UNIVERSITY OF PIRAEUS - DEPARTMENT OF INFORMATICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

MSc «Cybersecurity and Data Science»

ΠΜΣ «Κυβερνοασφάλεια και Επιστήμη Δεδομένων»

MSc Thesis

Μεταπτυχιακή Διατριβή

Thesis Title: Τίτλος Διατριβής:	**Digital forensics methods for recovering ransomware encryption keys** Μέθοδοι ψηφιακής εγκληματολογίας για την ανάκτηση κλειδιών κρυπτογράφησης ransomware
Student's name-surname: Ονοματεπώνυμο φοιτητή:	**Paraskevas Tassios** Παρασκευάς Τάσσιος
Father's name: Πατρώνυμο:	**Nikolaos** Νικόλαος
Student's ID No: Αριθμός Μητρώου:	ΜΠΚΕΔ 2240
Supervisor: Επιβλέπων:	**Konstantinos Patsakis, Associate Professor** Κωνσταντίνος Πατσάκης, Αναπληρωτής Καθηγητής

July 2024/ Ιούλιος 2024

3-Member Examination Committee

Τριμελής Εξεταστική Επιτροπή

Konstantinos Patsakis	**Panagiotis Kotzanikolaou**	**Efthymios Alepis**
Associate Professor	**Professor**	**Professor**
Κωνσταντίνος Πατσάκης	Παναγιώτης Κοτζανικολάου	Ευθύμιος Αλέπης
Αναπληρωτής Καθηγητής	Καθηγητής	Καθηγητής

Acknowledgements

I would like to express my gratitude to Professor Konstantinos Patsakis, who has been a help, during the research phase of my master’s thesis. His insightful suggestions, guidance and unwavering support have been truly invaluable.

Contents

Περίληψη

Το ransomware έχει αυξηθεί σταθερά σε κλίμακα, κόστος, πολυπλοκότητα και αντίκτυπο από τότε που πρωτοεμφανίστηκε πριν από σχεδόν 35 χρόνια. Οι ειδικοί ασφαλείας εμπλέκονται συνεχώς σε μια μάχη με τους προγραμματιστές ransomware, προσπαθώντας να προστατεύσουν την ψηφιακή τους υποδομή από αυτές τις επιθέσεις. Πρόσφατες παραλλαγές ransomware έχουν αρχίσει να χρησιμοποιούν έναν συνδυασμό συμμετρικής και ασύμμετρης κρυπτογράφησης για το κλείδωμα των αρχείων των χρηστών. Αυτή η μεταπτυχιακή διατριβή διερευνά εάν μπορούν να χρησιμοποιηθούν ψηφιακές εγκληματολογικές τεχνικές για την αποκάλυψη των κλειδιών κρυπτογράφησης που χρησιμοποιούνται από τέτοιο κακόβουλο λογισμικό. Για τη διεξαγωγή αυτής της έρευνας, δημιουργήθηκε ένα ασφαλές και απομονωμένο εικονικό περιβάλλον όπου εκτελέστηκαν διάφορα δείγματα ransomware. Στη συνέχεια, η μνήμη από τα μολυσμένα συστήματα καταγράφηκε και εξετάστηκε χρησιμοποιώντας δύο διαφορετικά εγκληματολογικά εργαλεία για τον εντοπισμό των συμμετρικών κλειδιών κρυπτογράφησης που χρησιμοποιούνται από το ransomware.Επιπλέον, όταν οι εγκληματολογικές αναλύσεις μνήμης δεν απέδωσαν αποτελέσματα, χρησιμοποιήθηκε μια εναλλακτική μέθοδος που περιλαμβάνει το CryptoAPI hooking με το εργαλείο Frida. Η μελέτη εξέτασε δείγματα ransomware, συμπεριλαμβανομένων των Jigsaw, NotPetya, Thanos, Gpcode, WannaCry και Phobos σε δύο διαφορετικά λειτουργικά συστήματα. Αυτά τα δείγματα επιλέχθηκαν λόγω της υψηλής δημοσιότητας τους, των σημαντικών απαιτήσεων λύτρων και της σημαντικής διατάραξης που προκάλεσαν σε πολλούς οργανισμούς. Η έρευνα έδειξε με επιτυχία ότι είναι δυνατό να ανακαλυφθούν τα κλειδιά κρυπτογράφησης που χρησιμοποιούνται από αυτά τα δείγματα ransomware. Τα ευρήματα, μαζί με τις προκλήσεις που αντιμετωπίστηκαν κατά τη διάρκεια της έρευνας, παρουσιάζονται στην παρούσα διατριβή.

Abstract

Ransomware has steadily increased in scale, cost, complexity, and impact since it first appeared nearly 35 years ago. Security experts are constantly engaged in a battle with ransomware developers, striving to protect their digital infrastructure from these attacks. Recent variants of ransomware have begun using a combination of symmetric and asymmetric encryption to lock users' files.This master thesis investigates whether digital forensic techniques can be used to uncover the encryption keys utilized by such malicious software. To conduct this research, a secure and isolated virtual environment was set up where various ransomware samples were executed. Memory from the infected systems was then captured and examined using two different forensic tools to identify the symmetric encryption keys used by the ransomware. Additionally, an alternative method involving CryptoAPI hooking with the Frida tool was employed when memory forensics did not yield results.The study tested ransomware samples including Jigsaw, NotPetya, Thanos, Gpcode, WannaCry, and Phobos on two different operating systems. These samples were selected due to their high-profile nature, significant ransom demands, and substantial disruption to numerous organizations.The investigation successfully demonstrated that it is possible to discover the encryption keys used by these ransomware samples. The findings, along with the challenges faced during the investigation, are presented in this thesis.

List Of Figures

List Of Tables

CHAPTER 1

Introduction

Among the prevailing malware causing significant disruptions across the Internet, ransomware stands out as one of the most formidable threats. Exploiting user negligence, ransomware primarily proliferates through phishing techniques, such as the dissemination of malware-laden attachments via spam emails and the exploitation of vulnerable internet-facing devices [34]. When unwitting users open these malicious attachments, substantial damage can occur, leading to data loss through the encryption of critical data.Ransomware has evolved beyond simple encryption tactics [38]. Initially, single extortion involved encrypting files and demanding a ransom for their decryption. However, double extortion emerged, where attackers also exfiltrate data and threaten to publicize it, adding pressure on the victims. Triple extortion further escalates the threat by incorporating DDoS attacks alongside encryption and data exposure, aiming to disrupt operations further. The most severe form, quadruple extortion, targets not only the victim organization but also its customers and stakeholders, increasing the pressure to comply with ransom demands [11]. It's crucial to note that paying a ransom is not only illegal but also offers no guarantee that the attacker will provide the decryption key. Additionally, threat actors often demand payments in Bitcoin, which is the most prominent cryptocurrency to this date, further complicating the traceability and recovery efforts [36][46]. The primary targets of these attacks are often companies and corporations, as the data they possess holds high value. The encryption of such data can result in service disruptions and financial losses for these entities. While backups offer a potential countermeasure to ransomware attacks, they have limitations, particularly concerning the validity of data and the frequency of backup procedures. Thus, the objective is not only to recover from a ransomware attack but also to promptly detect the initiation of data encryption to mitigate the attack [1].

In the ongoing battle against cyberattacks, digital forensics is essential in uncovering malicious activities. Memory forensics, in particular, is a powerful strategy for revealing hidden information within the vast landscape of random-access memory, providing conclusive evidence that unravels the sequence of events on a system. This thesis explores the key management and cryptography models utilized by ransomware, highlighting potential vulnerabilities in cryptoviral infections. By exploiting the transparency of physical memory, the aim is to extract decryption keys and other critical insights from the ransomware process memory during execution. Additionally, leveraging CryptoAPI calls made by ransomware helps extract keys for ransomware mitigation [40]. The comprehensive analysis includes tracing injected dynamic link libraries (DLLs), investigating process hollowing, and employing reverse engineering techniques. A key technical challenge is the volatile nature of physical memory, which poses difficulties in extracting crucial findings due to its ephemeral characteristics. Despite this challenge, the exclusive insights derived from the analysis pave the way for data recovery, offering a viable alternative to ransom payments. The findings and challenges encountered are thoroughly presented to provide a comprehensive understanding of the ransomware landscape.

CHAPTER 2

Background

Ransomware has posed a significant threat to system security for more than a decade, evolving to execute sophisticated targeted attacks on organizations. Consequently, numerous proposals for solutions have emerged, all aiming to safeguard user data from the impact of ransomware-induced unavailability attacks. In this chapter, we systematically organize and assess the current array of solutions devised to counteract ransomware.

2.1 Introduction

It is imperative to subject current solutions to an objective analysis to assess their effectiveness in delivering a comprehensive and pragmatic response to emerging ransomware variants. For example, the use of signature-based detection in antivirus software provides optimal protection by identifying ransomware statically (prior to execution), but it proves inadequate against novel ransomware strains. In the subsequent sections, we organize and objectively evaluate prevailing solutions based on predefined criteria to grasp their genuine capabilities. It’s essential to note that the suitability of a specific solution is contingent upon the deployment environment.

2.2 The current advancements in available solutions

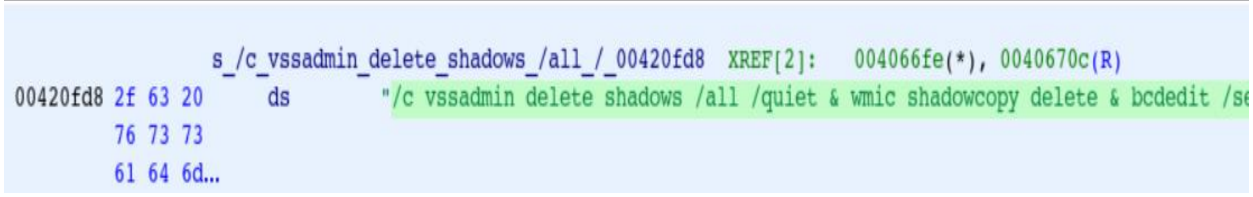
Numerous proposed countermeasures aim to address cryptoviral extortions and can be categorized as follows:

1. Backup solutions
2. Solutions based on static signatures
3. Solutions based on dynamic behavior
4. Solutions oriented towards user training
5. Vulnerability management solutions
6. Solutions centered around cryptography

Subsequently, we provide a brief overview of each of these existing ransomware countermeasures.

2.2.1 Backup solutions

Backups are proposed as the ultimate remedy against all malware infections. In theory, they prove effective as ransomware essentially executes a denial-of-control attack on the victim’s resources. By rendering the data inaccessible to the victim, ransomware operators gain the necessary leverage to demand a ransom. Therefore, when backups are accessible, the victim can simply erase the machine, reinstall the host OS, and reload the data onto the system. Consequently, the ransomware threat can be reduced to a mere inconvenience. However, a significant challenge with this approach is that backups are frequently unavailable, incomplete, and irregularly performed. Sustaining comprehensive and regularly updated data copies offsite is a intricate and costly process, and ransomware developers exploit this understanding. Furthermore, contemporary ransomware has been observed explicitly targeting the encryption of backups within the internal network and the cloud, along with executing discreet commands to obliterate shadow files on the host, thereby preventing the victim from recovering any data (Figure 2.1). Shadow files are inherently maintained on a Windows host to facilitate restoration in the event of failures [2].



```
                    s_/c_vssadmin_delete_shadows_/all_/_00420fd8  XREF[2]:   004066fe(*), 0040670c(R)
00420fd8 2f 63 20      ds        "/c vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /se
         76 73 73
         61 64 6d...
```

Figure 2.1: Wana Decrypt0r 2.0 ransomware deleting shadow volume copies on host [31].

2.2.2 Solutions based on static signatures

Similar to other forms of malware, ransomware can be recognized through static signatures integrated into virus definition files utilized by antivirus solutions. This established method is effective for detecting known threats. However, the fundamental challenge lies in the assumption for detection: a ransomware variant must have been previously identified and analyzed to create a signature, making it detectable and neutralizable. This implies that newly emerging ransomware variants can go undetected, rendering the system consistently ineffective against novel ransomware families [39][41]. Moreover, malware developers commonly employ packers to obfuscate malware, altering its signatures and evading signature-based detection. In essence, static signature-based detection methods prove inadequate against contemporary novel ransomware threats and should only be implemented as part of a comprehensive layered defense strategy.

2.2.3 Solutions based on dynamic behavior

The fundamental goal of cryptographic ransomware is to encrypt a user's data using a unique secret held for ransom by the attacker. Consequently, ransomware undertakes a series of anticipated tasks on the host, constituting a partially distinctive dynamic signature that mirrors its behavior during execution. This behavior is only partially unique because legitimate applications can exhibit similar conduct on the host. The resulting signature is dynamic as it stems from identifying common patterns established by the ransomware process during its execution on the host.While solutions based on dynamic behavior appear promising initially, their primary vulnerability lies in the significant number of false positives they generate. Real-world applications can produce dynamic footprints resembling ransomware behavior, creating challenges in distinguishing between malicious and legitimate activities. For instance, applications like archiving utilities and valid encryption software may sequentially increase the entropy of files in directories, mimicking the behavior of a ransomware process. Consequently, implementing these solutions outside controlled laboratory conditions proves challenging. Various noteworthy approaches within this category are elaborated below:

2.2.3.1 Approaches based on file access patterns

As cryptographic ransomware encrypts files on the host, these malicious programs alter the status of existing files to an encrypted state. Encryption is a high-entropy operation, meaning that encrypted data exhibits a higher degree of randomness than the original data. Various solutions have been suggested to leverage the identifiable file access patterns expected from cryptographic ransomware. However, the sequential mass modification of files, increasing data entropy to a higher state, is not exclusive to ransomware. This similarity often results in a significant number of false positives in practical implementations.

2.2.3.2 Approaches employing machine learning

These strategies rely on figuring out if a program's actions mimic those of ransomware based on certain traits [41]. A feature set used to teach a machine learning model might include a mix of behaviors, and the presence or absence of these traits can suggest the program's likelihood of being ransomware. For example, the feature set might cover aspects like how files are accessed, the order in which files are modified, and the types of files getting altered (like system files versus user data files).

However, the main hiccup with these methods is the significant number of false alarms in the real world, as mentioned earlier. Legit applications also exhibit behaviors similar to ransomware, leading monitoring tools to mistakenly tag them as threats and causing users to get overwhelmed with alerts. Moreover, there are instances of ransomware purposefully designed to outsmart machine learning solutions. Take Cerber ransomware, for instance; its packaging and loading tricks are specifically crafted to throw off machine learning systems [3].

2.2.4 Solutions oriented towards user training

Traditionally, ransomware assaults, much like other forms of malware attacks, have primarily relied on social engineering tactics, particularly phishing, to persuade unsuspecting users to download and execute malicious content. For instance, ransomware distributors commonly entice victims using phishing emails containing attachments like "invoice.docx.exe" or "resume.txt.js." Consequently, user awareness and training play a crucial role in empowering individuals within the security chain to recognize prevalent social engineering tactics employed by attackers.

However, the primary limitation of relying solely on user awareness and training is that, although it reduces the likelihood of a successful social engineering attack, it doesn't eradicate the risk entirely. Utilizing user awareness and training as a supplementary strategy that complements other defense measures is recommended [4] [5]. Notably, certain ransomware attacks, such as WannaCry and NotPetya, exploit known vulnerabilities and spread like worms without requiring human involvement, rendering such solutions ineffective. Additionally, targeted ransomware attacks have employed alternative, more manual tactics to infiltrate host systems, including brute-forcing Remote Desktop Protocol (RDP).

2.2.5 Vulnerability management solutions

Consistent system patching is a crucial practice for administrators to thwart malware that exploits recognized vulnerabilities for initial infiltration. Notably, recent ransomware instances like WannaCry and Petya garnered attention for capitalizing on a well-known SMB vulnerability [6], specifically utilizing the EternalBlue exploit. Implementing robust vulnerability management can effectively circumvent such ransomware attacks. While regularly applying updates and patches to address known security issues enhances overall security measures, it doesn't offer a conclusive solution for ransomware. This approach primarily guards against one of the multiple attack vectors employed by contemporary ransomware.

2.2.6 Solutions based on cryptography

In the realm of ransomware, where encryption plays a crucial role the focus of these methods revolves around addressing vulnerabilities within the implemented cryptosystem of the ransomware. For example lets consider the 'No More Ransom' initiative [7] , which involves collaboration, between security entities to create customized 'decryptors' for ransomware that have cryptosystems. One such vulnerability is when a ransomware includes a key in its code, which becomes exposed during reverse engineering. However as Ransomware as a Service (RaaS) advances and becomes more sophisticated these implementation flaws become less common.

A recent defensive strategy against ransomware proposes dynamically intercepting CryptoAPI generation. Besides relying on backups this approach serves as the way to regain control over data once files are encrypted by the ransomware on a computer. The main idea behind this proposed escrow system is to safeguard all keys generated on the computer so that file recovery can be possible at a later stage [8]. However, there are challenges associated with this escrow approach. One major obstacle is assuming that ransomware will use an API for generation without considering other possible paths that it may take and not necessarily relying solely on Windows CryptoAPI. Additionally suggesting signatures for APIs adds complexity due to the programming languages used by different types of ransomware each offering numerous choices, for cryptographic tasks.

Essentially the methods used for generating encryption keys are not completely effective, in providing a solution to combat ransomware attacks.

2.3 Summary

While a significant portion of malware research is geared toward understanding how it infiltrates systems for preventive measures, our approach deviates from the norm. We focus on examining the actions taken by malware after it successfully evades all the prevention and detection defenses in place on the host. Given that ransomware primarily tweaks the data landscape through file encryption, our aim is to retrieve decryption keys and glean insights from the ransomware’s process memory during its execution.

CHAPTER 3

Key management in ransomware

3.1 Introduction

Ransomware operates by locking up your files and the management of the keys, to this lock is crucial for the attackers. In order to enhance our defense against attacks it is important to comprehend how these keys are handled. This chapter explores the methods employed by ransomware to manage these keys and identifies vulnerabilities that can help us safeguard ourselves. Our objective is to demonstrate how the techniques for handling these keys in ransomware have evolved over time uncovering any points in their systems to mitigate the threat.

While it would be ideal to prevent attacks we are assuming that the damage has already been done. Therefore we delve into what can be done beyond restoration from backups as these backups may not always be accessible or up to date. Some ransomware has become intelligent enough to locate and encrypt backups over networks. To better defend against scenarios it becomes essential to identify weaknesses in the design and usage of locks in ransomware. This underlines the importance of understanding how these key management systems have evolved in ransomware. However before delving into details, about these management systems used by ransomware lets briefly explore the fundamentals of how digital locks operate.

3.2 Exploring the Fundamentals of Cryptography

This section acts as a reminder, revisiting the common cryptographic types employed by contemporary ransomware. In general, cryptographic algorithms fall into the following two categories:

3.2.1 Delving into Symmetric Keys

Symmetric key cryptography, aptly named, utilizes a single key for both the encryption and decryption processes. A notable instance is the Advanced Encryption Standard (AES), frequently employed by various strains of ransomware [9]. The distinct advantage of symmetric key encryption lies in its expeditious nature compared to asymmetric algorithms. Analogous to a swift criminal endeavor, the primary objective is to swiftly coerce the victim and extract ransom before countermeasures can be enacted. For instance, an antivirus program may detect and isolate the ransomware based on file access and modification patterns. The faster the encryption process occurs, the greater the leverage the ransomware attains by encrypting more user data before detection. Consequently, symmetric key cryptography holds allure for ransomware developers. However, it is imperative to note that improper management of the key could result in inadvertent disclosure. Ransomware must adeptly deploy the key for encryption while concealing it to remain beyond the victim’s reach until the ransom is paid.

3.2.2 Unpacking the World of Asymmetric Keys

Asymmetric key cryptography, also recognized as public key cryptography, employs a pair of interconnected keys—one for encryption (public key) and the other for decryption (private key). Ransomware commonly utilizes the RSA algorithm as an example of such a cipher for decrypting encrypted data. Decrypting the data solely based on the public key and algorithm is currently infeasible. When implemented correctly, this method provides attackers with flexibility while rendering decryption impossible without knowledge of their key. However, a drawback for attackers is that asymmetric encryption can be slower than encryption, leading to larger ciphertexts compared to the original plaintext. This elongates the encryption process, demanding additional storage space on the host system and raising the likelihood of ransomware detection. Consequently, asymmetric encryption is primarily employed to encrypt a session key after it has been used to encrypt user data in what we refer to as a hybrid approach.

3.2.3 Blending Symmetry and Asymmetry: The Hybrid Approach

Typically, more recent iterations of ransomware adopt a hybrid approach, integrating both symmetric and asymmetric encryption techniques to leverage their respective strengths. Initially, user data undergoes swift encryption using a cipher. Subsequently, an additional layer of encryption is applied to the key used in this process, utilizing the attacker’s key. This efficient hybrid key model amalgamates elements from both asymmetric encryption methods and follows a specific sequence of steps:

1. The ransomware infiltrates the targeted system and initiates its operation.
2. Cryptographic Application Programming Interfaces (APIs) on the system generate an encryption key, such as AES-256.
3. The ransomware encrypts this key using a predetermined key like RSA-2048 and then transmits the encrypted version to the attacker.

4. The user’s data gets encrypted using the key.

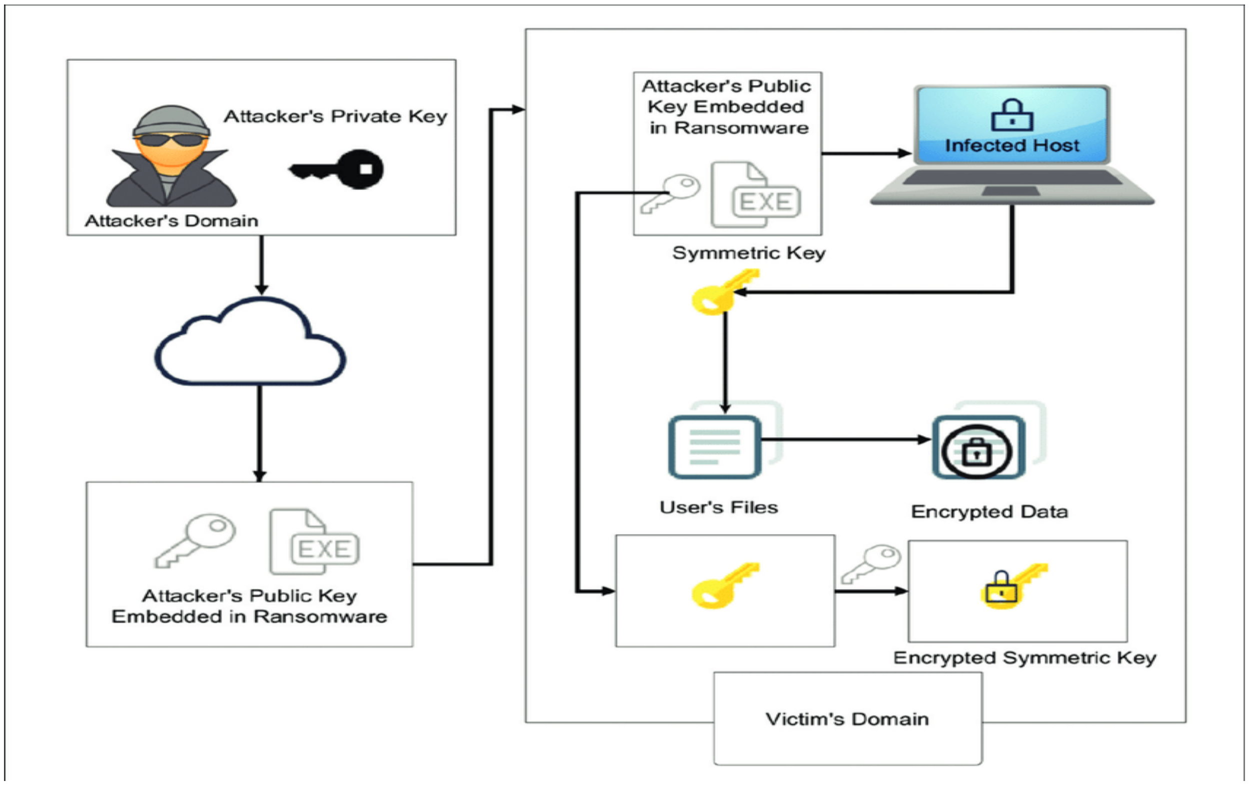


Figure 3.1: Fundamental hybrid encryption structure in ransomware [24]

5. The ransomware systematically erases the key from the system, leaving only the attacker with access to the decryption key.
6. A ransom message is displayed to the user, awaiting payment.

It’s important to note that variations may exist, where the encrypted key is securely stored on the system itself, with any interaction with the attacker limited solely to matters related to payment for decryption.

3.3 Key Management Approaches in Ransomware

Over the years, the approach to managing keys in ransomware has evolved as developers assimilate lessons from their errors. This continual evolution in cybercrime renders it a lucrative endeavor when executed adeptly. Essentially, all cryptoviral attacks adhere to a set of steps:

1. Infiltrate the target and initiate the attack.
2. Obtain the secret encryption key.
3. Encrypt user data.

4. Culminate the operation by demanding a ransom.

The “encryption secret” typically involves the use of a key, and safeguarding this key is paramount for attackers to sustain control over their victims, underscoring the criticality of effective management practices. This discussion delves into observed techniques for managing keys in cryptoviral extortion schemes.

3.3.1 Absence of Key or Encryption

Certain scareware employs deceptive tactics to create a false perception of compromised security, inducing users to make impulsive decisions under duress. Some fake scareware capitalizes on the success of widespread ransomware attacks, masquerading as authentic ransomware. Despite its deceptive appearance, this type of software does not genuinely encrypt files. Instead, it may obfuscate or delete user data, presenting a ransom note soliciting payment. For instance, the pseudo-ransomware AnonPop falsely claimed file encryption and demanded $125 for “decryption.” In reality, there is no legitimate file restoration process in this deceptive scareware. Since files are not securely deleted, recovery is feasible without ransom payment. Due to the absence of actual encryption, key management is irrelevant. The primary objective of such fake ransomware is to profit swiftly without undertaking the complexities of secure file encryption, decryption, and associated key management. This represents a low-effort operation for cybercriminals, particularly when authorities are focused on addressing more substantial malware threats. Furthermore, the absence of encryption operations enhances the likelihood of evading heuristics-based detection methods employed by antivirus solutions, such as triggering alerts related to CryptoAPI access in Windows. Instances of ransomware following this model include AnonPop, original versions of ConsoleCrypt and Nemucod, and certain WannaCry imitators like Aron WanaCrypt0r 2.0 [10].

3.3.2 Essential Decryption in User Domain

“Essential Decryption in User Domain” in the context of ransomware denote critical components required for the decryption process that may be susceptible to user access. This susceptibility arises when users can obtain or reveal the decryption key, essential for decrypting files encrypted by ransomware. Such exposure may occur through activities like scrutinizing the ransomware’s code or examining files within the system or network where the decryption key is stored. If these decryption components are readily accessible to users, it categorizes that ransomware variant as having decryption essentials within the user’s reach.

3.3.2.1 Decryption Necessities on Host Machine

If the decryption key can be obtained by scrutinizing the computer, either during or after the encryption process, it falls into this category. This classification encompasses scenarios where a distinct symmetric key is generated on the compromised machine and then safeguarded using a coded key(public key) embedded in the ransomware. The attacker possesses the corresponding key (private key), but due to inadequate measures, it is labeled as ‘key on host machine,’ making recovery relatively straightforward for victims. Since the symmetric encryption key originated within the user’s domain, there might be a possibility of accessing it without paying any ransom. In some instances, programming errors in coding have inadvertently facilitated key retrieval. For example, the CryptoDefense ransomware overlooked the step of destroying the key on the infected machine, making it easily recoverable from a specific folder [16].

3.3.2.2 Decryption Necessities Distributed Among Peers

In this approach, attackers seek to conceal the decryption key by fragmenting it, potentially encrypting these fragments, and dispersing them among a peer group, such as compromised hosts within an organizational network [12]. The significant advantage for attackers in this strategy is that the key is not centralized on one host, making reverse engineering more challenging. Additionally, attackers can avoid the reliance on successful communication with a C&C server post-infection, which is crucial for the ransomware's functionality, as explained later in this paper. However, there is a risk that a user restoring their host machine from a backup may lose their portion of the key, making it impossible to decrypt other infected peers, as the key cannot be reconstructed. This poses a serious concern for attackers, as the overall success of a cryptoviral extortion campaign hinges on successful decryption upon payment. Without this, future victims lack motivation to pay. In ransom notes, ransomware authors now stress the risk of attempted restoration, warning that such actions may result in the loss of critical decryption information and potential data loss for other network nodes, a point highlighted by Young and Yung [12].Examples of ransomware employing this model: (None observed to date).

3.3.3 Essential Decryption in Attacker Domain

This model encompasses situations where the attacker exclusively possesses the decryption essentials. Overall, it provides the attacker with a strategic advantage by ensuring the key remains secure in their possession. Two variations of this model will be detailed.

3.3.3.1 Decryption Necessities on a Command and Control (C&C) Server: Single Encryption

In this ransomware model, certain variants rely solely on public key cryptography. They encrypt user files using a hardcoded or infection-specific public key upon initial infection. After displaying a ransom note, they send the private decryption key upon receiving payment. While simple, this model has weaknesses, including a single key pair for all victims and slower asymmetric key encryption with increased file size.

CryptoLocker exemplifies a single encryption method, securing user data with an exclusive host-specific asymmetric public key, preventing key sharing among victims as shown in Figure 3.2 [13].

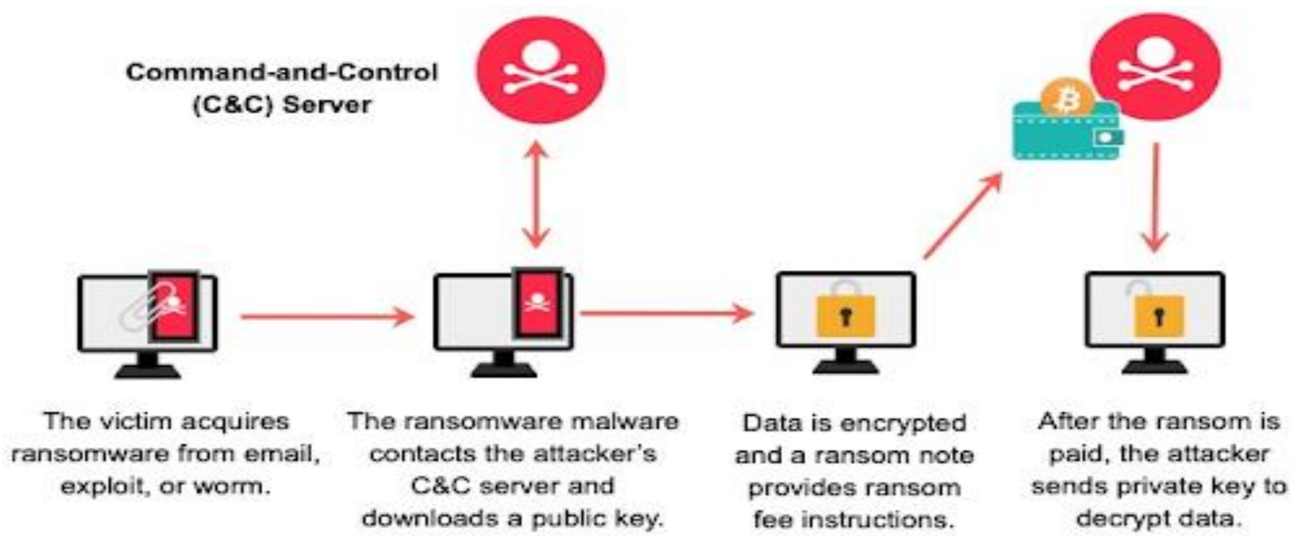

Command-and-Control (C&C) Server
The victim acquires ransomware from email, exploit, or worm.
The ransomware malware contacts the attacker's C&C server and downloads a public key.
Data is encrypted and a ransom note provides ransom fee instructions.
After the ransom is paid, the attacker sends private key to decrypt data.

Figure 3.2: Cryptolocker encryption process [32]

The model employed by CryptoLocker, while free of cryptographic flaws when implemented correctly, is becoming less prevalent in modern ransomware variants like WannaCry. This shift is attributed to the slow nature of asymmetric key encryption and a fundamental operational constraint—the dependence on a connection to a C&C server. Encryption doesn't initiate until the ransomware receives the public key from the C&C server. The potential disruption of this communication by blocking requests to potential C&C servers is feasible, as network administrators maintain blacklists of known C&C server IP addresses [14][33]. Crowd-sourced lists contribute to effective border firewall blocks, causing dormant cryptoviral infections and disrupting the overall ransomware operation.

3.3.3.2 Decryption Necessities on a C&C Server: Hybrid Encryption

Earlier, we explained a hybrid encryption model. Now, we introduce a ransomware instance utilizing a slightly adapted hybrid model.

In the hybrid encryption model demonstrated by WannaCry, the ransomware employs a series of steps to ensure layers of encryption;

1. First the ransomware infiltrates a host. Generates a RSA key pair (Ks, Kp), for the infection.

2. It then uses a predefined key (KA) to encrypt the generated private key (Ks).

3. With the help of a pseudorandom number generator the ransomware creates AES keys for encrypting files.

4. The infection specific public key (Kp) is applied to encrypt all AES keys (S = {K1, K2...Kn}).

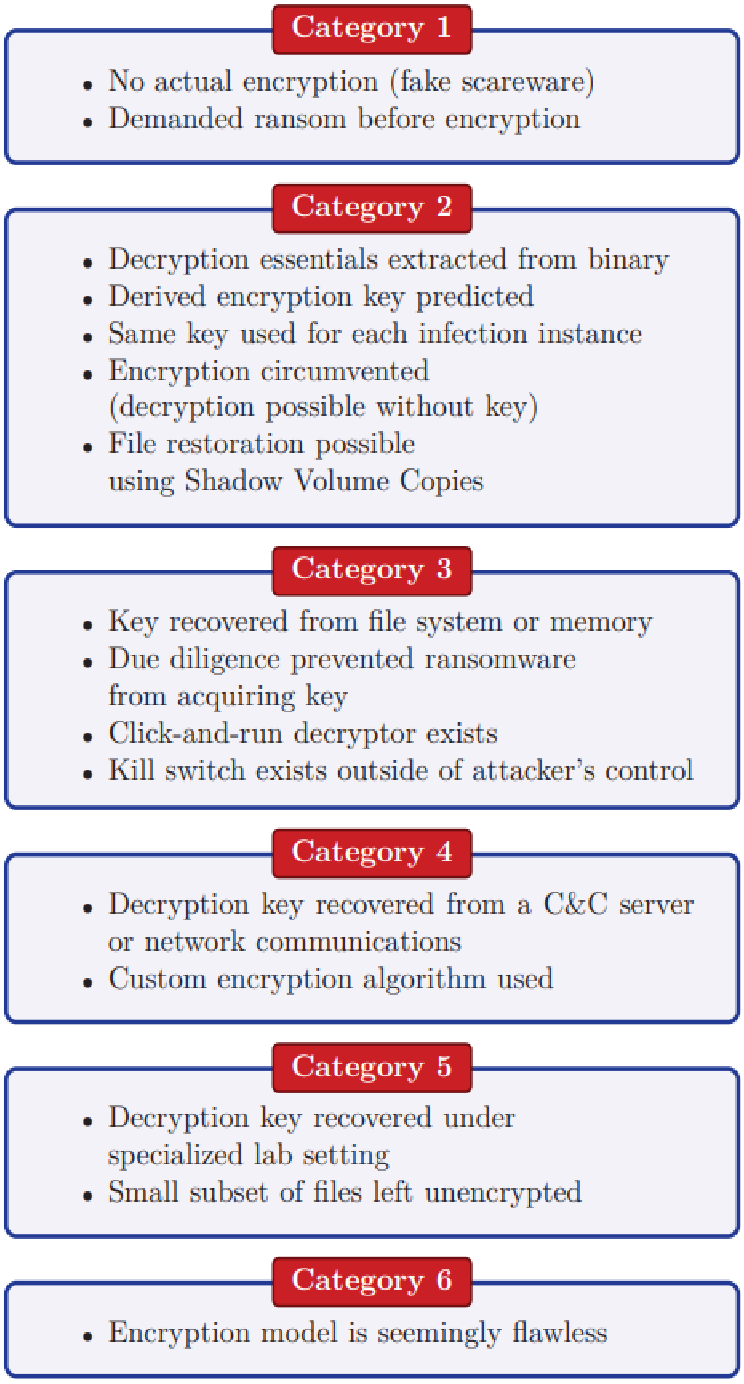
5. To prevent any chance of recovery all AES keys are immediately deleted from memory.

6. Finally the ransomware displays a message demanding payment.

This particular model offers advantages. It ensures encryption using techniques like AES while limiting communication with external entities solely to payment related matters. Moreover it securely retains the attackers key. Notably this model addresses limitations by utilizing distinct AES keys for each file thereby enhancing security, against interruptions and attacks.

3.4 Taxonomy of Ransomware Based on Key Management Approaches

We propose a ransomware classification inspired by the Saffir-Simpson hurricane scale, introducing six categories based on the complexity of reversing encryption without paying the ransom [15]. This system aims to quickly inform security professionals and end-users about the likelihood of decrypting data without payment, considering observed flaws. Analyzing samples from 25 ransomware families, we categorized them similar to hurricane classifications, considering their impact and including recent variants. The classification is based on the time and difficulty of reversing encryption, using static and dynamic analysis to understand functionality and behavior. The goal is to assess the virulence of ransomware infections in terms of their encryption models and provide insights into the challenges of decryption without payment. Note that a ransomware strain may shift categories over time based on discoveries of vulnerabilities in its encryption model. See Figure 3.3 for a summary of ransomware categories.



Category 1

- No actual encryption (fake scareware)
- Demanded ransom before encryption

Category 2

- Decryption essentials extracted from binary
- Derived encryption key predicted
- Same key used for each infection instance
- Encryption circumvented (decryption possible without key)
- File restoration possible using Shadow Volume Copies

Category 3

- Key recovered from file system or memory
- Due diligence prevented ransomware from acquiring key
- Click-and-run decryptor exists
- Kill switch exists outside of attacker's control

Category 4

- Decryption key recovered from a C&C server or network communications
- Custom encryption algorithm used

Category 5

- Decryption key recovered under specialized lab setting
- Small subset of files left unencrypted

Category 6

- Encryption model is seemingly flawless

Figure 3.3: Summary of ransomware categories [24]

3.4.1 Category 1

• Deceptive scareware: pretends to be ransomware but doesn’t encrypt files.

• Operational flaw: some ransomware reveals the ransom note before starting encryption, allowing quick preventive action by victims or antivirus solutions.

3.4.2 Category 2

Decryption essentials in ransomware can be extracted through reverse engineering, like disassembling the binary if a hard-coded key is used. In cases like Linux.Encoder.A, the ransomware used system timestamps to create keys, making decryption easy if the timestamp is accessible [16]. Weaknesses include using the same key for all victims, poor encryption algorithm choices (as seen in desuCrypt’s use of vulnerable RC4 stream cipher) [17], and the possibility of file restoration from neglected system backups like Shadow Volume Copies on NTFS.

3.4.3 Category 3

The decryption key in ransomware can be discovered by an average user from the host machine’s file structure or memory, as seen in CryptoDefense where keys were not securely deleted [16]. Users can also prevent ransomware by interrupting its encryption process, blocking known C&C servers, using decryptors created by the security community [7], or exploiting external kill switches like the one in WannaCry, where a global kill switch in the form of a domain name could abort encryption if registered, rendering the ransomware ineffective [18].

3.4.4 Category 4

The decryption key can be obtained either from a centralized location, such as a compromised C&C server, or extracted through the complex process of monitoring communication between the ransomware and the C&C server. An illustration of this is seen in the CryptoLocker case, where authorities managed to take control of a network of compromised hosts, gaining entry to the decryption essentials for approximately 500,000 victims [19].

Ransomware that employs unique encryption techniques often goes against the fundamental principle of cryptography: “do not create your own encryption method.” While the idea of devising a distinct cipher that seems secure may be tempting, amateur-designed cryptographic methods are likely to be exposed under the scrutiny of professional cryptanalysts [20]. Instances like the early version of the GPCoder ransomware in 2005, which utilized poorly designed custom encryption, serve as examples of the risks associated with this approach [21].

3.4.5 Category 5

Acquiring the decryption key is a rare occurrence and usually requires specific laboratory conditions. Take, for instance, WannaCry, where an unpatched Windows XP system with a cryptographic API vulnerability allowed users to extract prime numbers from RAM, leading to decryption key retrieval [22]. However, this situation is highly dependent on the victim using a particular Windows XP version and being lucky enough that memory space hasn’t been reassigned to another process. Another theoretical avenue involves exploiting a pseudo-random-number-generator (PRNG) flaw in an unpatched Windows XP system, revealing previously generated keys and potentially reversing WannaCry encryption [23]. It’s crucial to understand that these unique conditions are not applicable to most victims.

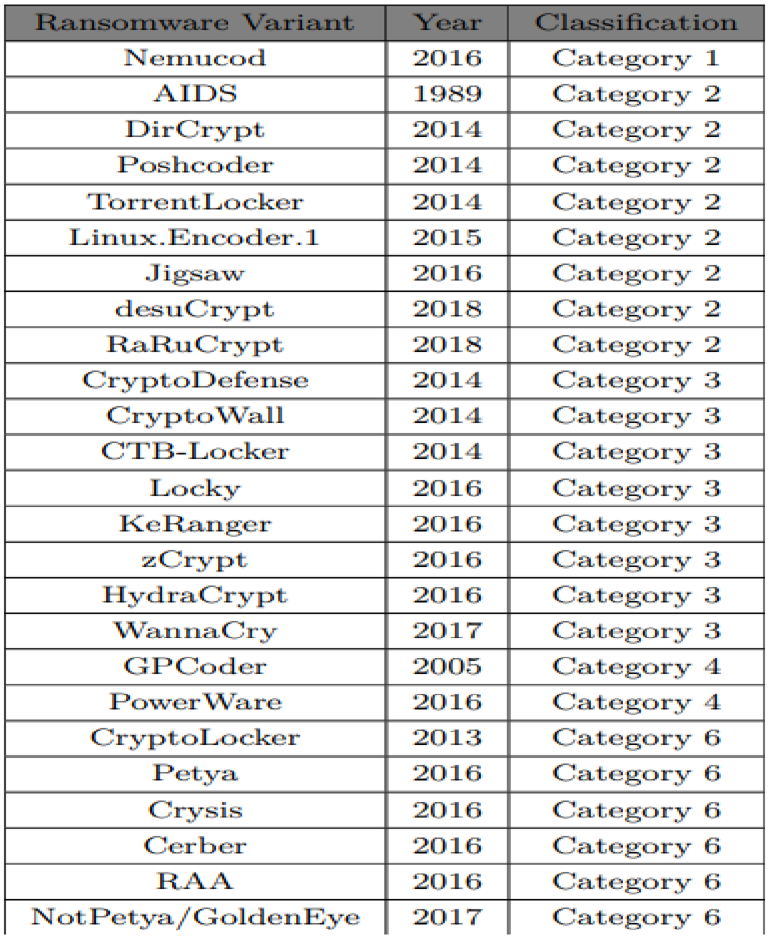
In certain cases, ransomware may leave a small subset of files unencrypted for various reasons. Some ransomware selectively encrypt files based on size, while others decrypt a few files for free to demonstrate their decryption capability. Consequently, only a limited number of victims might be fortunate enough to require only these unencrypted files, tolerating the loss of the remaining ones.

3.4.6 Category 6

The encryption method is highly resistant to cryptographic attacks, implemented flawlessly, and currently has no known vulnerabilities. In simpler terms, there is currently no established method to decrypt the files without fulfilling the ransom payment.

3.4.7 Results of Categorization

We categorized 25 ransomware samples, as depicted in Figure 3.4, utilizing the methodology outlined earlier [24]. The classification was based on identifying vulnerabilities in their encryption models, and we provided insights into the primary reasons each sample was assigned to a specific category.



Ransomware Variant	Year	Classification
Nemucod	2016	Category 1
AIDS	1989	Category 2
DirCrypt	2014	Category 2
Poshcoder	2014	Category 2
TorrentLocker	2014	Category 2
Linux.Encoder.1	2015	Category 2
Jigsaw	2016	Category 2
desuCrypt	2018	Category 2
RaRuCrypt	2018	Category 2
CryptoDefense	2014	Category 3
CryptoWall	2014	Category 3
CTB-Locker	2014	Category 3
Locky	2016	Category 3
KeRanger	2016	Category 3
zCrypt	2016	Category 3
HydraCrypt	2016	Category 3
WannaCry	2017	Category 3
GPCoder	2005	Category 4
PowerWare	2016	Category 4
CryptoLocker	2013	Category 6
Petya	2016	Category 6
Crysis	2016	Category 6
Cerber	2016	Category 6
RAA	2016	Category 6
NotPetya/GoldenEye	2017	Category 6

Figure 3.4: Categorization of Ransomware [24]

3.5 Methodology for Coping with Ransomware

To cope with ransomware effectively,in this thesis, the following systematic approach was employed:

1. **Verification of Threat**: Begin by verifying if the ransomware threat is real or merely scareware. If preventive action is possible, isolate the affected machine to prevent further spread. For virtual machines (VMs), pause the VM and take a memory dump before considering a shutdown. This helps preserve crucial data and enables further analysis of the ransomware.
2. **Reverse Engineering**: Decompile the ransomware binary to locate hard-coded keys. This process involved utilizing tools such as Volatility to perform memory analysis and identify suspicious processes. Subsequently, these processes were dumped and the executable was decompiled to search for embedded keys.
3. **Memory Forensics**: If hard-coded keys are not found, search for encryption keys stored in memory. This involves using live memory forensic tools, as was done in the experiments detailed in this thesis.
4. **CryptoAPI Hooking**: If memory forensics does not yield results, employ an alternative method involving CryptoAPI hooking. The Frida tool was utilized to intercept and analyze cryptographic API calls, potentially revealing the encryption keys.
5. **Monitoring C&C Communication**: Search for communications between the ransomware and its Command & Control (C&C) server to intercept decryption keys. Collaborate with authorities to take control of these servers if feasible. This step was deemed too risky and thus not examined in this thesis.
6. **Exploiting System Vulnerabilities**: If the above methods are unsuccessful, use specific laboratory conditions, such as unpatched system vulnerabilities, to extract decryption keys from memory. This scenario was not examined in this thesis due to its specialized requirements.
7. **Assessment of Encryption Strength**: Finally, assess if the ransomware uses strong encryption methods with no known vulnerabilities. If it does, this indicates that conventional decryption attempts may be futile without paying the ransom.

3.6 Distinctive features identified in contemporary ransomware

Contemporary ransomware poses diverse challenges extending beyond mere data loss. These malicious programs encompass functionalities like deploying trojans and cryptocurrency mining modules. Some leverage sophisticated elliptic curve cryptography, advanced key management systems, novel methods of infection, backup elimination, and other advanced techniques [24]. In this section, we explore the anticipated evolution of highly impactful cryptoviral extortions based on empirical analysis of real-world ransomware samples, showcasing their divergence from conventional trends.

Contemporary ransomware introduces threats that extend beyond basic data encryption, as discussed earlier. These sophisticated variants encompass additional complexities, and we elaborate on the intricacies of these associated threats below.

3.6.1 Integration of cryptojacking routines

The rise of cryptojacking is evident, with ransomware developers now integrating illicit cryptocurrency mining into their tactics. A recent trend involves combining a mining operation with ransomware to generate extra income. BlackRuby ransomware, for instance, conceals a mining process in the background, activating it while waiting for ransom payment [25]. This dual strategy aims to maximize gains by demanding a ransom for encrypted files and utilizing the victim's computing power for cryptocurrency mining.

3.6.2 Utilization of elliptic curve cryptography

Recent examples of ransomware, such as Petya and PetrWrap, opt for the ECIES algorithm over the traditional RSA algorithm to secure their encryption keys [26].

3.6.3 Deliberate destruction of backups

Increasingly, ransomware variations are proactively targeting and encrypting network backups, along with permanently erasing VSS files to eliminate any chances of recovering files.

3.6.4 Introduction of spyware

Some ransomware types, like RAA, go beyond encryption and introduce additional malware, like trojans, to spy on users [27]. Although these ransomware strains may release the decryption key upon payment, there is no confirmation of trojan removal post-payment.

3.6.5 Diversification across multiple attack vectors

Historically, malware primarily infiltrated systems through social engineering in emails, relying on human interaction. However, this method is less efficient than exploiting known vulnerabilities. Notably, WannaCry gained notoriety for its worm-like spread, exploiting the EternalBlue vulnerability [6]. In recent targeted ransomware attacks, sophisticated manual reconnaissance is employed to infiltrate hosts and propagate within internal networks. Another emerging attack vector involves targeting inadequately authenticated RDP services, a method increasingly favored by ransomware operators.

3.7 Summary

A fundamental distinction between cryptoviral extortion programs and regular on-the-fly encryption programs like TrueCrypt or VeraCrypt is the unknown decryption key, unauthorized encryption, and the need for a unique key for each victim. Modern ransomware generates distinct encryption keys to prevent collaboration among victims and facilitate effective decryption. This chapter explores the evolution of key management in ransomware, highlighting novel characteristics in modern variants. The classification methodology introduced assesses technical prowess, excluding overall effectiveness, with plans to expand to reflect overall effectiveness in the future. The focus is on post-execution aspects, assuming successful infiltration, emphasizing the critical role of key management in ransomware threat mitigation.

CHAPTER 4

Extracting Encryption Keys with Memory Forensics

4.1 Introduction

The objective of this chapter was to assess the viability of live forensic methodologies in combating ransomware attacks. It evaluates the attainment of objectives and the fulfillment of the primary aim.

The chapter is structured into distinct sections, each dedicated to the analysis of a specific ransomware sample. These sections present the findings of the conducted experiments. A key focus of live forensics is the scrutiny of system memory, wherein malicious code operates. This examination typically occurs offline to preserve memory integrity, necessitating the capture and preservation of the system's memory for analysis.

4.2 Experiment design

The experimentation phase involved testing ransomware samples within a VirtualBox virtual machine running Windows 7. Ransomware specimens were sourced from reputable repositories, namely https://github.com/ytisf/theZoo and https://bazaar.abuse.ch/ in February 2024. These samples, initially in binary format, were extracted from encrypted ZIP files before use, often requiring manual addition of file extensions prior to execution.

To ensure the safe testing of these ransomware samples, precautions were implemented. The virtual machine's network adapter was set to host-only mode, shared folders between the guest and host were removed, and on the host side, data was backed up externally, and internet connectivity was severed to prevent ransomware escape. Various test folders were strategically placed across the file system, including Desktop, Documents, Pictures, Program Files, Program Files (x86), and Windows. Additionally, a folder was introduced into the Recycle Bin to assess if the ransomware scanned this location. These test folders encompassed diverse file formats—rich-text, text, PDF, and image files—each having a non-zero size.

At an abstract level, the experiments followed a structured approach: executing a ransomware sample in a controlled virtual environment, capturing copies of the machine's volatile memory during execution, and subsequently analyzing these memory captures using forensic tools. The focus of the investigation was on identifying the encryption key employed during the symmetric encryption phase, typically represented by the Advanced Encryption Standard (AES) key. Modern crypto ransomware often utilizes a hybrid encryption approach, combining symmetric and asymmetric encryption. While the public key for asymmetric encryption is delivered with the ransomware, the private key remains in the hands of the attacker. Since the private key is not present on the infected machine, this research concentrated on detecting the key utilized during the symmetric encryption phase, particularly the AES key.

Dynamic analysis forensic tools, identified through a comprehensive literature review, were employed to scrutinize the captured memory samples. The objective was to identify potential candidate AES keys, shedding light on the encryption mechanisms employed by the ransomware during its execution.

4.3 Ransomware sample selection

Three ransomware examples were selected for analysis, all categorized as HCR strains utilizing AES for symmetric encryption:

- Phobos: Emerged in early 2019, Phobos ransomware bears a strong resemblance to the Dharma (a.k.a. CrySis) family, likely distributed by the same group.

- NotPetya: Notorious for its devastating impact, NotPetya stands as one of the most costly cyberattacks in history, estimated at over $10 billion. Unlike WannaCry, NotPetya employs various propagation techniques to infect networked computers.

- Jigsaw: Originating in 2016, Jigsaw encrypts files and progressively deletes them, demanding ransom for decryption and file preservation. Initially dubbed “BitcoinBlackmailer,” it garnered its name from its association with the Saw film franchise, featuring Billy the Puppet.”

4.3.1 Other Ransomware

Several other ransomware samples were initially considered before deciding to use the three examples mentioned above. The following were evaluated before being excluded:

- WannaCry, Thanos & Gpcode: Despite conducting multiple tests on the memory acquired during the execution of this malware using all live forensics tools, no recoverable AES keys were found.
- Cerber: This ransomware does not appear to use AES encryption.
- Locky: The sample of this ransomware required internet access to download the file encryption modules, as they are not included with the initial sample. Allowing this external network access was considered too risky.
- Satan, SamSam & GrandCrab: It was not possible to trigger these samples of ransomware to encrypt any of the control files or display the ransom message.

4.4 Tools

The following tools were used during execution of the experiments:

- REMnux – A complimentary Linux toolkit, REMnux serves as a pivotal resource for malware analysis and reverse engineering endeavors. Offering a clutter-free interface and robust features, REMnux facilitates the seamless examination of malware files.
- Volatility – As an open-source memory forensics framework, Volatility plays a vital role in incident response and malware analysis tasks. Crafted in Python, it boasts cross-platform compatibility, catering to Microsoft Windows, Mac OS X, and Linux environments.
- Process Hacker – An open-source process viewer, Process Hacker is a versatile tool equipped with a suite of functionalities. From aiding in debugging to malware detection and system monitoring, it boasts potent capabilities such as process termination, memory manipulation, and other specialized features.
- PE Studio – Designed for static investigation of Windows executable binaries, PE Studio is a complimentary tool that provides insights into the inner workings of executable files. Offering a range of analysis capabilities, PE Studio aids in identifying potential security risks and vulnerabilities within binaries.
- IDA Pro – It is used as disassembler to parse Windows OS executable files

- dnSpy – It is an open-source tool for reverse engineering .NET applications. It serves as an assembly editor, debugger, and decompiler, enabling users to analyze and modify compiled .NET assemblies. With support for various .NET languages, such as C#, it allows for inspecting and understanding the source code of .NET applications. dnSpy is widely used in security research and debugging activities to examine and analyze the inner workings of .NET applications.

- Winpmem – It is an open source framework that serves for the extraction of volatile memory. It can be found in GitHub and is written in Python.

- FTK imager – It is a virtual memory imaging and data preview tool used to acquire information (memory dumps) in a forensic way by creating copies without making changes in the state of the original evidence. It is a tool widely used for both extraction and memory analysis, thanks to its graphical environment that facilitates its use for the user.

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

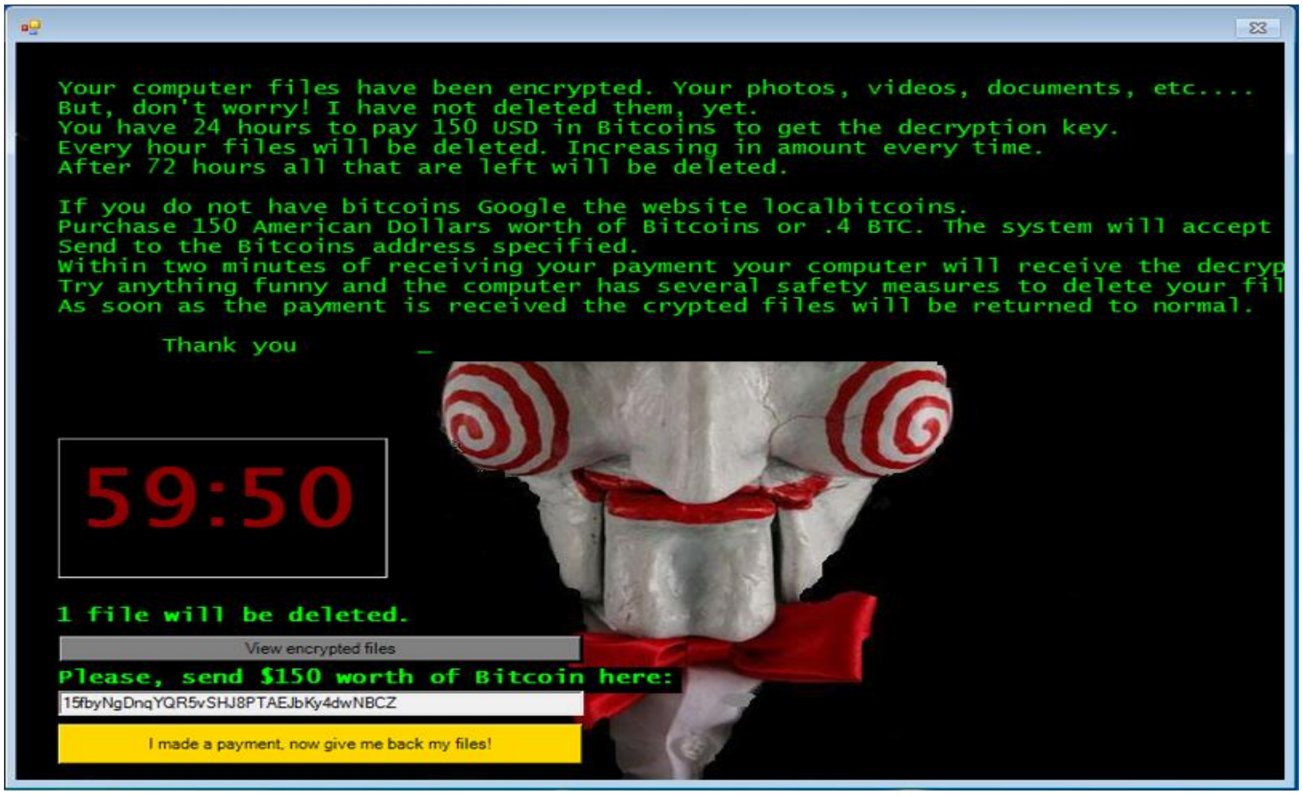
- Interrogate – This was a tool developed by Maartmann-Moe(Maartmann Moeetal.,2009) and was used during their research to investigate both RSA and AES keys in cryptographic applications such as disk encryption and PGP. This is one of the two tools that will be used to try and discover the ransomware's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

4.5 Experiment 1

A safe, isolated virtual environment was created and Jigsaw ransomware sample described in Table 4.1 was executed within it. After approximately 2 minutes the ransom note shown in Figure 4.1 is displayed. Memory was captured from the infected system and its contents were examined using different live forensic tools in an attempt to identify the symmetric encryption keys being used by the ransomware.

Name	Jigsaw
SHA 256	86a391fe7a237f4f17846c53d71e45820411d1a9a6e0c16f22a11ebc491ff9ff
URL	https://github.com/ytisf/theZoo/blob/master/malware/Binaries/Ransomware.Jigsaw/Ransomware.Jigsaw.zip

Table 4.1: Jigsaw Sample Details

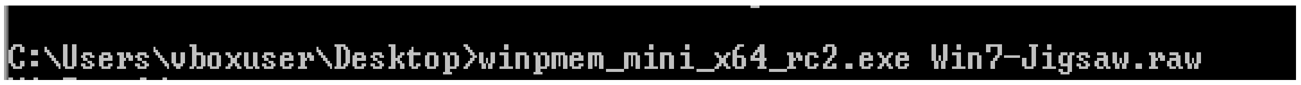


```
Your computer files have been encrypted. Your photos, videos, documents, etc....
But, don't worry! I have not deleted them, yet.
You have 24 hours to pay 150 USD in Bitcoins to get the decryption key.
Every hour files will be deleted. Increasing in amount every time.
After 72 hours all that are left will be deleted.

If you do not have bitcoins Google the website localbitcoins.
Purchase 150 American Dollars worth of Bitcoins or .4 BTC. The system will accept
Send to the Bitcoins address specified.
Within two minutes of receiving your payment your computer will receive the decryp
Try anything funny and the computer has several safety measures to delete your fil
As soon as the payment is received the crypted files will be returned to normal.

        Thank you

59:50

1 file will be deleted.
View encrypted files
Please, send $150 worth of Bitcoin here:
15fbyNgDnqYQR5vSHJ8PTAEJbKy4dwNBCZ
I made a payment, now give me back my files!
```

Figure 4.1: Jigsaw Ransom Message

To obtain a memory capture, the virtual machine was subjected to the following command:

Winpmem_mini_x64_rc2.exe Win7-Jigsaw.raw

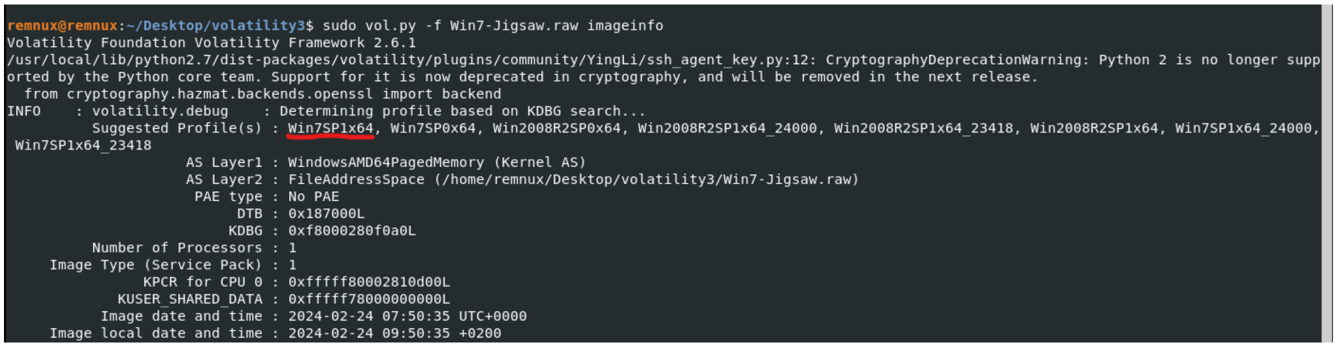


```
C:\Users\vboxuser\Desktop>winpmem_mini_x64_rc2.exe Win7-Jigsaw.raw
```

Figure 4.2: Jigsaw Memory Capture

We then analyzed the memory dump (.raw) obtained in the previous step using Volatility. Using the following commands one can determine the operating system, hardware architecture, and service pack version utilized.

Vol.py -f Win7-Jigsaw.raw imageinfo

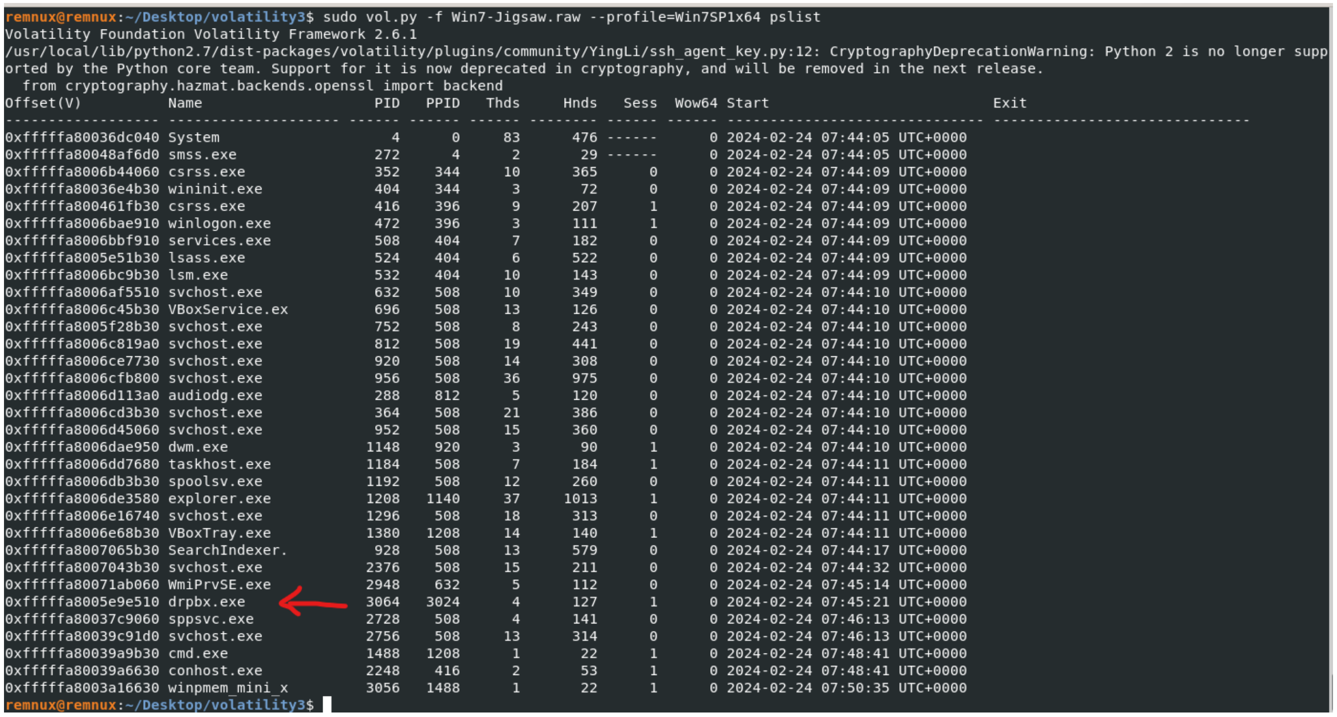


```
remnux@remnux:~/Desktop/volatility3$ sudo vol.py -f Win7-Jigsaw.raw imageinfo
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: CryptographyDeprecationWarning: Python 2 is no longer supp
orted by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography.hazmat.backends.openssl import backend
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_24000, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_24000,
 Win7SP1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/remnux/Desktop/volatility3/Win7-Jigsaw.raw)
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf8000280f0a0L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xfffff80002810d00L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2024-02-24 07:50:35 UTC+0000
     Image local date and time : 2024-02-24 09:50:35 +0200
```

Figure 4.3: Running volatility, which is a tool for memory forensics analysis

The pslist lists all the processes running on that system when we acquired the RAM dump on the memory dump file Win7-Jigsaw.raw . Type the following command to list all the processes running on that system when the RAM dump was acquired:

vol.py -f Win7-Jigsaw.raw —profile= Win7SP1x64 pslist

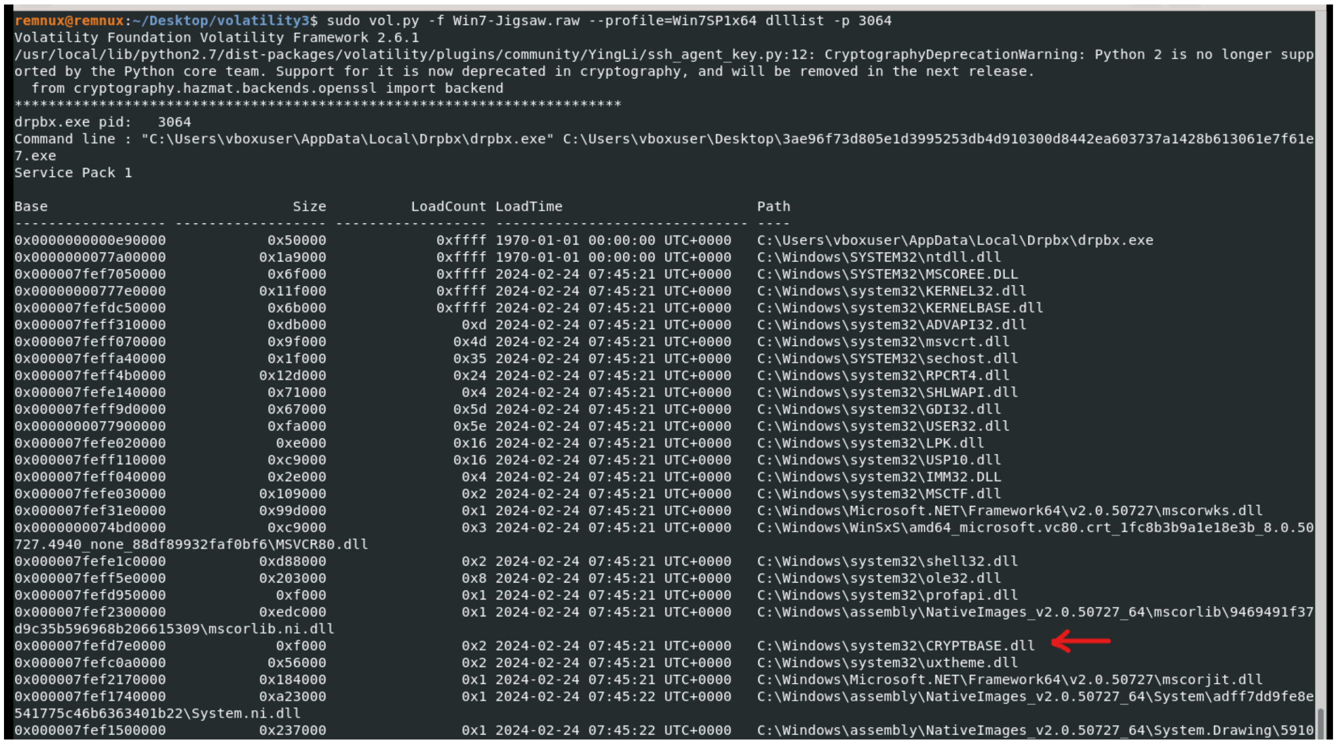


```
remnux@remnux:~/Desktop/volatility3$ sudo vol.py -f Win7-Jigsaw.raw --profile=Win7SP1x64 pslist
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: CryptographyDeprecationWarning: Python 2 is no longer supp
orted by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography.hazmat.backends.openssl import backend
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0xfffffa80036dc040 System                    4      0     83      476 ------      0 2024-02-24 07:44:05 UTC+0000
0xfffffa80048af6d0 smss.exe                272      4      2       29 ------      0 2024-02-24 07:44:05 UTC+0000
0xfffffa8006b44060 csrss.exe               352    344     10      365      0      0 2024-02-24 07:44:09 UTC+0000
0xfffffa80036e4b30 wininit.exe             404    344      3       72      0      0 2024-02-24 07:44:09 UTC+0000
0xfffffa800461fb30 csrss.exe               416    396      9      207      1      0 2024-02-24 07:44:09 UTC+0000
0xfffffa8006bae910 winlogon.exe            472    396      3      111      1      0 2024-02-24 07:44:09 UTC+0000
0xfffffa8006bbf910 services.exe            508    404      7      182      0      0 2024-02-24 07:44:09 UTC+0000
0xfffffa8005e51b30 lsass.exe               524    404      6      522      0      0 2024-02-24 07:44:09 UTC+0000
0xfffffa8006bc9b30 lsm.exe                 532    404     10      143      0      0 2024-02-24 07:44:09 UTC+0000
0xfffffa8006af5510 svchost.exe             632    508     10      349      0      0 2024-02-24 07:44:10 UTC+0000
0xfffffa8006c45b30 VBoxService.ex          696    508     13      126      0      0 2024-02-24 07:44:10 UTC+0000
0xfffffa8005f28b30 svchost.exe             752    508      8      243      0      0 2024-02-24 07:44:10 UTC+0000
0xfffffa8006c819a0 svchost.exe             812    508     19      441      0      0 2024-02-24 07:44:10 UTC+0000
0xfffffa8006ce7730 svchost.exe             920    508     14      308      0      0 2024-02-24 07:44:10 UTC+0000
0xfffffa8006cfb800 svchost.exe             956    508     36      975      0      0 2024-02-24 07:44:10 UTC+0000
0xfffffa8006d113a0 audiodg.exe             288    812      5      120      0      0 2024-02-24 07:44:10 UTC+0000
0xfffffa8006cd3b30 svchost.exe             364    508     21      386      0      0 2024-02-24 07:44:10 UTC+0000
0xfffffa8006d45060 svchost.exe             952    508     15      360      0      0 2024-02-24 07:44:10 UTC+0000
0xfffffa8006dae950 dwm.exe                1148    920      3       90      1      0 2024-02-24 07:44:10 UTC+0000
0xfffffa8006dd7680 taskhost.exe           1184    508      7      184      1      0 2024-02-24 07:44:11 UTC+0000
0xfffffa8006db3b30 spoolsv.exe            1192    508     12      260      0      0 2024-02-24 07:44:11 UTC+0000
0xfffffa8006de3580 explorer.exe           1208   1140     37     1013      1      0 2024-02-24 07:44:11 UTC+0000
0xfffffa8006e16740 svchost.exe            1296    508     18      313      0      0 2024-02-24 07:44:11 UTC+0000
0xfffffa8006e68b30 VBoxTray.exe           1380   1208     14      140      1      0 2024-02-24 07:44:11 UTC+0000
0xfffffa8007065b30 SearchIndexer.          928    508     13      579      0      0 2024-02-24 07:44:17 UTC+0000
0xfffffa8007043b30 svchost.exe            2376    508     15      211      0      0 2024-02-24 07:44:32 UTC+0000
0xfffffa80071ab060 WmiPrvSE.exe           2948    632      5      112      0      0 2024-02-24 07:45:14 UTC+0000
0xfffffa8005e9e510 drpbx.exe              3064   3024      4      127      1      0 2024-02-24 07:45:21 UTC+0000
0xfffffa80037c9060 sppsvc.exe             2728    508      4      141      0      0 2024-02-24 07:46:13 UTC+0000
0xfffffa80039c91d0 svchost.exe            2756    508     13      314      0      0 2024-02-24 07:46:13 UTC+0000
0xfffffa80039a9b30 cmd.exe                1488   1208      1       22      1      0 2024-02-24 07:48:41 UTC+0000
0xfffffa80039a6630 conhost.exe            2248    416      2       53      1      0 2024-02-24 07:48:41 UTC+0000
0xfffffa8003a16630 winpmem_mini_x         3056   1488      1       22      1      0 2024-02-24 07:50:35 UTC+0000
remnux@remnux:~/Desktop/volatility3$
```

Figure 4.4: We see a suspicious process running

We can now take a look with the DllList plugin at the dynamic libraries that are associated with drpbx.exe(process id 3064):

vol.py -f Win7-Jigsaw.raw —profile= Win7SP1x64 dlllist -p 3064

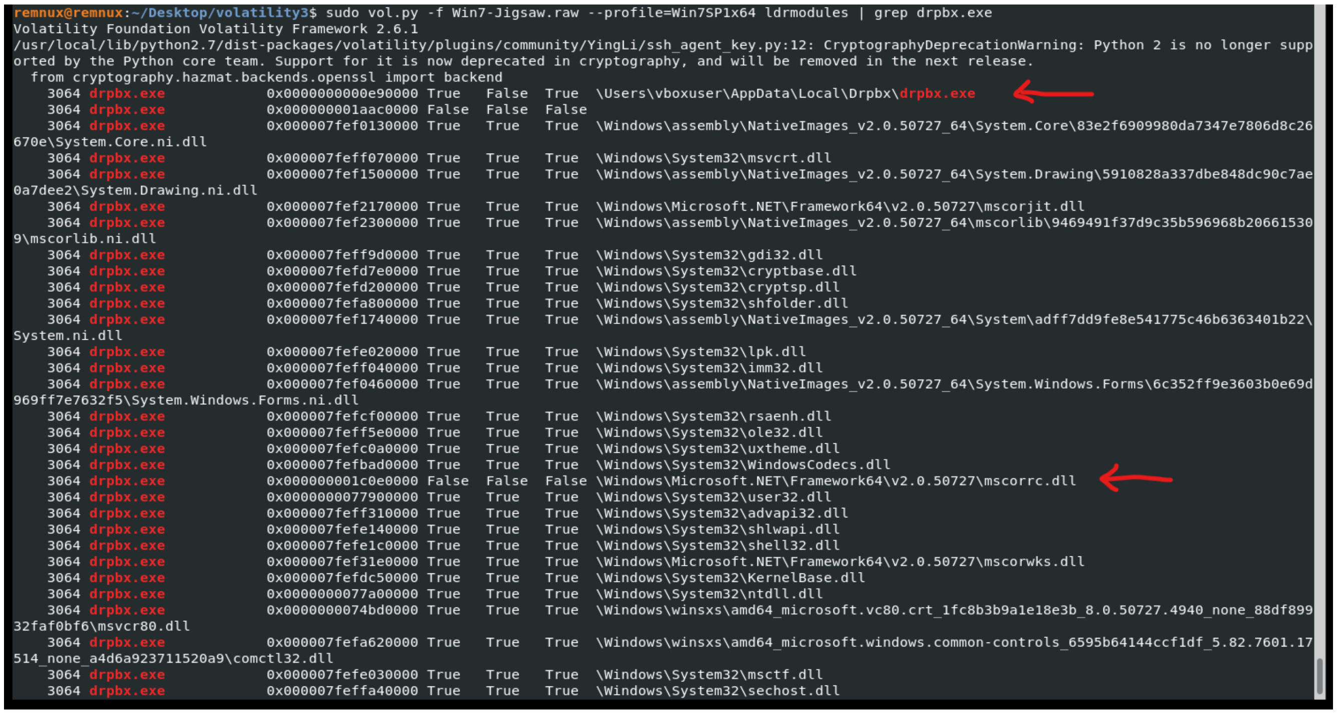


```
remnux@remnux:~/Desktop/volatility3$ sudo vol.py -f Win7-Jigsaw.raw --profile=Win7SP1x64 dlllist -p 3064
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: CryptographyDeprecationWarning: Python 2 is no longer supp
orted by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography.hazmat.backends.openssl import backend
************************************************************************
drpbx.exe pid:   3064
Command line : "C:\Users\vboxuser\AppData\Local\Drpbx\drpbx.exe" C:\Users\vboxuser\Desktop\3ae96f73d805e1d3995253db4d910300d8442ea603737a1428b613061e7f61e
7.exe
Service Pack 1

Base                             Size          LoadCount LoadTime                       Path
------------------ ------------------ ------------------ ------------------------------ ----
0x0000000000e90000            0x50000             0xffff 1970-01-01 00:00:00 UTC+0000   C:\Users\vboxuser\AppData\Local\Drpbx\drpbx.exe
0x0000000077a00000           0x1a9000             0xffff 1970-01-01 00:00:00 UTC+0000   C:\Windows\SYSTEM32\ntdll.dll
0x000007fef7050000            0x6f000             0xffff 2024-02-24 07:45:21 UTC+0000   C:\Windows\SYSTEM32\MSCOREE.DLL
0x00000000777e0000           0x11f000             0xffff 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\KERNEL32.dll
0x000007fefdc50000            0x6b000             0xffff 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\KERNELBASE.dll
0x000007feff310000            0xdb000                0xd 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\ADVAPI32.dll
0x000007feff070000            0x9f000               0x4d 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\msvcrt.dll
0x000007feffa40000            0x1f000               0x35 2024-02-24 07:45:21 UTC+0000   C:\Windows\SYSTEM32\sechost.dll
0x000007feff4b0000           0x12d000               0x24 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\RPCRT4.dll
0x000007fefe140000            0x71000                0x4 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\SHLWAPI.dll
0x000007feff9d0000            0x67000               0x5d 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\GDI32.dll
0x0000000077900000            0xfa000               0x5e 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\USER32.dll
0x000007fefe020000             0xe000               0x16 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\LPK.dll
0x000007feff110000            0xc9000               0x16 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\USP10.dll
0x000007feff040000            0x2e000                0x4 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\IMM32.DLL
0x000007fefe030000           0x109000                0x2 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\MSCTF.dll
0x000007fef31e0000           0x99d000                0x1 2024-02-24 07:45:21 UTC+0000   C:\Windows\Microsoft.NET\Framework64\v2.0.50727\mscorwks.dll
0x0000000074bd0000            0xc9000                0x3 2024-02-24 07:45:21 UTC+0000   C:\Windows\WinSxS\amd64_microsoft.vc80.crt_1fc8b3b9a1e18e3b_8.0.50
727.4940_none_88df89932faf0bf6\MSVCR80.dll
0x000007fefe1c0000           0xd88000                0x2 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\shell32.dll
0x000007feff5e0000           0x203000                0x8 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\ole32.dll
0x000007fefd950000             0xf000                0x1 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\profapi.dll
0x000007fef2300000           0xedc000                0x1 2024-02-24 07:45:21 UTC+0000   C:\Windows\assembly\NativeImages_v2.0.50727_64\mscorlib\9469491f37
d9c35b596968b206615309\mscorlib.ni.dll
0x000007fefd7e0000             0xf000                0x2 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\CRYPTBASE.dll
0x000007fefc0a0000            0x56000                0x2 2024-02-24 07:45:21 UTC+0000   C:\Windows\system32\uxtheme.dll
0x000007fef2170000           0x184000                0x1 2024-02-24 07:45:21 UTC+0000   C:\Windows\Microsoft.NET\Framework64\v2.0.50727\mscorjit.dll
0x000007fef1740000           0xa23000                0x1 2024-02-24 07:45:22 UTC+0000   C:\Windows\assembly\NativeImages_v2.0.50727_64\System\adff7dd9fe8e
541775c46b6363401b22\System.ni.dll
0x000007fef1500000           0x237000                0x1 2024-02-24 07:45:22 UTC+0000   C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Drawing\5910
```

Figure 4.5: The dll responsible for the encryption of files

We aim to list the modules (loaded libraries or executables) that are associated with the process containing the string “drpbx.exe.”:

vol.py -f Win7-Jigsaw.raw —profile= Win7SP1x64 ldrmodules | grep drpbx.exe

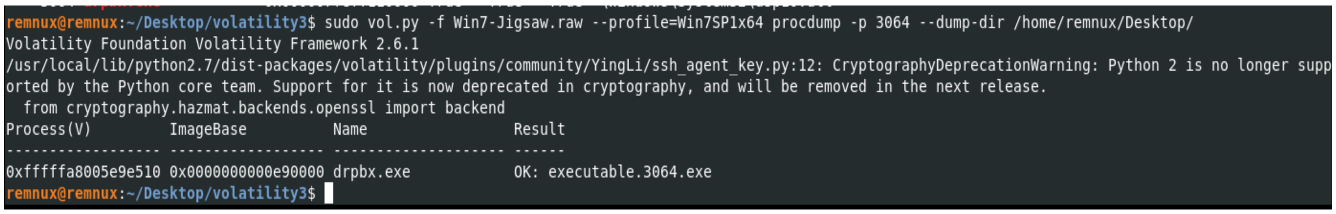


```
remnux@remnux:~/Desktop/volatility3$ sudo vol.py -f Win7-Jigsaw.raw --profile=Win7SP1x64 ldrmodules | grep drpbx.exe
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: CryptographyDeprecationWarning: Python 2 is no longer supp
orted by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography.hazmat.backends.openssl import backend
    3064 drpbx.exe            0x0000000000e90000 True   False  True  \Users\vboxuser\AppData\Local\Drpbx\drpbx.exe
    3064 drpbx.exe            0x000000001aac0000 False  False  False
    3064 drpbx.exe            0x000007fef0130000 True   True   True  \Windows\assembly\NativeImages_v2.0.50727_64\System.Core\83e2f6909980da7347e7806d8c26
670e\System.Core.ni.dll
    3064 drpbx.exe            0x000007feff070000 True   True   True  \Windows\System32\msvcrt.dll
    3064 drpbx.exe            0x000007fef1500000 True   True   True  \Windows\assembly\NativeImages_v2.0.50727_64\System.Drawing\5910828a337dbe848dc90c7ae
0a7dee2\System.Drawing.ni.dll
    3064 drpbx.exe            0x000007fef2170000 True   True   True  \Windows\Microsoft.NET\Framework64\v2.0.50727\mscorjit.dll
    3064 drpbx.exe            0x000007fef2300000 True   True   True  \Windows\assembly\NativeImages_v2.0.50727_64\mscorlib\9469491f37d9c35b596968b20661530
9\mscorlib.ni.dll
    3064 drpbx.exe            0x000007feff9d0000 True   True   True  \Windows\System32\gdi32.dll
    3064 drpbx.exe            0x000007fefd7e0000 True   True   True  \Windows\System32\cryptbase.dll
    3064 drpbx.exe            0x000007fefd200000 True   True   True  \Windows\System32\cryptsp.dll
    3064 drpbx.exe            0x000007fefa800000 True   True   True  \Windows\System32\shfolder.dll
    3064 drpbx.exe            0x000007fef1740000 True   True   True  \Windows\assembly\NativeImages_v2.0.50727_64\System\adff7dd9fe8e541775c46b6363401b22\
System.ni.dll
    3064 drpbx.exe            0x000007fefe020000 True   True   True  \Windows\System32\lpk.dll
    3064 drpbx.exe            0x000007feff040000 True   True   True  \Windows\System32\imm32.dll
    3064 drpbx.exe            0x000007fef0460000 True   True   True  \Windows\assembly\NativeImages_v2.0.50727_64\System.Windows.Forms\6c352ff9e3603b0e69d
969ff7e7632f5\System.Windows.Forms.ni.dll
    3064 drpbx.exe            0x000007fefcf00000 True   True   True  \Windows\System32\rsaenh.dll
    3064 drpbx.exe            0x000007feff5e0000 True   True   True  \Windows\System32\ole32.dll
    3064 drpbx.exe            0x000007fefc0a0000 True   True   True  \Windows\System32\uxtheme.dll
    3064 drpbx.exe            0x000007fefbad0000 True   True   True  \Windows\System32\WindowsCodecs.dll
    3064 drpbx.exe            0x000000001c0e0000 False  False  False \Windows\Microsoft.NET\Framework64\v2.0.50727\mscorrc.dll
    3064 drpbx.exe            0x0000000077900000 True   True   True  \Windows\System32\user32.dll
    3064 drpbx.exe            0x000007feff310000 True   True   True  \Windows\System32\advapi32.dll
    3064 drpbx.exe            0x000007fefe140000 True   True   True  \Windows\System32\shlwapi.dll
    3064 drpbx.exe            0x000007fefe1c0000 True   True   True  \Windows\System32\shell32.dll
    3064 drpbx.exe            0x000007fef31e0000 True   True   True  \Windows\Microsoft.NET\Framework64\v2.0.50727\mscorwks.dll
    3064 drpbx.exe            0x000007fefdc50000 True   True   True  \Windows\System32\KernelBase.dll
    3064 drpbx.exe            0x0000000077a00000 True   True   True  \Windows\System32\ntdll.dll
    3064 drpbx.exe            0x0000000074bd0000 True   True   True  \Windows\winsxs\amd64_microsoft.vc80.crt_1fc8b3b9a1e18e3b_8.0.50727.4940_none_88df899
32faf0bf6\msvcr80.dll
    3064 drpbx.exe            0x000007fefa620000 True   True   True  \Windows\winsxs\amd64_microsoft.windows.common-controls_6595b64144ccf1df_5.82.7601.17
514_none_a4d6a923711520a9\comctl32.dll
    3064 drpbx.exe            0x000007fefe030000 True   True   True  \Windows\System32\msctf.dll
    3064 drpbx.exe            0x000007feffa40000 True   True   True  \Windows\System32\sechost.dll
```

Figure 4.6: These are injected with ransomware

Now, we aim to dump this process to our system and analyze it ,doing some kind of reverse engineering or manual analysis to understand its behavior

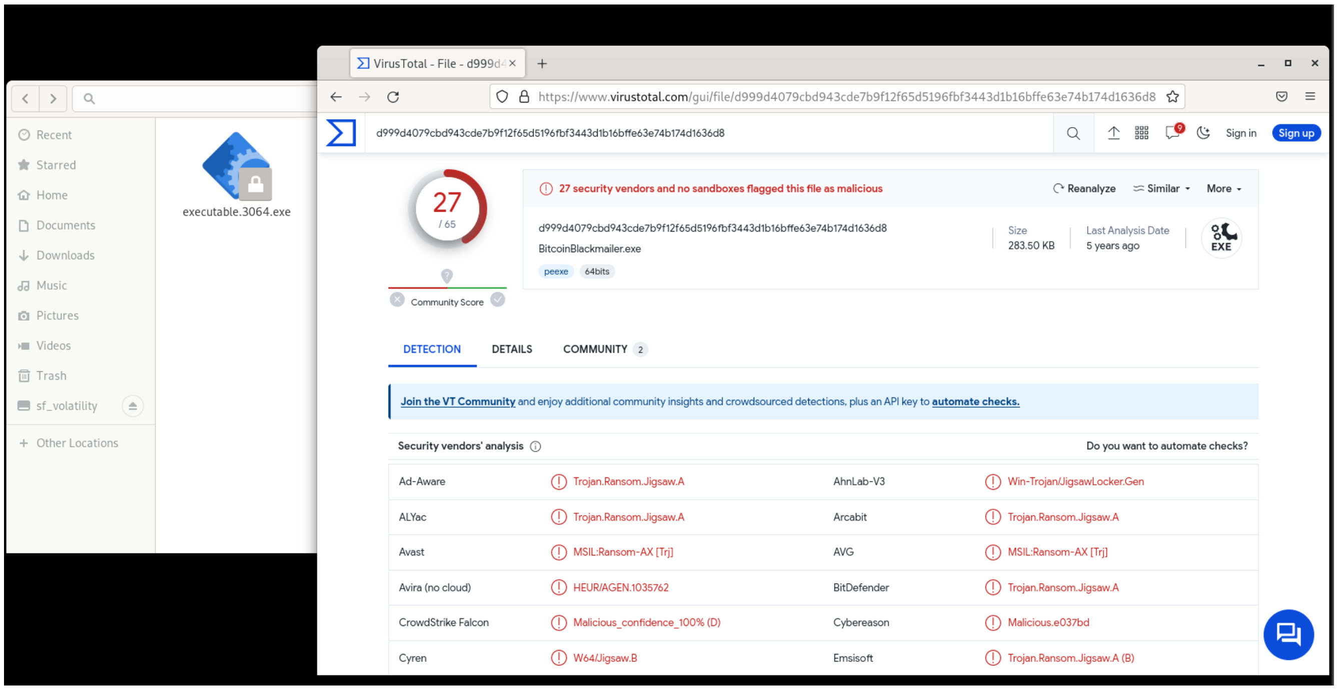
vol.py -f Win7-Jigsaw.raw —profile= Win7SP1x64 procdump -p 3064 —dump-dir /home/remnux/Desktop/



```
remnux@remnux:~/Desktop/volatility3$ sudo vol.py -f Win7-Jigsaw.raw --profile=Win7SP1x64 procdump -p 3064 --dump-dir /home/remnux/Desktop/
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: CryptographyDeprecationWarning: Python 2 is no longer supp
orted by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography.hazmat.backends.openssl import backend
Process(V)         ImageBase          Name                 Result
------------------ ------------------ -------------------- ------
0xfffffa8005e9e510 0x0000000000e90000 drpbx.exe            OK: executable.3064.exe
remnux@remnux:~/Desktop/volatility3$ 
```

Figure 4.7: Memory dump of that particular process

Now we aim to upload our malicious file on the VirusTotal website and check whether it is malicious.



https://www.virustotal.com/gui/file/d999d4079cbd943cde7b9f12f65d5196fbf3443d1b16bffe63e74b174d1636d8

27 / 65

Community Score

27 security vendors and no sandboxes flagged this file as malicious

d999d4079cbd943cde7b9f12f65d5196fbf3443d1b16bffe63e74b174d1636d8
BitcoinBlackmailer.exe
peexe 64bits

Size 283.50 KB

Last Analysis Date 5 years ago

EXE

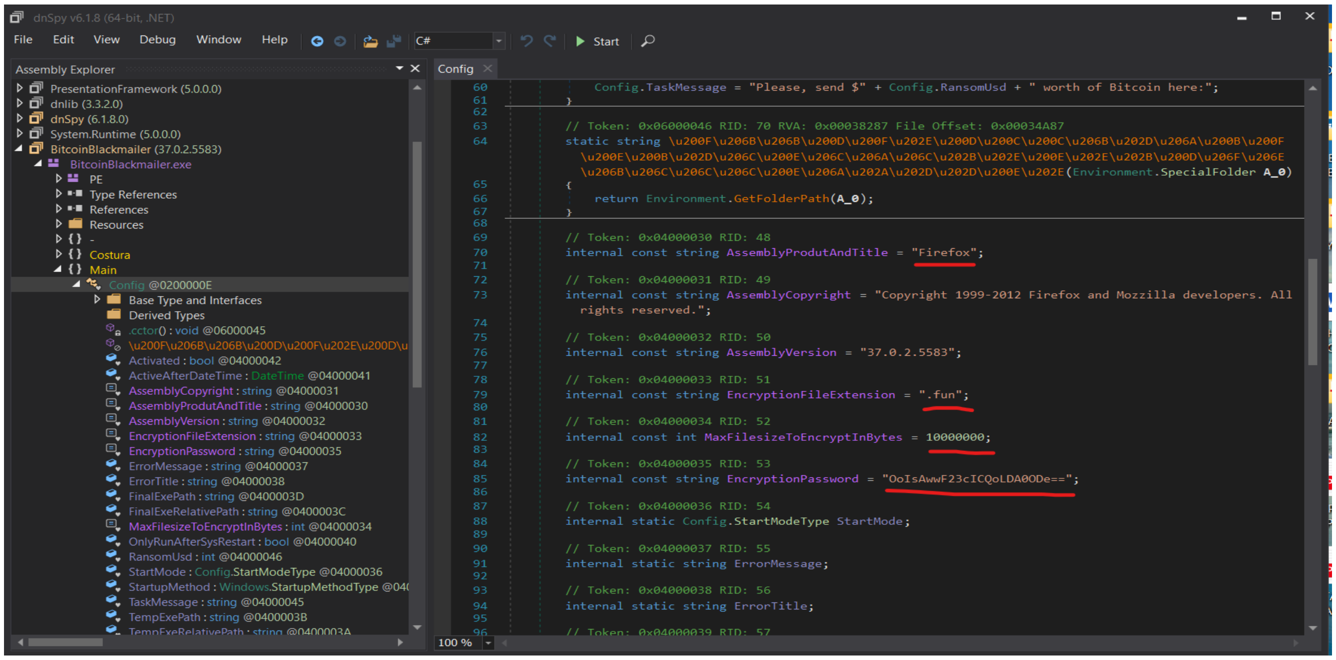
Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Security vendors' analysis — Do you want to automate checks?

Ad-Aware	Trojan.Ransom.Jigsaw.A	AhnLab-V3	Win-Trojan/JigsawLocker.Gen
ALYac	Trojan.Ransom.Jigsaw.A	Arcabit	Trojan.Ransom.Jigsaw.A
Avast	MSIL:Ransom-AX [Trj]	AVG	MSIL:Ransom-AX [Trj]
Avira (no cloud)	HEUR/AGEN.1035762	BitDefender	Trojan.Ransom.Jigsaw.A
CrowdStrike Falcon	Malicious_confidence_100% (D)	Cybereason	Malicious.e037bd
Cyren	W64/Jigsaw.B	Emsisoft	Trojan.Ransom.Jigsaw.A (B)

Figure 4.8: VirusTotal

Now we aim to try and analyze this executable with dnSpy which is going to allow us to decompile it and try to read parts of the source code



```
PresentationFramework (5.0.0.0)
dnlib (3.3.2.0)
dnSpy (6.1.8.0)
System.Runtime (5.0.0.0)
BitcoinBlackmailer (37.0.2.5583)
  BitcoinBlackmailer.exe
    PE
    Type References
    References
    Resources
    { } -
    { } Costura
    { } Main
      Config @0200000E
        Base Type and Interfaces
        Derived Types
        .cctor() : void @06000045
        \u200F\u206B\u206B\u200D\u200F\u202E\u200D\u
        Activated : bool @04000042
        ActiveAfterDateTime : DateTime @04000041
        AssemblyCopyright : string @04000031
        AssemblyProdutAndTitle : string @04000030
        AssemblyVersion : string @04000032
        EncryptionFileExtension : string @04000033
        EncryptionPassword : string @04000035
        ErrorMessage : string @04000037
        ErrorTitle : string @04000038
        FinalExePath : string @0400003D
        FinalExeRelativePath : string @0400003C
        MaxFilesizeToEncryptInBytes : int @04000034
        OnlyRunAfterSysRestart : bool @04000040
        RansomUsd : int @04000046
        StartMode : Config.StartModeType @04000036
        StartupMethod : Windows.StartupMethodType @04(
        TaskMessage : string @04000045
        TempExePath : string @0400003B
        TempExeRelativePath : string @0400003A
```

```csharp
        Config.TaskMessage = "Please, send $" + Config.RansomUsd + " worth of Bitcoin here:";
    }

    // Token: 0x06000046 RID: 70 RVA: 0x00038287 File Offset: 0x00034A87
    static string \u200F\u206B\u206B\u200D\u200F\u202E\u200D\u200C\u200C\u206B\u202D\u206A\u200B\u200F
      \u200E\u200B\u202D\u206C\u200E\u206C\u206A\u206C\u202B\u202E\u200E\u202E\u202B\u200D\u206F\u206E
      \u206B\u206C\u206C\u206C\u200E\u206A\u202A\u202D\u202D\u200E\u202E(Environment.SpecialFolder A_0)
    {
        return Environment.GetFolderPath(A_0);
    }

    // Token: 0x04000030 RID: 48
    internal const string AssemblyProdutAndTitle = "Firefox";

    // Token: 0x04000031 RID: 49
    internal const string AssemblyCopyright = "Copyright 1999-2012 Firefox and Mozzilla developers. All
      rights reserved.";

    // Token: 0x04000032 RID: 50
    internal const string AssemblyVersion = "37.0.2.5583";

    // Token: 0x04000033 RID: 51
    internal const string EncryptionFileExtension = ".fun";

    // Token: 0x04000034 RID: 52
    internal const int MaxFilesizeToEncryptInBytes = 10000000;

    // Token: 0x04000035 RID: 53
    internal const string EncryptionPassword = "OoIsAwwF23cICQoLDA0ODe==";

    // Token: 0x04000036 RID: 54
    internal static Config.StartModeType StartMode;

    // Token: 0x04000037 RID: 55
    internal static string ErrorMessage;

    // Token: 0x04000038 RID: 56
    internal static string ErrorTitle;

    // Token: 0x04000039 RID: 57
```

Figure 4.9: Jigsaw Decompilation

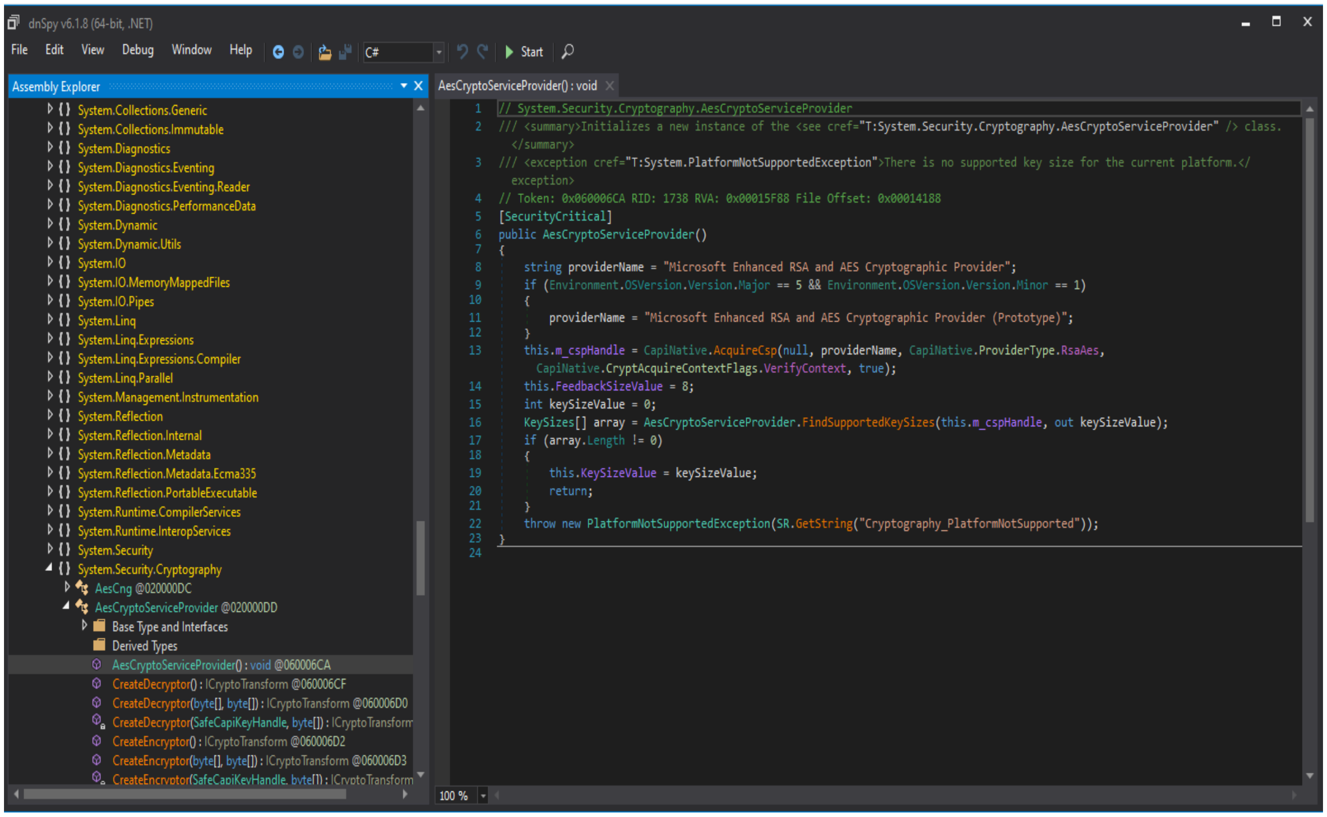
We see that we've got a program called BitcoinBlackmailer and if we open it up we can see that we've got a 'Main' function and under that we've got different forms but most importantly we've got a 'Config' and if we go ahead and open that, we will see that we've got the product title which is Firefox that's what this masquerades us

We've got the encryption file extension ".fun"

We know the max file size to encrypt "10000000" bytes

Then, we have here the encryption password "OoIsAwwF23cICQoLDA0Ode==" which is a static key which means that in this case they key is hardcoded within the actual ransomware executable.

I discovered a function named AesCrytoServiceProvider, indicating that AES encryption is utilized.


dnSpy v6.1.8 (64-bit, .NET)
File
Edit
View
Debug
Window
Help
C#
Start
Assembly Explorer
AesCryptoServiceProvider() : void
System.Collections.Generic
System.Collections.Immutable
System.Diagnostics
System.Diagnostics.Eventing
System.Diagnostics.Eventing.Reader
System.Diagnostics.PerformanceData
System.Dynamic
System.Dynamic.Utils
System.IO
System.IO.MemoryMappedFiles
System.IO.Pipes
System.Linq
System.Linq.Expressions
System.Linq.Expressions.Compiler
System.Linq.Parallel
System.Management.Instrumentation
System.Reflection
System.Reflection.Internal
System.Reflection.Metadata
System.Reflection.Metadata.Ecma335
System.Reflection.PortableExecutable
System.Runtime.CompilerServices
System.Runtime.InteropServices
System.Security
System.Security.Cryptography
AesCng @020000DC
AesCryptoServiceProvider @020000DD
Base Type and Interfaces
Derived Types
AesCryptoServiceProvider() : void @060006CA
CreateDecryptor() : ICryptoTransform @060006CF
CreateDecryptor(byte[], byte[]) : ICryptoTransform @060006D0
CreateDecryptor(SafeCapiKeyHandle, byte[]) : ICryptoTransform
CreateEncryptor() : ICryptoTransform @060006D2
CreateEncryptor(byte[], byte[]) : ICryptoTransform @060006D3
CreateEncryptor(SafeCapiKeyHandle, byte[]) : ICryptoTransform
100 %

```
// System.Security.Cryptography.AesCryptoServiceProvider
/// <summary>Initializes a new instance of the <see cref="T:System.Security.Cryptography.AesCryptoServiceProvider" /> class.
  </summary>
/// <exception cref="T:System.PlatformNotSupportedException">There is no supported key size for the current platform.</
  exception>
// Token: 0x060006CA RID: 1738 RVA: 0x00015F88 File Offset: 0x00014188
[SecurityCritical]
public AesCryptoServiceProvider()
{
    string providerName = "Microsoft Enhanced RSA and AES Cryptographic Provider";
    if (Environment.OSVersion.Version.Major == 5 && Environment.OSVersion.Version.Minor == 1)
    {
        providerName = "Microsoft Enhanced RSA and AES Cryptographic Provider (Prototype)";
    }
    this.m_cspHandle = CapiNative.AcquireCsp(null, providerName, CapiNative.ProviderType.RsaAes,
      CapiNative.CryptAcquireContextFlags.VerifyContext, true);
    this.FeedbackSizeValue = 8;
    int keySizeValue = 0;
    KeySizes[] array = AesCryptoServiceProvider.FindSupportedKeySizes(this.m_cspHandle, out keySizeValue);
    if (array.Length != 0)
    {
        this.KeySizeValue = keySizeValue;
        return;
    }
    throw new PlatformNotSupportedException(SR.GetString("Cryptography_PlatformNotSupported"));
}
```

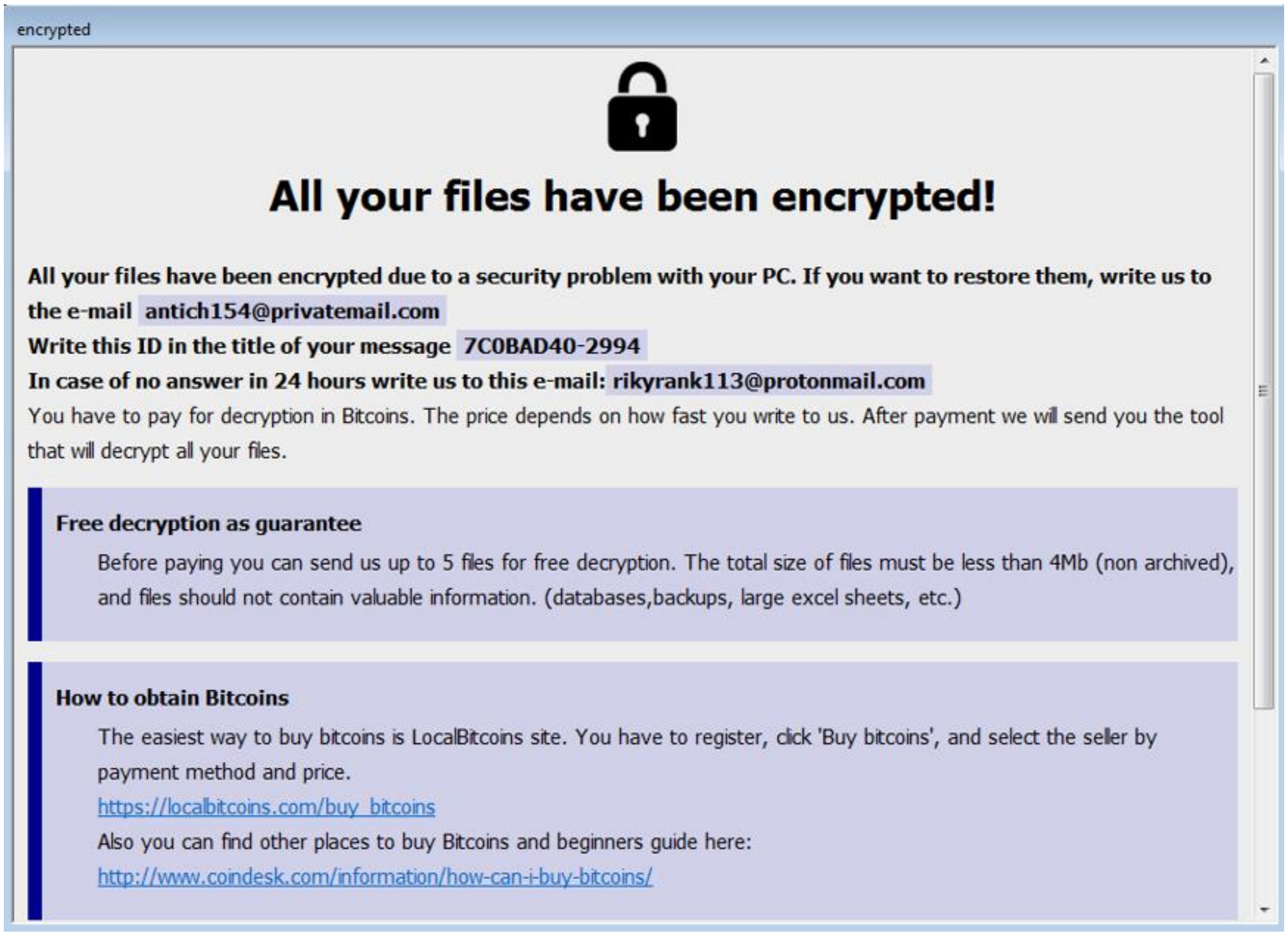
Figure 4.10: Jigsaw Decompilation

4.6 Experiment 2

A safe, isolated virtual environment was created and Phobos ransomware sample described in Table 4.2 was executed within it. After approximately 2 minutes the ransom note shown in Figure 4.11 is displayed. Memory was captured from the infected system and its contents was examined using two different live forensic tools in an attempt to identify the symmetric encryption keys being used by the ransomware.

Name	Phobos
SHA 256	9bd421c6f7f7d8278036944fcad3e04db408619678acf1b2024ef69d85c3932b
URL	https://bazaar.abuse.ch/sample/9bd421c6f7f7d8278036944fcad3e04db408619678acf1b2024ef69d85c3932b/

Table 4.2: Phobos Sample Details



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 antich154@privatemail.com

Write this ID in the title of your message 7C0BAD40-2994

In case of no answer in 24 hours write us to this e-mail: rikyrank113@protonmail.com

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 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 4Mb (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

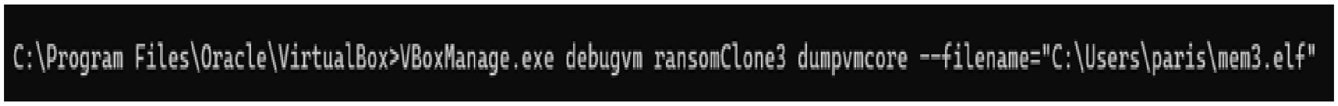
Also you can find other places to buy Bitcoins and beginners guide here:

http://www.coindesk.com/information/how-can-i-buy-bitcoins/

Figure 4.11: Phobos Ransom Message

To capture the memory, the following command was executed on the host machine:

VboxManage.exe debugvm <Vbox Machine Name> dumpvmcore—filename
<filename>.elf

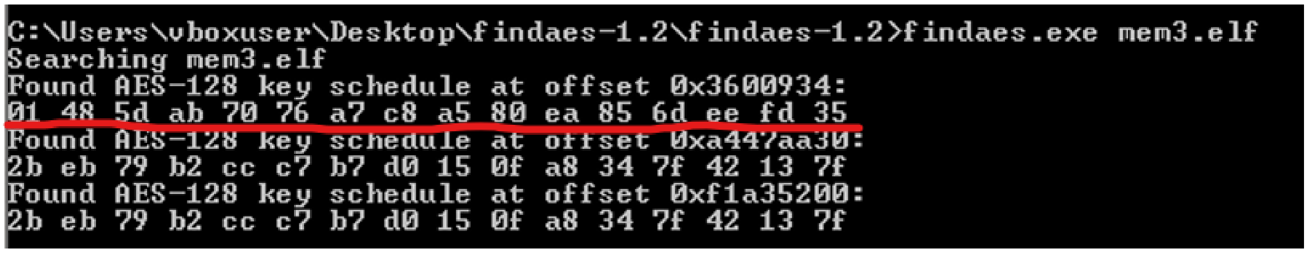


```
C:\Program Files\Oracle\VirtualBox>VBoxManage.exe debugvm ransomClone3 dumpvmcore --filename="C:\Users\paris\mem3.elf"
```

Figure 4.12: Memory Capture

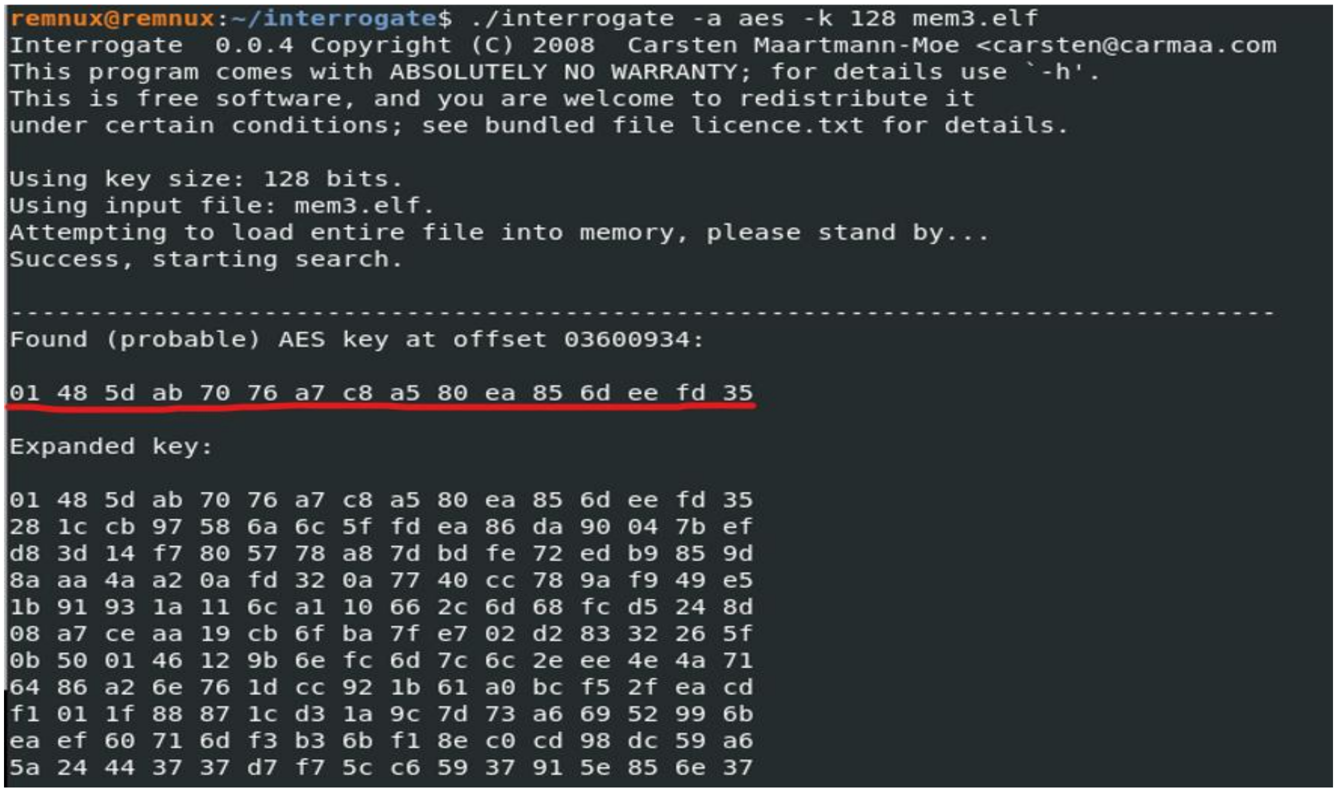
Both live forensics tools employed to analyze the memory dumps successfully detected AES keys in memory. However, some of these identified keys were disregarded as they existed before the ransomware execution. However all the tools also successfully identified the 128 bit key used by the ransomware to encrypt the files using the following commands:

findaes.exe mem3.elf
or
interrogate -a aes -k 128 mem3.elf



```
C:\Users\vboxuser\Desktop\findaes-1.2\findaes-1.2>findaes.exe mem3.elf
Searching mem3.elf
Found AES-128 key schedule at offset 0x3600934:
01 48 5d ab 70 76 a7 c8 a5 80 ea 85 6d ee fd 35
Found AES-128 key schedule at offset 0xa447aa30:
2b eb 79 b2 cc c7 b7 d0 15 0f a8 34 7f 42 13 7f
Found AES-128 key schedule at offset 0xf1a35200:
2b eb 79 b2 cc c7 b7 d0 15 0f a8 34 7f 42 13 7f
```

Figure 4.13: Phobos findaes Output



```
remnux@remnux:~/interrogate$ ./interrogate -a aes -k 128 mem3.elf
Interrogate  0.0.4 Copyright (C) 2008  Carsten Maartmann-Moe <carsten@carmaa.com
This program comes with ABSOLUTELY NO WARRANTY; for details use `-h'.
This is free software, and you are welcome to redistribute it
under certain conditions; see bundled file licence.txt for details.

Using key size: 128 bits.
Using input file: mem3.elf.
Attempting to load entire file into memory, please stand by...
Success, starting search.

--------------------------------------------------------------------------------
Found (probable) AES key at offset 03600934:

01 48 5d ab 70 76 a7 c8 a5 80 ea 85 6d ee fd 35

Expanded key:

01 48 5d ab 70 76 a7 c8 a5 80 ea 85 6d ee fd 35
28 1c cb 97 58 6a 6c 5f fd ea 86 da 90 04 7b ef
d8 3d 14 f7 80 57 78 a8 7d bd fe 72 ed b9 85 9d
8a aa 4a a2 0a fd 32 0a 77 40 cc 78 9a f9 49 e5
1b 91 93 1a 11 6c a1 10 66 2c 6d 68 fc d5 24 8d
08 a7 ce aa 19 cb 6f ba 7f e7 02 d2 83 32 26 5f
0b 50 01 46 12 9b 6e fc 6d 7c 6c 2e ee 4e 4a 71
64 86 a2 6e 76 1d cc 92 1b 61 a0 bc f5 2f ea cd
f1 01 1f 88 87 1c d3 1a 9c 7d 73 a6 69 52 99 6b
ea ef 60 71 6d f3 b3 6b f1 8e c0 cd 98 dc 59 a6
5a 24 44 37 37 d7 f7 5c c6 59 37 91 5e 85 6e 37
```

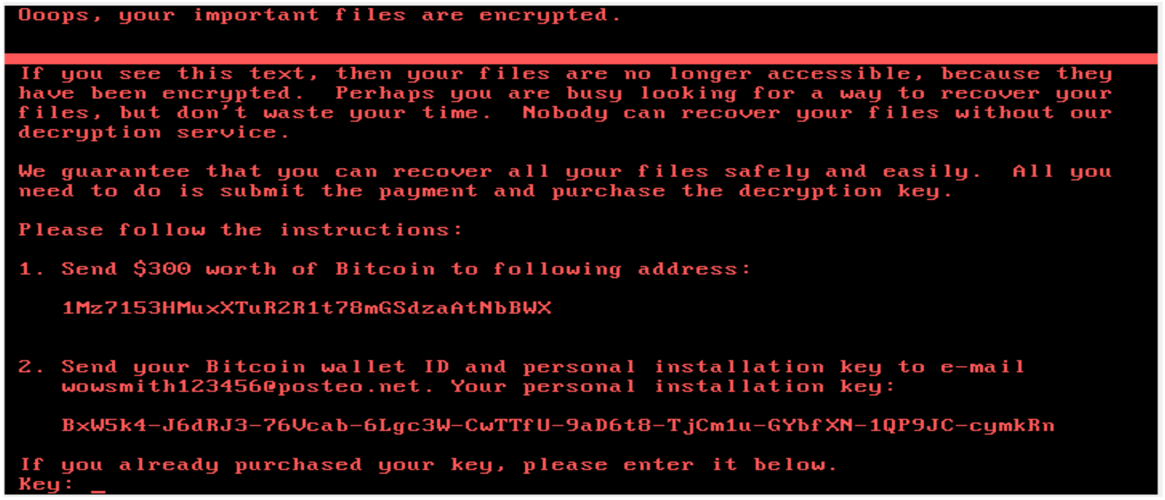
Figure 4.14: Phobos Interrogate Output

4.7 Experiment 3

A safe, isolated virtual environment was created and NotPetya ransomware sample described in Table 4.3 was executed within it. After approximately 2 minutes the ransom note shown in Figure 4.15 is displayed.We captured memory from the infected system and analyzed its contents using two distinct live forensic tools. Our goal was to identify the symmetric encryption keys utilized by the ransomware.

Name	NotPetya
SHA 256	027cc450ef5f8c5f653329641ec1fed91f694e0d229928963b30f6b0d7d3a745d7d3a745
URL	https://github.com/fabrimagic72/malware-samples/tree/master/Ransomware/NotPetya

Table 4.3: NotPetya Sample Details

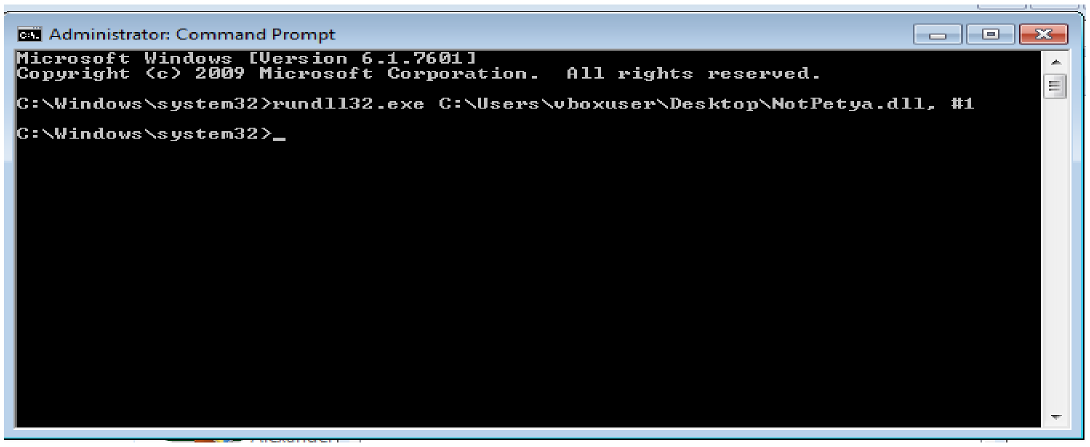


```
Ooops, your important files are encrypted.

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:

   BxW5k4-J6dRJ3-76Vcab-6Lgc3W-CwTTfU-9aD6t8-TjCm1u-GYbfXN-1QP9JC-cymkRn

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

Figure 4.15: NotPetya Ransom Message

The test ransomware sample was started, using the command below:

C:\Windows\system32\rundll32.exe C:\Users\vboxuser\Desktop\NotPetya.dll, #1

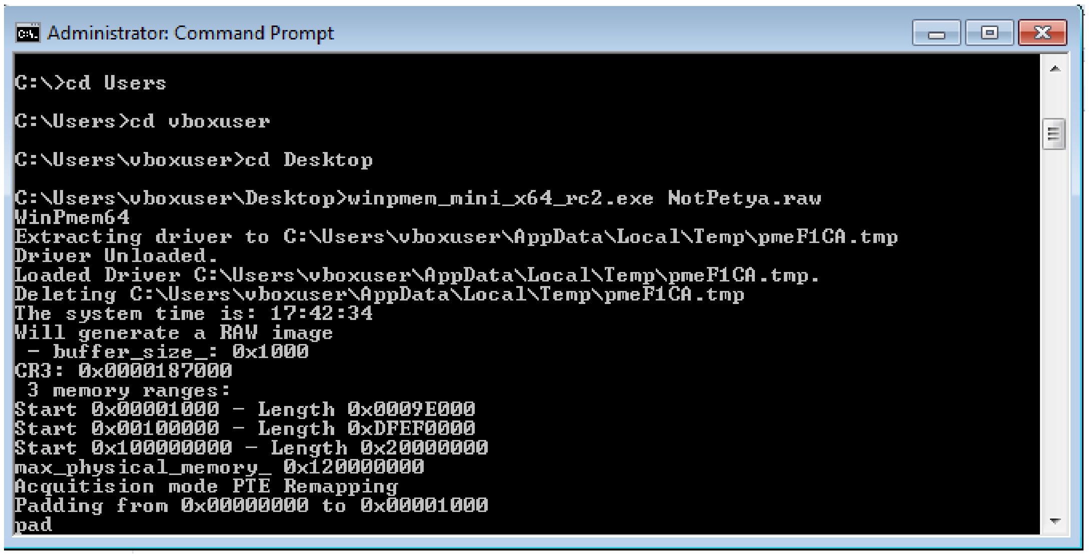


```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>rundll32.exe C:\Users\vboxuser\Desktop\NotPetya.dll, #1

C:\Windows\system32>_
```

Figure 4.16: NotPetya execution command

To capture the memory, the following command was executed on the virtual machine :

winpmem_mini_x64_rc2.exe NotPetya.raw


Administrator: Command Prompt

```
C:\>cd Users

C:\Users>cd vboxuser

C:\Users\vboxuser>cd Desktop

C:\Users\vboxuser\Desktop>winpmem_mini_x64_rc2.exe NotPetya.raw
WinPmem64
Extracting driver to C:\Users\vboxuser\AppData\Local\Temp\pmeF1CA.tmp
Driver Unloaded.
Loaded Driver C:\Users\vboxuser\AppData\Local\Temp\pmeF1CA.tmp.
Deleting C:\Users\vboxuser\AppData\Local\Temp\pmeF1CA.tmp
The system time is: 17:42:34
Will generate a RAW image
 - buffer_size_: 0x1000
CR3: 0x0000187000
 3 memory ranges:
Start 0x00001000 - Length 0x0009E000
Start 0x00100000 - Length 0xDFEF0000
Start 0x100000000 - Length 0x20000000
max_physical_memory_ 0x120000000
Acquitision mode PTE Remapping
Padding from 0x00000000 to 0x00001000
pad
```

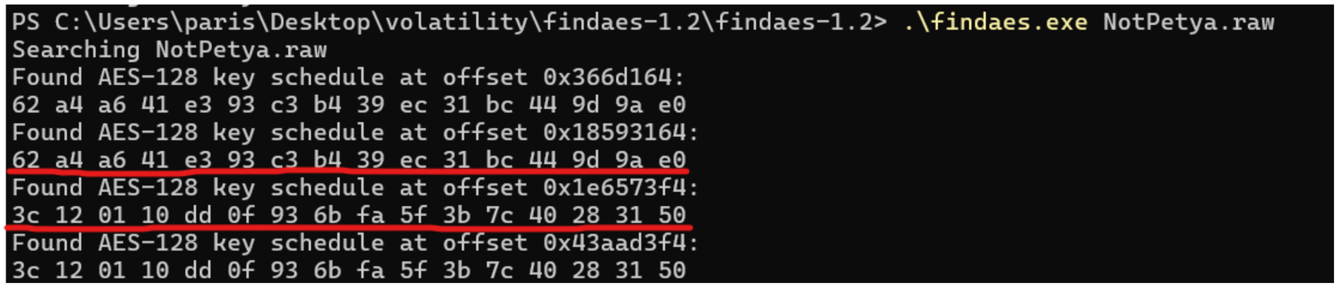
Figure 4.17: NotPetya memory capture

All two 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 128 bit keys used by the ransomware to encrypt the files using the following commands:

findaes.exe NotPetya.raw

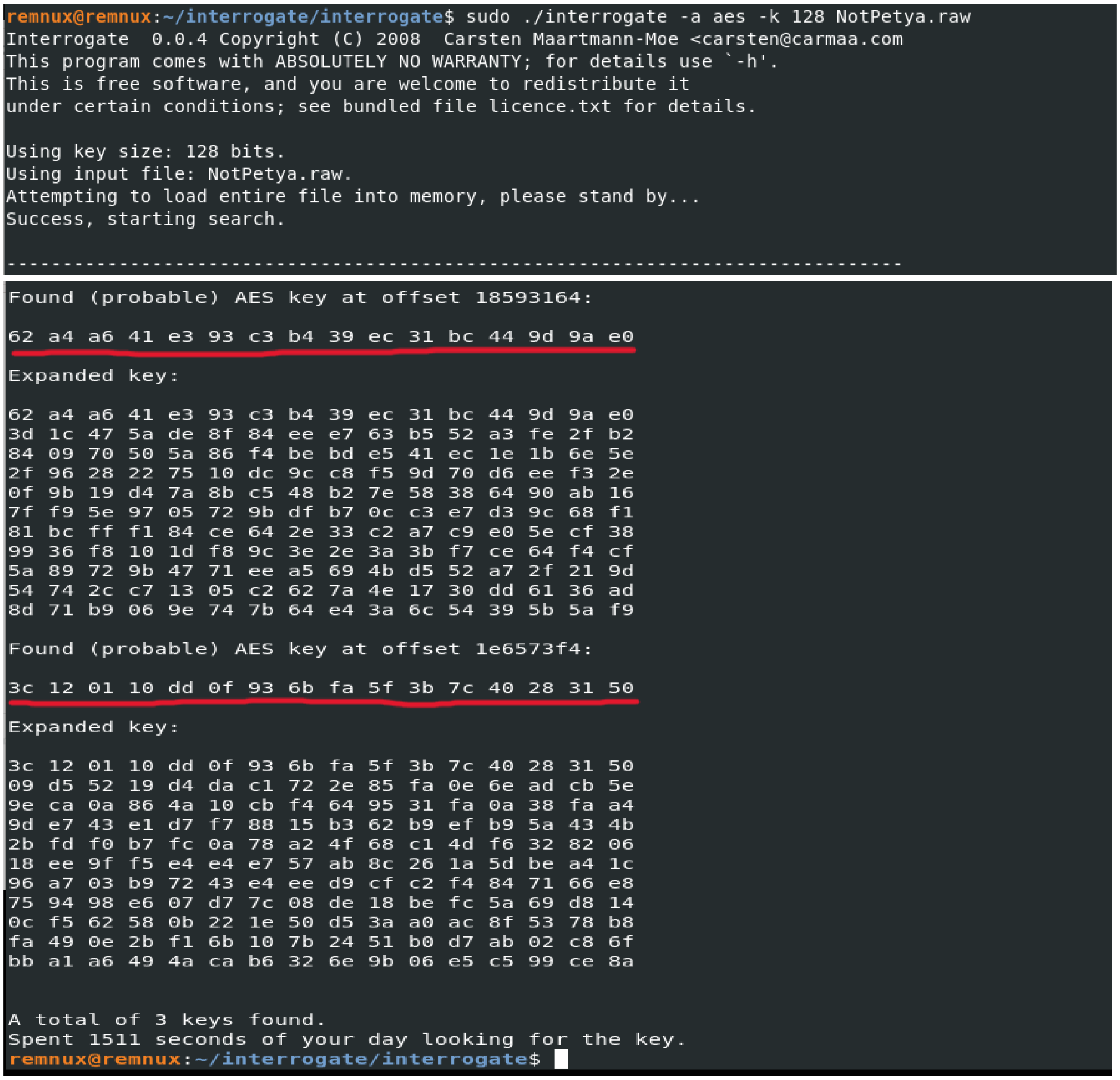
or

interrogate -a aes -k 128 NotPetya.raw



```
PS C:\Users\paris\Desktop\volatility\findaes-1.2\findaes-1.2> .\findaes.exe NotPetya.raw
Searching NotPetya.raw
Found AES-128 key schedule at offset 0x366d164:
62 a4 a6 41 e3 93 c3 b4 39 ec 31 bc 44 9d 9a e0
Found AES-128 key schedule at offset 0x18593164:
62 a4 a6 41 e3 93 c3 b4 39 ec 31 bc 44 9d 9a e0
Found AES-128 key schedule at offset 0x1e6573f4:
3c 12 01 10 dd 0f 93 6b fa 5f 3b 7c 40 28 31 50
Found AES-128 key schedule at offset 0x43aad3f4:
3c 12 01 10 dd 0f 93 6b fa 5f 3b 7c 40 28 31 50
```

Figure 4.18: NotPetya findaes Output



```
remnux@remnux:~/interrogate/interrogate$ sudo ./interrogate -a aes -k 128 NotPetya.raw
Interrogate  0.0.4 Copyright (C) 2008  Carsten Maartmann-Moe <carsten@carmaa.com
This program comes with ABSOLUTELY NO WARRANTY; for details use `-h'.
This is free software, and you are welcome to redistribute it
under certain conditions; see bundled file licence.txt for details.

Using key size: 128 bits.
Using input file: NotPetya.raw.
Attempting to load entire file into memory, please stand by...
Success, starting search.

-------------------------------------------------------------------------------
```

```
Found (probable) AES key at offset 18593164:

62 a4 a6 41 e3 93 c3 b4 39 ec 31 bc 44 9d 9a e0

Expanded key:

62 a4 a6 41 e3 93 c3 b4 39 ec 31 bc 44 9d 9a e0
3d 1c 47 5a de 8f 84 ee e7 63 b5 52 a3 fe 2f b2
84 09 70 50 5a 86 f4 be bd e5 41 ec 1e 1b 6e 5e
2f 96 28 22 75 10 dc 9c c8 f5 9d 70 d6 ee f3 2e
0f 9b 19 d4 7a 8b c5 48 b2 7e 58 38 64 90 ab 16
7f f9 5e 97 05 72 9b df b7 0c c3 e7 d3 9c 68 f1
81 bc ff f1 84 ce 64 2e 33 c2 a7 c9 e0 5e cf 38
99 36 f8 10 1d f8 9c 3e 2e 3a 3b f7 ce 64 f4 cf
5a 89 72 9b 47 71 ee a5 69 4b d5 52 a7 2f 21 9d
54 74 2c c7 13 05 c2 62 7a 4e 17 30 dd 61 36 ad
8d 71 b9 06 9e 74 7b 64 e4 3a 6c 54 39 5b 5a f9

Found (probable) AES key at offset 1e6573f4:

3c 12 01 10 dd 0f 93 6b fa 5f 3b 7c 40 28 31 50

Expanded key:

3c 12 01 10 dd 0f 93 6b fa 5f 3b 7c 40 28 31 50
09 d5 52 19 d4 da c1 72 2e 85 fa 0e 6e ad cb 5e
9e ca 0a 86 4a 10 cb f4 64 95 31 fa 0a 38 fa a4
9d e7 43 e1 d7 f7 88 15 b3 62 b9 ef b9 5a 43 4b
2b fd f0 b7 fc 0a 78 a2 4f 68 c1 4d f6 32 82 06
18 ee 9f f5 e4 e4 e7 57 ab 8c 26 1a 5d be a4 1c
96 a7 03 b9 72 43 e4 ee d9 cf c2 f4 84 71 66 e8
75 94 98 e6 07 d7 7c 08 de 18 be fc 5a 69 d8 14
0c f5 62 58 0b 22 1e 50 d5 3a a0 ac 8f 53 78 b8
fa 49 0e 2b f1 6b 10 7b 24 51 b0 d7 ab 02 c8 6f
bb a1 a6 49 4a ca b6 32 6e 9b 06 e5 c5 99 ce 8a


A total of 3 keys found.
Spent 1511 seconds of your day looking for the key.
remnux@remnux:~/interrogate/interrogate$ 
```

Figure 4.19: NotPetya Interrogate Output

4.8 Conclusions

This chapter details the experimental processes and commands executed to test the hypothesis that live memory forensics techniques can mitigate ransomware attacks.

During the execution of the ransomware samples, no external effects were observed, and no assets outside the experimental environment were affected, demonstrating the quality of the implemented experimental setup.

For the selected ransomware samples, no network traffic was detected, indicating that these samples were self-contained and did not require network access to function. However, network traffic was generated when executing some of the rejected ransomware samples, such as 'Locky', which attempted to contact its command and control (C&C) server to download the necessary encryption modules [49].

Some challenges were encountered in determining which ransomware samples to include in the investigation. For instance, no keys were captured during the execution of 'WannaCry', 'Gpcode', and 'Thanos' ransomware samples. Consequently, these samples were excluded from the study, and the focus was shifted to samples where key capture was successful. Nonetheless, further investigation into these samples was conducted using the Frida tool via API hooking, as discussed in the next chapter. Additionally, difficulties were faced in getting some ransomware samples to execute correctly leading to their exclusion from the study.

It is noteworthy that Jigsaw falls under category 2 in the ransomware classification, indicating that the encryption key is hardcoded. Consequently, after identifying and dumping the suspicious process from the victim machine, as detailed in paragraph 4.5, the static key can be found. This is why a different approach was employed in the other two experiments for extracting encryption keys. However, Jigsaw will also be examined in the next chapter using the Frida tool to validate my findings.

CHAPTER 5

Extracting Encryption Keys via CryptoAPI Hooking with Frida

5.1 Introduction

In this chapter, our goal is to investigate the effectiveness of employing live forensic methods, particularly utilizing Frida for CryptoAPI hooking, to mitigate ransomware attacks. We evaluate the attainment of our objectives and determine if our primary aim has been achieved. The chapter is organized into separate sections, each dedicated to analyzing different ransomware samples tested. Within these sections, we discuss and analyze the results of four conducted experiments.

5.2 Experiment design

For the experimentation phase, ransomware samples were tested within a VirtualBox virtual machine running Windows 10. These samples were sourced from reputable repositories such as https://github.com/ytisf/theZoo and https://bazaar.abuse.ch/ in April 2024. Initially, the ransomware specimens were in binary format and were extracted from encrypted ZIP files before use, often requiring manual addition of file extensions prior to execution.

To ensure safe testing, several precautions were taken. The virtual machine’s network adapter was set to host-only mode, shared folders between the guest and host were removed, and on the host side, data was backed up externally, with internet connectivity severed to prevent ransomware escape. Various test folders were strategically placed across the file system, including Desktop, Documents, Pictures, Program Files, Program Files (x86), and Windows. Additionally, a folder was introduced into the Recycle Bin to assess if the ransomware scanned this location. These test folders contained diverse file formats—rich-text, text, PDF, and image files—each having a non-zero size.

At an abstract level, the experiments followed a structured approach: executing a ransomware sample in a controlled virtual environment, intercepting API calls using Frida for CryptoAPI hooking, and analyzing the intercepted data to identify encryption keys, particularly the

AES key used during the symmetric encryption phase. Modern crypto ransomware often employs a hybrid encryption approach, combining symmetric and asymmetric encryption. While the public key for asymmetric encryption is delivered with the ransomware, the private key remains in the hands of the attacker. Since the private key is not present on the infected machine, this research focused on detecting the key utilized during the symmetric encryption phase, particularly the AES key.

5.3 Ransomware sample selection

The following three ransomware examples were selected for further analysis due to the previous lack of results from memory forensics. Additionally, the Jigsaw ransomware sample was analyzed to validate prior findings. All of which were examples of the HCR type of ransomware strains using AES for the symmetric portion of the encryption.

- Thanos: Thanos ransomware, named after the Marvel supervillain, is a malicious program developed by Moises Luis Zagala Gonzalez, a Venezuelan-French cardiologist. It emerged around February 2020, coded in C#. Thanos encrypts victim files and demands payment, usually in cryptocurrency like Bitcoin. It’s highly sophisticated, bypassing antivirus by rebooting the system in safe mode. The ransomware offers customization options for attackers, including message modification and self-deletion after attack.

- Gpcode: Gpcode ransomware spreads mainly through email attachments or by tricking users into downloading it disguised as a legitimate software update. Once activated on a victim’s computer, it encrypts files using robust encryption methods like RSA-1024 and AES-256, making decryption without the key nearly impossible. Originating around 2005, Gpcode gained notoriety as one of the earliest ransomware variants, earning the nickname “$20 ransomware.” Despite its age, Gpcode remains active today. However, its creators are notorious for not providing decryption keys even after receiving ransom payments, and attempts to contact them have been futile.

- WannaCry: The WannaCry ransomware attack, which occurred in May 2017, was a global cyberattack unleashed by the WannaCry ransomware cryptoworm. It specifically targeted computers running on the Microsoft Windows operating system, encrypting their data and demanding ransom payments in Bitcoin. The attack spread rapidly due to the use of EternalBlue, a vulnerability initially developed by the United States National Security Agency (NSA) for Windows systems. EternalBlue had been stolen and leaked by a group known as The Shadow Brokers a month prior to the attack. Despite Microsoft having released patches to fix this vulnerability earlier, many organizations had not applied them, either due to operational demands, concerns about compatibility issues with existing software, shortage of staff or time for installation, or other reasons. This failure to apply patches left systems vulnerable to attack, highlighting the critical importance of timely cybersecurity measures.

5.4 Tools

The following tools were used during execution of the experiments:

- Frida: Frida is a dynamic instrumentation toolkit renowned for its capability to intercept and manipulate function calls in real-time, particularly within the Windows environment where it can hook into the CryptoAPI. By injecting custom JavaScript scripts into running processes, Frida enables researchers to intercept CryptoAPI function calls, such as BcryptOpenAlgorithmProvider, BcryptSetProperty, BcryptGenerateSymmetricKey, and BcryptEncrypt, allowing for the monitoring and modification of cryptographic operations. This powerful framework is widely utilized in security research, malware analysis, and penetration testing to analyze and understand the behavior of ransomware and other cryptographic applications. Frida's flexibility and ease of use make it an indispensable tool for dynamic analysis and reverse engineering tasks focused on CryptoAPI-related activities.

- Visual Studio Code: VS Code provides a streamlined environment for creating JavaScript scripts to intercept CryptoAPI calls with Frida. With its user-friendly interface and powerful features like syntax highlighting and debugging support, developers can efficiently craft, test, and debug Frida scripts within VS Code. This integration enhances the development experience, allowing for seamless script creation and validation in real-time.

5.5 Experiment 1

A safe, isolated virtual environment was created and Jigsaw ransomware sample described in Table 5.1 was executed within it.Following this, Frida was deployed to intercept CryptoAPI calls made by the ransomware. Specifically, the focus was on hooking into CryptoAPI functions to trace the generation and usage of symmetric encryption keys. Through this method, the goal was to monitor the ransomware's encryption process and identify the keys involved, providing insights into its encryption mechanisms and aiding in potential mitigation strategies.

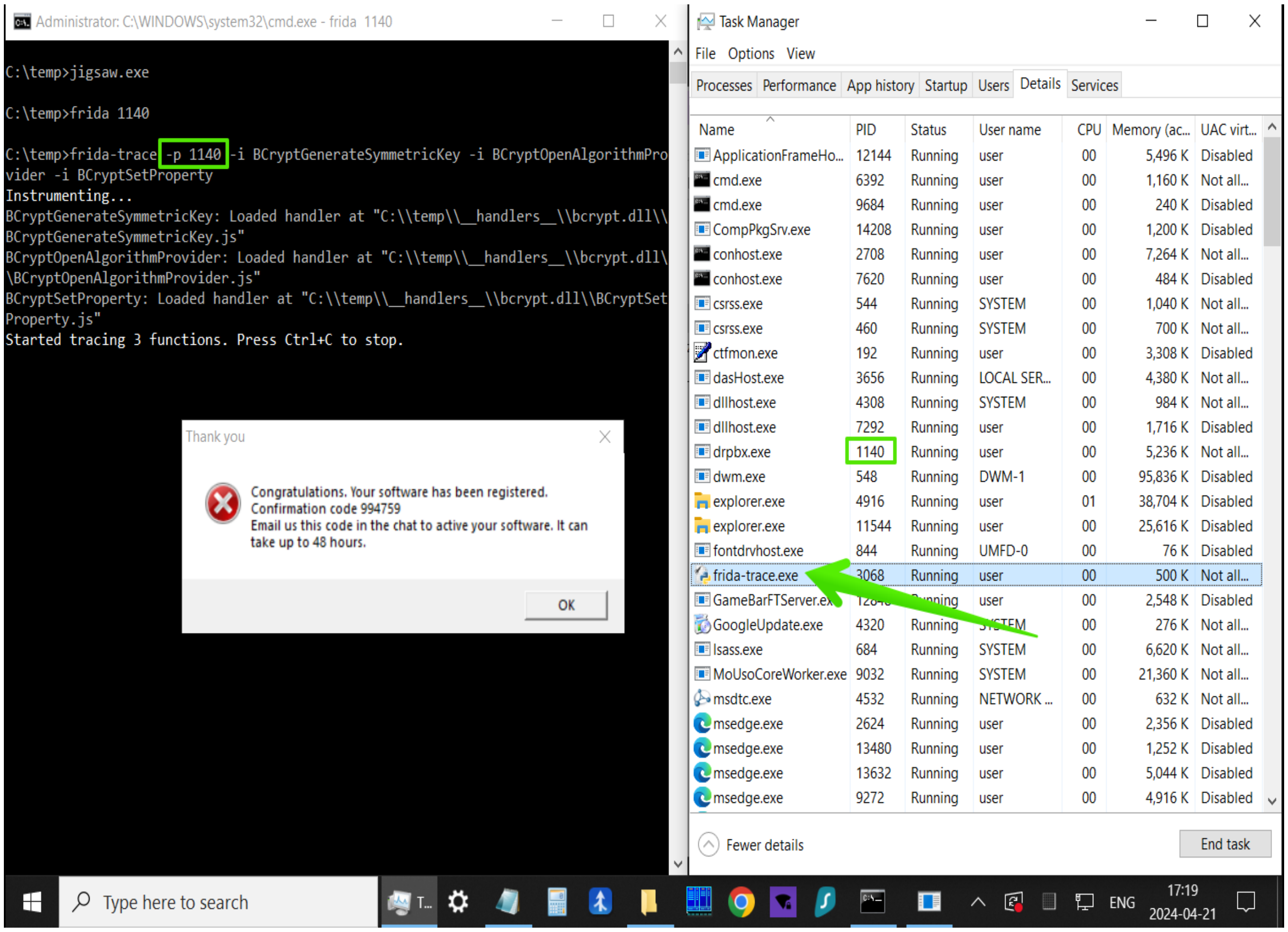
Name	Jigsaw
SHA 256	86a391fe7a237f4f17846c53d71e45820411d1a9a6e0c16f22a11ebc491ff9ff
URL	https://github.com/ytisf/theZoo/blob/master/malware/Binaries/Ransomware.Jigsaw/Ransomware.Jigsaw.zip

Table 5.1: Jigsaw Sample Details

Below is the suspicious process associated with the Jigsaw ransomware running. The Frida tracing command executed is as follows:

```
frida-trace -p <process_id_of_drpbx.exe> -i BCryptGenerateSymmetricKey -i
BCryptOpenAlgorithmProvider -i BCryptSetProperty
```

The Jigsaw sample employs the AesCryptoServiceProvider .NET API, which leverages the Cryptography API: Next Generation (CNG) framework provided by Windows. This cryptographic framework is implemented within the bcrypt.dll library. To configure the encryption process, the ransomware utilizes various functions from the CNG API. Specifically, the encryption algorithm, along with its associated parameters and encryption key, is established through the invocation of functions such as BcryptOpenAlgorithmProvider, BcryptSetProperty (for setting the Initialization Vector and Chaining Mode), and BcryptGenerateSymmetricKey.


Administrator: C:\WINDOWS\system32\cmd.exe - frida 1140

```
C:\temp>jigsaw.exe

C:\temp>frida 1140

C:\temp>frida-trace -p 1140 -i BCryptGenerateSymmetricKey -i BCryptOpenAlgorithmPro
vider -i BCryptSetProperty
Instrumenting...
BCryptGenerateSymmetricKey: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\\
BCryptGenerateSymmetricKey.js"
BCryptOpenAlgorithmProvider: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\
\BCryptOpenAlgorithmProvider.js"
BCryptSetProperty: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\\BCryptSet
Property.js"
Started tracing 3 functions. Press Ctrl+C to stop.
```

Thank you

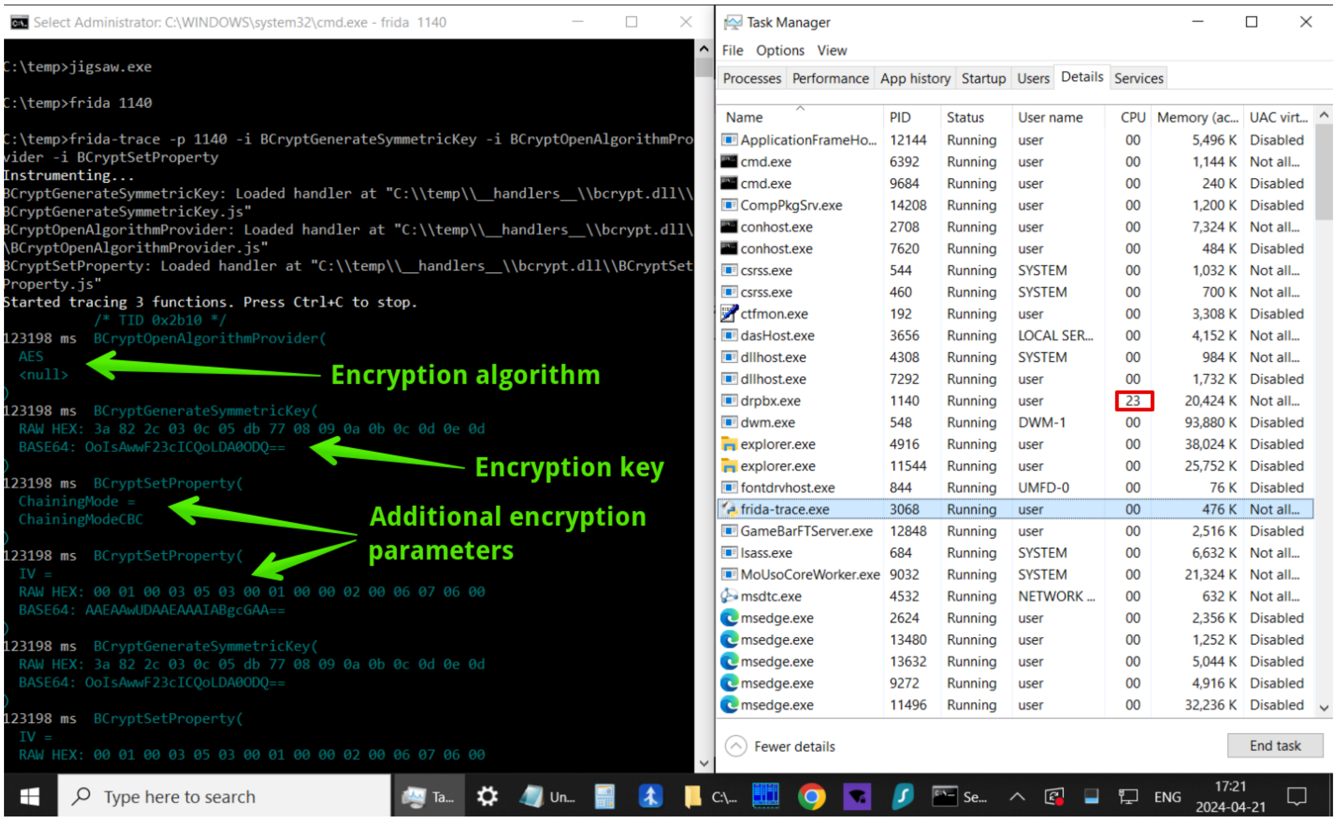
Congratulations. Your software has been registered.
Confirmation code 994759
Email us this code in the chat to active your software. It can take up to 48 hours.

Task Manager

File Options View
Processes Performance App history Startup Users Details Services

Name	PID	Status	User name	CPU	Memory (ac...	UAC virt...
ApplicationFrameHo...	12144	Running	user	00	5,496 K	Disabled
cmd.exe	6392	Running	user	00	1,160 K	Not all...
cmd.exe	9684	Running	user	00	240 K	Disabled
CompPkgSrv.exe	14208	Running	user	00	1,200 K	Disabled
conhost.exe	2708	Running	user	00	7,264 K	Not all...
conhost.exe	7620	Running	user	00	484 K	Disabled
csrss.exe	544	Running	SYSTEM	00	1,040 K	Not all...
csrss.exe	460	Running	SYSTEM	00	700 K	Not all...
ctfmon.exe	192	Running	user	00	3,308 K	Disabled
dasHost.exe	3656	Running	LOCAL SER...	00	4,380 K	Not all...
dllhost.exe	4308	Running	SYSTEM	00	984 K	Not all...
dllhost.exe	7292	Running	user	00	1,716 K	Disabled
drpbx.exe	1140	Running	user	00	5,236 K	Not all...
dwm.exe	548	Running	DWM-1	00	95,836 K	Disabled
explorer.exe	4916	Running	user	01	38,704 K	Disabled
explorer.exe	11544	Running	user	00	25,616 K	Disabled
fontdrvhost.exe	844	Running	UMFD-0	00	76 K	Disabled
frida-trace.exe	3068	Running	user	00	500 K	Not all...
GameBarFTServer.ex	1284[illegible]	Running	user	00	2,548 K	Disabled
GoogleUpdate.exe	4320	Running	SYSTEM	00	276 K	Not all...
lsass.exe	684	Running	SYSTEM	00	6,620 K	Not all...
MoUsoCoreWorker.exe	9032	Running	SYSTEM	00	21,360 K	Not all...
msdtc.exe	4532	Running	NETWORK ...	00	632 K	Not all...
msedge.exe	2624	Running	user	00	2,356 K	Disabled
msedge.exe	13480	Running	user	00	1,252 K	Disabled
msedge.exe	13632	Running	user	00	5,044 K	Disabled
msedge.exe	9272	Running	user	00	4,916 K	Disabled

Figure 5.1: Frida execution command


Select Administrator: C:\WINDOWS\system32\cmd.exe - frida 1140
Encryption algorithm
Encryption key
Additional encryption parameters
Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
End task
Type here to search
ENG
17:21
2024-04-21

```
C:\temp>jigsaw.exe

C:\temp>frida 1140

C:\temp>frida-trace -p 1140 -i BCryptGenerateSymmetricKey -i BCryptOpenAlgorithmPro
vider -i BCryptSetProperty
Instrumenting...
BCryptGenerateSymmetricKey: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\\
BCryptGenerateSymmetricKey.js"
BCryptOpenAlgorithmProvider: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\
\BCryptOpenAlgorithmProvider.js"
BCryptSetProperty: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\\BCryptSet
Property.js"
Started tracing 3 functions. Press Ctrl+C to stop.
           /* TID 0x2b10 */
123198 ms  BCryptOpenAlgorithmProvider(
  AES
  <null>
)
123198 ms  BCryptGenerateSymmetricKey(
  RAW HEX: 3a 82 2c 03 0c 05 db 77 08 09 0a 0b 0c 0d 0e 0d
  BASE64: OoIsAwwF23cICQoLDA0ODQ==
)
123198 ms  BCryptSetProperty(
  ChainingMode =
  ChainingModeCBC
)
123198 ms  BCryptSetProperty(
  IV =
  RAW HEX: 00 01 00 03 05 03 00 01 00 00 02 00 06 07 06 00
  BASE64: AAEAAwUDAAEAAAIABgcGAA==
)
123198 ms  BCryptGenerateSymmetricKey(
  RAW HEX: 3a 82 2c 03 0c 05 db 77 08 09 0a 0b 0c 0d 0e 0d
  BASE64: OoIsAwwF23cICQoLDA0ODQ==
)
123198 ms  BCryptSetProperty(
  IV =
  RAW HEX: 00 01 00 03 05 03 00 01 00 00 02 00 06 07 06 00
```

Name	PID	Status	User name	CPU	Memory (ac...	UAC virt...
ApplicationFrameHo...	12144	Running	user	00	5,496 K	Disabled
cmd.exe	6392	Running	user	00	1,144 K	Not all...
cmd.exe	9684	Running	user	00	240 K	Disabled
CompPkgSrv.exe	14208	Running	user	00	1,200 K	Disabled
conhost.exe	2708	Running	user	00	7,324 K	Not all...
conhost.exe	7620	Running	user	00	484 K	Disabled
csrss.exe	544	Running	SYSTEM	00	1,032 K	Not all...
csrss.exe	460	Running	SYSTEM	00	700 K	Not all...
ctfmon.exe	192	Running	user	00	3,308 K	Disabled
dasHost.exe	3656	Running	LOCAL SER...	00	4,152 K	Not all...
dllhost.exe	4308	Running	SYSTEM	00	984 K	Not all...
dllhost.exe	7292	Running	user	00	1,732 K	Disabled
drpbx.exe	1140	Running	user	23	20,424 K	Not all...
dwm.exe	548	Running	DWM-1	00	93,880 K	Disabled
explorer.exe	4916	Running	user	00	38,024 K	Disabled
explorer.exe	11544	Running	user	00	25,752 K	Disabled
fontdrvhost.exe	844	Running	UMFD-0	00	76 K	Disabled
frida-trace.exe	3068	Running	user	00	476 K	Not all...
GameBarFTServer.exe	12848	Running	user	00	2,516 K	Disabled
lsass.exe	684	Running	SYSTEM	00	6,632 K	Not all...
MoUsoCoreWorker.exe	9032	Running	SYSTEM	00	21,324 K	Not all...
msdtc.exe	4532	Running	NETWORK ...	00	632 K	Not all...
msedge.exe	2624	Running	user	00	2,356 K	Disabled
msedge.exe	13480	Running	user	00	1,252 K	Disabled
msedge.exe	13632	Running	user	00	5,044 K	Disabled
msedge.exe	9272	Running	user	00	4,916 K	Disabled
msedge.exe	11496	Running	user	00	32,236 K	Disabled

Figure 5.2: Frida results

5.6 Experiment 2

A safe, isolated virtual environment was created and Thanos ransomware sample described in Table 5.2 was executed within it.Following this, Frida was deployed to intercept CryptoAPI calls made by the ransomware. Specifically, the focus was on hooking into CryptoAPI functions to trace the generation and usage of symmetric encryption keys. Through this method, the goal was to monitor the ransomware's encryption process and identify the keys involved, providing insights into its encryption mechanisms and aiding in potential mitigation strategies.

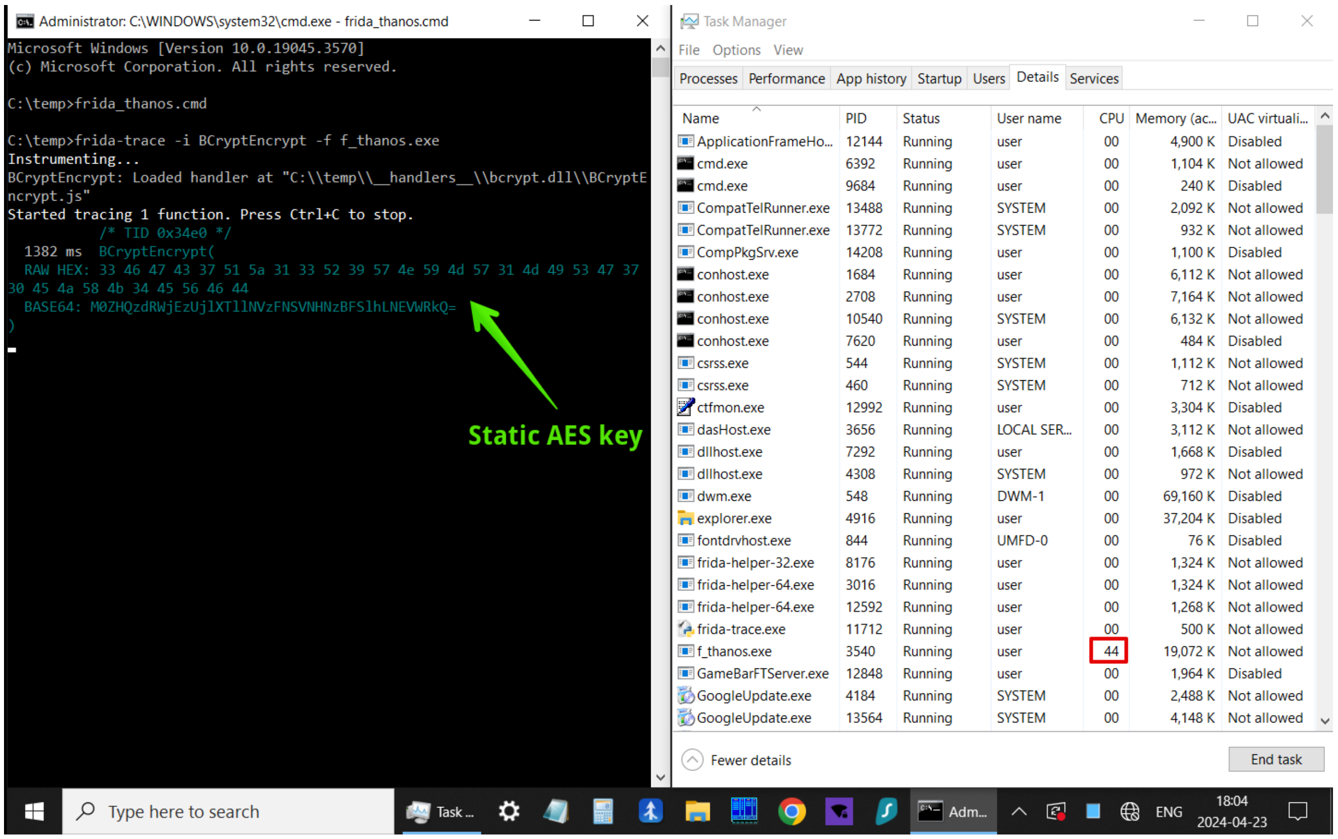
Name	Thanos
SHA 256	5d40615701c48a122e44f831e7c8643d07765629a83b15d090587f469c77693d
URL	https://github.com/ytisf/theZoo/tree/master/malware/Binaries/Ransomware.Thanos

Table 5.2: Thanos Sample Details

The Thanos ransomware employs a single key for encrypting all files. However, unlike Jigsaw ransomware, which relies on a hardcoded key, Thanos generates a random key for encryption [28].

The Frida tracing command executed is as follows:

frida-trace -i BCryptEncrypt -f f_thanos.exe


Administrator: C:\WINDOWS\system32\cmd.exe - frida_thanos.cmd
Static AES key
Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
End task
Type here to search
Task ...
Adm...
ENG
18:04
2024-04-23

```
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\temp>frida_thanos.cmd

C:\temp>frida-trace -i BCryptEncrypt -f f_thanos.exe
Instrumenting...
BCryptEncrypt: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\\BCryptE
ncrypt.js"
Started tracing 1 function. Press Ctrl+C to stop.
           /* TID 0x34e0 */
  1382 ms  BCryptEncrypt(
  RAW HEX: 33 46 47 43 37 51 5a 31 33 52 39 57 4e 59 4d 57 31 4d 49 53 47 37
30 45 4a 58 4b 34 45 56 46 44
  BASE64: M0ZHQzdRWjEzUjlXTllNVzFNSVNHNzBFSlhLNEVWRkQ=
)
```

Name	PID	Status	User name	CPU	Memory (ac...	UAC virtuali...
ApplicationFrameHo...	12144	Running	user	00	4,900 K	Disabled
cmd.exe	6392	Running	user	00	1,104 K	Not allowed
cmd.exe	9684	Running	user	00	240 K	Disabled
CompatTelRunner.exe	13488	Running	SYSTEM	00	2,092 K	Not allowed
CompatTelRunner.exe	13772	Running	SYSTEM	00	932 K	Not allowed
CompPkgSrv.exe	14208	Running	user	00	1,100 K	Disabled
conhost.exe	1684	Running	user	00	6,112 K	Not allowed
conhost.exe	2708	Running	user	00	7,164 K	Not allowed
conhost.exe	10540	Running	SYSTEM	00	6,132 K	Not allowed
conhost.exe	7620	Running	user	00	484 K	Disabled
csrss.exe	544	Running	SYSTEM	00	1,112 K	Not allowed
csrss.exe	460	Running	SYSTEM	00	712 K	Not allowed
ctfmon.exe	12992	Running	user	00	3,304 K	Disabled
dasHost.exe	3656	Running	LOCAL SER...	00	3,112 K	Not allowed
dllhost.exe	7292	Running	user	00	1,668 K	Disabled
dllhost.exe	4308	Running	SYSTEM	00	972 K	Not allowed
dwm.exe	548	Running	DWM-1	00	69,160 K	Disabled
explorer.exe	4916	Running	user	00	37,204 K	Disabled
fontdrvhost.exe	844	Running	UMFD-0	00	76 K	Disabled
frida-helper-32.exe	8176	Running	user	00	1,324 K	Not allowed
frida-helper-64.exe	3016	Running	user	00	1,324 K	Not allowed
frida-helper-64.exe	12592	Running	user	00	1,268 K	Not allowed
frida-trace.exe	11712	Running	user	00	500 K	Not allowed
f_thanos.exe	3540	Running	user	44	19,072 K	Not allowed
GameBarFTServer.exe	12848	Running	user	00	1,964 K	Disabled
GoogleUpdate.exe	4184	Running	SYSTEM	00	2,488 K	Not allowed
GoogleUpdate.exe	13564	Running	SYSTEM	00	4,148 K	Not allowed

Figure 5.3: Frida execution command and results

5.7 Experiment 3

A safe, isolated virtual environment was created and Gpcode ransomware sample described in Table 5.3 was executed within it. Following this, Frida was deployed to intercept CryptoAPI calls made by the ransomware. Specifically, the focus was on hooking into CryptoAPI functions to trace the generation and usage of symmetric encryption keys. Through this method, the goal was to monitor the ransomware's encryption process and identify the keys involved, providing insights into its encryption mechanisms and aiding in potential mitigation strategies.

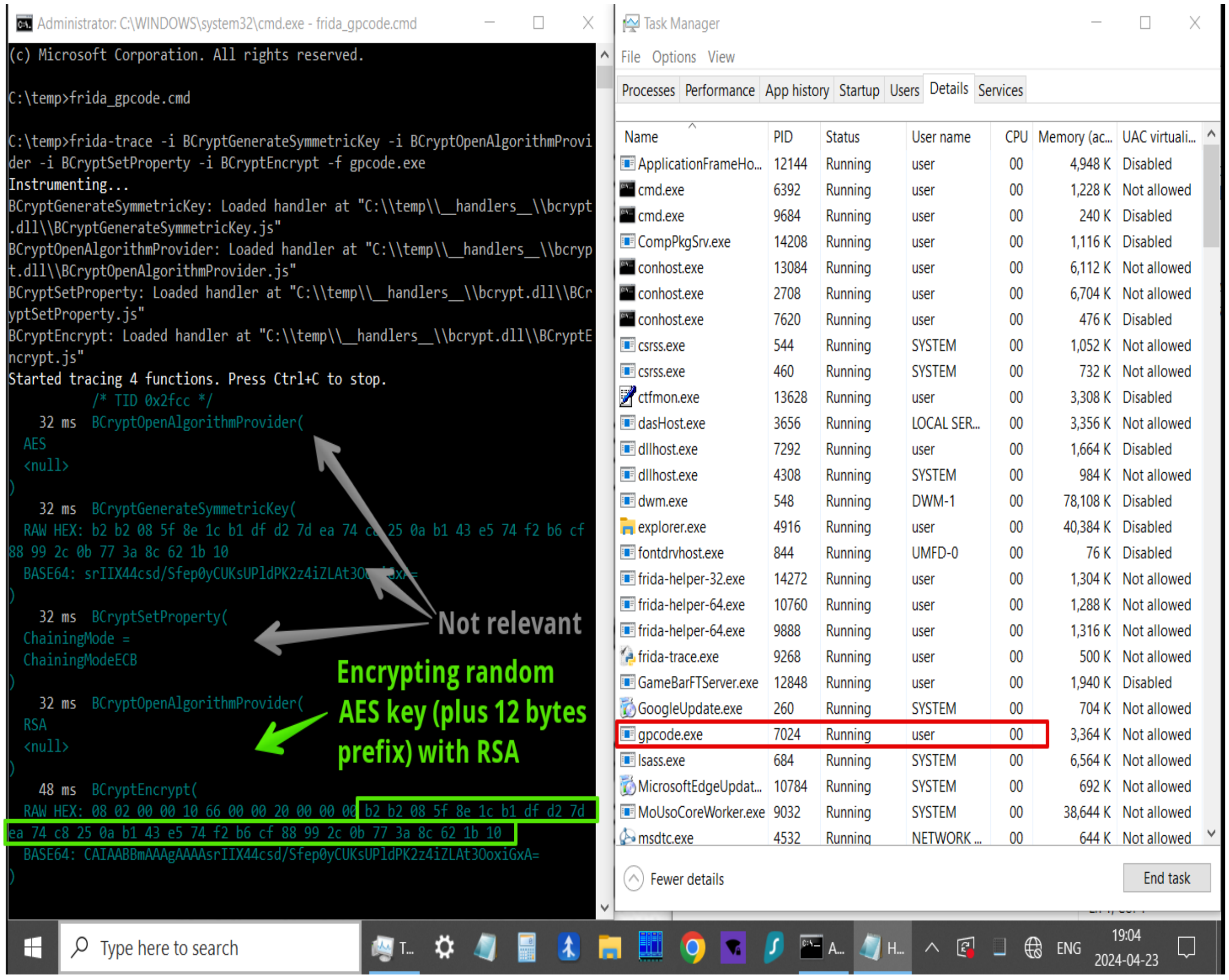
Name	Gpcode
SHA 256	e9ffda70e3ab71ee9d165abec8f2c7c52a139b71666f209d2eaf0c704569d3b1
URL	https://bazaar.abuse.ch/browse.php?search=signature%3Agpcode

Table 5.3: Gpcode Sample Details

Initially, the ransomware generates a 256-bit AES key, represented as a random 32-byte sequence. This key is then encrypted with a public RSA key pair using the legacy CryptEncrypt API, which internally invokes the modern BCryptEncrypt cryptography API [29]. Subsequently, the ransomware utilizes the same AES key and API calls for encrypting files. This process involves CryptEncrypt, which then triggers BCryptEncrypt, possibly with a zero Initialization Vector (IV).

The Frida tracing command executed is as follows:

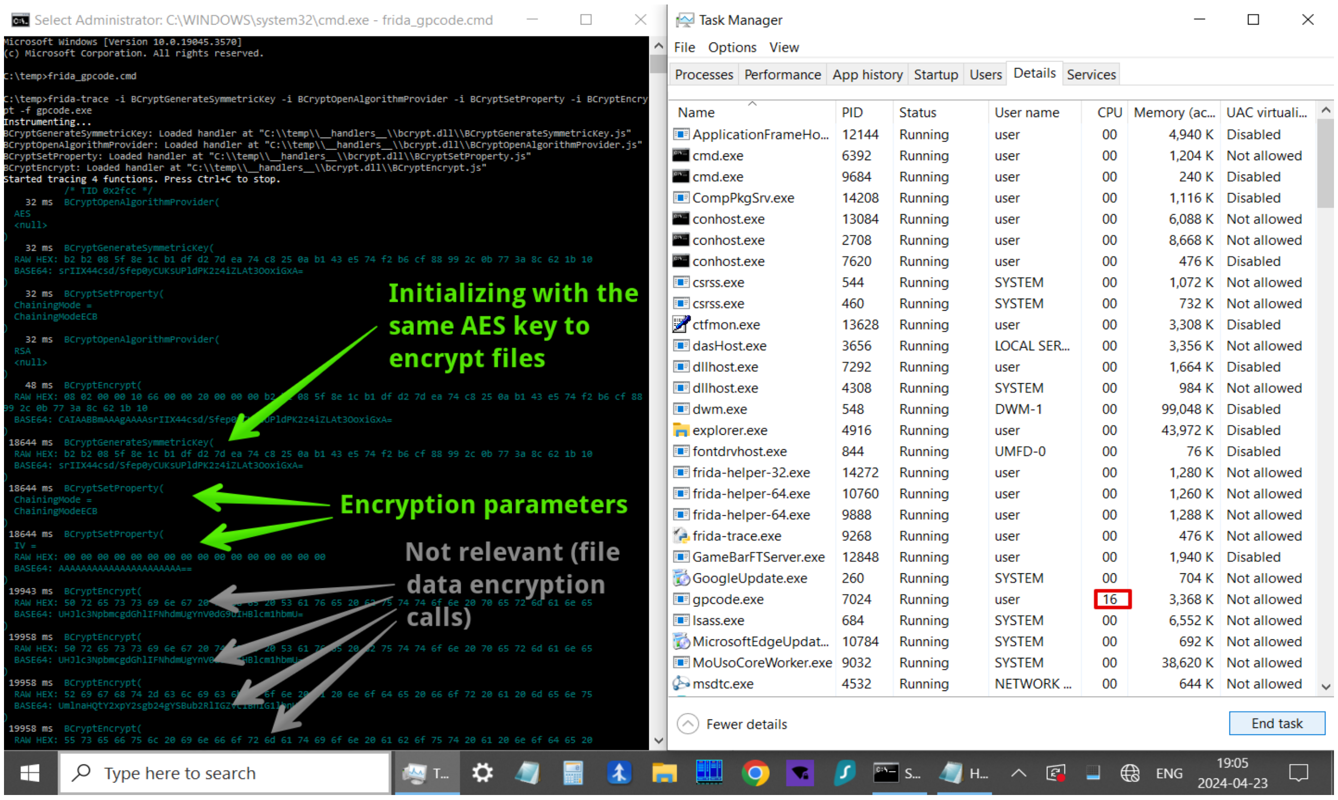
frida-trace -i BCryptGenerateSymmetricKey -i BCryptOpenAlgorithmProvider -i BCryptSetProperty -i BCryptEncrypt -f gpcode.exe


Administrator: C:\WINDOWS\system32\cmd.exe - frida_gpcode.cmd
Not relevant
Encrypting random AES key (plus 12 bytes prefix) with RSA
Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
End task
Type here to search
ENG
19:04
2024-04-23

```
(c) Microsoft Corporation. All rights reserved.

C:\temp>frida_gpcode.cmd

C:\temp>frida-trace -i BCryptGenerateSymmetricKey -i BCryptOpenAlgorithmProvi
der -i BCryptSetProperty -i BCryptEncrypt -f gpcode.exe
Instrumenting...
BCryptGenerateSymmetricKey: Loaded handler at "C:\\temp\\__handlers__\\bcrypt
.dll\\BCryptGenerateSymmetricKey.js"
BCryptOpenAlgorithmProvider: Loaded handler at "C:\\temp\\__handlers__\\bcryp
t.dll\\BCryptOpenAlgorithmProvider.js"
BCryptSetProperty: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\\BCr
yptSetProperty.js"
BCryptEncrypt: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\\BCryptE
ncrypt.js"
Started tracing 4 functions. Press Ctrl+C to stop.
           /* TID 0x2fcc */
    32 ms  BCryptOpenAlgorithmProvider(
  AES
  <null>
)
    32 ms  BCryptGenerateSymmetricKey(
  RAW HEX: b2 b2 08 5f 8e 1c b1 df d2 7d ea 74 c8 25 0a b1 43 e5 74 f2 b6 cf
88 99 2c 0b 77 3a 8c 62 1b 10
  BASE64: srIIX44csd/Sfep0yCUKsUPldPK2z4iZLAt3OoxiGxA=
)
    32 ms  BCryptSetProperty(
  ChainingMode =
  ChainingModeECB
)
    32 ms  BCryptOpenAlgorithmProvider(
  RSA
  <null>
)
    48 ms  BCryptEncrypt(
  RAW HEX: 08 02 00 00 10 66 00 00 20 00 00 00 b2 b2 08 5f 8e 1c b1 df d2 7d
ea 74 c8 25 0a b1 43 e5 74 f2 b6 cf 88 99 2c 0b 77 3a 8c 62 1b 10
  BASE64: CAIAABBmAAAgAAAAsrIIX44csd/Sfep0yCUKsUPldPK2z4iZLAt3OoxiGxA=
)
```

Name	PID	Status	User name	CPU	Memory (ac...	UAC virtuali...
ApplicationFrameHo...	12144	Running	user	00	4,948 K	Disabled
cmd.exe	6392	Running	user	00	1,228 K	Not allowed
cmd.exe	9684	Running	user	00	240 K	Disabled
CompPkgSrv.exe	14208	Running	user	00	1,116 K	Disabled
conhost.exe	13084	Running	user	00	6,112 K	Not allowed
conhost.exe	2708	Running	user	00	6,704 K	Not allowed
conhost.exe	7620	Running	user	00	476 K	Disabled
csrss.exe	544	Running	SYSTEM	00	1,052 K	Not allowed
csrss.exe	460	Running	SYSTEM	00	732 K	Not allowed
ctfmon.exe	13628	Running	user	00	3,308 K	Disabled
dasHost.exe	3656	Running	LOCAL SER...	00	3,356 K	Not allowed
dllhost.exe	7292	Running	user	00	1,664 K	Disabled
dllhost.exe	4308	Running	SYSTEM	00	984 K	Not allowed
dwm.exe	548	Running	DWM-1	00	78,108 K	Disabled
explorer.exe	4916	Running	user	00	40,384 K	Disabled
fontdrvhost.exe	844	Running	UMFD-0	00	76 K	Disabled
frida-helper-32.exe	14272	Running	user	00	1,304 K	Not allowed
frida-helper-64.exe	10760	Running	user	00	1,288 K	Not allowed
frida-helper-64.exe	9888	Running	user	00	1,316 K	Not allowed
frida-trace.exe	9268	Running	user	00	500 K	Not allowed
GameBarFTServer.exe	12848	Running	user	00	1,940 K	Disabled
GoogleUpdate.exe	260	Running	SYSTEM	00	704 K	Not allowed
gpcode.exe	7024	Running	user	00	3,364 K	Not allowed
lsass.exe	684	Running	SYSTEM	00	6,564 K	Not allowed
MicrosoftEdgeUpdat...	10784	Running	SYSTEM	00	692 K	Not allowed
MoUsoCoreWorker.exe	9032	Running	SYSTEM	00	38,644 K	Not allowed
msdtc.exe	4532	Running	NETWORK ...	00	644 K	Not allowed

Figure 5.4: Frida execution command and results


Select Administrator: C:\WINDOWS\system32\cmd.exe - frida_gpcode.cmd
Initializing with the same AES key to encrypt files
Encryption parameters
Not relevant (file data encryption calls)
Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
End task
Type here to search
ENG
19:05
2024-04-23

```
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\temp>frida_gpcode.cmd

C:\temp>frida-trace -i BCryptGenerateSymmetricKey -i BCryptOpenAlgorithmProvider -i BCryptSetProperty -i BCryptEncry
pt -f gpcode.exe
Instrumenting...
BCryptGenerateSymmetricKey: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\\BCryptGenerateSymmetricKey.js"
BCryptOpenAlgorithmProvider: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\\BCryptOpenAlgorithmProvider.js"
BCryptSetProperty: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\\BCryptSetProperty.js"
BCryptEncrypt: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\\BCryptEncrypt.js"
Started tracing 4 functions. Press Ctrl+C to stop.
           /* TID 0x2fcc */
    32 ms  BCryptOpenAlgorithmProvider(
  AES
  <null>
)
    32 ms  BCryptGenerateSymmetricKey(
  RAW HEX: b2 b2 08 5f 8e 1c b1 df d2 7d ea 74 c8 25 0a b1 43 e5 74 f2 b6 cf 88 99 2c 0b 77 3a 8c 62 1b 10
  BASE64: srIIX44csd/Sfep0yCUKsUPldPK2z4iZLAt3OoxiGxA=
)
    32 ms  BCryptSetProperty(
  ChainingMode =
  ChainingModeECB
)
    32 ms  BCryptOpenAlgorithmProvider(
  RSA
  <null>
)
    48 ms  BCryptEncrypt(
  RAW HEX: 08 02 00 00 10 66 00 00 20 00 00 00 b2 [illegible] 08 5f 8e 1c b1 df d2 7d ea 74 c8 25 0a b1 43 e5 74 f2 b6 cf 88
99 2c 0b 77 3a 8c 62 1b 10
  BASE64: CAIAABBmAAAgAAAAsrIIX44csd/Sfep0[illegible]UPldPK2z4iZLAt3OoxiGxA=
)
 18644 ms  BCryptGenerateSymmetricKey(
  RAW HEX: b2 b2 08 5f 8e 1c b1 df d2 7d ea 74 c8 25 0a b1 43 e5 74 f2 b6 cf 88 99 2c 0b 77 3a 8c 62 1b 10
  BASE64: srIIX44csd/Sfep0yCUKsUPldPK2z4iZLAt3OoxiGxA=
)
 18644 ms  BCryptSetProperty(
  ChainingMode =
  ChainingModeECB
)
 18644 ms  BCryptSetProperty(
  IV =
  RAW HEX: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  BASE64: AAAAAAAAAAAAAAAAAAAAAA==
)
 19943 ms  BCryptEncrypt(
  RAW HEX: 50 72 65 73 73 69 6e 67 20 [illegible] 65 20 53 61 76 65 20 6[illegible] 75 74 74 6f 6e 20 70 65 72 6d 61 6e 65
  BASE64: UHJlc3NpbmcgdGhlIFNhdmUgYnV0dG9[illegible]IHBlcm1hbmU=
)
 19958 ms  BCryptEncrypt(
  RAW HEX: 50 72 65 73 73 69 6e 67 20 74[illegible] 20 53 61 7[illegible] 20 [illegible]2 75 74 74 6f 6e 20 70 65 72 6d 61 6e 65
  BASE64: UHJlc3NpbmcgdGhlIFNhdmUgYnV0[illegible]HBlcm1hbmU[illegible]
)
 19958 ms  BCryptEncrypt(
  RAW HEX: 52 69 67 68 74 2d 63 6c 69 63 6[illegible] 6f 6e 20 [illegible]1 20 6e 6f 64 65 20 66 6f 72 20 61 20 6d 65 6e 75
  BASE64: UmlnaHQtY2xpY2sgb24gYSBub2RlIGZ[illegible]G1[illegible]
)
 19958 ms  BCryptEncrypt(
  RAW HEX: 55 73 65 66 75 6c 20 69 6e 66 6f 72 6d 61 74 69 6f 6e 20 61 62 6f 75 74 20 61 20 6e 6f 64 65 20
```

Name	PID	Status	User name	CPU	Memory (ac...	UAC virtuali...
ApplicationFrameHo...	12144	Running	user	00	4,940 K	Disabled
cmd.exe	6392	Running	user	00	1,204 K	Not allowed
cmd.exe	9684	Running	user	00	240 K	Disabled
CompPkgSrv.exe	14208	Running	user	00	1,116 K	Disabled
conhost.exe	13084	Running	user	00	6,088 K	Not allowed
conhost.exe	2708	Running	user	00	8,668 K	Not allowed
conhost.exe	7620	Running	user	00	476 K	Disabled
csrss.exe	544	Running	SYSTEM	00	1,072 K	Not allowed
csrss.exe	460	Running	SYSTEM	00	732 K	Not allowed
ctfmon.exe	13628	Running	user	00	3,308 K	Disabled
dasHost.exe	3656	Running	LOCAL SER...	00	3,356 K	Not allowed
dllhost.exe	7292	Running	user	00	1,664 K	Disabled
dllhost.exe	4308	Running	SYSTEM	00	984 K	Not allowed
dwm.exe	548	Running	DWM-1	00	99,048 K	Disabled
explorer.exe	4916	Running	user	00	43,972 K	Disabled
fontdrvhost.exe	844	Running	UMFD-0	00	76 K	Disabled
frida-helper-32.exe	14272	Running	user	00	1,280 K	Not allowed
frida-helper-64.exe	10760	Running	user	00	1,260 K	Not allowed
frida-helper-64.exe	9888	Running	user	00	1,288 K	Not allowed
frida-trace.exe	9268	Running	user	00	476 K	Not allowed
GameBarFTServer.exe	12848	Running	user	00	1,940 K	Disabled
GoogleUpdate.exe	260	Running	SYSTEM	00	704 K	Not allowed
gpcode.exe	7024	Running	user	$\boxed{16}$	3,368 K	Not allowed
lsass.exe	684	Running	SYSTEM	00	6,552 K	Not allowed
MicrosoftEdgeUpdat...	10784	Running	SYSTEM	00	692 K	Not allowed
MoUsoCoreWorker.exe	9032	Running	SYSTEM	00	38,620 K	Not allowed
msdtc.exe	4532	Running	NETWORK ...	00	644 K	Not allowed

Figure 5.5: Frida results

5.8 Experiment 4

A safe, isolated virtual environment was created and WannaCry ransomware sample described in Table 5.4 was executed within it.Following this, Frida was deployed to intercept CryptoAPI calls made by the ransomware. Specifically, the focus was on hooking into CryptoAPI functions to trace the generation and usage of symmetric encryption keys. Through this method, the goal was to monitor the ransomware's encryption process and identify the keys involved, providing insights into its encryption mechanisms and aiding in potential mitigation strategies.

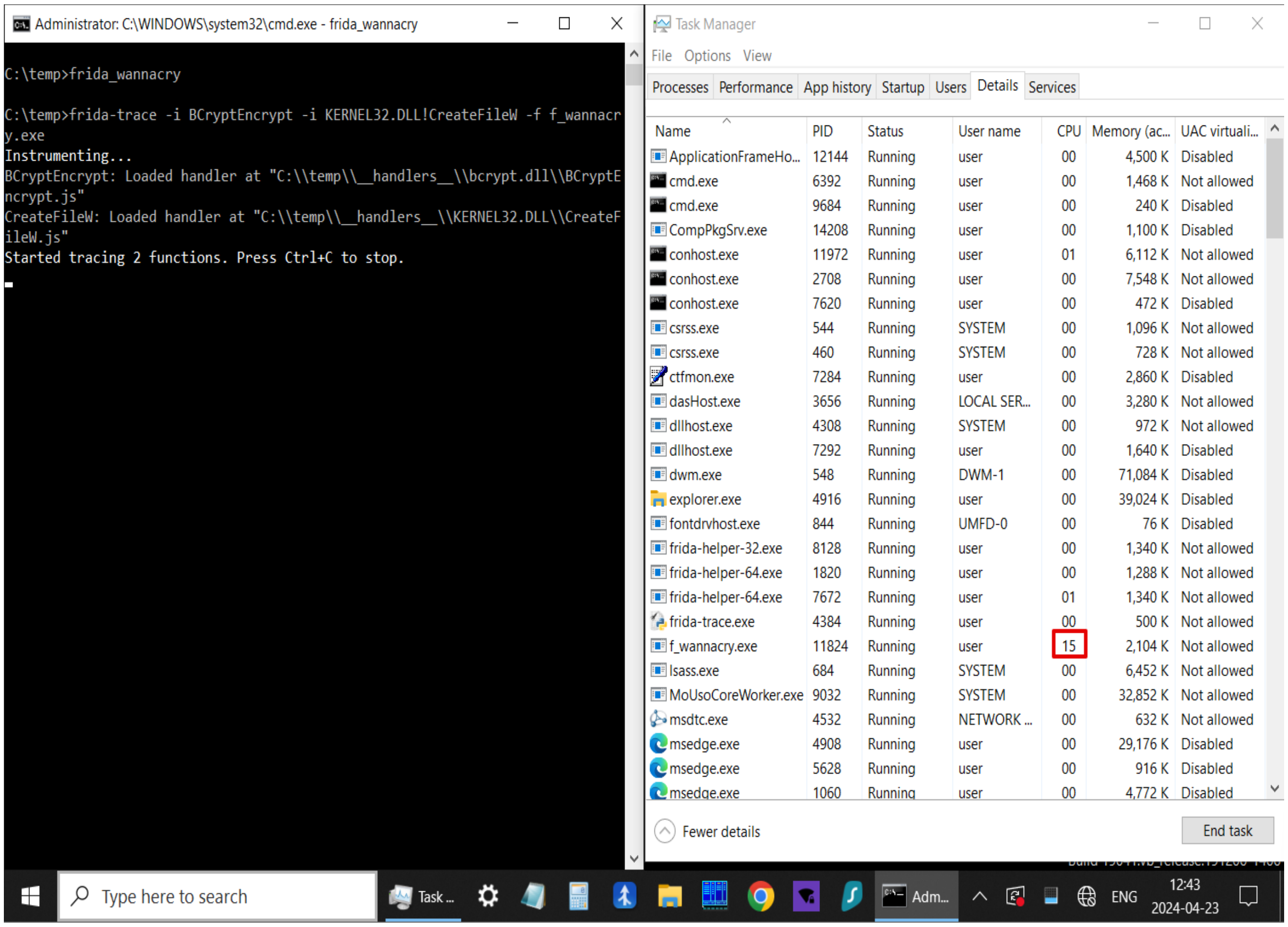
Name	WannaCry
SHA 256	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
URL	https://github.com/ytisf/theZoo/tree/master/malware/Binaries/Ransomware.WannaCry

Table 5.4: WannaCry Sample Details

WannaCry ransomware employs a unique approach by generating a private RSA-2048 key pair for each infected machine [45]. These keys are stored locally with an '.eky' extension, such as '00000000.eky', following encryption with an embedded RSA public key. The generated RSA key pair is then utilized to encrypt individual AES-128 keys, which in turn are assigned to each encrypted file [30]. In the results below, we observe the distinct AES keys for each file, which have not yet undergone encryption. Additionally, our analysis reveals the presence of the private key created for this infected machine with additional metadata or information beyond just the raw RSA-2048 keys.

The Frida tracing command executed is as follows:
frida-trace -i BCryptEncrypt -i KERNEL32.DLL!CreateFileW -f f_wannacry.exe

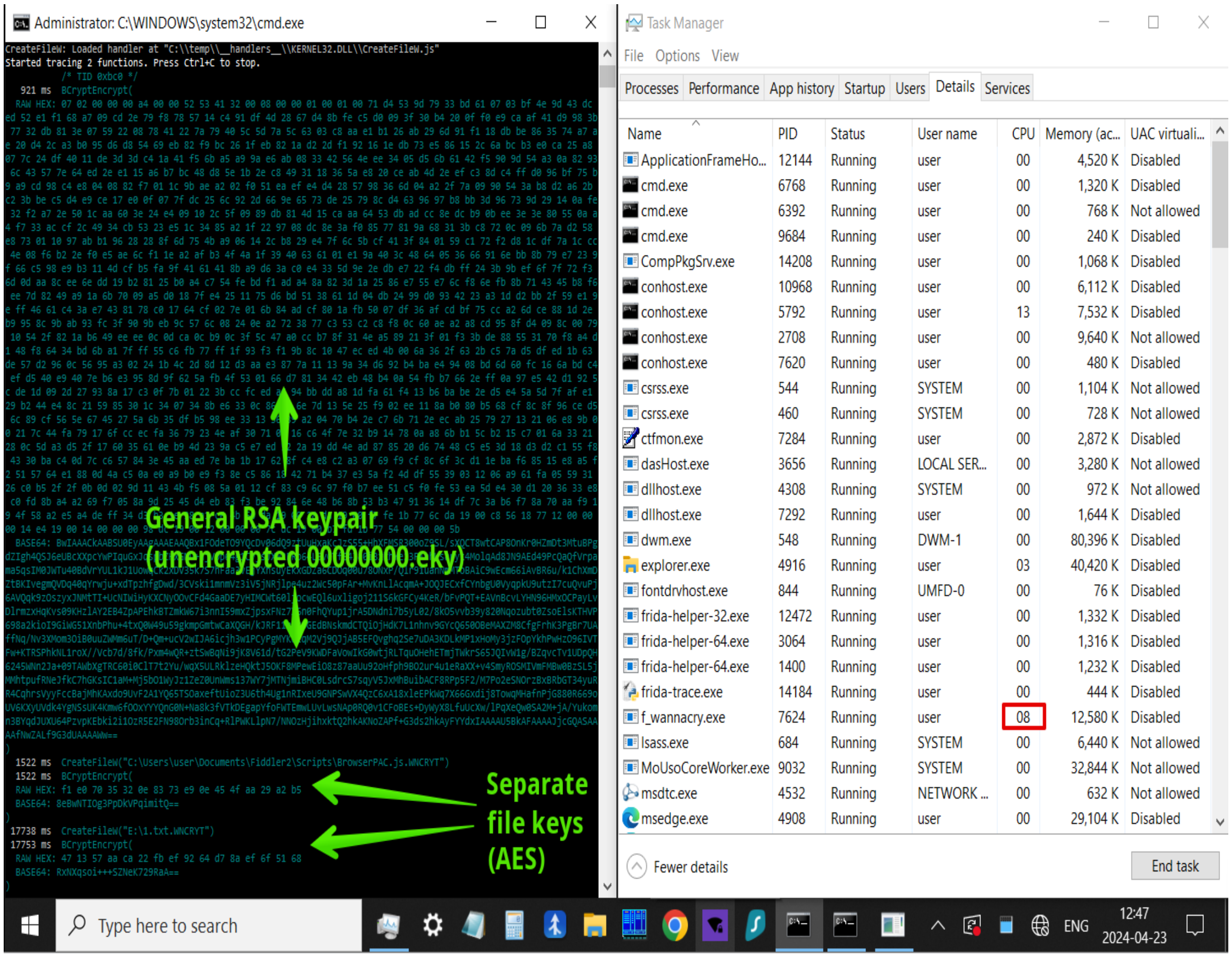
We also use CreateFileW handler to show the filename for which AES key is generated.


Administrator: C:\WINDOWS\system32\cmd.exe - frida_wannacry
Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
End task
Type here to search
Task ...
Adm...
ENG
12:43
2024-04-23

```
C:\temp>frida_wannacry

C:\temp>frida-trace -i BCryptEncrypt -i KERNEL32.DLL!CreateFileW -f f_wannacr
y.exe
Instrumenting...
BCryptEncrypt: Loaded handler at "C:\\temp\\__handlers__\\bcrypt.dll\\BCryptE
ncrypt.js"
CreateFileW: Loaded handler at "C:\\temp\\__handlers__\\KERNEL32.DLL\\CreateF
ileW.js"
Started tracing 2 functions. Press Ctrl+C to stop.
```

Name	PID	Status	User name	CPU	Memory (ac...	UAC virtuali...
ApplicationFrameHo...	12144	Running	user	00	4,500 K	Disabled
cmd.exe	6392	Running	user	00	1,468 K	Not allowed
cmd.exe	9684	Running	user	00	240 K	Disabled
CompPkgSrv.exe	14208	Running	user	00	1,100 K	Disabled
conhost.exe	11972	Running	user	01	6,112 K	Not allowed
conhost.exe	2708	Running	user	00	7,548 K	Not allowed
conhost.exe	7620	Running	user	00	472 K	Disabled
csrss.exe	544	Running	SYSTEM	00	1,096 K	Not allowed
csrss.exe	460	Running	SYSTEM	00	728 K	Not allowed
ctfmon.exe	7284	Running	user	00	2,860 K	Disabled
dasHost.exe	3656	Running	LOCAL SER...	00	3,280 K	Not allowed
dllhost.exe	4308	Running	SYSTEM	00	972 K	Not allowed
dllhost.exe	7292	Running	user	00	1,640 K	Disabled
dwm.exe	548	Running	DWM-1	00	71,084 K	Disabled
explorer.exe	4916	Running	user	00	39,024 K	Disabled
fontdrvhost.exe	844	Running	UMFD-0	00	76 K	Disabled
frida-helper-32.exe	8128	Running	user	00	1,340 K	Not allowed
frida-helper-64.exe	1820	Running	user	00	1,288 K	Not allowed
frida-helper-64.exe	7672	Running	user	01	1,340 K	Not allowed
frida-trace.exe	4384	Running	user	00	500 K	Not allowed
f_wannacry.exe	11824	Running	user	15	2,104 K	Not allowed
lsass.exe	684	Running	SYSTEM	00	6,452 K	Not allowed
MoUsoCoreWorker.exe	9032	Running	SYSTEM	00	32,852 K	Not allowed
msdtc.exe	4532	Running	NETWORK ...	00	632 K	Not allowed
msedge.exe	4908	Running	user	00	29,176 K	Disabled
msedge.exe	5628	Running	user	00	916 K	Disabled
msedge.exe	1060	Running	user	00	4,772 K	Disabled

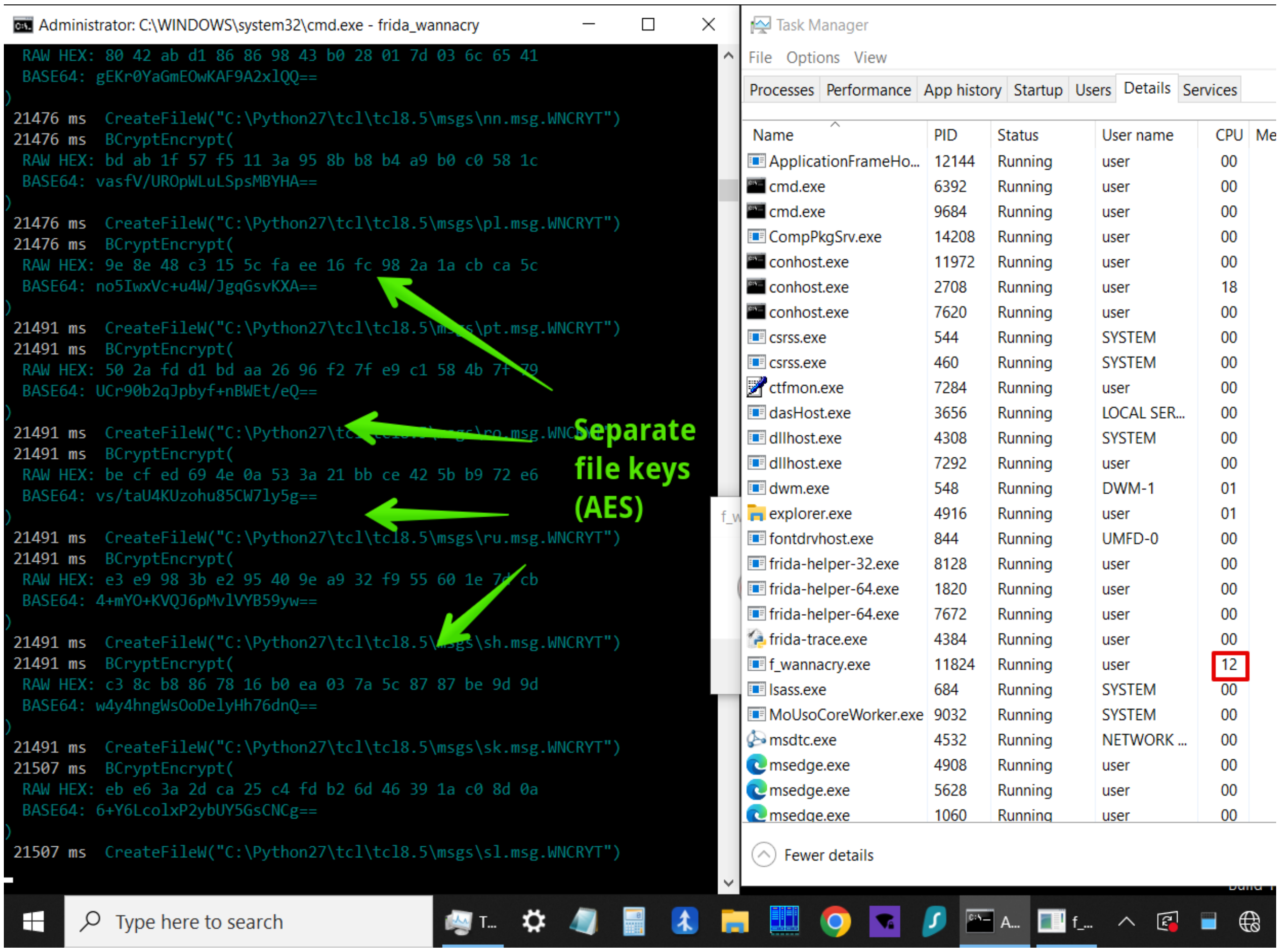
Figure 5.6: Frida execution command


Administrator: C:\WINDOWS\system32\cmd.exe
General RSA keypair
(unencrypted 00000000.eky)
Separate file keys (AES)
Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
End task
Type here to search
ENG
12:47
2024-04-23

```
CreateFileW: Loaded handler at "C:\\temp\\__handlers__\\KERNEL32.DLL\\CreateFileW.js"
Started tracing 2 functions. Press Ctrl+C to stop.
           /* TID 0xbc0 */
   921 ms  BCryptEncrypt(
  RAW HEX: 07 02 00 00 00 a4 00 00 52 53 41 32 00 08 00 00 01 00 01 00 71 d4 53 9d 79 33 bd 61 07 03 bf 4e 9d 43 dc
ed 52 e1 f1 68 a7 09 cd 2e 79 f8 78 57 14 c4 91 df 4d 28 67 d4 8b fe c5 d0 09 3f 30 b4 20 0f f0 e9 ca af 41 d9 98 3b
 77 32 db 81 3e 07 59 22 08 78 41 22 7a 79 40 5c 5d 7a 5c 63 03 c8 aa e1 b1 26 ab 29 6d 91 f1 18 db be 86 35 74 a7 a
e 20 d4 2c a3 b0 95 d6 d8 54 69 eb 82 f9 bc 26 1f eb 82 1a d2 2d f1 92 16 1e db 73 e5 86 15 2c 6a bc b3 e0 ca 25 a8
07 7c 24 df 40 11 de 3d 3d c4 1a 41 f5 6b a5 a9 9a e6 ab 08 33 42 56 4e ee 34 05 d5 6b 61 42 f5 90 9d 54 a3 0a 82 93
 6c 43 57 7e 64 ed 2e e1 15 a6 b7 bc 48 d8 5e 1b 2e c8 49 31 18 36 5a e8 20 ce ab 4d 2e ef c3 8d c4 ff d0 96 bf 75 b
9 a9 cd 98 c4 e8 04 08 82 f7 01 1c 9b ae a2 02 f0 51 ea ef e4 d4 28 57 98 36 6d 04 a2 2f 7a 09 90 54 3a b8 d2 a6 2b
c2 3b be c5 d4 e9 ce 17 e0 0f 07 7f dc 25 6c 92 2d 66 9e 65 73 de 25 79 8c d4 63 96 97 b8 bb 3d 96 73 9d 29 14 0a fe
 32 f2 a7 2e 50 1c aa 60 3e 24 e4 09 10 2c 5f 09 89 db 81 4d 15 ca aa 64 53 db ad cc 8e dc b9 0b ee 3e 3e 80 55 0a a
4 f7 33 ac cf 2c 49 34 cb 53 23 e5 1c 34 85 a2 1f 22 97 08 dc 8e 3a f0 85 77 81 9a 68 31 3b c8 72 0c 09 6b 7a d2 58
e8 73 01 10 97 ab b1 96 28 28 8f 6d 75 4b a9 06 14 2c b8 29 e4 7f 6c 5b cf 41 3f 84 01 59 c1 72 f2 d8 1c df 7a 1c cc
 4e 08 f6 b2 2e f0 e5 ae 6c f1 1e a2 af b3 4f 4a 1f 39 40 63 61 01 e1 9a 40 3c 48 64 05 36 66 91 6e bb 8b 79 e7 23 9
f 66 c5 98 e9 b3 11 4d cf b5 fa 9f 41 61 41 8b a9 d6 3a c0 e4 33 5d 9e 2e db e7 22 f4 db ff 24 3b 9b ef 6f 7f 72 f3
6d 0d aa 8c ee 6e dd 19 b2 81 25 b0 a4 c7 54 fe bd f1 ad a4 8a 82 3d 1a 25 86 e7 55 e7 6c f8 6e fb 8b 71 43 45 b8 f6
 ee 7d 82 49 a9 1a 6b 70 09 a5 d0 18 7f e4 25 11 75 d6 bd 51 38 61 1d 04 db 24 99 d0 93 42 23 a3 1d d2 bb 2f 59 e1 9
e ff 46 61 c4 3a e7 43 81 78 c0 17 64 cf 02 7e 01 6b 84 ad cf 80 1a fb 50 07 df 36 af cd bf 75 cc a2 6d ce 88 1d 2e
b9 95 8c 9b ab 93 fc 3f 90 9b eb 9c 57 6c 08 24 0e a2 72 38 77 c3 53 c2 c8 f8 0c 60 ae a2 a8 cd 95 8f d4 09 8c 00 79
 10 54 2f 82 1a b6 49 ee ee 0c 0d ca 0c b9 0c 3f 5c 47 a0 cc b7 8f 31 4e a5 89 21 3f 01 f3 3b de 88 55 31 70 f8 a4 d
1 48 f8 64 34 bd 6b a1 7f ff 55 c6 fb 77 ff 1f 93 f3 f1 9b 8c 10 47 ec ed 4b 00 6a 36 2f 63 2b c5 7a d5 df ed 1b 63
de 57 d2 96 0c 56 95 a3 02 24 1b 4c 2d 8d 12 d3 aa e3 87 7a 11 13 9a 34 d6 92 b4 ba e4 94 08 bd 6d 60 fc 16 6a bd c4
 ef d5 40 e9 40 7e b6 e3 95 8d 9f 62 5a fb 4f 53 01 66 d7 81 34 42 eb 48 b4 0a 54 fb b7 66 2e ff 0a 97 e5 42 d1 92 5
c de 1d 09 2d 27 93 8a 17 c3 0f 7b 01 22 3b cc fc ed a[illegible] 94 bb dd a8 1d fa 61 f4 13 b6 ba be 2e d5 e4 5a 5d 7f af e1
29 b2 44 e4 8c 21 59 85 30 1c 34 07 34 8b e6 33 0c 86 [illegible] 6e 7d 13 5e 25 f9 02 ee 11 8a b0 80 b5 68 cf 8c 8f 96 ce d5
 6c 89 cf 56 5e 67 45 27 5a 6b 35 df b5 98 ee 33 13 [illegible] a2 04 70 b4 2e c7 6b 71 2e ec ab 25 79 27 13 21 06 e8 9b 0
0 21 7c 44 fa 79 17 6f cc ec fa 36 79 23 4e af 30 71 0[illegible] 16 c6 4f 7e 32 b9 14 78 0a a8 6b b1 5c b2 15 c7 01 6a 33 21
28 0c 5d a3 d5 2f 17 60 35 61 0e b9 4d 23 9a c5 e7 ed [illegible]2 2a 19 dd 4e ad 87 85 20 d6 74 48 c5 e5 3d 18 d3 d2 c1 55 f8
 43 30 ba c4 0d 7c c6 57 84 3e 45 aa ed 7e ba 1b 17 62 8f c4 e8 c2 a3 07 69 f9 cf 8c 6f 3c d1 1e ba f6 85 15 e8 a5 f
2 51 57 64 e1 88 0d 4a c5 0a e0 a9 b0 e9 f3 8e c5 86 18 42 71 b4 37 e3 5a f2 4d df 55 39 03 12 06 a9 61 fa 05 59 31
26 c0 b5 2f 2f 0b 0d 02 9d 11 43 4b f5 08 5a 01 12 cf 83 c9 6c 97 f0 b7 ee 51 c5 f0 fe 53 ea 5d e4 30 d1 20 36 33 e8
 c0 fd 8b a4 a2 69 f7 05 8a 9d 25 45 d4 eb 83 f3 be 92 84 6e 48 b6 8b 53 b3 47 91 36 14 df 7c 3a b6 f7 8a 70 aa f9 1
9 4f 58 a2 e5 a4 de ff 34 d[illegible] fe 1b 77 6c da 19 00 c8 56 18 77 12 00 00
00 14 e4 19 00 14 00 00 00 98 [illegible] 77 54 00 00 00 5b
  BASE64: BwIAAACkAABSU0EyAA[illegible]AAAEAAQBx1FOdeTO9YQcDv06dQ9ztUuHxaKcJzS55+HbXFMSR300oZ9SL/sXQCT8wtCAP8OnKr0HZmDt3MtuBPg
dZIgh4QSJ6eUBcXXpcYwPIquGxJ[illegible]4MolqAd8JN9AEd49PcQaQfVrpa
ma5qsIM0JWTu40BdVrYUL1kJ1Uow[illegible]BAiC9wEcm66iAvBR6u/k1ChXmD
ZtBKIvegmQVDq40qYrwju+xdTpzhfgDwd/3CVski1mnmVz3iV5jNRjlpe4uz2Wc50pFAr+MvKnLlAcqmA+JOQJECxfCYnbgU0VyqpkU9utzI7cuQvuPj
6AVQqk9zOszyxJNMtTI+UcNIWiHyKXCNyOOvCFd4GaaDE7yHIMCWt60l[illegible]pcwEQl6uxligoj211S6kGFCy4KeR/bFvPQT+EAVnBcvLYHN96HMxOCPayLv
DlrmzxHqKvs09KHzlAY2EB4ZpAPEhkBTZmkW67i3nnI59mxZjpsxFNz7[illegible]n0FhQYup1jrA5DNdni7b5yL02/8kO5vvb39y820Nqozubt0ZsoElsKTHVP
698a2kioI9GiWG51XnbPhu+4txQ0W49u59gkmpGmtwCaXQGH/kJRF1[illegible]GEdBNskmdCTQiOjHdK7L1nhnv9GYcQ650OBeMAXZM8CfgFrhK3PgBr7UA
ffNq/Nv3XMom3OiB0uuZWMm6uT/D+Qm+ucV2wIJA6icjh3w1PCyPgMYK[illegible]qM2Vj9QJjAB5EFQvghq2Se7uDA3KDLkMP1xHoMy3jzFOpYkhPwHzO96IVT
Fw+KTRSPhkNL1roX//Vcb7d/8fk/Pxm4wQR+ztSwBqNi9jK8V61d/tG2PeV9KWDFaVowIkG0wtjRLTquOHehETmjTWkrS65JQIvW1g/BZqvcTv1UDpQH
6245WNn2Ja+09TAWbXgTRC60i0ClT7t2Yu/wqX5ULRklzeHQktJ5OKF8MPewEiO8z87aaUu92oHfph9BO2ur4u1eRaXX+v4SmyROSMIVmFMBw0BzSL5j
MMhtpufRNeJfkC7hGKsIC1aM+Mj5bO1WyJz1ZeZ0UnWms137WY7jMTNjmiBHC0LsdrcS7sqyV5JxMhBuibACF8RPp5F2/M7Po2eSNOrzBxBRbGT34yuR
R4CqhrsVyyFccBajMhKAxdo9UvF2A1YQ65TSOaxeftUioZ3U6th4Ug1nRIxeU9GNPSwVX4QzC6xA18xleEPkWq7X66Gxdij8TowqMHafnPjG880R669o
UV6KXyUVdk4YgNSsUK4Kmw6fOOxYYYQnG0N+Na8k3fVTkDEgapYfoFWTEmwLUvLwsNAp0RQ0v1CFoBEs+DyWyX8LfuUcXw/lPqXeQw0SA2M+jA/Yukom
n3BYqdJUXU64PzvpKEbki2i1OzR5E2FN98Orb3inCq+RlPWKLlpN7/NNOzHjihxktQ2hkAKNoZAPf+G3ds2hkAyFYYdxIAAAAU5BkAFAAAAJjcGQASAA
AAfNwZALf9G3dUAAAAWw==
)
  1522 ms  CreateFileW("C:\Users\user\Documents\Fiddler2\Scripts\BrowserPAC.js.WNCRYT")
  1522 ms  BCryptEncrypt(
  RAW HEX: f1 e0 70 35 32 0e 83 73 e9 0e 45 4f aa 29 a2 b5
  BASE64: 8eBwNTIOg3PpDkVPqimitQ==
)
 17738 ms  CreateFileW("E:\1.txt.WNCRYT")
 17753 ms  BCryptEncrypt(
  RAW HEX: 47 13 57 aa ca 22 fb ef 92 64 d7 8a ef 6f 51 68
  BASE64: RxNXqsoi+++SZNeK729RaA==
)
```

Name	PID	Status	User name	CPU	Memory (ac...	UAC virtuali...
ApplicationFrameHo...	12144	Running	user	00	4,520 K	Disabled
cmd.exe	6768	Running	user	00	1,320 K	Disabled
cmd.exe	6392	Running	user	00	768 K	Not allowed
cmd.exe	9684	Running	user	00	240 K	Disabled
CompPkgSrv.exe	14208	Running	user	00	1,068 K	Disabled
conhost.exe	10968	Running	user	00	6,112 K	Disabled
conhost.exe	5792	Running	user	13	7,532 K	Disabled
conhost.exe	2708	Running	user	00	9,640 K	Not allowed
conhost.exe	7620	Running	user	00	480 K	Disabled
csrss.exe	544	Running	SYSTEM	00	1,104 K	Not allowed
csrss.exe	460	Running	SYSTEM	00	728 K	Not allowed
ctfmon.exe	7284	Running	user	00	2,872 K	Disabled
dasHost.exe	3656	Running	LOCAL SER...	00	3,280 K	Not allowed
dllhost.exe	4308	Running	SYSTEM	00	972 K	Not allowed
dllhost.exe	7292	Running	user	00	1,644 K	Disabled
dwm.exe	548	Running	DWM-1	00	80,396 K	Disabled
explorer.exe	4916	Running	user	03	40,420 K	Disabled
fontdrvhost.exe	844	Running	UMFD-0	00	76 K	Disabled
frida-helper-32.exe	12472	Running	user	00	1,332 K	Disabled
frida-helper-64.exe	3064	Running	user	00	1,316 K	Disabled
frida-helper-64.exe	1400	Running	user	00	1,232 K	Disabled
frida-trace.exe	14184	Running	user	00	444 K	Disabled
f_wannacry.exe	7624	Running	user	08	12,580 K	Disabled
lsass.exe	684	Running	SYSTEM	00	6,440 K	Not allowed
MoUsoCoreWorker.exe	9032	Running	SYSTEM	00	32,844 K	Not allowed
msdtc.exe	4532	Running	NETWORK ...	00	632 K	Not allowed
msedge.exe	4908	Running	user	00	29,104 K	Disabled

Figure 5.7: Frida results


Administrator: C:\WINDOWS\system32\cmd.exe - frida_wannacry
Separate file keys (AES)
Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Fewer details
Type here to search

```
 RAW HEX: 80 42 ab d1 86 86 98 43 b0 28 01 7d 03 6c 65 41
 BASE64: gEKr0YaGmEOwKAF9A2x1QQ==
)
21476 ms  CreateFileW("C:\Python27\tcl\tcl8.5\msgs\nn.msg.WNCRYT")
21476 ms  BCryptEncrypt(
 RAW HEX: bd ab 1f 57 f5 11 3a 95 8b b8 b4 a9 b0 c0 58 1c
 BASE64: vasfV/UROpWLuLSpsMBYHA==
)
21476 ms  CreateFileW("C:\Python27\tcl\tcl8.5\msgs\pl.msg.WNCRYT")
21476 ms  BCryptEncrypt(
 RAW HEX: 9e 8e 48 c3 15 5c fa ee 16 fc 98 2a 1a cb ca 5c
 BASE64: no5IwxVc+u4W/JgqGsvKXA==
)
21491 ms  CreateFileW("C:\Python27\tcl\tcl8.5\msgs\pt.msg.WNCRYT")
21491 ms  BCryptEncrypt(
 RAW HEX: 50 2a fd d1 bd aa 26 96 f2 7f e9 c1 58 4b 7f 79
 BASE64: UCr90b2qJpbyf+nBWEt/eQ==
)
21491 ms  CreateFileW("C:\Python27\tcl\tcl8.5\msgs\ro.msg.WNCRYT")
21491 ms  BCryptEncrypt(
 RAW HEX: be cf ed 69 4e 0a 53 3a 21 bb ce 42 5b b9 72 e6
 BASE64: vs/taU4KUzohu85CW7ly5g==
)
21491 ms  CreateFileW("C:\Python27\tcl\tcl8.5\msgs\ru.msg.WNCRYT")
21491 ms  BCryptEncrypt(
 RAW HEX: e3 e9 98 3b e2 95 40 9e a9 32 f9 55 60 1e 7d cb
 BASE64: 4+mYO+KVQJ6pMvlVYB59yw==
)
21491 ms  CreateFileW("C:\Python27\tcl\tcl8.5\msgs\sh.msg.WNCRYT")
21491 ms  BCryptEncrypt(
 RAW HEX: c3 8c b8 86 78 16 b0 ea 03 7a 5c 87 87 be 9d 9d
 BASE64: w4y4hngWsOoDelyHh76dnQ==
)
21491 ms  CreateFileW("C:\Python27\tcl\tcl8.5\msgs\sk.msg.WNCRYT")
21507 ms  BCryptEncrypt(
 RAW HEX: eb e6 3a 2d ca 25 c4 fd b2 6d 46 39 1a c0 8d 0a
 BASE64: 6+Y6LcolxP2ybUY5GsCNCg==
)
21507 ms  CreateFileW("C:\Python27\tcl\tcl8.5\msgs\sl.msg.WNCRYT")
```

Name	PID	Status	User name	CPU	Me
ApplicationFrameHo...	12144	Running	user	00	
cmd.exe	6392	Running	user	00	
cmd.exe	9684	Running	user	00	
CompPkgSrv.exe	14208	Running	user	00	
conhost.exe	11972	Running	user	00	
conhost.exe	2708	Running	user	18	
conhost.exe	7620	Running	user	00	
csrss.exe	544	Running	SYSTEM	00	
csrss.exe	460	Running	SYSTEM	00	
ctfmon.exe	7284	Running	user	00	
dasHost.exe	3656	Running	LOCAL SER...	00	
dllhost.exe	4308	Running	SYSTEM	00	
dllhost.exe	7292	Running	user	00	
dwm.exe	548	Running	DWM-1	01	
explorer.exe	4916	Running	user	01	
fontdrvhost.exe	844	Running	UMFD-0	00	
frida-helper-32.exe	8128	Running	user	00	
frida-helper-64.exe	1820	Running	user	00	
frida-helper-64.exe	7672	Running	user	00	
frida-trace.exe	4384	Running	user	00	
f_wannacry.exe	11824	Running	user	12	
lsass.exe	684	Running	SYSTEM	00	
MoUsoCoreWorker.exe	9032	Running	SYSTEM	00	
msdtc.exe	4532	Running	NETWORK ...	00	
msedge.exe	4908	Running	user	00	
msedge.exe	5628	Running	user	00	
msedge.exe	1060	Running	user	00	

Figure 5.8: Frida results

CHAPTER 6

Conclusions

Ransomware poses a significant threat to organizations globally, resulting in extensive disruption and financial harm. In our endeavor to counter this menace, we have leveraged identifiable traces left by ransomware within compromised systems. Employing advanced digital forensics techniques, notably memory forensics and CryptoAPI hooking, we aimed to facilitate timely detection and mitigation of ransomware attacks.

Our exploration has underscored the indispensable role of digital forensics in combating ransomware. Memory forensics emerged as a formidable tool, enabling the revelation of concealed information within system memory. This capability provided crucial insights into ransomware behavior and revealed potential vulnerabilities. Through meticulous analysis of memory dumps and CryptoAPI function calls, we amassed comprehensive evidence and successfully extracted encryption keys employed by various ransomware strains.

Our practical endeavors encompassed the analysis of diverse ransomware samples, including Jigsaw, Phobos, NotPetya, Thanos, Gpcode, and WannaCry. Despite encountering challenges such as the volatility of physical memory and the intricate nature of ransomware behavior, our study underscores the efficacy of digital forensics in mitigating ransomware threats. By presenting our findings, we contribute to the expanding knowledge base in ransomware mitigation, emphasizing the criticality of proactive cybersecurity measures.

In conclusion, our study serves as a testament to the pivotal role of digital forensics, particularly memory forensics and CryptoAPI hooking, in addressing ransomware threats. Through the adept utilization of advanced forensic methodologies, organizations can bolster their resilience against ransomware attacks and safeguard their vital data and systems.

Bibliography

[1] Smorti, Marco. Analysis and improvement of ransomware detection techniques. Diss. Politecnico di Torino, 2023.
[2] Microsoft Learn: Volume Shadow Copy Service, 2022. (https://learn.microsoft.com/en-us/windows-server/storage/file-server/volume-shadow-copy-service)
[3] Trend Micro: Cerber Starts Evading Machine Learning, 2017. (https://www.trendmicro.com/en_us/research/17/c/cerber-starts-evading-machine-learning.html)
[4] Kumaraguru, Ponnurangam, et al. "School of phish: a real-world evaluation of anti-phishing training." Proceedings of the 5th Symposium on Usable Privacy and Security. 2009.
[5] Luo, Xin, and Qinyu Liao. "Awareness education as the key to ransomware prevention." Information Systems Security 16.4 (2007): 195-202.
[6] ThreatDown by Malwarebytes: How did the WannaCry ransomworm spread? , 2017. (https://www.threatdown.com/blog/how-did-the-wannacry-ransomworm-spread/)
[7] No More Ransom. [Online]. Available: https://www.nomoreransom.org/el/index.html
[8] Kolodenker, Eugene, et al. "Paybreak: Defense against cryptographic ransomware." Proceedings of the 2017 ACM on Asia conference on computer and communications security. 2017.
[9] Dworkin, Morris J., et al. "Advanced encryption standard (AES)." (2001).
[10] Triangle InfoSeCon: Ransomware-Success Stories, 2018. (https://www.triangleinfosecon.com/wp-content/uploads/2018/11/Karishma-Mehta-Ransomware-Success-Stories.pdf)
[11] Agcaoili, Janus, et al. "Ransomware double extortion and beyond: REvil, Clop, and Conti." Trend MICRO. Available at https://www. trendmicro. com/vinfo/us/security/news/cybercrime-and-digital-threats/ransomware-double-extortion-and-beyond-revil-clop-and-conti. Accessed 20 (2022).
[12] Young, Adam, and Moti Yung. "Cryptovirology: Extortion-based security threats and countermeasures." Proceedings 1996 IEEE Symposium on Security and Privacy. IEEE, 1996.
[13] Cannell, Joshua. "Cryptolocker Ransomware: What you need to know." Malwarebytes Labs 2013 (2013).
[14] Zeltser, L. (n.d.). Free Blocklists of Suspected Malicious IPs and URLs. (https://zeltser.com/malicious-ip-blocklists/)
[15] Aboud, Marah A., and K. Mariyappn. "Investigation of Modern Ransomware Key Generation Methods: A Review." 2021 International Conference on Computer Communication and Informatics (ICCCI). IEEE, 2021.
[16] Herzog, Ben, and Yaniv Balmas. "Great crypto failures." Virus Bulletin Conference. 2016.
[17] Knudsen, Lars R., et al. "Analysis methods for (alleged) RC4." Advances in Cryptology—ASIACRYPT'98: International Conference on the Theory and Application of Cryptology and Information Security Beijing, China, October 18–22, 1998 Proceedings. Springer Berlin Heidelberg, 1998.
[18] Newman, Lily Hay. "How an accidental'kill switch'slowed Friday's massive ransomware attack." Wired 13 (2017).
[19] Ward, Mark. "Cryptolocker victims to get files back for free." BBC News, August 6 (2014).
[20] Blessing, Jenny, Michael A. Specter, and Daniel J. Weitzner. "You Really Shouldn't Roll Your Own Crypto: An Empirical Study of Vulnerabilities in Cryptographic Libraries." arXiv preprint arXiv:2107.04940 (2021).
[21] Savage, Kevin, Peter Coogan, and Hon Lau. "The evolution of ransomware." Symantec, Mountain View (2015).
[22] Adrien Guinet, "Wannacry in-memory key recovery" [Online]. Available (https://github.com/aguinet/wannakey)

[23] Dorrendorf, Leo, Zvi Gutterman, and Benny Pinkas. "Cryptanalysis of the random number generator of the windows operating system." ACM Transactions on Information and System Security (TISSEC) 13.1 (2009): 1-32.
[24] Bajpai, Pranshu, Aditya K. Sood, and Richard Enbody. "A key-management-based taxonomy for ransomware." 2018 APWG Symposium on Electronic Crime Research (eCrime). IEEE, 2018.
[25] PCrisk: BlackRuby Ransomware. Decryption, Removal, and Lost Files Recovery (Updated), 2021. (https://www.pcrisk.com/removal-guides/12266-blackruby-ransomware)
[26] SECURELIST by Kaspersky: PetrWrap: the new Petya-based ransomware used in targeted attacks, 2017. (https://securelist.com/petrwrap-the-new-petya-based-ransomware-used-in-targeted-attacks/77762/)
[27] PCrisk: RAA Ransomware. Decryption, Removal, and Lost Files Recovery (Updated), 2021. (https://www.pcrisk.com/removal-guides/10112-raa-ransomware)
[28] Cysiv: Threat Report: Thanos Ransomware, 2020.(https://www.forescout.com/resources/thanos-ransomware/)
[29] Threatpost: New GPCode Ransomware Raises Encryption Bar and Price For Key, 2011. (https://threatpost.com/new-gpcode-ransomware-raises-encryption-bar-and-price-key-032811/75075/)
[30] Secureworks: WCry Ransomware Analysis, 2017. (https://www.secureworks.com/research/wcry-ransomware-analysis)
[31] VMware by Broadcom: Threat Report: Illuminating Volume Shadow Deletion, 2022. (https://blogs.vmware.com/security/2022/09/threat-report-illuminating-volume-shadow-deletion.html)
[32] EXTRAHOP: How Ransomware Works and How to Prevent It, 2020. (https://www.extrahop.com/blog/ransomware-explanation-and-prevention)
[33] Feodo Tracker | Blocklist. (n.d.). (https://feodotracker.abuse.ch/blocklist/)
[34] Patsakis, Constantinos, David Arroyo, and Fran Casino. "The Malware as a Service ecosystem." arXiv preprint arXiv:2405.04109 (2024).
[35] Gilbert, Stephen, et al. "Can we learn from an imagined ransomware attack on a hospital at home platform?." NPJ Digital Medicine 7.1 (2024): 65.
[36] Patsakis, Constantinos, et al. "Cashing out crypto: state of practice in ransom payments." International Journal of Information Security 23.2 (2024): 699-712.
[37] Al-Rimy, Bander Ali Saleh, Mohd Aizaini Maarof, and Syed Zainudeen Mohd Shaid. "Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions." Computers & Security 74 (2018): 144-166.
[38] McIntosh, Timothy, et al. "Ransomware mitigation in the modern era: A comprehensive review, research challenges, and future directions." ACM Computing Surveys (CSUR) 54.9 (2021): 1-36.
[39] Moussaileb, Routa, et al. "A survey on windows-based ransomware taxonomy and detection mechanisms." ACM Computing Surveys (CSUR) 54.6 (2021): 1-36.
[40] Palisse, Aurélien, et al. "Ransomware and the legacy crypto API." Risks and Security of Internet and Systems: 11th International Conference, CRiSIS 2016, Roscoff, France, September 5-7, 2016, Revised Selected Papers 11. Springer International Publishing, 2017.
[41] Bae, Seong Il, Gyu Bin Lee, and Eul Gyu Im. "Ransomware detection using machine learning algorithms." Concurrency and Computation: Practice and Experience 32.18 (2020): e5422.
[42] Weckstén, Mattias, et al. "A novel method for recovery from Crypto Ransomware infections." 2016 2nd IEEE International Conference on Computer and Communications (ICCC). IEEE, 2016.
[43] Lee, Kyungroul, Kangbin Yim, and Jung Taek Seo. "Ransomware prevention technique using key backup." Concurrency and Computation: Practice and Experience 30.3 (2018): e4337.
[44] Kim, Giyoon, et al. "A method for decrypting data infected with hive ransomware." Journal of Information Security and Applications 71 (2022): 103387.
[45] Bajpai, Pranshu, and Richard Enbody. "Memory forensics against ransomware." 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security). IEEE, 2020.

[46] Paquet-Clouston, Masarah, Bernhard Haslhofer, and Benoit Dupont. "Ransomware payments in the bitcoin ecosystem." Journal of Cybersecurity 5.1 (2019): tyz003.
[47] Mehnaz, Shagufta, Anand Mudgerikar, and Elisa Bertino. "Rwguard: A real-time detection system against cryptographic ransomware." International Symposium on Research in Attacks, Intrusions, and Defenses. Cham: Springer International Publishing, 2018.
[48] Chen, Jing, et al. "Uncovering the face of android ransomware: Characterization and real-time detection." IEEE Transactions on Information Forensics and Security 13.5 (2017): 1286-1300.
[49] Dargahi, Tooska, et al. "A Cyber-Kill-Chain based taxonomy of crypto-ransomware features." Journal of Computer Virology and Hacking Techniques 15 (2019): 277-305.
[50] Gómez-Hernández, José Antonio, L. Álvarez-González, and Pedro García-Teodoro. "R-Locker: Thwarting ransomware action through a honeyfile-based approach." Computers & Security 73 (2018): 389-398.