point of view. 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 ransomware sample, where AES keys were not able to be identified.

It was also observed that in case of a ransomware attack where the user was still able to interact with the operating system after the initial encryption has completed, such as the Phobos ransomware, a second AES key looks to be in memory. This key is being used to perform any subsequent encryption, such as on any new files created. This approach being possibly utilised to frustrate the recovery of the original key and protect the original encryption. The keys found during the work seem to align with suggestions from previous work [4, 17], which indicate that it was likely that these secondary AES keys in theory should also be available in memory. Further research is planned to analyse various ransomware samples to determine if capture of these secondary keys is also possible and useful in a forensic investigation.

References

[1] 2017 Malwarebytes LABS, 2017. Bad Rabbit: a closer look at the new version of Petya/NotPetya. URL: https://blog.malwarebytes.com/threat-analysis/2017/10/badrabbit-closer-look-new-version-petyanotpetya/.

[2] 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 , 629–635doi:10.1109/ICACCCT.2016.7831716.

[3] 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. URL: https://doi.org/10.1016/j.cose.2018.01.001, doi:10.1016/j.cose.2018.01.001.

[4] 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, 759–763. doi:10.1109/IDAACS.2011.6072872.

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

[6] Berry, A., Homan, J., Eitzman, R., 2017. Threat Research WannaCry Malware Profile. URL: https://www.fireeye.com/blog/threat-research/2017/05/wannacry-malware-profile.html.

[7] Bose, M., 2018. A Complete Comparison of VMware and VirtualBox. URL: https://www.nakivo.com/blog/vmware-vs-virtual-box-comprehensive-comparison/.

[8] Bradley, S., 2016. Information Security Reading Room Ransomware. Technical Report. SANS Institute. URL: https://www.sans.org/reading-room/whitepapers/awareness/paper/37317.

[9] Carvey, H., Casey, E., 2009. Windows Forensic analysis DVD toolkit. 2nd ed., Burlington, MA : Syngress.

[10] CERT-EU, 2017. WannaCry Ransomware Campaign Exploiting SMB Vulnerability. Technical Report. Computer Emergency Response Team - EU. URL: https://cert.europa.eu/static/SecurityAdvisories/2017/CERT-EU-SA2017-012.pdf.

[11] Comodo, 2018. 8 Ransomware Attacks that have Occurrred Recently. URL: https://enterprise.comodo.com/forensic-analysis/ransomware-attacks.php.

[12] Dinaburg, A., Royal, P., Sharif, M., Lee, W., 2008. Ether: Malware Analysis via Hardware Virtualization Extension. Operating Systems]: Security and Protection , 51–62URL: http://ether.gtisc.gatech.edu/ether{_}ccs{_}2008.pdf.

[13] Egele, M., Scholte, T., Kirda, E., Kruegel, C., 2012. A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Surveys 44. doi:10.1145/2089125.2089126.

[14] Europol, 2016. INTERNET ORGANISED CRIME 2016 IOCTA. Technical Report. Europol. URL: https://www.europol.europa.eu/activities-services/main-reports/internet-organised-crime-threat-assessment-iocta-2016, doi:10.2813/275589.

[15] Europol, 2018. INTERNET ORGANISED CRIME 2018 IOCTA. Technical Report. Europol. URL: https://www.europol.europa.eu/internet-organised-crime-threat-assessment-2018, doi:10.2813/858843.

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

[17] 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 , 1369–1376doi:10.1109/ARES.2008.109.

[18] Hautala, L., 2019. States brace for ransomware assaults on voter registries. URL: https://www.cnet.com/news/wi-fi-6-is-barely-here-but-wi-fi-7-is-already-on-the-way/.

[19] Heninger, N., Feldman, A., 2008. AESKeyFind. URL: https://github.com/eugenekolo/sec-tools/tree/master/crypto/aeskeyfind/aeskeyfind.

[20] Hoopes, J., 2009. Chapter 6. Malware Analysis Solutions, in: Virtualization Security Protecting Virtualized Environments. 1st ed.. O'Reilly. chapter Chapter 6. URL: https://learning.oreilly.com/library/view/virtualization-for-security/9781597493055/{#}toc.

[21] Intelligence, T., Analysis, I., 2019. 2019 SonicWall Cyber Threat Report. Technical Report July. SonicWall. URL: www.sonicwall.com.

[22] Issa, J., 2019. A deep dive into Phobos ransomware. URL: https://blog.malwarebytes.com/threat-analysis/2019/07/a-deep-dive-into-phobos-ransomware/.

[23] Kapersky, 2018. Top 5 most notorious cyberattacks. URL: https://www.kaspersky.com/blog/five-most-notorious-cyberattacks/24506/.

[24] Kornblum, J., 2019. findaes. URL: http://jessekornblum.com/tools/findaes/.

[25] Ligh, M.H., Case, A., Levy, J., Walters, A., 2014. The Art of memory Forensics. Wiley.

[26] 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.

[27] Malwarebytes, 2019. Cybercrime Tactics and Techniques Q1 2019. Technical Report. MalwareBytes. URL: https://www.malwarebytes.com/pdf/labs/Cybercrime-Tactics-and-Techniques-Q1-2017.pdf.

[28] Mamedov, O., Sinitsyn, F., Ivanov, A., 2018. Bad Rabbit ransomware. URL: https://securelist.com/bad-rabbit-ransomware/82851/.

[29] McAfee Labs, 2016. Understanding Ransomware and Strategies to Defeat it. Network Security , 1–18URL: https://www.mcafee.com/us/resources/white-papers/wp-understanding-ransomware-strategies-defeat.pdf.

[30] McLaren, P., Buchanan, W.J., Russell, G., Tan, Z., 2019a. Deriving ChaCha20 Key Streams From Targeted Memory Analysis. Journal of Information Security and Applications 48. URL: http://arxiv.org/abs/1907.11941{%}0Ahttp://dx.doi.org/10.1016/j.jisa.2019.102372, doi:10.1016/j.jisa.2019.102372, arXiv:1907.11941.

[31] McLaren, P., Russell, G., Buchanan, W.J., Tan, Z., 2019b. Decrypting live SSH traffic in virtual environments. Digital Investigation 29, 109–117. doi:10.1016/j.diin.2019.03.010, arXiv:arXiv:1907.10835v1.

[32] 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. URL: scmr.com.

[33] 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.

[34] O'Donnall, L., 2019. Coordinated Ransomware Attack Hits 23 Texas Government Agencies. URL: https://threatpost.com/coordinated-ransomware-attack-hits-23-texas-government-agencies/

147457/.

[35] O'Donnell, L., 2020. ThreatList: Ransomware Costs Double in Q4, Sodinokibi Dominates. URL: https://threatpost.com/threatlist-ransomware-costs-double-in-q4-sodinokibi-dominates/152200/.

[36] Panda Security, 2017. Technical Analysis of Bad Rabbit. Panda Security 5, 1–5.

[37] Perekalin, A., 2018. Bad Rabbit: A new ransomware epidemic is on the rise. URL: https://www.kaspersky.com/blog/bad-rabbit-ransomware/19887/.

[38] Ptacek, T., 2008. Recover a Private Key from Process Memory. URL: http://www.matasano.com/log/178/recovera-{%}0Aprivate-key-from-process-memory.

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

[40] Ruff, N., 2008. Windows memory forensics. Journal in Computer Virology 4, 83–100. doi:10.1007/s11416-007-0070-0.

[41] Salvi, H.U., 2015. Ransomware : A Cyber Extortion. Asian Journal of Convergence in Technology II.

[42] Sanabria, A., 2007. Malware Analysis : Environment Design and Architecture. Technical Report. SANS Institute. URL: https://www.sans.org/reading-room/whitepapers/threats/paper/1841.

[43] Saravanan, M., Mukesh, K., 2014. Forensic Recovery of Fully Encrypted Volume. International Journal of Computer Applications 91, 18–21. doi:10.5120/15892-4896.

[44] 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.

[45] Sikorski, A., Hong, A., 2012. Practical Malware Analysis. No Starch Press, San Francisco.

[46] SonicWall, 2019. Unmasking the threats that target glkobal enterprises, governments abd SMBs. Journal of Chemical Information and Modeling 53.

[47] Sood, K., Hurley, S., 2017. NotPetya Technical Analysis A Triple Threat: File Encryption, MFT Encryption, Credential Theft. URL: https://www.crowdstrike.com/blog/petrwrap-ransomware-technical-analysis-triple\-threat-file-encryption-mft-encryption-credential-theft/.

[48] Sophos, 2019. Ransomware: How an attack works - Sophos Community. Sophos , 1–2URL: https://community.sophos.com/kb/en-us/124699.

[49] Trenholme, S., . Rijndael's key schedule. URL: https://www.samiam.org/key-schedule.html.

[50] Trenholme, S., 2014. findaes. URL: https://sourceforge.net/projects/findaes/.

[51] Vanderburg, E., 2019. A Timeline of Ransomware Advances. URL: https://www.tcdi.com/ransomware-timeline/.

[52] Vipre Security, 2017. WannaCry Technical Analysis : Support. URL: https://support.threattracksecurity.com/support/solutions/articles/1000250396-wannacry-technical-analysis.

[53] Volatility, 2019. Volatility Foundation. URL: https://www.volatilityfoundation.org/.

[54] Walters, A., Petroni, N.L., 2007. Volatools: Integrating Volatile Memory Forensics into the Digital Investigation Process. Black Hat DC , 1–18.