Log analysis for malicious software detection

Routa Moussaileb

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Routa MOUSSAILEB

Log Analysis for Malicious Software Detection

Analyse de Logs pour les Besoins de Détection de Logiciels Malveillants

Thèse présentée et soutenue à Rennes, le 08 Octobre 2020
Unité de recherche : Institut de Recherche en Informatique et Système Aléatoires (IRISA)
Thèse N° : 2020IMTA0211

Rapporteurs avant soutenance :

Jean-Yves MARION	Professeur, Directeur du LORIA, Campus Scientifique Vandoeuvre-lès-Nancy
Hervé DEBAR	Professeur, Télécom SudParis, Institut Mines-Télécom

Composition du Jury :

Président :	Pascal URIEN	Professeur, Télécom ParisTech
Examinateurs :	Maroun CHAMOUN	Professeur, ESIB
	Nora CUPPENS	Professeur, Polytechnique Montréal
	José FERNANDEZ	Professeur, Polytechnique Montréal
	Jean-Yves MARION	Professeur, Directeur du LORIA
Dir. de thèse :	Yann BUSNEL	Professeur, IMT Atlantique Bretagne-Pays de la Loire
Co-dir. de thèse :	Jean-Louis LANET	Professeur, INRIA
Encadrante de thèse :	Hélène LE BOUDER	Maître de conférences, IMT Atlantique Bretagne-Pays de la Loire

Acknowledgement

This thesis is the outcome of collaboration and support of many people, to whom I am sincerely grateful and appreciate their help.

I would like to thank my dad and my beautiful sister Nisrine for being my best support system throughout this journey. You have always been a source of joy, hope and inspiration on so many levels.

To my Lebanese and Lithuanian families: you are a blessing!

I would like to express my gratitude to my research directors, Professor Nora Cuppens and Professor Jean-Louis Lanet for their trust, time and dedication. Their valuable comments and guidance helped me during my research activities and shaped me into the best version of the researcher I am today. I also appreciate the sleepless nights to submit my papers within tight deadlines ;) I am thankful to my supervisor Hélène Le Bouder for guiding me throughout my journey and showering me with chocolate boxes to celebrate milestones in my thesis.

Thank you Professor Yann Busnel for all your efforts to organize my thesis defense, and a special thanks to the professors Jean-Yves Marion and Hervé Debar for being my thesis rapporteurs dedicating their time for reading my manuscript. I would like to thank the rest of my thesis committee: Professors Maroun Chamoun, José Fernandez and Pascal Urien for their insightful comments and encouragement.

Many thanks to IMT Atlantique and INRIA department and staff for providing me the right tools and equipments to pursue my research in a pleasant and friendly environment.

I would like to thank my students for their tremendous work and effort that helped me during my thesis.

I would like to thank my colleagues and friends for bringing joy into my life. A special thanks to my lovely girls Farah, Rahaf, Tania, and Nisrine. Thank you Salima, Renzo, Romain, Juan Carlos, Aris, Aimilia, Thomas, Edwin, Raphaël, Tima, Carel, Carla, Marya, Kevin, Léo, Léa, Ludo, Louis, Alex, Hadi, Keren, Dimitrios, Georgios, Bassem, Nagham, Yassine, Yara, Marco, Hassan, Nadine, Mohamad, Joe, Mariella, Nathalie, Lara, Jean-Louis, Jack and Nour. I spent the best moments with you during those three years.

To everyone who supported me during this jounrey, **Thank you** again.

Lastly and most importantly, I dedicate all my work to the loving memory of my beautiful mom Regina. You are always on my mind and in my heart!

List of Publications

1. Routa Moussaileb, Benjamin Bouget, Aurélien Palisse, Hélène Le Bouder, Nora Cuppens, and Jean-Louis Lanet. Ransomware's early mitigation mechanisms. *In Proceedings of the 13th International Conference on Availability, Reliability and Security*, page 2. ACM, 2018.

2. Routa Moussaileb, Nora Cuppens, Jean-Louis Lanet, and Hélène Le Bouder. Ransomware network traffic analysis for pre-encryption alert. *In International Symposium on Foundations and Practice of Security*, pages 20–38. Springer, 2019.

3. Routa Moussaileb, Charles Berti, Guillaume Deboisdeffre, Nora Cuppens, and Jean-Louis Lanet. Watch out! doxware on the way... *In International Conference on Risks and Security of Internet and Systems*, pages 279–292. Springer, 2019.

4. Routa Moussaileb, Renzo E. Navas, and Nora Cuppens. Watch out! doxware on the way... *Journal of Information Security and Applications*, 55:102668, 2020.

Contents

[1] A zoom into the types of network intrusion detection systems and the datasets available is presented in section 5.

List of Figures

List of Tables

Abstract

Ransomware remains the number one cyberthreat for individuals, enterprises, and governments. Malware's aftermath can cause irreversible casualties if the requirements of the attackers are not met in time. Cyber attackers have a history of mounting attacks on hospitals; for example, the ransomware attack carried out on the cardiology center of Acadiana in April 2017. Similarly, Ryuk resurgence in 2019 forced Australian health service providers to shut down their services and systems. The goal of earlier attacks was to gain money. However, these attacks are threatening the lives of millions around the world in the midst of the COVID-19 pandemic. Besides hospitals, the WannaCry attack in late 2017 affected many enterprises such as FedEx, Nissan, railway companies in Germany, and more than 200 000 computers worldwide. For all the reasons cited above, we focused on ransomware, and we considered it as an exquisite malicious software that should be scrutinized.

I start the thesis by providing a systematic review of ransomware countermeasures starting from its deployment on the victim machine until the ransom payment via cryptocurrency. After a thorough analysis of the white papers and journals collected from the past few years, we define four stages of this malware attack: Delivery, Deployment, Destruction, and Dealing. Then, we assign the corresponding countermeasures to each phase of the attack and cluster them according to the techniques used. Once this step has been accomplished, we noticed that some aspects were not treated previously in the literature, or there was not a conclusive decision about the provided technique. Accordingly, we present our countermeasure based on the system level or network level.

The first contribution introduces a ransomware detection technique that serves as an Intrusion Detection System (IDS). More precisely, it targets crypto-ransomware since it presents a higher threat than locking ransomware. It is based on a file system exploration before the attack (encryption) takes place. Indeed, once it is unpacked, ransomware's payload goal is to explore the file system to find files to encrypt. As for the exploration phase, threads that traverse the file system behave similarly and predictably, enabling the possibility of an early detection of the ransomware.

Moving on to the network part, we propose an analysis of various ransomware families based on the collection of system and network logs from a computer. We delve into analyzing malicious network traffic generated by these samples to perform a packet-level detection. Our goal is to reconstruct ransomware's full activity to check if its network communication is distinguishable from benign traffic. Then, we examine if the first packet sent occurs before data encryption to alert the administrators or afterward. We aim to define the first occurrence of the alert raised by malicious network traffic and where it takes place in a ransomware workflow.

The final contribution of the thesis provides an insight into plausible attacks, especially Doxware (also called leakware). We present a quantification model that explores the Windows file system in search of valuable data. It is based on the Term Frequency-Inverse Document Frequency (TF-IDF) solution provided in the literature for information retrieval. The top n files considered are then exfiltrated over the Internet to the attacker's server. Our approach delivers an observation of the evolution of malware throughout the last years. It enables users to prevent their sensitive information from being exposed to potential data exfiltration attacks.

Résumé

Les rançongiciels demeurent la menace informatique principale pour les particuliers, les entreprises et les gouvernements. Les conséquences de ces attaques peuvent causer des pertes irréversibles si les exigences des attaquants ne sont pas satisfaites à temps. Les cyber-attaquants ont l'habitude de monter des attaques contre les hôpitaux ; par exemple, l'attaque menée sur le centre de cardiologie d'Acadiana en avril 2017. De même, la résurgence de Ryuk en 2019 a forcé les prestataires de services de santé australiens à arrêter leurs services et systèmes. Les attaques ont pour objectif un gain monétaire. Cependant, aujourd'hui, des millions de vies sont en jeu. Outre les hôpitaux, l'attaque de WannaCry fin 2017 a touché de nombreuses entreprises telles que FedEx, Nissan, des compagnies de chemin de fer en Allemagne et plus de 200 000 ordinateurs dans le monde entier. Pour toutes les raisons citées ci-dessus, nous nous sommes concentrés sur les rançongiciels, et nous les avons considérés comme un type de logiciel malveillant qui devrait être surveillé.

Nous commençons la thèse par une analyse systématique des contre-mesures de rançongiciels, depuis leur déploiement sur la machine victime jusqu'au paiement de la rançon par crypto-monnaie. Après une analyse approfondie des papiers et des journaux recueillis ces dernières années, nous définissons quatre étapes pour ce type d'attaque: la distribution, le déploiement, la destruction et la transaction. Ensuite, nous assignons les contre-mesures correspondantes à chaque phase de l'attaque et les regroupons selon les techniques utilisées. Une fois cette étape accomplie, nous avons constaté que certains aspects n'étaient pas traités auparavant dans la littérature, ou il n'y avait pas de décision concluante sur la technique fournie. En conséquence, nous présentons nos contre-mesures en fonction de leurs implémentation soit au niveau du système soit au réseau.

La première contribution présente une technique de détection de rançongiciel qui remplit le rôle de Système de Détection d'Intrusion (IDS). Cette technique est basée sur une exploration du système de fichiers avant que l'attaque (le chiffrement) n'ait lieu. En effet, le but de la charge utile du rançongiciel est d'explorer le système de fichiers pour trouver des fichiers à chiffrer. Quant à la phase d'exploration, les processus qui traversent le système de fichiers se comportent de manière similaire et prévisible, permettant la possibilité d'une détection précoce du rançongiciel.

Passant à la partie réseau, nous proposons une analyse des différentes familles de rançongiciels basée sur la collecte des journaux système et réseau d'un ordinateur. Nous nous penchons sur le trafic réseau malveillant généré par ces échantillons pour effectuer une détection au niveau des paquets. Notre objectif est de reconstituer l'activité complète du rançongiciel pour vérifier si la communication réseau se distingue du trafic bénin. Ensuite, nous examinons si le premier paquet envoyé se produit avant le chiffrement des données pour alerter les administrateurs ou après. Nous visons à définir la première occurrence de l'alerte déclenchée par le trafic réseau malveillant.

La dernière contribution de la thèse donne un aperçu des attaques plausibles, en particulier les Doxware (également appelés "leakware"). Nous présentons un modèle de quantification qui explore le système de fichiers Windows à la recherche de données importantes. Le modèle est basé sur Term Frequency-Inverse Document Frequency (TF-IDF), une solution fournie dans la littérature pour la recherche d'information. Les fichiers retrouvés et classés cruciaux sont ensuite exfiltrés sur l'Internet vers le serveur de l'attaquant. Notre approche permet d'observer l'évolution des logiciels malveillants au cours des dernières années. Elle permet aux utilisateurs d'éviter que leurs informations sensibles ne soient exposées à des potentielles attaques d'exfiltration de données.

Introduction

Malware remains a recurring threat that affected people decades ago and still is in an exponential rise. The trade-off prevails in what researchers and security analysts try to protect and attackers that always find a way around. Advances in cybercrime technology, like the dark-web marketplace, help attackers to buy the required tools to carry out their attacks, facilitating malware's propagation [2]. Some of the essential malware families are worms, virus, Trojan horse, logic bombs, and ransomware. This thesis focuses on ransomware, our main research study.

The first ransomware appeared in 1989. Since 2012, the number of ransomware victims has increased significantly. Ransomware attacks represent a widespread phenomenon of this decade. Our motivation to join this arms race against malware is the increased number of attacks in recent years. Besides, the polymorphism of such ransomware registered by antivirus software led to an undetected/unmitigated threat in the early stages. Ransomware attacks are no longer affecting users' data or computers, but they undermine many public services. For example, a hospital has been hit by ransomware, and its servers have been encrypted, exposing more than 9k patients [3]. Symantec, one of the leading cybersecurity companies worldwide, has been able to block in the first half of 2017, 319k ransomware, as shown in their annual report [4]. According to the latest study completed by Malwarebytes, the top industries affected by ransomware include but are not limited to consulting, education, manufacturing, retail, and government [5]. There is a high demand to mitigate the ransomware infection process since the economic incentive is at risk.Victims infected by ransomware suffer considerable monetary loss. De facto, sixteen million US dollars are traced back to ransomware payment via Bitcoin over two years [6]. Also, some ransomware authors create eBay-like auction site for stolen data [7]. It is auctioned to the highest bidder. Besides, Ransomware-as-a-Service (RaaS) concept provides the required tools for cybercriminals to carry out their attacks [2]. No prior knowledge of the end-to-end system nor the infrastructure of the victim's machine is required using RaaS. It represents a significant reason for this business growth in the previous years. Even though former operating system targets were mostly Windows computers, nowadays a more comprehensive range of infected equipment and OS is noticed: MacOS, IoT devices and Cellphones (Android OS) [8–14]. We chose to analyze ransomware in Windows-based operating systems since it is the most deployed and used OS worldwide, thus, the most impacted.

Bond *et al.* state: "The arms race between propagation and defense will continue ad infinitum" [15]. We face several challenges in analyzing Windows ransomware. There is an ongoing work on this subject. The ransomware life cycle is limited (it can be active today but inactive tomorrow). The use of cryptocurrency such as bitcoin for the trade is nearly impossible to trace. It remains a valid model since attackers are peer pressuring victims who are willing to pay any amount to retrieve their data. Evasion techniques are spreading at a high rate. It is a challenge for antivirus software to adjust to the ongoing ransomware evolution. This kind of global economy is beneficial for cybercriminals and is fed by people's lack of information about spam emails and other mechanisms that enable the spread of ransomware at a very high rate. Our goal is to restrain file losses if no prior detection was achievable and to evaluate the possibility of new types of attacks. Researchers are teaming up to prevent ransomware exponential proliferation and impact on users. Meanwhile, attackers are keeping up the pace too forming an extortion cartel to share resources and increase the success rate of the attacks, thus, the monetary gain [16].

Outline

The thesis takes place at IMT Atlantique Bretagne Pays de la Loire engineering school and Inria research center in Rennes. It is funded by the French government defense called Direction Générale de l'Armement Maitrise de l'Information (DGA MI). The experiments are carried out in the restricted area of IMT (zone à régime restrictif ZRR) and in the High Security Laboratory (HSL, Laboratoire de Haute sécurité LHS). This thesis extends the work achieved by Aurélien Palisse by analyzing the system and network logs [17]. This thesis offered me the possibility to guide students throughout various labs including intrusion detection systems, network topology discovery, access control policies, Android application development, network packets analysis. In addition, I was able to collaborate with my colleagues and students broadening my research scope.

This thesis is divided into two parts. The literature review is introduced in Part I. An in-depth look at ransomware's workflow is proposed in Chapter 1. It includes the initial infection means, the current detection mechanisms employed to protect the system from such prominent attacks, and finally, the counter-countermeasures undertaken by attackers preserving the malicious ransomware. A detailed overview of the proposed techniques developed in the literature is presented in Chapter 2. A classification of those elements based on ransomware's workflow is given. To the best of our knowledge, this aspect was not previously covered in research areas. This survey serves as an entry point to any person concerned with ransomware's lifecycle and the actions undertaken to thwart it. We conclude the Part I by introducing our chosen dynamic analysis platform, and we provide an overview of the ransomware samples acquired from public databases (Chapter 3).

Part II consists of our contributions divided in three chapters. A ransomware detection technique that serves as an Intrusion Detection System (IDS) is introduced. It is based on a file system exploration before the attack (encryption) takes place (Chapter 1). Indeed, once it is unpacked, ransomware's payload goal is to explore the file system to find files to encrypt. This search is done from the root of the file system, or directly from the user's folder, with a depth-first or a breadth-first search. As for the exploration phase, threads that traverse the file system behave similarly and predictably, enabling the possibility of an early detection of ransomware. The method mentioned above can help a user to protect the confidentiality and availability of his/her data while limiting the probability of an attack and minimizing losses.

Afterward, a way to spot the same traceability, however, based on network analysis is proposed in Chapter 2. A mechanism for data filtering based on open source tools is provided. Then, ransomware models are created via machine learning on network flows. Finally, ransom notes and encrypted files are evaluated to check whether the detection occurred at a time t inferior at the start of the encryption. Our work does not represent a real-time based solution, but rather a study on ransomware families to extract an additional signature besides the visible encryption phase.

Lastly, an insight into plausible attacks, especially Doxware (also called leakware) is proposed. A quantification model that explores the Windows file system in search of valuable data is presented in Chapter 3. Our approach delivers an observation of the evolution of malware throughout the last years. It enables users to prevent their sensitive information from being exposed to potential risks.

An overall analysis of ransomware in the last decade and its impacts on humans and society is evaluated. We answer the following question on Windows-Based Ransomware: What to Expect/What is Next?

Part I

Literature Review

1 About Ransomware

Chapter 1 describes the current state of the art of ransomware workflow as one of the most threatening malware to date. A detailed anatomy of ransomware in presented in section 1. Then, an overview of the different types of ransomware in the wild and the encryption scheme used is given. The ransomware attack is divided into four stages based on the information gathered from the literature review and our experiments carried out in the last three years. Ransomware timeline is shown in section 2, displaying the evolution of this malware from being a simple scare tactic into encrypting files using robust cryptographic algorithms. This approach enables a distinct classification of ransomware countermeasures.

1 Ransomware's Workflow

Ransomware is a malicious software that holds the data of the victims hostage and proceeds with the release if the ransom payment is made in time. Two types of ransomware can infect a computer nowadays: a locking or an encrypting ransomware. Locking ransomware does not alter your data but blocks a person's access to his/her personal computer. Crypto-ransomware encrypts specific files from the file system, making recovery impossible if the victim does not have the decryption keys or previously installed patches [18].

Crypto-ransomware is also divided into subparts. The encryption mechanism is at the root of the payload of the ransomware. An achievable exchange (key-money) between the attacker and the victim is only possible if data are not retrievable by any other methods. Therefore, the stronger the encryption scheme, the less are the chances to recover the locked files. In addition, the key generation must be irreversible. If the analysts perform reverse engineering on the samples and extract the key generation algorithm, the business model is going to be unsuccessful.

Outlining accurately any malware behavior and defining its characteristics require both static and dynamic analysis. Even though every ransomware has its characterization, (samples belonging to the same family might also slightly diverge), the overall steps performed by any ransomware are similar. Multiple variations of the ransomware attack have been presented in the literature; they consist of 4, 5, or 6 phases [6, 19–21]. The majority of the ransomware samples can be grouped as follows: a multi-phased attack compromising 4 phases (Pi, where i is the number of the phase) named the 4 Ds: Delivery (P1), Deployment (P2), Destruction (P3) and Dealing (P4). The Data Recovery portrayed by (P5) depends on the victim's action after the infection. It represents the aftermath of an attack. Some users refuse to pay the ransom since they have backed up data or due to ethical reasons. The attack is summarized in Figure 1.1.

Delivery. Initially, the ransomware searches for a vulnerability and relies on all the available mechanisms to penetrate the target system. Zimba *et al.* present different means of ransomware infection (spam, web-server, server message block, macro, backdoor, flash, zero-day vulnerability) [22,23].

Deployment. Once the malware infiltrates the system, it loads all the required libraries to perform its destructive intent. Some of them might perform a kill switch domain check[1] to stop the ransomware

[1]It is a hardcoded domain name into the malware in case the creator wants it to stop spreading. The ransomware executable makes a request to this specific domain name, if the connection succeeds, the kill switch takes effect and the malware stops spreading.

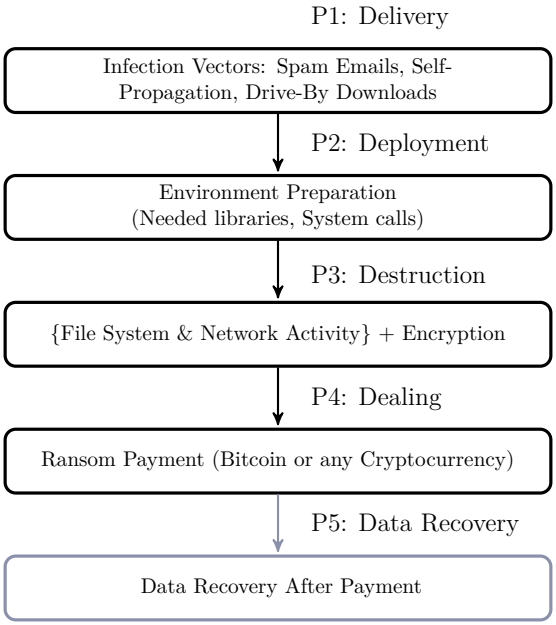

Figure 1.1: Ransomware's Workflow.

attack.

Destruction. Then, the querying of volumes on the target machine by alphabetical order begins. Different file extensions are targeted: .xls, .jpg, .pdf. Some folders can be omitted from the search, such as ProgramFiles and Windows [24]. After the search, the ransomware tries to communicate with the Command-and-Control (C&C) to receive some information (encryption keys). The encryption process begins using API calls to AES or embedded AES encrypting algorithm.

Formerly, attackers opted for symmetric encryption (standard AES). However, through reverse engineering, researchers are able to provide decryption tools for the encrypted files since ransomware authors used weak crypto [25–27]. Subsequently, attackers relied on the combination of symmetric and asymmetric encryption for an invincible malware design. Each symmetric key is generated locally on the targeted device and helps to encrypt a specific file or multiple documents (depending on the implemented algorithm). Then, the symmetric key is encrypted with the attacker's public key and added to the targeted file [28]. This scheme is known as hybrid cryptography.

Three types of encrypting ransomware or classes exist [29, 30]:

- **Class A** represents the set of ransomware that performs the encryption in-place; it opens the file, reads its content, performs encryption then closes the file with a possibility of renaming it.
- **Class B** is a further extension where the file is moved to another directory before performing the encryption and moved back once the task is accomplished.
- **Class C** opens the original file, then creates another one to write the enciphered data. The original file is deleted.

The aim of removing the original files and Microsoft Shadow Volumes is to make data retrieval impossible. A new entry is created on the Windows registry[2] enabling the execution of the ransomware every time the computer is restarted, ensuring its persistence [33].

Dealing. Finally, the ransom note is displayed to the user providing him/her the steps required to retrieve the locked files. Mostly, ransomware authors display the ransom note at the end of the infection

[2]The Windows Registry is a hierarchal database used to store information about the system. The "autostart" locations allow applications to be launched without any conscious or direct user interaction. Even if victims restart their PC, the ransomware will re-launch itself at startup, when the user logs into the system [31, 32].

phase since they do not want to be detected/stopped during their destruction process. Usually, ransom notes are written in the same language of the victims configured PC. They can have many formats like images, texts, and Hypertext Markup Language (HTML). Ransom notes explain to the users what happened to their PC and the required procedure to get the decryption keys. They display the following information:

- If it is not the mother language used by the victim, the attackers propose using *https://translate.google.com*.
- They inform users that all of the files are protected by a strong encryption with RSA 4096, and decrypting them is only possible with the help of the private key and decrypt program stored on their "Secret Server".
- The attackers offer two options: either wait until the price is doubled or start obtaining bitcoin.
- Finally, the attacker provides further instructions via multiple URL addresses like *http://t54ndnku456ngkwsudqer.wallymac.com/68052649D4FFA17*.

Data Recovery. At this stage, the attacker successfully carried out the attack. Paying the ransom or not remains a debate to date, however, researchers and security agencies strongly recommend not paying any ransom [34, 35]. In fact, if the required sum is paid, the users are supporting the criminals' business model and thus partially responsible for the increased number of infected people. Some organizations possess sensitive data, and no backups, therefore, proceed with the ransom payment. Nonetheless, there is no guarantee that the attackers are going to give them back the tools (decryptor and the decryption keys) to restore the data.

The No More Ransom Project helps ransomware victims to get back access to their files [36]. It is done by providing decryption tools with specific instructions regarding more than 100 ransomware families. Teslacrypt ransomware authors regretted their actions and released the master decryption key in 2016 [37].

2 Ransomware Timeline

Putting security on sounder footing, a thorough analysis grants insight into this malicious software and its potential future targets. Since the early stages of the computer conception, the world had known numerous attacks. The timeline shown in Figure 1.2 displays a glimpse of ransomware families' initial release dates and their targeted systems. It represents the majority of the samples discussed and analyzed in the literature detailed in the following paragraphs.

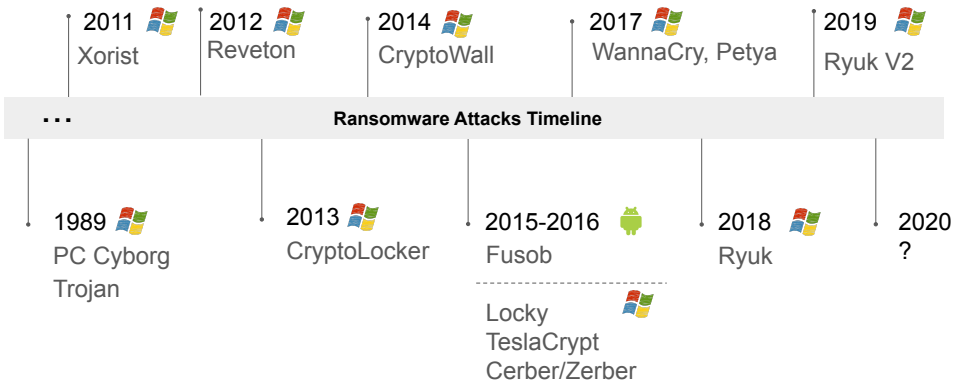

Figure 1.2: Ransomware Attacks Timeline.

AIDS Trojan is the first known malware extortion attack developed by Joseph Popp in 1989. It infects systems through a floppy disk. However, it has a significant pitfall since it encrypts the file's name and do not alter its content. Thus, restoring the encrypted files is possible if the extension and filename encryption tables are known. Following AIDS Trojan, a calm period supervened that lasted for more than two decades until Xorist ransomware emerged in 2011.

Xorist ransomware targets Windows OS, where encrypted files have diverse extensions (EnCiPhErEd, .73i87A, .errorfiles, .xdata, .HELLO, .cryptedx). It also displays a ransom note instructing the victims on the required steps to retrieve the data. The tradeoff consists of sending an ID via SMS to a specific phone number. Then, the code received is used by the victim to begin the decryption process. The 2019 and 2020 versions of Xorist show an evolution in the encryption scheme to avoid detection via entropy changes using statistical tools.

Over the following years, widespread ransom attacks infect users daily with an average of one attack every 40 seconds [38]. Many families and subfamilies are created as a medium of polymorphic and metamorphic malware. Polymorphic and metamorphic malware constantly change their identifiable features to evade detection. Polymorphic malware maintains the same functional purpose (for example, encryption), however, metamorphic malware completely re-writes the code by adding new/modified functionalities.

In 2012, **Reveton** ransomware begins to spread. It differs from Xorist by using scare tactics to pressure the victims to pay. For example, its payload displays a warning, revealing that the victim has been downloading illegal or unlicensed software. In addition, the IP addresses of victims and an actual footage of their webcam are displayed pushing them to pay. The fine is paid using MoneyPak card.

CryptoLocker ransomware emerges in late 2013. It propagates via infected email attachments and uses the RSA public-key cryptography for encrypting victims' files. It is one of the pioneers in using digital currencies such as Bitcoin for ransom payment to receive the decryption key. Furthermore, ransomware authors threaten to delete the private key used for encryption if no transfer is made upon the required deadline.

Shortly after, **CryptoWall** emerges in the first quarter of 2014. It maintains an asymmetric encryption scheme similar to CryptoLocker. To prove the efficiency of the attack scheme, attackers provide victims a free single-use decryption tool that decrypts some files to prove that they hold the encryption keys, and they are the only ones capable of restoring files [39].

Even though previous ransomware targeted Windows operating systems, **Fusob** strikes in the mid of 2015 and continues until March 2016, targeting mobile devices. Similarly to Reveton, scare tactics are used as extortion means forcing victims to pay. Up to date, at least 22k ransomware samples can be found crawling online malware databases like VirusTotal and Kaspersky [40]. The extensive analysis carried out by researchers targets especially ransomware that infected people on a large scale such as Xorist, CryptoWall. Therefore, an accurate description of each malware sample is not available in the security blogs, forums, literature review, for example, for the **SilentCrypt** sample.

Mukesh performs a combination of static and dynamic analysis of the **TeslaCrypt** ransomware [41]. It is spread via spam emails, especially during late 2015 and mid-2016. Common libraries used by such ransomware are ole32.dll, kernel32.dll, apphelp.dll. The upgraded model attacks a wider variety of file extensions (.py, .ptx, .jpeg) even though the initial version of this malware encrypts files related to 40 different games. In May 2016, the master decryption key is released by the attackers.

Locky is discovered at the beginning of 2016. It is delivered by an email attachment or via a spam campaign. It has a Microsoft document containing malicious macros [42]. Once downloaded, Locky renames itself to svchost.exe to be considered as a trustworthy application. Also, it creates other processes to delete the backup files before moving to the encryption phase. It encrypts 164 file types infecting different categories of users or companies using hybrid cryptography [43].

The initial version of **Cerber** ransomware is spotted in March 2016. Cerber has a particularity in the usage of the RaaS business model. RaaS enables launching ransom attacks by novice cybercriminals, providing them the required tools in exchange for a percentage of the collected money from the victims [44]. Another particularity of Cerber resides in an increasing ransom sum as time passes. Most importantly, there is currently no Cerber ransomware decryption tools available online [45]. Some authors classify **Zerber** as a new variant of Cerber ransomware or a new family [46] that shares specific file system traversal and network behavior [47, 48], while others believe that it belongs to the same family having an alternative name [49].

Petya ransomware is first discovered in 2016. Its particularity resides in modifying Windows Master

Boot Record (MBR), causing the system to crash. It resurges in June 2017, affecting machines having different versions of Windows from XP through 8.1 [50].

WannaCry attacks have wider target options. Not only end users are targeted, but agencies are also affected at a large scale such as FedEx, Renault, in addition to parts of the British National Health Service (NHS). Kao *et al.* performs reverse engineering to extract the leading characteristics of WannaCry ransomware [21]. It is a weaponized malware: sophisticated by design and intended for malicious purposes. This specific ransomware exploits the MS17-010 vulnerability to inject itself on the target machine [51]. Then, a kill switch domain is queried; if a successful connection is achieved, the ransomware stops its payload else, it proceeds with the malicious functionality. Ransomware authors rely on hybrid encryption to securely lock the files. The original files are deleted or erased. The worm propagation capability, defined by utilizing the EternalBlue exploit and the DoublePulsar backdoor, is used in WannaCry attacks [52].

A snapshot of ransomware characteristics is presented in Table 1.1. All ransomware families have the static IP address used by the C&C in their binaries except for TeslaCrypt that relies on the domain generation algorithm (DGA) to contact the C&C. The AES is used for all the samples except CryptoWall that utilizes RSA-2048 bits. The encryption mechanisms used, the infection process, and the type of platform infected are detailed in [1, 53, 54].

Year	Family	C&C		Encryption		Changed	Infection
		Static	DGA	Symm.	Asymm.	Extensions	Vectors
2011	Xorist	✓	-	✓	-	✓	Spam
2013	CryptoLocker	✓	-	✓	-	✓	Spam & Corrupted web pages
2014	CryptoWall	✓	-	-	✓	✓	Spam & malicious ads or sites
2015	TeslaCrypt	✓	✓	✓	-	✓	Spam & Phishing
2016	Cerber	✓	-	✓	-	✓	Spam & Fake software
2016	Petya	✓	-	✓	-	✓	Fake software. Worm
2017	WannaCry	✓	-	✓	-	✓	Worm (EternalBlue)

Table 1.1: Ransomware characteristics [1].

3 Conclusion

This overview presents two facts: ransomware attacks target specifically Windows since it is one of the most used and ergonomic OS worldwide. Also, advanced features are being continuously added to the payload of the ransomware changing its behavior. However, a shift to IoT, SCADA, and Android devices is recently noticed [11–14]. We present in the following Chapter 2, ransomware countermeasures that scale down the success of this ongoing business. They are clustered based on the methodology used. The categorized countermeasures are assigned to the corresponding ransomware phases.

2 Ransomware Detection Mechanisms

Chapter 2 presents a systematic literature review of ransomware countermeasures. It describes the available solutions developed by researchers to protect users' data. We highlight the fact that this sequence of events is necessary for administrators. Indeed, it provides an initial alert mechanism to warn the user of a potential threat. It is crucial since early detection mechanisms such as signature-based can spare file losses. Thus, it protects the whole infrastructure avoiding later on "a pact with the devil" [15] to retrieve the encrypted files. The following sections represent the defense mechanism taxonomy clustered corresponding to each phase of the ransomware workflow, as presented in Fig. 1.1 of Chapter 1. We gather the existent solutions that cover a specific phase of the intrusion and the methods deployed. The countermeasures corresponding to each phase are presented in Fig. 2.1 and detailed in the Delivery, Deployment, Destruction, and Dealing phases.

1 P1: Delivery

Infection vectors are usually hard to trace since researchers do not analyze malware on-the-fly. They download the malicious samples from online databases and execute them in a Sandbox or bare metal environment. One of the best defense mechanisms at the delivery stage is to raise awareness about ransomware cyberthreats, for example, by deleting a suspicious email immediately or filtering spam ones. These aspects corresponding to the initial phase of the ransomware attack are developed in this part. Table 2.1 clusters the following studies in five sections. The awareness part includes the best practices to be taken into consideration as a protection measure on a computer (section 1.1). It is portrayed by a safe web browsing, having an up-to-date system including the recent patches and frequent backups [55, 56]. A proposition of an incidence response is portrayed in [57] that could involve using software-defined networking [58]. Data backup is further explained in section 1.2 that is complemented by an access control list measure in place to protect specific assets on the file system (section 1.3). To enable a clear restoration point, renaming the Volume Shadow copies executable in Windows is presented in section 1.4. Finally, the signature-based detection that helps to flag malicious software without their execution on any system is described in section 1.5.

1.1 Awareness

Han *et al.* develop a proof of concept to check whether a requested website is SSL certified or not [55]. Their prototype is added as an extension to a popular browser such as Google Chrome. Also, it checks the downloaded files searching for common ransomware patterns. Moreover, to alert the user of a hidden threat, a pop-up warning is displayed. Even though their proposed concept relies on raising awareness strategies to eradicate ransomware's danger; it is not an exhaustive solution. Similarly, Ganorkar *et al.* raise awareness of the plausible threat encountered by a ransomware attack [56]. They show clearly that a communication is done between the server and the PC of the victim to retrieve the encryption key.

Before infecting the machine, an attacker should be able to gain access to the latter, fulfilled by various attack vectors, once tracked down, attacks numbers can be scaled down. Attackers are always seeking to find optimal ways to invade a target system minimizing the chances of being detected and therefore stopped. Administrators should be aware of the broad spectrum of possible penetration to protect the data's confidentiality, integrity, and availability (CIA).

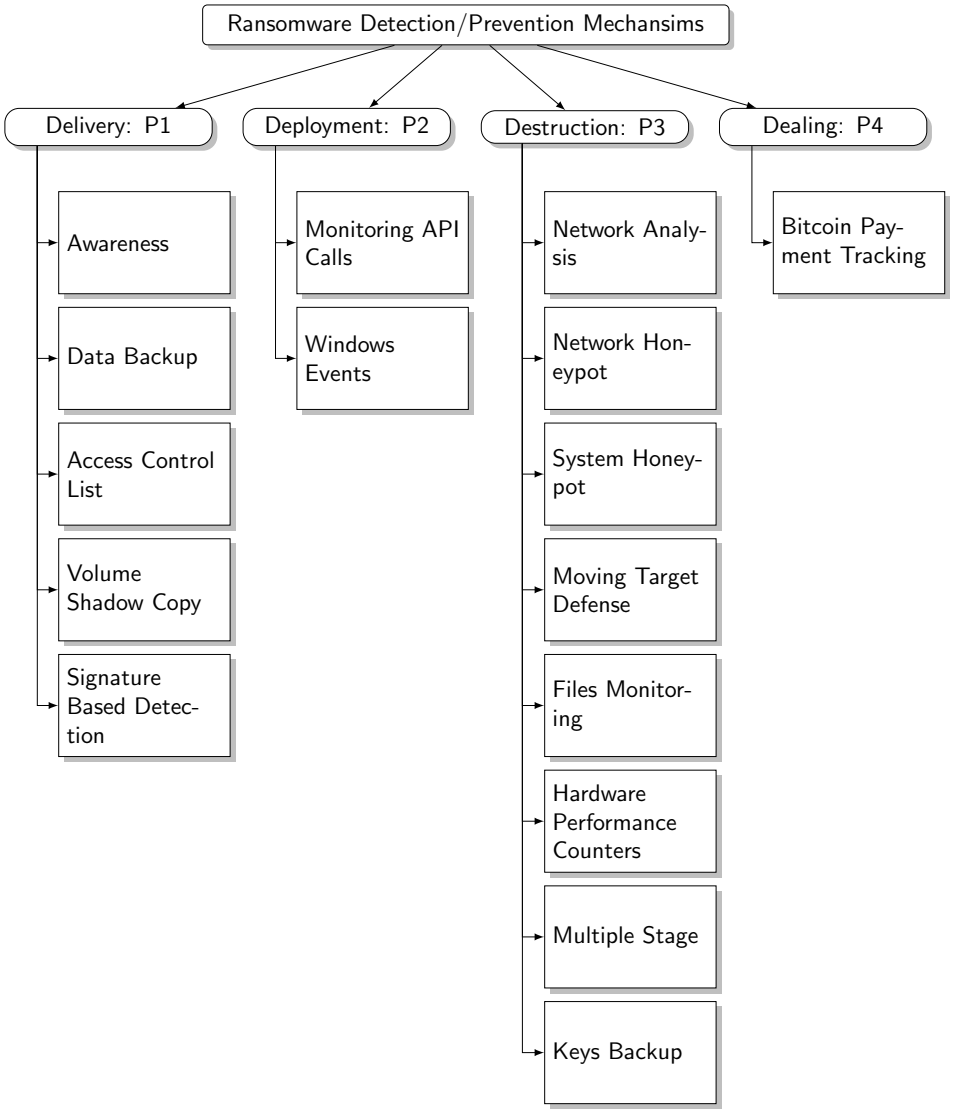

Figure 2.1: Ransomware Detection Mechanisms Summary.

The authors in [59] implement a test scenario using Endian, a software-based firewall equipped with specific functionalities (port blocking, IPs/IDS, content control, mail filtering). The testbed consists of sending emails attached with malicious PDF files to 150 employees of a pharmaceutical company where 85% were tricked by this fake attack. Infected PDFs enable the simulation of a ransomware attack without altering the file system of the victim. They state that education remains a fundamental pillar to protect the assets of a company.

The most common attack vector is the spam email. It is a form of social engineering method that lures the victim to perform an action like opening an infected PDF, image, etc. Zimba analyzed various infection vectors such as malicious emails, brute-force authentication credentials, and exploit kits (EK)[1] [20]. He described the attack model of the ransomware that consists of an attacking agent (EK or a human) using specific assets (resources, open ports, addresses) to perform the required actions (requests with an expected return) to attain its goals (ransomware payload execution). A graph translated by a matrix represents these elements. The tested ransomware performed reconnaissance attacks, checking for available backdoors. Attackers are constantly seeking malware-free intrusions since it requires no

[1]EKs are "tools used by cybercriminals to perform drive-by-download attacks" [60]. A drive-by download attack represents the unintended download of a computer software from the Internet.

actions from the end-user. Ransomware relies on this attack vector to penetrate a system and execute its payload. Besides using an exploit kit based on the vulnerability fetched [57, 58], attackers use various techniques like drive-by downloads, malvertising to penetrate the target system.

The authors in [61] carry out a comprehensive analysis (system and network-level) of distinct ransomware families in a simulated environment. They state that multiple attack vectors exist, such as social engineering methods, outdated systems, unpatched known vulnerabilities (service message block), nonexistent antivirus solutions on the target system, and finally, the absence of regular backup.

To cope with all these potential threats, researchers should be aware of the likely system breaches representing an attacker's way into the system. Most of the research carried out in this phase represents an explanation of the ransomware threat rather than developing specific countermeasures or detecting it.

1.2 Data Backup

Castiglione and Pavlovic show that a better defense is provided when an economic incentive is on the line. Consequently, strengthening the infrastructure ecosystem is essential to encumber the proliferation of those crimes [62]. They address an important issue favoring a proper backup regularly: a cost-effective solution to paying the ransom at a given time. Therefore, they suggest an encryption algorithm using a One-Time Pad with deletions. It is an effective one since encryption should frequently occur, whereas decryption is only taken into consideration if an attack arises or the system is down. There must be a coordination between the servers to keep the data synchronized and up to date, achieving a resilient distributed storage. Besides, it is immune to ransomware targeted attacks on servers: even if a particular server is down, others distribute the information among the rest of the nodes having a balanced and resilient system.

Baykara *et al.* introduce the safe zone concept where all the critical files are moved to protect them [63]. Critical data are zipped and stored in this safe zone, and the files are kept open in non-stop write mode to prevent any alteration by the ransomware. Besides, an integrity check is made to examine any changes that occur. Their approach is not tested using ransomware. A noticeable drawback is the single point of failure. Crucial information is detained in a single zone; if attacked, the user loses everything. In addition, their solution tackles only the availability and integrity of the data: if the information gets exfiltrated, the confidentiality is compromised. Thus considerable losses affect the end-user or the company.

1.3 Access Control List (ACL)

Kumari *et al.* propose a locking file mechanism to prevent any alteration of the data [64]. Their main idea relies on securing the memory location of confidential files. Their mechanism is divided into three phases. Initially, an authentication with an input password is required. Then, the file extension is checked before storing the data; if it is valid, the file is locked and hidden. Certainly, it presents a white list of possible file manipulation that could be done by a user. However, their solution is not scalable: individuals need to gain access to their sensitive files regularly. They have to accomplish the same procedure for all the important files done manually to register them in a safe "database". Also, they need to authenticate each time they want to access the files. Moreover, their solution was not tested using ransomware samples.

1.4 Microsoft Volume Shadow Copy (VSS)

Weckstén *et al.* analyze multiple ransomware samples and conclude that they rely on the tools available in the infected OS to carry out their attacks [65]. The proposed idea consists of renaming the Volume Shadow copies (VSS) executable in Windows, so the ransomware will not be able to access or modify it. They run their experiments in a virtualized environment, and all files are retrieved. If the restoration point schedule is correctly and frequently configured, malicious attacks can be defeated. As the authors state, it is a simple solution for unfamiliar users; however, unsustainable.

1.5 Signature Based Detection (SBD)

The static analysis enables a signature-based code classification. For example, if a malicious piece of code is found within the executable, an antivirus will drop the complete package. Nevertheless, this mechanism is not immune to code obfuscation. The behavioral analysis extends this part, where malevolent patterns are examined. Dynamic analysis limitations are some stealth and anti-debugging techniques [66]. This part outlines ransomware characteristics extracted mostly from static analysis. Signature-based detection belongs to the Delivery phase since the payload (encryption) is not executed. The major drawback of signature-based detection is its inability to detect zero-day attacks. The protection of any system is valid only after updating the signatures database with the ones published of unseen malware.

Medhat *et al.* present a static-based framework having a multi-level alert system to detect ransomware [67]. Their work relies on the concept of shared patterns/code among ransomware that represents static features. Four elements are kept: cryptographic signatures, API functions, file keywords, and file extension. Their detection tool is based on Yara rules [68]. A limitation of their work is the omission of obfuscated or packed samples representing a significant number of ransomware samples in the wild.

Subedi *et al.* utilize reverse engineering tools to provide distinct identifiers for various ransomware families [54]. For a given ransomware, they extract assembly instruction level, libraries used, and functions called. The association rule mining is deployed for DLLs (Dynamic-Link Library) identification to construct a known signature (sequence of DLLs) of the malicious software. Cosine similarity is used to measure the similarity between the frequency vector of the assembly code of a benign and malevolent software [69]. Their implemented CRSTATIC tool can detect crypto-ransomware without executing the sample, based on the features provided above.

The authors in [70] focus on distinguishing a benevolent application from ransomware established on discriminant characteristics of the Portable Executable (PE) file. In the static analysis part, the PE file is disassembled and unpacked to extract the metadata from the header fields. Accordingly, 60 static properties are identified to enable an accurate classification (bytes on the last page, pages in file and relocations in the DOS header; size of optional header in the file header and number of sections) and nine ransomware specific (presence of packer, DLLs used for network communication, command for registry modification). Howbeit, if obfuscation was used, the authors performed a dynamic analysis to extract the rest of the features. The sample is then executed in an isolated environment having the sys-internal tools in place. An extended analysis revealed suspicious DLLs at run-time, the windows registry changes, and the alteration of directories.

The work done in [71] performs a comparison between binaries checking for a similarity amongst the samples. To this end, import hashing, fuzzy hashing, and YARA rules have been used. Even though each of these methods has its limitation, 92% of similarity was achieved among the same families. Fuzzy hashing outperformed the rest of the fuzzing algorithms based on time efficiency, memory, and hash/rule size. A continuation of their work is the analysis of the polymorphic aspect of various samples acquired. It is done by performing ransomware clustering using the combination of two fuzzy techniques: fuzzy hashing and FCM clustering method [72]. They are able to aggregate multiple samples of the same family in different distinct clusters. The accuracy of the clustering varies between the families and the number of clusters chosen.

2 P2: Deployment

The following step in ransomware mitigation relies on monitoring the API calls. They show the interaction between the malware and the computer of the victim. Many attackers rely on the services provided by Microsoft Cryptographic API to complete their payload execution, such as random number generator, AES encryption. Writing a specific code is prone to errors. Thus, attackers prefer to use built-in services to accomplish their tasks. Therefore, researchers analyzed the API calls, including their patterns and frequency, to classify processes (section 2.1). Monitoring carefully Windows events helps to extract patterns to describe the habitual behavior of any user accurately compared to ransomware (section 2.2). These methods are summarized in Table 2.2.

Articles	Type	Approaches		Tested	Detection/Protection/Prevention Mechanism
		Static	Dynamic	Solution	
[55]	Awareness	-	-	X	Web browser extension to alert users of potential threat
[56]	Awareness	✓	✓	✓	Best Practices Proposal: Disabling RDP, frequent backup
[58]	Awareness	✓	✓	✓	SDN usage to detect and alert the user of malicious intent
[57]	Awareness	-	-	X	Proposition of incidence response hacks
[62]	Data Backup	-	✓	X	Providing a Dynamic Distributed Storage
[63]	Data Backup	✓	-	X	Data stored in a Safe Zone System
[64]	ACL	-	-	X	Authentication + File Locking
[65]	VSS	-	-	✓	Renaming Windows VSS
[67]	SBD	✓	-	✓	Yara rules to detect ransomware
[54]	SBD	✓	✓	✓	Ransomware signature extracted based on reverse engineering tools
[70]	SBD	✓	✓	✓	Discriminant characteristics of the Portable Executable extracted to flag ransomware
[71]	SBD	✓	-	✓	Fuzzy hashing to compare binaries and detect ransomware

Table 2.1: Ransomware Detection Mechanisms Overview for the Delivery Phase P1.

2.1 API Calls

Chen *et al.* monitor the API calls made by ransomware to generate API calls flow graphs (CFG) [73]. It is a proactive solution that provides an early stage detection while the ransomware is still setting its environment. They improve ransomware detection by analyzing the API call flow graph utilizing machine learning techniques. They develop their API Monitor tool to gather the calls made during the experiments executed on a virtual machine. A weighted directed graph represents the sequence of API calls. The weight corresponds to the frequencies of a specific API 1, followed by API 2. The CFG is converted to a feature vector where its values are normalized and rescaled from zero to one. Subsequently, feature selection is performed to retain certain features enabling a distinct separation between malicious and benevolent software. The Simple Logistic (SL) algorithm outperforms the rest of the classifiers (decision tree DT, random forest RF, support vector machine SVM).

In a like manner, Maniath *et al.* rely on the sequence of API calls to flag ransomware behavior. They utilize a modified version of the Cuckoo sandbox to extract those calls from the JSON report of 157 ransomware samples [74]. The sequence of API calls is converted to a chain of integers (each integer refers to a specific system call). Missing inputs in the dataset occur because each malware is programmed to operate distinctly. Thus, to handle those missing inputs (for example, five sequence calls compared to 200), 0s are appended to the record since they do not influence the record's value. By applying the LSTM algorithm (Long Sort-Term Memory), prominent results are achieved.

Takeuchi *et al.* also rely on the sequence of API calls to depict ransomware-like behavior [75]. Their contribution is the representation of API calls by a vector where they quantified the sequence of those API calls (including the number of q-grams in the execution logs). A mapping is performed using n-grams. For a software using 2 distinct API calls A= a, b, the possible 2-gram would be (a, a), (a, b), (b, a) and (b, b) . The final vector is [0,1,1,0] since it does not include (a, a) nor (b, b). A major drawback is that two distinct API call strings can have the same output vector. Therefore, they solve this issue by adding the count of the performed calls. Since the number of API calls diverges exponentially between applications, standardized vectors are calculated to have a balanced set. SVM is used to differentiate between the created vectors belonging to ransomware or to a benign application.

Similarly, Vinayakumar *et al.* and Hampton *et al.* analyze ransomware activity considering API call patterns and their frequency [76, 77]. Tests performed on the sequence of API calls show that

ransomware identification is possible through its frequency. Additionally, some system calls are made solely by ransomware (InternetOpen, CryptDeriveKey, CryptGenKey) [77].

Al-Rimy *et al.* propose a 0-Day aware crypto-ransomware behavioral detection framework [78]. Their model is divided into three submodules: preprocessing, features engineering, and detection. They do not rely only on API calls collected during the preprocessing phase for early detection. They added a layer consisting of data-centric detection (this method focuses on the data using entropy or similarity) and anomaly detection based on a deviation of normal behavior. However, no tests were performed to prove the validity nor the accuracy of their framework even though it has promising characteristics.

Al Rimi *et al.* propose a combination of behavioral and anomaly-based mechanisms to achieve accurate ransomware detection rate and maintain low false alarms [79]. Cuckoo sandbox is used for the experiments where all the samples are executed for 5 seconds to collect the API calls information. Each API call is treated as a feature. Term Frequency–Inverse Document Frequency (TF-IDF) is used to build a vector for the training and test phase. The vector contains the weight (calculated by applying TF-IDF formula) of each API. Mutual Information (MI) is adopted to extract significant features. As for the anomaly-based estimator, only benign software is used to carry out the experiments. This estimator flags a deviation compared to normal behavior. The fusion of both mechanisms shifted the detection results providing better classification. In some cases, specific user actions (for example, a mouse click) trigger the execution of ransomware. Therefore, the duration of 5 seconds is not adequate for API calls collection.

Al Rimy *et al.* overcome information limitation in the early phases of a ransomware attack by using two novel techniques incremental bagging (iBagging) and enhanced semi-random subspace selection (ESRS) [80]. iBagging represents a progressive stage of the attacks rather than having it all at a specific time, while the ESRS builds various subspaces maintaining the diversity in each one. Three main components constitute their mechanism: initially, the subspace creation then features selection and, finally, the choice of the best combination of base classifiers. Their database consists solely of API calls captured during the execution of each sample in a sandboxed environment. A pre-encryption boundary vector represents the stage that occurs just before the attack takes place. For the data subsets, N-gram and Term Frequency–Inverse Document Frequency (TF-IDF) are employed to decrease the similarity between two adjacent subsets. Taking into account only 10% of the APIs in the training set, they achieved higher detection rate using iBagging with ESRS rather than using solely iBagging.

Palisse *et al.* have implemented a Cryptographic Service Provider [81]. It contains the required functions for the end-to-end encryption process. This mechanism would help restore encrypted data. Adopting this method, a user can protect himself from 50% of ransomware attacks. This solution takes place during the encryption phase, at the end of the infection process of the ransomware. The authors use bare-metal hosts during the experiments.

2.2 Windows Events

The initial stage of a ransomware is similar to reconnaissance for Advanced Persistent Threats (APT) [82]. Both malware rely on social engineering techniques to perform the required tasks of an attacker (opening an infected PDF). Ransomware gathers information about the environment (language used, IP addresses, installed libraries) in order to carry out the attack. The malware executed on the machine proceeds with a sequence of specific events that is explored by Homayoun *et al.* [83]. They gather the first 10 seconds of logs collected from any goodware or malware downloaded on their virtual machines. Analyzed ransomware samples are from three different categories Locky, Cerber, and Teslacrypt. The logs consist of data gathered from the ProcessMonitor application that has records, including loaded dynamic linked libraries (D), file system activities (F), and registry activities (R). Thus, the authors converted their data into a sequential dataset and applied the sequential pattern mining technique Mind the Gap: Frequent Sequence Mining (MG-FSM). The best features from the maximum sequential pattern are selected: R, D, and FD (file system to DLL).

Using random forests, authors achieve a clear distinction between the events accomplished by a goodware versus the ransomware (for example, ransomware applications conduct a wider range of Registry activities). Their experiments are done on virtual machines. The main advantage of such solution is the early detection of an infected PC without any prior encryption process. However, any change in the current analyzed sequence of events would modify the detection rates prone to increased false positive and negative rates.

Articles	Type	Approaches		Tested	Detection/Protection Mechanism
		Static	Dynamic	Solution	
[73–76, 79, 80]	API Calls	-	✓	✓	API Calls sequence &/or frequency features used to detect ransomware via applying ML algorithms
[81]	API Calls	-	✓	✓	Intercepting calls made to MS-CAPI
[83]	Windows Events	-	✓	✓	Maximal frequent patterns extracted from (registry, DLL, transition file to DLL) events then ML applied to detect ransomware

Table 2.2: Ransomware Detection Mechanisms for the Deployment Phase P2.

3 P3: Destruction

The destruction phase is characterized by the encryption process that affects a significant number of user files. Initially, researchers flag the malicious communication with the C&C of the attacker that represents a critical element of the ransomware attack (sections 3.1 and 3.2). Then, the honeypot countermeasures are developed in section 3.3 to detect ransomware that queries the file system to collect specific file extensions (.doc, .xls, .txt, .jpg). The moving target defense technique that regularly changes file extensions omitting consequential file types from the ransomware search is presented in section 3.4. Massive operations, including open, read, and write, portray the encryption phase. Encrypted information has higher entropy[2]. The statistical tools adopted in the literature that distinguishes a non-encrypted text from an encrypted one are discussed in section 3.5. This step consumes resources; therefore, the hardware events can depict the ongoing ransomware attack (section 3.6). Some authors combine multiple indicators of compromise to detect malicious behavior (section 3.7). Finally, if no real-time solution can stop the encryption process, the restoration of keys can save user files (section 3.8). The methods presented in the destruction phase are summarized in Table 2.3.

3.1 Network Traffic Analysis[3]

Wang *et al.* propose a mechanism for remote desktop protocol tracing and tracking down [84]. The authors resort to cyber deception technology to lure ransomware attacks. They create a deception environment to log and analyze the actions completed by the attacker. It consists of a login with weak passwords and known vulnerabilities enabled. The collected information relies on IP addresses, shared folder path, and clipboard strings. An automated analysis is carried out to filter and obtain the required results. To accomplish this task, the Markov model is trained to distinguish gibberish words from existing ones. Windows 7 virtual machines are used for the experiments. The question remains if this traceability is enough to stop ransom attacks and if it is sufficient to physically traceback the attacker.

The authors in [85] propose a novel detection mechanism of highly survivable ransomware. They target hybrid ransomware since they represent the highest threat. They define the Highly Survivable Ransomware as ransomware that infects users; ransomware writers can only reverse the encryption process and restore the data. Finally, freeing one victim does not include freeing the rest of the targeted devices. The authors focus their detection mechanism on the public key received from the C&C to perform the encryption on the infected system. It targets the domain generation algorithm to contact C&C candidates for key retrieval. They propose a mechanism able to detect DNS requests generated by domain generation algorithms (DGA [86]). Markov chains are used to define the transitions from one letter to another. A gibberish query is more likely to be generated by DGA. They added another layer

[2]Recent Xorist samples encrypt files maintaining an unaltered entropy before and after the encryption.

[3]A zoom into the types of network intrusion detection systems and the datasets available is presented in section 5.

of protection by signing the applications. Their detection is completed before the encryption process, so all files are saved.

Tsen *et al.* find that ransomware share common communication patterns that enable an early detection [87]. Their solution based on deep packet inspection is fed to a deep learning algorithm to distinguish between malicious and benign traffic. The data consist of HTTP requests and raw payloads downloaded from malware-traffic-analysis.net.

Alhawi *et al.* perform also supervised learning on the network traffic downloaded from VirusTotal [88] using Weka [89]. Different ransomware families like Cerber, CryptoWall, and Teslacrypt are included in the training phase. The features selected for the learning algorithm are protocol type, addresses and ports (source and destination), the number of packets exchanged, the total number of packets, time relative to the start of the conversation, and the duration of the conversation flow. Surprisingly, feature selection did not influence the overall sensitivity and specificity of the detection (for example, decision trees provided the same results with/without feature selection). Also, there was no separation between the TCP and UDP protocols, which might lead to an unbalanced dataset. For instance, Zerber ransomware (7bbb346484186447fb1d085e6942b56b MD5 hash) made up to 40 000 different UDP requests while TeslaCrypt (05330fff36ad3e359be8bb2b33f09436 MD5 hash) only 2 TCP requests [48]. Besides, an administrator should know whether this detection occurs before or after the encryption process that was not discussed in the paper.

The authors in [90] develop a Compromise Detection System (CDS) also based on machine learning applied to network traffic to detect new variants of ransomware. They perform an analysis of WannaCry ransomware once with the network configuration enabled and the second without any connection. The malware contacts the C&C via TOR (The Onion Router), which is complex to trace. Their CDS also inspects the DNS requests generated by the ransomware using DGA algorithms. In addition, their tool can interact with a firewall to block the source of compromise, thus restraining the propagation to other systems on the network.

Almashhadani *et al.* presume that the majority of ransomware samples can be detected via network communication with the C&C. Having analyzed 4 Locky samples, the authors conclude that the communication occurs before any payload execution [43]. They analyze multiple features as potential discriminating characteristics of malicious traffic, including RST, POST, GET, and DNS requests. They are able to extract 18 detectable features divided into two subsets behavioral (number of HTTP-POSTs, DNS-NE, and MDN, DNS-NE, MDN, MNBNS) and non-behavioral parameters (dns-ipv6, dns-ipv4, dns-time, dns-resp-ttl). However, another classification is taken into consideration for the training algorithm, whether it is packet level (MDN, DNS-NE, dns-ipv6, dns-ipv4) or flow level (number of HTTP-POSTs, DNS-NE, and MDN).

Cusack *et al.* monitor network traffic searching for communication patterns between the victim and the C&C [91]. Their module consists of two building blocks the stream processing and then the classifier. They can reduce important features from 28 to 8 that is sufficient for a proper classification (inflow and outflow number of bytes, length, outflow to inflow packet ratio). Their settings are not tested in a real-world scenario.

The authors in [92] develop a Software-Defined Networking (SDN) based on common ransomware patterns as an effective ransomware countermeasure. Even though detecting those signals might come behindhand; however, they can save other users from being infected by the same executable. They analyze the two corresponding families CryptoWall and Locky. Although they both communicate via HTTP requests, some specific characteristics define each malware. Their detection mechanism is solely based on the size of the data in the three POST messages. Then, for each family, the centroid and the limit distance (distance square) are established. For an unknown triple, the distance to the centroid is calculated, and if it is below the limit distance set up previously, a ransomware communication is encountered. The benign traffic is downloaded from maccdc.org. Tests are performed on Cuckoo guest with Microsoft Windows 7 to validate the proposed methodology.

Akbanov *et al.* resort also to SDN as a means of detection and mitigation of ransomware attacks based on OpenFlow [93]. From their static analysis using Petstudio, both worm and encrypting components of WannaCry samples use DLLs. The worm component gets the information about the network environment, while the encrypting one utilizing Windows Cryptographic API is used for keys generation. The authors are able to find that WannaCry tries to connect to an unregistered domain name via performing dynamic analysis. If it receives an answer, it stops its execution, else the encryption process begins. A simple string search can extract two hardcoded IP addresses found in the samples. Besides having

an initial list of blacklisted IP addresses, the OpenFlow switch of the SDN-based mechanism redirects the traffic generated by any machine connected to the network. The controller parses and extracts IP addresses from packets received to compare them to the existent DB (port numbers and IP addresses) or updates it with a new entry if malicious communication is detected. At the end, the corresponding traffic is blocked, and propagation opportunities eliminated. Three Windows 7 VMs are used for the experiments where one is infected. Their mechanism can detect the traffic incoming from the infected one and blocks it, which makes their approach successful. However, they block only the worm component rather than stopping an active infection on the PC. Another point worth mentioning as a limitation is DB poisoning with real/fake IP addresses.

3.2 Network Honeypot

Cabaj *et al.* use a honeypot technique in addition to an automatic runtime system to analyze and detect ransomware through the network activity [94]. Their approach is built on virtual machines to download and test ransomware samples on Windows XP. They reveal that CryptoWall uses domain names rather than IP addresses. Multiple actions are carried out by the sample, such as getting the IP address of the victim's machine and contacting the hardcoded servers. Therefore, by blocking the DNS requests made by CryptoWall, the authors are able to enumerate all the contacted servers. The parties maintained encrypted communication. All the proxies hosting malicious scripts are identified.

3.3 File System Honeypot

Monitoring file system activity, apart from system calls, is crucial for an overall detection mechanism. In fact, if an attacker learns different patterns or sequences of the system calls made to bypass security measures deployed on the system, an early detection of the malware is improbable.

A **honeypot** is a resource used by administrators to detect unauthorized access to a system [95].

Lee and Hong introduce a novel mechanism to make efficient decoy files [96]. Two search methods are extracted from malware's source codes. The first one consists of performing a search looking for specific file extension hence .pptx, .docx, .txt. Then, it saves the location of these files, encrypting them one by one at the end of this process. The second method is encrypting a file as soon as it is found. Since the search is performed in order or reverse order of Windows-1252 (character encoding of the Latin alphabet), consequently, decoy files are created using the first or the last character in Windows-1252. Preferably, they should be located in the parent folder rather than in sub-folders due to ransom traversal patterns. The size and attributes of decoy folders can be updated to meet the new requirements of the ransomware in the wild and flag them as soon as possible.

Lee and Hong's work is complemented by Moore and Al Kossairi *et al.* investigations [97,98]. Moore's work relies on a honeypot folder that a File Server Resource Manager (FSRM) monitors, followed by changes analysis of the Windows Event Logs. A tiered response to detection is developed based on the number of modified files. FSRM is a tool that prevents an already executing malware from infecting the entire file server. The EventSentry makes a warning if an attempt of modification is made to a specific object. The threshold is defined based on a regular observation of users' behavior. Any abnormality noticed is a deviation of double or three times the normal activity. A practical method certainly, however, it can be bypassed if the malware does not attempt to access these areas.

Whereas Al Kossairi *et al.* monitor decoy folders by Watching File System Event Handler watcher applicable only on Windows OS. Decoy folders properties have been identified (variability, differentiability from benign ones). Low (contains random data) and High (contains fake data) Interaction Decoy files are used for the proof of concept. They are monitored by Watching File System Event Handler watcher. The decoy folders are positioned at the beginning of each directory to be first intercepted by the ransomware. These files contain misleading information about credentials or even IP addresses. They provide an efficient detection mechanism. However, it is dependent on Find First File & Find Next File functions used in Windows OS to get the files or search directories. In addition, if an attacker used a reversed search, the victim would be alerted at the end of the encryption process leaving only the decoy files intact.

The authors in [99] propose two quality measures to evaluate decoy strategies, the first one is based on the level of deceptfullness of the method, while the second is related to the efficiency. They state that a good decoy generation strategy should blend in with the rest of the files on the file system. Thus, the

file generated will have a high probability of being selected by an attacker. Besides, a user should not access a file from those generated decoys. They proposed distinguishing decoys using statistical methods applied to the attributes of the MFT that we will analyze in-depth in Chapter 4.

3.4 Moving Target Defense (MTD)

Lee *et al.* come up with a mechanism based on MTD applied to file extension to prevent losses on the victim system, most importantly, without a performance overhead [100]. MTD increases the complexity of the attack surface. They randomly change 7 file extensions (.docx, .hwp, .pdf, .pptx, .txt, .xlsx, and.zip) over one iteration to protect them. They randomly generate a four-digit file extension using the Cryptographically Secure Pseudo-Random Number Generator (CSPRNG), then if it was not previously used, it replaces the existing extension in the registry. The experiments realized are on VMware with Windows 7 installed. The overall modification of 1000 files (extension + registry) does not exceed 3.6 seconds. However, two limitations are found in this work, a non-ergonomic work environment, as the authors stated in addition to the encryption of a specific directory regardless of the file extension. Not to mention, each file type is associated with its corresponding magic number (pdf: 25 50 44 46 2D, ppt: D0 CF 11 E0 A1 B1 1A E1, 7-zip: 37 7A BC AF 27 1C) consequently if a ransomware scans just the first couple of bytes of any file, it can get its format and encrypt it if it belongs to the whitelist of files [101].

3.5 Files Monitoring (Encryption, I/O requests)

Kharaz *et al.* present a dynamic based approach to detect ransomware by identifying any tampering of users files in a created artificial environment [102]. Their solution is built on top of Cuckoo Sandbox using Windows XP as an OS, where each sample runs for 20 minutes. Generating a different artificial user environment for each run is essential since the malware is not going to be able to fingerprint the user-generated content. The generated documents should be indistinguishable from normal ones, including valid content (headers, file archives, passwords, meaningful content), randomly generated directories with a set of sub-folders, and finally, different time attributes. Monitoring of the file system activity is performed by converting all the calls to a sequence of I/O requests and returning the file's entropy in demand. Also, they identify the common patterns for accessing the files and performing the encryption based on write and delete requests. Three main classes of attacks are identified: whether the attacker overwrites the original data by the encrypted one, creates a new encrypted file, and unlinks or deletes the original one. If more than five created files have higher entropy than the read file option, then a ransomware is detected. Furthermore, they can detect zero-day ransomware (SilentCrypt) since their mechanism is based on behavioral analysis. A limitation of their work is new ransomware variants that could shuffle the data content having a slightly modified entropy; thus, it evades detection.

Palisse *et al.* create two contributions in their work Data-Aware Defense (DaD) [103]. Unlike previous test environments based on virtualization techniques, the authors build their platform to perform the analysis using Clonezilla and Viper. At each run, a clean image of Windows 7 or 10 is loaded, and samples run for 15 minutes. This automatic analysis enables a larger scope of malware investigation, removing the possibility of potential evasion techniques used by malware in a virtual machine. Furthermore, they rely on the chi-square goodness-of-fit test (χ^2) to distinguish between encrypted (aka random data) and non-encrypted files on the system. They develop a kernel driver that captures the I/O requests and calculates the χ^2 of the file being used. If the median of the last 50 operations exceeds a predefined threshold, an alert is raised, and the thread is stopped. Another advantage is the slight overhead that does not exceed 12 μs per operation. At most, 70 MB of data was lost. Besides Shannon entropy and χ^2, Mbol *et al.* rely on Kullback-Liebler divergence to detect randomness of data, in their case, ransomware encryption [104]. They focus solely on JPEG files. They show that it is impossible to compare an encrypted versus a non-encrypted file based on Shannon entropy since the output is practically the same (7.99 versus 7.96). Although it seems to be a prominent solution, the only file type taken into consideration was JPEG. It represents less than 1% of the entirely possible infected file types [105]. Therefore, an extensive work should be done to compare at least some of the important file types before and after the encryption process.

Lee *et al.* propose an extension of the detection system that covers the files in the cloud [106]. Their technique identifies the encrypted files on the cloud before synchronizing the system and losing the actual file. They utilize the collision test estimate, the Markov test estimate, and the compression test estimate

to measure the uniformity of a specific file. They calculate these statistical values of 6 different file types (system file, document, image, source code file, executable and compressed files) before and after the encryption process. Machine learning is used to derive the entropy reference value. The decision tree is selected as the most suitable classifier to distinguish between an encrypted and a non-encrypted file. These files thus are not synchronized to the backup system. However, their solution does not tackle the core of the subject: a ransomware attack is not stopped; they relatively propose a clean system recovery. In addition, the use of machine learning for statistical tests is skeptical since the values fluctuate around specific numbers that are theoretically known. Thus, a dynamic change is not seen in these cases that require the usage of an adaptation algorithm. Furthermore, no real-time tests are made to prove the accuracy of their end-to-end process.

Agrawal *et al.* opt for sequence learning module specifically LSTM (Long Short-Term Memory) to detect ransomware [107]. They incorporate attention to learn ransomware sequence known as Attended Recent Inputs (ARI). Ransomware has a significant repetition of small local patterns: the encryption process. They introduce recent input attention within a larger cell. Their dataset consists of 12,500 sequences of ransomware and benign executables for Windows OS. Their proposed algorithm ARI-LSTM outperforms normal LSTM.

3.6 Hardware Performance Counters (HPC)

Alam *et al.* present RAPPPER a two-step mechanism based on unsupervised learning to flag malicious activity. RAPPPER relies on Artificial Neural Network and Fast Fourier Transformation [108]. The hardware events selected for the study are instruction, cache-references, cache-misses, branches, and branch-misses. An observation of the hardware performance counter is done so that the tool learns the normal behavior of the system that is going to be fed to the learning algorithm for further inspection. As a final step, Fast Fourier Transformation (FFT) is applied to understand the repeatability of data over time. The experiments are carried out in a Linux Sandbox environment, and a precise threshold is set to differentiate between malicious ransomware behavior and benign ones. RAPPER can flag ransomware 4 seconds from its launch. No similar approach is proposed for the Windows OS; therefore, it is kept as an idea for defending victims from potential attacks.

3.7 Multiple Stage/ IOC (indicators of compromise)

Chew *et al.* propose a behavioral-based approach to detect ransomware to thwart its malicious intent [109]. Their work is based on multiple malware characteristics that represent indicators of compromise. These indicators are based on monitoring file changes by checking the added extensions to the encrypted files. Besides, an increased file entropy indicates possible encrypted data. Decoy files and Access Control List (ACL) Authentication help flagging ransomware if decoys are altered or unauthorized modification, deletion of a specific folder is noticed. Five seconds interval is maintained to increment the counter of the comprised action performed by the ransomware. The authors use SigCheck to check whether the file format has also been encrypted or not to resolve the high entropy problem of zip files and DLLs. Notwithstanding, ransomware authors are currently sparing the encryption of file headers, so the false positive rate increases. Windows 8.1 running on Virtual Box is used for the experiments. Results are satisfactory for all the ransomware families except Petya that triggered a Bluescreen of Death and encrypted the Master Boot Record.

Kharraz *et al.* studied 15 different ransomware families released from 2006 until 2014 [110]. They state that initial attacks were not sophisticated since they used scare tactics rather than encrypting the file system having irreversible actions. Experiments are performed in a controlled environment using Cuckoo sandbox. Their analysis is divided into three parts. For the file system activity, the authors developed a minifilter driver to capture the I/O request to perform the analysis afterward. It is deployed in the kernel mode to avoid being altered by the ransomware. Then, they look into the encryption mechanism, searching for standard Windows API calls and libraries used to encrypt a file on the disk (CryptoAPI). The deletion mechanism is also taken into account since 35% of the samples did not perform any encryption mechanism. Some samples altered the master boot record (MBR) and made persistent screen locks. The authors employ multiple mitigation strategies. They consisted of monitoring API calls, the file system activity (creation, deletion, or encryption of files), and finally using decoy resources

that should not have been altered normally by a user. All these elements propose an additional level of defense against crypto attacks.

Similarly, Scaif *et al.* develop a detection mechanism based on a set of behavior indicators [29]. Their solution relies on monitoring changes to the magic numbers (it corresponds to the type of the data stored) of the files, hash similarity measurements taken before and after a modification process, and Shannon entropy, which increases with encrypted information. As for secondary indicators, they checked the numbers of actions taken to read/write/delete files. They opt for the union of these indicators to achieve improved results than using each indicator separately. Cuckoo sandbox is also used for the experiments. In the worst-case scenario, 30 files are encrypted before an alarm is raised, and the process is stopped. In the median case, only 0.2% of the files are lost. Continella *et al.* present ShieldFS, a file system minifilter driver, that protects users from ransomware attacks [111]. The authors analyze I/O request packets for benign software and ransomware to set an initial detection threshold that indicates an ongoing attack. They check if both software interact with the file system in a like manner or not by taking into consideration the process level activity as well as the system activity. The approach is based on portraying the habits of normal users, including the entropy of write operations, the frequency of read, write, and folder-listing operations, dispersion of per-file writes, the fraction of files renamed, and the file-type usage statistics. Moreover, they scan the memory of processes looking for cryptographic primitives. Random forest is applied to the features presented above gathered during intervals. These intervals are defined as the fraction of files accessed by the monitored process. Furthermore, ShieldFS proposes a remediation aspect that shadows the original file if a malicious behavior is suspected. A drawback of ShieldFS is the difficulty in distinguishing JPEG files from encrypted files relying solely on entropy, as discussed in [104]. Besides, if a ransomware equally distributes the tasks on different processes using multithreading techniques, some files are going to be lost before detecting the malware.

The authors in [112] presented new variants of ransomware attacks that can go unnoticed. For instance, writing the encrypted data in an SQL database, then deleting all the files. Multithreading attacks for reading, writing, and removing files to maintain a low variation between the entropy of the data read and written. A set of features is added to block such attacks (file attributes, path diversity, rate of creation, modification, size, and mime change). They perform their experiments on real machines that detect all the ransomware without losing more than 20 files.

3.8 Keys Backup

Lee *et al.* present a prevention mechanism based on the encryption keys used to restore the data after the encryption process [113]. They assume that ransomware authors rely on the Microsoft CNG library to import or generate encryption tools. They develop their ransomware and test the effectiveness of their solution. They are indeed able to retrieve the keys on a Windows 7 machine. However, if the malicious software has a built-in cryptographic function or uses Microsoft CryptoAPI, no keys are backed up.

Kolodenker *et al.* propose PayBreak, a reactive solution that saves the information related to the symmetric keys generated to decrypt the files locked after the infection process [25]. Their proactive solution relies on a key escrow that stores the encryption keys securely, where only the user has exclusive access. Windows 7 is the target machine running on Cuckoo SandBox. Paybreaks consists of three major components. The crypto function hooking in CryptEncrypt to export the symmetric keys via CryptExport used or created by Microsoft's Crypto APIs, then the control is returned to the application. Further hooks are required to get additional attributes such as the initialization vector and cipher mode. As for Crypto++ , the memory of each executable is scanned for function signatures, and if a match is found, a hook is placed. Then, the key vault is used to store the symmetric encryption in an append-only file protected by a private key created by the user using the same hybrid cryptosystem as the ransomware. Finally, the file recovery is achieved by testing multiple decryption schemes at different offsets since the encrypted file contains metadata of the ransomware. Twelve out of twenty families are successfully defeated. The rest of the samples could be identified by hooking to various statically linked libraries used during the encryption process.

Articles	Type	Approaches		Tested	Detection/Protection Mechanism
		Static	Dynamic	Solution	
[85]	Network Analysis	-	✓	✓	DNS requests generated by DGA detection via Markov chains
[43, 87, 89, 91]	Network Analysis	-	✓	✓	Flagging suspicious communication via machine learning
[92]	Network Analysis	-	✓	✓	SDN mechanism capable of flagging malicious POST requests
[94, 114]	Network Honeypot	-	✓	X	Proposition of using Network Honeypot
[96]	System Honeypot	✓	✓	X	Novel mechanism to make efficient decoy files
[97, 98]	System Honeypot	-	✓	X ✓	Detection via monitoring honeypot folders
[100]	MTD	-	✓	✓	MTD applied to file extension to prevent the encryption process
[102]	Files Monitoring	-	✓	✓	Monitoring I/O requests and files' Shannon entropy for ransomware detection
[103]	Files Monitoring	-	✓	✓	Chi-squared test to check encrypted files
[105]	Files Monitoring	-	✓	✓	Kullback-Liebler divergence to locate JPEG encrypted files
[106]	Files Monitoring	-	✓	✓	Ransomware detection (in the backup system) by applying ML on file format and entropy
[108]	HPC	-	✓	✓	ANN applied on cache events to flag ransomware
[109]	Multiple IOC	-	✓	✓	Monitoring file changes and entropy, manipulation of decoy files to detect ransomware
[111]	Multiple IOC	-	✓	✓	ML applied to the entropy of write operations, the frequency of read, write, and folder-listing operations, dispersion of per-file writes, the fraction of files renamed, and the file-type usage statistics + files recovery
[110]	Multiple IOC	-	✓	✓	Monitoring I/O request and changes in the MFT to detect ransomware
[29, 112]	Multiple IOC	-	✓	✓	File attributes, modification, features used to flag ransomware
[25, 113]	Keys Backup	-	✓	✓	Hooking to Microsoft cryptographic function to restore the keys and decrypt the files

Table 2.3: Ransomware Detection Mechanisms for the Destruction Phase P3.

4 P4: Dealing

The final stage in the ransomware attack consists of an exchange between the attacker and the victim. It is the most critical phase of the intrusion. Indeed, the cyber attacker displays a ransom note indicating the steps he/she has to follow for the payment to receive the decryption keys. State-of-the-art papers delve into extracting and clustering the addresses of ransomware found in the blockchain. The goal is to monitor the bitcoin flow, visualize the transactions, and provide an estimate of the infected people. The methods presented in the dealing phase are summarized in Table 2.4.

4.1 Bitcoin Tracking

Spagnuolo *et al.* developed BitIodine, a framework that helps to track the irreversible transactions publically available on the blockchain [115]. Correlating the information extracted from the blockchain and its metadata allows an accurate description of the cryptocurrency flow between two addresses. BitIodine scheme relies on parsing the blocks and transactions found in the .bitcoin folder and exporting them into a database. Then, a cluster of addresses is based on the multi-input transactions (assuming multiple input addresses belong to the same wallet) and change (the "unspent" output of a transaction delivered back to the user). A set of scrapers crawl the web, collecting information associated with the bitcoin addresses like usernames, physical coins, scammers, and shareholders. Finally, the transaction and user graphs are generated corresponding to the assembled information above. The classifier labels each cluster to its corresponding potential owner. Spagnuolo *et al.* investigate CryptoLocker using BitIodine. The tool they developed can gather 1467 CryptoLocker addresses belonging to 12 clusters. This step is carried out by analyzing flows of 0.5, 2, or 10 BTC (bitcoin), the ransom demanded by the attackers extracted from [116, 117]. Two key elements are presented in BitIodine. It is possible to track and identify ransomware based on the ransom amount, and multiple clusters can represent the same family. Different ransomware families can be studied based on these characteristics. Additionally, new unidentified clusters can be analyzed to check the possibility of classifying them as ransomware.

Kuzuno and Karma propose as well an analytical process environment for bitcoin [118]. It is divided into four steps. Initially, their mechanism has to find the bitcoin address (target is already known or search for example via the amount spent and the date). Then, the indexer collects and stores each transaction ID and Block ID in the private database of the authors. Next, the visualizer displays the relation between the collected addresses (transactions made). Finally, the clustering process associates a known address with another that might belong to the same wallet operator. Applying this process to Cryptolocker's case, two addresses quickly stood out, since they received 2.0 BTC from many other addresses.

Similarly, Huang *et al.* trace financial transactions related to ransomware [6]. The authors collect the seed addresses (ransom addresses) from real victims or by executing the ransomware. For the real victims, the authors check public forums such as Bleeping Computer. Once they find the screenshots of ransom notes, they perform image and or text analysis. As for the experiments carried out to by synthetic victims (authors executing the ransomware), they are executed for 20 min on four independent platforms VmRay, VMware-based sandbox, Cuckoo, and Windows XP on a bare-metal machine. To be able to collect more ransom addresses, clustering and micropayments methods are used. Clustering by co-spending helps to expend the range of "malicious addresses" if two wallet addresses are used as the input to the same transaction. Augmenting clustering with micropayments consists of paying 0.001 bitcoins to the ransom address and observe bitcoins flow. To cover the limitations of those two methods (for example, micropayment did not result in subsequent bitcoin movement), the authors incorporate the timing of payments.

Harlev *et al.* focus on predicting if a previously unidentified cluster belongs to one of the following predeïňĄned categories: exchange, gambling, ransomware, etc [119]. To accomplish this step, the authors apply machine learning algorithms (k-nearest neighbors, random forests, decision tree, extra trees) on the bitcoin dataset provided by Chainalysis, a bitcoin analysis company [120]. Significant features are extracted and kept (timestamp of the transaction, the amount of BTC received/sent, total BTC amount sent to a given cluster, equivalent USD amount at the point in time). In a like manner, Akcora *et al.* detect new ransomware addresses using topological data analysis (TDA) and machine learning [121]. TDA helps to extract hidden patterns fundamental elements to distinguish a ransomware transaction in the blockchain (income, number of addresses, and unique addresses, neighbors).

The model can predict new ransomware families with 27.53 false positives for each true positive.

Articles	Type	Approaches		Tested	Tracking Mechanism
		Static	Dynamic	Solution	
[115]	Bitcoin Payment Tracking	-	✓	✓	Clustering and visualizing bitcoin addresses and flows based on the multi-input transactions and change heuristics
[118]	Bitcoin Payment Tracking	-	✓	✓	Ransom addresses traced, then bitcoin flow visualized; lastly, addresses are clustered.
[6]	Bitcoin Payment Tracking	-	✓	✓	Ransom addresses traced via clustering by co-spending augmented with micropayments
[119]	Bitcoin Payment Tracking	-	✓	✓	Categorize yet-unidentified clusters via supervised ML
[121]	Bitcoin Payment Tracking	-	✓	✓	Extract features related to ransomware Bitcoin to detect new addresses associated with known ransomware families or new ones.

Table 2.4: Ransomware Detection Mechanisms for the Dealing Phase P4.

5 Network Intrusion Detection System (NIDS) and Datasets

The second contribution of the thesis tackles network communication to detect ransomware. Hence, this section introduces the types of NIDS found in the literature and the datasets used as a baseline comparison or for benchmark testing. We conclude by the need for generating our own dataset presenting ransomware network traffic.

5.1 Types of NIDS

Signature & divergence-based (or anomaly-based) NIDS represent a device or software application that monitors a network or system for malicious activity or policy violations. It detects, if possible, any attempt to:

- gain unauthorized access to the system
- compromise network availability
- install malicious applications

Signature-based detection compares the current network traffic to previously well-known malware patterns called the signature. If any matched signature is noticed, the administrator is alerted to take the corresponding measures. Howbeit, this strategy is limited since it is not capable of detecting new malware. Another drawback is the necessity to maintain the malware database up to date and synchronized.

Divergence-based NIDS intricate architecture characterizes this technique. Henceforth, it can expose new threats or attempts to compromise the system. It sets a baseline for network behavior, enabling it to identify any abnormality. Any deviation is going to alert the administrator if it exceeds a threshold. Different types of anomalies exist:

- point anomaly
- contextual anomaly

- collective anomaly

Therefore, an administrator has to define anomaly's scope and set various rules based upon the output of the analysis [122].

5.2 Datasets

Network intrusion detection systems are widely researched and adopted to protect individuals and companies from being attacked. When referring to machine learning in a network environment, an administrator's primary concern is the input data that is fed to the algorithm to classify or distinguish between various classes.

The evolution of the different datasets is presented below. The Knowledge Discovery and Data Mining (KDD) data have been extensively used to perform those validation tests. The dataset includes four major categories of attacks: probing, Denial-of-Service (DoS), user-to-root (U2R), and remote-to-local (R2L) attacks. A set of pattern recognition and machine learning algorithms is carried out on the KDD dataset to check the improvement of the performance [123].

Since the initial KDD dataset contains 41 features, Olusola *et al.* select relevant ones using rough set mathematical tools and discretization based on entropy [124]. They identify the most relevant features for each type of attack: the wrong fragment for teardrop attack whereas source bytes for the back attack. Overall, KDD dataset analysis and feature selection was a widespread phenomenon in the first decade of the 21st century [125–127]. However, it has some problems regarding the distribution of the attacks and redundant records. Therefore, to solve these issues, Tavallaee *et al.* create the NSL-KDD dataset [128]. Researchers then shifted their study on the modified version of the KDD dataset [129–131]. Real traffic traces are available in WITS (*https://wand.net.nz/wits/*), MAWI (*https://mawi.wide.ad.jp/mawi/*), CAIDA (*http://www.caida.org/data/*) datasets and others [132].

Ring *et al.* focused literature survey on network-based intrusion detection datasets presents a polished study about these records [133]. They shed light on five key properties to acquire or generate a proper dataset. These elements are:

- general information: year of traffic creation, publicly available, normal/attack traffic.
- nature of the data: metadata, format, anonymity.
- data volume: count, duration.
- recording environment: kind of traffic, type of network, complete network.
- evaluation: predefined splits, balanced, labeled.

Nonetheless, these datasets do not contain ransomware traffic except *https://www.malware-traffic-analysis.net/* (1) and *https://www.virustotal.com/* (2). (1) provides full capture of network communication, yet, the sequence of the events performed on the machine of the victims is missing and (2) is not freely available. Therefore, we proceed by generating our ransomware network traffic.

Stratosphere IPS dataset is used for normal captures only [134]. It contains recent normal traffic captured from 2013 until 2017. An additional information is the description of the behavior captured, making the labeling process feasible. Moreover, in the project's malware section, it contains ransomware packet capture: it is a means of comparison between the ransomware traces provided by their dataset and our own generated in a bare-metal platform explained in the following sections.

6 Conclusion

This chapter displays the contributions to combat ransomware from 2012 up to date and gives an insight into NIDS. The state of the art section is submitted to ACM Computing Surveys. An extensive work is achieved in the literature covering all the aspects of a ransomware attack from the delivery until the dealing phase. Blocking ransomware in the delivery phase is not a trivial task. The user is held responsible (to some extent) at this stage for performing the malicious intent of the attacker, even though done unknowingly/unwillingly. Therefore, raising awareness reduces the potential risk of being infected

by a previously unknown malware. API calls are thoroughly studied in the deployment phase. They are extensively analyzed in all shapes and forms, including their frequency and the n-grams sequence implemented in the code of the malware. Thus, we focused on the destruction step to provide a prototype for ransomware detection as soon as it is installed and it has set the required environment on the victim's machine. No previous studies have been conducted on the traversal of the malware; consequently, we present our findings in the second part of the thesis. Besides, we analyze the network communication and the ransom notes to check the chronology of ransomware events. Finally, we show the presence of a new threat Doxware that could render victims' data unavailable to them yet shared with third parties (for example, competitors). The dealing phase represents the tracking of the ransom amount paid by the victim. Indeed, it enables labeling and clustering some bitcoin addresses. However, we solely present future perspectives in this part rather than developing a ransom tracking methodology.

3 Work Environment, Ransomware Samples and Evasion

In the previously developed countermeasures, ransomware binaries are mainly executed on virtual machines. Indeed, VMs are chosen for their simplicity enabling a large scale execution of numerous samples. Automated dynamic analysis is the adopted approach among researchers. It is impossible to inspect manually all the executables found in the open-source databases. Nevertheless, many malicious samples check the presence of such environments and try to evade detection. Malware authors use fingerprinting techniques to detect the presence of a controlled environment. Fingerprinting encompasses checking specific artifacts known in advance by the attackers. It includes but is not limited to registry keys, background processes, function hooks, or IP addresses. We present in this section the ready for use tools proposed in the literature for malware detection and show their limitations. However, many of the tested samples are not running correctly: no encryption is recorded. Indeed, some ransomware samples stop their execution if they are being run on VMs. One approach to detect malware evasion techniques is to execute samples on different platforms and observe the behavioral changes. Therefore, we present in this part, the available tools for malware analysis, and we explain our choices in adopting the bare-metal one.

1 Dynamic Analysis (DA) Tools

The dynamic analysis of ransomware is necessary to save the required logs (API calls, network communication) for a post mortem examination. Three analysis tools are accessible to scientists to achieve this step and are presented below.

1.1 Virtual Machine (VM)

A virtual machine provides "an efficient, isolated replica of a computer's system environment" as defined by Goldberg in [135]. VMs are limited to the same architecture as the host machine [136]. Resetting the VM to a clean state requires taking a snapshot before the execution of the malware, which is much faster than restoring a bare-metal system. However, it is always be possible for a malware to detect a virtualized environment as the authors state in [137]. Another threat occurs if a malware breaks out the virtual machine to execute arbitrary code on the host with the privileges of the hypervisor process, also known as virtual machine escape [138,139]. Common virtual machines are Cuckoo [140], VirtualBox [141] or VMware [142].

1.2 Hypervisor/Virtual Machine Monitor (VMM)

A VMM is a process that creates and runs virtual machines. Hypervisors are classified into two types.

- **Type-1 or bare-metal hypervisors** run directly on the host's hardware to control the hardware and to manage guest operating systems (for example, Microsoft Hyper-V and Xen).
- **Type-2 or hosted hypervisors** run on a conventional operating system like any other software (for example, VMs seen in section 1.1).

VMMs have relatively low overhead and are generally more transparent than other virtualization mechanisms [143]. However, VMMs are not immune to hyperjacking, where the attacker takes control over the hypervisor [144].

1.3 Bare-Metal (BM)

It is a malware analysis system that is indistinguishable from a real host [145]. The operating system runs on actual hardware. The particularities of a BM platform are the absence of virtualization and the efficient environment restore [137]. The main advantage is the inability to differentiate between a user host and a bare-metal platform. To date, BM is the best tool available for malware studies. However, some difficulties have arisen, for example, the system-restore technique that may require a reboot after each run is time and resource-consuming. It is not scalable since each sample is executed on a distinct machine. Some bare-metal analysis platforms have been proposed in the literature that address malware evasion methods [137, 146, 147].

2 What About Evasion?

The percentage of Reveton samples that uses anti-debugging techniques is 74.8%, and anti-VM techniques is 62.8% acquired from of the investigations carried out by Chen *et al.* [148]. This experiment shows that more than 50% of ransomware samples do not encrypt the files or perform their malicious intent if executed on a VM. We opt to perform our experiments using a bare-metal platform previously developed by Palisse *et al.* called MoM [17].

2.1 Chosen Bare-Metal Platform: MoM

MoM uses three open-source tools: Viper, Clonezilla, and Scrapy to perform the automated analysis [17]. It consists of a Linux master server and five test machines: 2 Windows 7 64 bits, 2 Windows 7 32 bits, and 1 Windows 10 64 bits. Each system image has a realistic environment, including files downloaded from the Digital Corpora [149]. A variety of file extensions is used, having in total more than 10k user files.

The platform works as follows.

1. The master machine hosts a Clonezilla server with a configured PXE allowing a network boot.
2. Each run is defined by restoring a clean/uninfected Windows image.
3. At the end of the restore phase, the client is configured to reboot on its local disk.
4. At startup, a python script is downloaded from the master machine. The script contains the required steps to collect the data or perform specific actions, including malware download.
5. Finally, the data is transmitted via FTP to the master server, and a clean reboot is set up for the next analysis.

Initially, we test the status of each ransomware sample, checking if it performs the encryption process or not. After running the sample, a hash function checks the integrity of the user's files. If it does not match the reference value, the sample is considered active. Then, for all the active samples, we define the actions that should be taken by the system to collect the network and system logs (for example, launching Wireshark, executing the kernel driver).

2.2 Ransomware Samples

The latest ransomware family names are collected from online forums, recently updated malware databases and scientific papers. Then, a crawler downloads the ransomware samples from two databases Virus Share [150] and Malwaredb.Malekal [151] (currently down), finally, it is executed on Windows 7 32 bits machines for a period of 2 to 3 minutes. A dump corresponding to this malware behavior is saved for further post mortem analysis. For scalability reasons, parallel machines are used to perform the tests as well as an improved disk image distribution. The discrepancy in ransomware samples throughout the thesis is due in part to inactive ransomware families after an epsilon time of their release.

For example, in our latest campaign launched in January-February 2019, 1054 ransomware were executed on Windows 7 OS (Table 3.1). **Howbeit, 100 ransomware executables are kept for the analysis phase since they were active** (encrypted the files of the victim). An increase of inactive samples is noticed through the experiments carried out by researchers. In the best case scenario, 76.8%

of samples are inactive ([152]). On average, 82.67% of binaries are inactive.

The reasons for **inactivity** of a given ransomware executable can be one of the following.

- The C&C or the external IP address/domain name is down.
- Inadequate work environment (missing DLLs, unmatched Windows version).
- Ransomware avoid being executed on particular environments (for example, GandCrab avoids infecting Syrian countries).
- Ransomware suspects being monitored/analyzed (VM or debugging tools).
- The footprint of the test environment is already registered in the attacker's server.

Family	Number of Downloaded Samples	Number of Working Samples
Teslacrypt	334	11
Yakes	252	2
Shade	139	56
Cerber	95	11
Deshacop	62	2
Zerber	57	5
TorrenLocker	38	0
Bitman	27	12
Razy	26	0
Locky	24	1
Total	1054	100
Losses		90.54%

Table 3.1: An overview of the active ransomware families (total of 100 active samples from 1054 tested), ranked in descending order according to their samples number.

3 Conclusion

The investigation of a considerable number of malware samples provides a better insight into its behavior. Hence, the dynamic analysis draws the workflow of any executed process. The dynamic analysis helps to provide a targeted solution for ransomware threats to detect it at the early stages. The bare-metal platform has the highest probability of achieving this exhaustive analysis; therefore, it is chosen for our experiments. A further investigation is required to indicate the appropriate elements that support the creation of a realistic environment. They are discussed in the future work, along with the reason for the inactivity of more than 80% of the executables. We proceed by presenting in the second part of the thesis, the developed countermeasures in the destruction phase. The first technique is based on the system level, the second one tackles the network part, finally, we present the plausible threats encountered by a Doxware attack.

Part II

Contributions

1 File System Based Solution for Ransomware Detection

This Chapter presents a system-based ransomware countermeasure to detect malicious threads carried out with Aurélien Palisse and Benjamin Bouget. It is a new mechanism that does not rely on previously used metrics in the literature to detect ransomware such as Shannon's entropy or system calls. The per-thread *file system traversal* is sufficient to highlight the malicious behaviors. To the best of our knowledge, no previous study has been conducted in this area. The ransomware collection used in our experiments contains more than 700 active examples of ransomware, that are analyzed in our bare-metal sandbox environment (section 2.1, Chapter 3, Part I).

Three methods are proposed to flag or cluster malicious ransomware threads. The first one is based on a decoy score, that is improved by a ML approach (2nd method). Finally, the possibility of identifying the ransomware family rather than maintaining just a binary classification (malicious or not) is proposed using graphs (3d method).

1 Need for Dynamic Analysis

Any prevention tool must be able to classify a given binary as suspicious or not. For this purpose, static or dynamic analysis can be used. Nevertheless, modern malware employs stealthy techniques to remain undetected on diverse systems making it difficult to analyze. Static analysis may quickly reveal the presence of obfuscation uncovering a glimpse of beneficial information to the analyst. Malware usually examines the environment it is operating in. It is equipped with the possibility of altering its behavior if an ongoing dynamic analysis is detected.

A new defense mechanism against ransomware is developed to avoid ransomware evasion. Our approach does not rely on code examination, code structure, or access to the system's calls. A module that analyses suspicious software behavior is designed while accessing the file system. All ransomware of our provided collection evaluate users' data by traversing all folders except for some specific ones. This traversal represents, thus, a signature of their original code. For this reason, execution traces and, in particular, traversal's order are collected.

In the following sections, different models to detect ransomware's attacks are presented. The novelty is the need for exclusively one information: *file system traversal*. Moreover, an accurate classification can be drawn with these observations. Thus, any binary can be flagged as malicious or not.

2 Decoy score

The first suggested prevention mechanism relies on the idea that ransomware scans specific files and folders that enable not only their detection but also categorization [96]. Some directories and files are rarely visited by the user or by one of the system's regular tools and thus can be considered as a trap. If a software manipulates these files, it can indicate an illegal and unwanted access. They are referred to as Decoy Folders. Crypto-ransomware does not attack the files that allow the proper functioning of the machine. Indeed, the user must be able to use it to pay the ransom.

2.1 Whitelists and Blacklists

Nowadays, most of the ransomware families available in multiple online databases implement a straightforward exploration of the file system. In other words, they explore the file system with well-known algorithms: depth-first or breadth-first search. Nevertheless, slight differences between them can be seen. Whitelisting and blacklisting strategies are widely adopted in the literature in several areas. Some authors rely on these techniques to propose crawling approaches for academic documents search engines as in [153], while others implement it as a control policy for applications [154]. The core remains the same regardless of the use case.

Ransomware's authors embed whitelists and blacklists of folders into malicious binaries. They represent two ways of filtering access to folders.

- **Whitelists** represent a set of folders that can be accessed by ransomware. For example, a Cerber sample especially targets multimedia folders (*e.g.*, C:/program files (x86)/steam/).
- **Blacklists** are the reverse of whitelists: folders that are omitted and denied from ransomware's path. For instance, C:/Windows/ system folders are avoided by malware to let the machine run normally.

The environment variables allow the attacker to attack users' documents directly rather than beginning the exploration from the root as most of the analyzed ransomware does. The above information is used for ransomware detection.

2.2 Proposed Algorithm

Our suggested solution checks if a thread passes in specific folders. If so, it marks them and then increments the decoy folder counter. In case a threshold is reached, the thread is recognized as malicious. The list of decoy folders that are used in our experiments are presented in Figure 1.1. The probability of having a benign thread that passes through at least 3 of these decoy folders is very low. Therefore, the counter that keeps track of the accessed repositories enables flagging suspicious behavior.

Recycle_bin	;	(C:\$Recycle.Bin)
Python	;	(C:\Python)
Perf_log	;	(C:\PerfLogs)
Prog_data	;	(C:\Prog_data)
Prog_files	;	(C:\Prog_files)

Figure 1.1: The list of decoy folders used to compute the per-thread score.

The main algorithm is the following:

Algorithm 1 Ransomware Detection

```
1:  procedure
2:      def detect_suspicious_behavior(thread, threshold):
3:      label_array ← {Decoy Folders}
4:      score_array ← {false, false, false, false, false}
5:      for path ∈ thread.paths do
6:          if path ∈ label_array then
7:              index ← label_array.index(path)
8:              score_array[index] ← true
9:      if evaluate_score(score_array) ⩾ threshold then
10:         return is_suspicious
11:     return is_not_suspicious
```

Algorithm 1 checks if a thread passes through specific folders. If so, it marks them and then increments the decoy folder counter. The number of decoy folders taken into consideration depends on the analyzed system. In the test environment, five decoys are sufficient to detect ransomware's activity.

Since five different decoy folders are considered for the experiments, the initial score array composed of five elements is set to false (on line 4 of **Algorithm 1**). The final score is normalized (*i.e.*, divided by the number of "decoy folders" that is added in the beginning). In case the threshold is reached, the thread is recognized as malicious. It is unlikely that a regular thread passes through at least 3 of these decoy folders. This is why the threshold is set to 0.6. If it is lower, some ransomware are not detected (for example, if a ransomware directly modifies the file, it does not go through the Recycle Bin folder). Also, many benign threads are going to be flagged. If it is higher, others also escape detection.

3 From Decoy Score to Supervised Learning

In this section, an improvement of the previous classification is made by using a supervised approach. Thread level granularity is maintained throughout the whole experiment. In addition to the access to previously mentioned decoy folders in Figure 1.1, other features are taken into consideration to perform supervised learning on the collected information. The features used to train the classifiers are presented in Table 1.1.

Paths_total	Total number of explored paths
Time_total	The file systems traversal duration
{Decoy_folder}_paths	The number of subfolders explored in the current decoy folder (Updated Decoy Folders: Recycle_bin, Perf_log, Windows, Python, Prog_data, Prog_files, Recovery)
{Decoy_folder}_time	The timestamp of the first subfolder explored in the current decoy folder

Table 1.1: The list of features used to train the classifiers.

This method is not limited to the file system's traversal, but overall and per decoy folder velocity is taken into consideration. It is not sufficient that a thread explores only decoy folders to be marked as malicious, the time spent in each decoy folder and in total is crucial and needs to be considered for a better classification.

3.1 Learning Phase

The holdout method is used to evaluate different supervised machine learning models. In order to apply those algorithms and evaluate their performance, Python is used [155]. More specifically, the scikit-learn library since it represents an efficient tool for data mining and data analysis, in our case classification and clustering [156]. Avoiding overfitting on the non-malicious threads is crucial for better results, which enables us to generalize our model.

The training set consists of partitioned data across all families of ransomware and different types of benign processes. It is challenging to have the same number of working samples belonging to each family type since ransomware executables are only active during an attack cycle. For example, the WannaCry attack lasted from the 12^{th} till the 15^{th} of May 2017. Multiple classifiers are trained: k-nearest neighbors [157], decision tree [158] and random forest [159].

- **k-nearest neighbors (k-NN):** It is a type of lazy learning (instance-based learning) where the generalization of the training data is delayed until the algorithm receives a new input. The hypotheses complexity increases with new data since it is directly constructed from the training instances. The classification phase consists of choosing a predefined constant k. The unseen record represented by a vector V is classified based on the nearest distance between V and the k neighbors. Euclidean distance is used for continuous variables, whereas Hamming distance is adopted for text classification. k-NN requires a large dataset, and it is best used when continuously updated/queried. The large space requirement represented by storing the whole dataset in memory can be reduced by having fewer attributes or eliminating similar records.

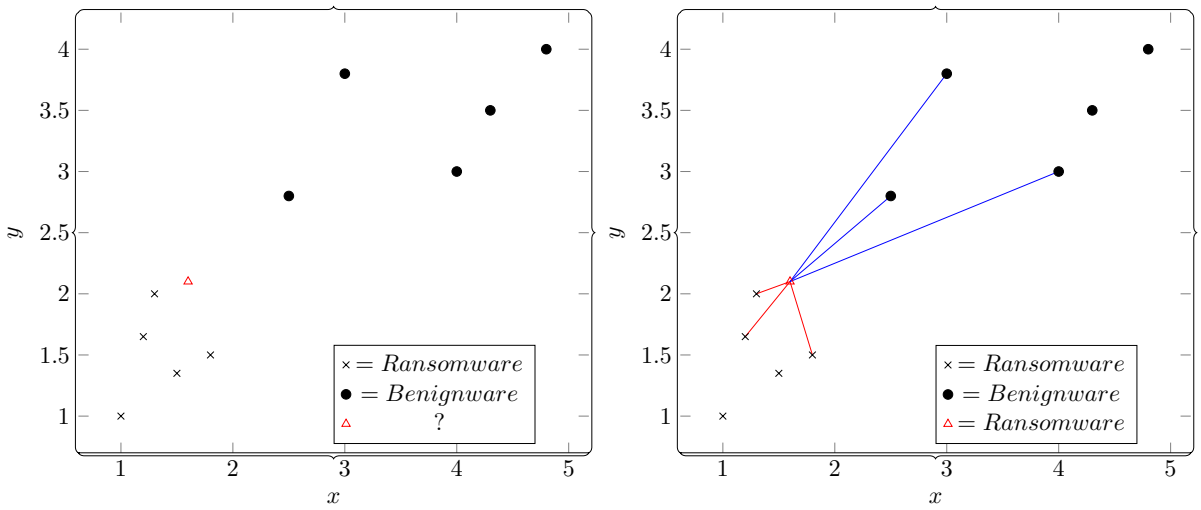

Figure 1.2: k-NN algorithm.

Figure 1.2 displays scattered points representing ransomware (symbolized by x) and benignware (symbolized by •) samples. Considering that the chosen number of neighbors k=3, k-NN calculates the Euclidean distance between the unknown sample (represented by the red triangle) and the rest of the points. Since the triangle is the closest to three samples belonging to ransomware, it is classified as ransomware as well.

- **Decision tree (DT):** The top-down approach is used to build decision trees by choosing the variable that provides the best split. Multiple DT are provided in the literature that differ on the metric chosen to perform the split. For example, the classification and regression trees (**CART**) method uses *Gini Impurity* to perform the best split. *Gini Impurity* is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. Gini Impurity is calculated as

$$G = \sum_{i=1}^{C} p(i) \cdot (1 - p(i))$$

In oure case, we have two possible classes ransomware (50% of the records) and benignware (the other 50% of the records): p(ransomware)=p(benignware)=0.5.

$$\begin{aligned} G &= p(ransomware) \cdot (1 - p(ransomware)) + p(benignware) \cdot (1 - p(benignware)) \\ &= 0.5 \cdot (1 - 0.5) + 0.5 \cdot (1 - 0.5) \\ &= 0.5 \end{aligned}$$

Considering the blue split in Figure 1.3, since the left branch contains only ransomware and the right one benignware, G_{left}=G_{right}=0. A Gini Impurity of 0 is the lowest and best possible impurity. It can only be achieved when everything is the same class. The best split is chosen by maximizing the Gini Gain, which is calculated by subtracting the weighted impurities of the branches from the original impurity.
Whereas **C4.5** uses relies on *information gain* (IG). IG represents the smallest number of bits, on average per symbol, needed to transmit a stream of symbols representing the values of a variable X. Unlike k-NN, DT requires training the model to perform the required classification, which is time and resource consuming. It might be prone to overfitting where the trained model is too closely fit to a limited set of data points.

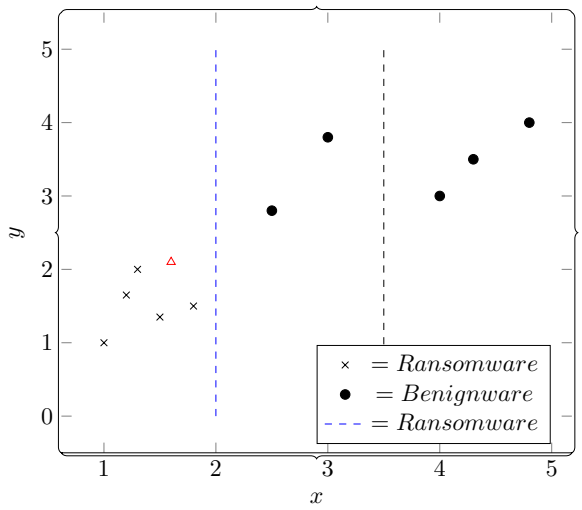

Figure 1.3: DT algorithm.

- **Random forest (RF):** RF is a versatile machine learning method applied to different types of data. The key concepts that characterize RF are the sampling of training data points when building trees and the random subsets of features considered when splitting nodes. The correlation of multiple trees outperforms the output of a singular one. The classification is based on combining multiple DT using *bagging*. Bagging helps to reduce the variance of a decision tree. It consists of selecting random samples with the replacement of the training set and fitting trees to these samples. Besides, a random subset of the features is selected at each candidate split in the learning process. The final prediction is made based on a majority voting.

4 File System Traversal Velocity

Another behavioral property is additionally investigated, the execution time of a family. Indeed, the samples issued from the same families have similar patterns. Each payload can be packed or obfuscated individually, but the system impact remains the same for a particular family, except for new versions.

5 Graph Similarity

Each ransomware's file system traversal can be compared to others to know if they belong to the same family or share some code concerning the exploration of the path. It saves time for the reverse-engineering task to know at which family belongs to a sample. As a first step, a directed graph of the explored folders for each sample is built, as displayed in Figure 1.4. Then, the similarity matrix corresponding to the ransomware dataset is computed, as seen in Figure 1.5. Finally, a classification of the samples is done based on the similarity matrix using a hierarchical clustering technique, which results in a dendrogram displayed in Figure 1.6. Hierarchical clustering is used since it merges clusters together which represents the ransomware samples belonging to the same family.

5.1 Hierarchical Graph

The machines used to collect the data have the same configuration and similar hardware. Each line of raw data used in the experiment corresponds to the nature of the operation on the file system (read file and open directory), the thread `pid`, and a timestamp. This data is sufficient to trace the complete graph traversal regardless of the write operation which is not taken into consideration at this phase.

Each node represents a file system folder that has been opened by a suspicious thread. File system traversals are represented as oriented graphs (via time stamps). Edges represent the transition from parent to child folders. In order to be scalable (*i.e.*, graph size) and generic (*i.e.*, distinct Windows

installations), subgraphs are created at specific file system levels (*e.g.*, C:/Program_Files/). The number of clusterized sub-folders is stored within the graphs' edges.

A graph is defined as the tuple $G = (V, E, \mu, \nu)$ where:

- V the set of nodes
- E the set of edges
- η the set of nodes covered by a ransomware
- θ the set of paired nodes with the number of clusterized sub-folders
- $\mu : V \to \eta$
- $\nu : E \to \theta$

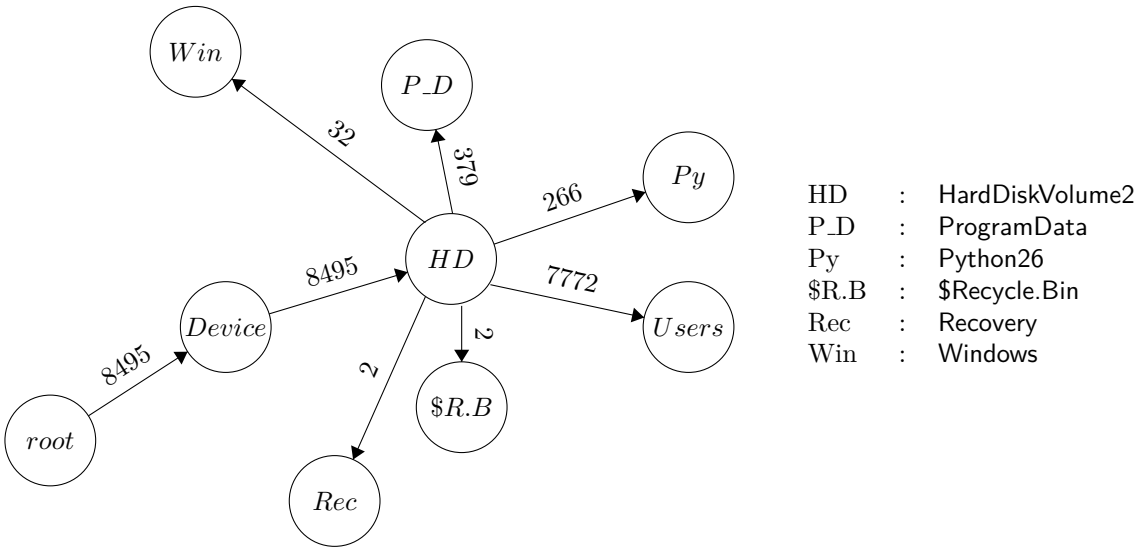

Figure 1.4: Xorist's File Traversal Subgraph G.

Xorist's subgraph is defined as $G = (V, E, \mu, \nu)$ where:

- V = {HardDiskVolume2, ProgramData, Users, Recovery, Windows, ... }
- E ={Device→ HD, HD→ Py, HD→ Users, ... }
- η = {HardDiskVolume2, ProgramData, Users, Recovery, Windows, ... }
- θ = {HD→ Users having 7772 traversals. The traversals include the sub-folders of Users like Desktop, Documents, Music, Pictures,... }
- $\mu : V \to \eta$
- $\nu : E \to \theta$

Xorist begins the exploration from the root of the Windows file system. 8495 threads pass from the root to the HardDiskVolume2 (Figure 1.4). Then, threads spread through different folders like the ProgramData and ProgramData. However, the main target remains the Users folder. More than 90% of the traversal is located in the Users folder (7772 traversals vs 266 in the Python26 repository).

5.2 Adjacency Similarity and Classification

A comparison between a given trace (*i.e.* a graph) with other graphs is needed. The cost to match two graphs is low if their structure is close. Graph similarity techniques can be classified into three main categories: edit distance/graph isomorphism, feature extraction, and iterative methods [160]. The drawback of graph isomorphism is that the algorithms are exponentially resource-consuming and, thus, not applicable to the large graphs that are of interest to us. The feature extraction approach relies on graph properties such as degree distribution, diameter, etc. This method scales well but depends on the chosen metrics; it is possible to get high similarity between two graphs with very different node-set sizes. Iterative methods are based on the fact that two nodes are similar if their neighborhoods are also similar. This latter method is chosen using an adjacency similarity algorithm.

Given two graphs $\mathrm{G}_1(\eta_1; \mathrm{E}_1)$ and $\mathrm{G}_2(\eta_2; \mathrm{E}_2)$ having the same or different number of nodes and edges, graph similarity determines the degree of similarity (a real number between 0 and 1) between these two graphs. It counts the number of edges that have the same source and target labels in both graphs. G_1 and G_2 are considered similar if the real number returned by the dedicated algorithm is close to 1, or any other threshold predefined by the user (for example, 0.96).

To compute the similarity matrix between all the graphs, Graph-tool is used [161]. It is a free framework for creating and manipulating graphs. The core of this framework is written in $C++$, which makes it fast even for large graphs. A built-in Graph-tool function computes the adjacency similarity between two graphs. The labels of vertices are used to build the adjacency matrices and thus make the comparison. The higher the score is, the higher the similarity.

Figure 1.5 presents the similarity matrix **S** between various file traversal graph of various ransomware defined by $\mathrm{R} = \{\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3, \mathrm{r}_4, \mathrm{r}_5\}$. The main diagonal of a matrix has the highest score since any ransomware sample is identical to itself. r_2 and r_4 do not share any elements concerning the traversal since their graph similarity is almost equivalent to zero, whereas r_2 and r_5 might belong to the same family since their similarity is 0.991.

$$\mathbf{S} = \begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{array} \begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_3 & r_4 & r_5 \end{array} \\ \begin{pmatrix} 1 & 0.238 & 0.896 & 0.468 & 0.991 \\ 0.238 & 1 & 0.273 & 0.001 & 0.991 \\ 0.896 & 0.273 & 1 & 0.536 & 0.423 \\ 0.468 & 0.001 & 0.536 & 1 & 0.513 \\ 0.991 & 0.991 & 0.423 & 0.513 & 1 \end{pmatrix} \end{array}$$

Figure 1.5: An example of similarity matrix **S**.

To classify the samples, unsupervised hierarchical clustering over this similarity matrix is used. A dendrogram represents the classification. It is a visual representation of the compound correlation data. The individual compounds are arranged along the bottom of the dendrogram and referred to as leaf nodes. Compound clusters are formed by joining individual compounds or existing compound clusters with the joining point referred to as a node. The leaves of the tree are the name of the classified ransomware.

Figure 1.6 shows the clusters formed representing in a different manner the same information conveyed by the similarity matrix. r_2 and r_5 are the most similar samples and are grouped together. Similarly, r_1 and r_3 belong to the same cluster. r_2 and r_4 are the furthest away.

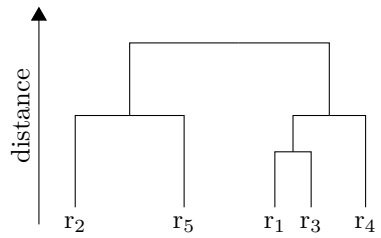

Figure 1.6: An example of a dendrogram.

6 Data Collection

Ransomware samples are downloaded from VirusTotal, VirusShare, and MalwareShare. Each sample is executed on Windows 7/10 machines for 15 minutes. A dump corresponding to this malware behavior is saved for analysis. The dump contains the folders traversed by the ransomware.
Benign data is collected from various users utilizing their computers for work purposes. The information gathered corresponds to web browsing, software development, file encryption.
The same information is collected for benign and malicious software: files traversed by threads.

The collected data is represented in the JSON format and provides a complete list of the explored folders for each userland thread of the system. Similarly to [29, 102, 111], a file system driver is used to monitor the runtime behavior of each thread.

As previously mentioned, 770 active ransomware are executed on Windows OS, both 7 and 10. However, 76 ransomware's binary hash correspond to multiple ransomware categories such as Yakes, Teslacrypt or Shade and Barys. Therefore, these records are omitted from ransomware family classification during the supervised learning phase. (1eb412a5f6400eb490a8698dc08129da) MD5 hash is an example of a sample belonging to different ransomware families. It is labeled by eight anti-virus software as Yakes and by six as Teslacrypt as analyzed by VirusTotal.

Since the overall database of malware collection contains 694 active ransomware, 417 ransomware records are used to act as a training set completed with 417 records of benign computer usage.

Benign data is collected on Windows 10 computers where a user is usually surfing the Internet, playing online games, developing a software, etc. They correspond to an end user's daily activity. Table 1.2 shows the distribution of ransomware and benign samples in the training set.

Ransomware Families	Samples	Benign Applications	Samples
Teslacrypt	115 (27.58%)	Firefox.exe	53 (12.74%)
Cerber	79 (18.94%)	Explorer.exe	47 (11.30%)
Xorist	74 (17.75%)	Svchost.exe	37 (8.89%)
Bitman	59 (14.15%)	Pnamain.exe	31 (7.45%)
Deshacop	13 (3.12%)	Receiver.exe	29 (6.97%)
Zerber	13 (3.12%)	Avp.exe	26 (6.25%)
Yakes	13 (3.12%)	Mscorsvw.exe	17 (4.09%)
Locky	7 (1.68%)	WmiPrvSE.exe	16 (3.85%)
Gpcode	6 (1.44%)	BackgroundTask	14 (3.37%)
Rest	38 (9.11%)	Rest	146 (35.10%)

Table 1.2: An overview of the active ransomware and goodware families used in the experiments, ranked in descending order according to their samples number.

7 Experimental results

7.1 Decoy Folders

The objective of this first detection mechanism is a binary classification of a record: benign or malicious. Three of the 694 ransomware pass through only the Perf_log and Python folders. They belong to Crpytxxx (1 sample) and Gpcode (2 samples) families. 99.56% of malicious threads are correctly detected. It highlights the fact that the majority of the ransomware collection begins the file system's exploration from the hard disk root.
None of the benign collected threads pass through at least 3 of the selected decoy folders. Eight benign applications visit the Prog_data and the Prog_files repositories. They are the SearchProtocol (twice), explorer.exe, svchost.exe (twice), avp.exe, klnagent.exe and vapm.exe.

7.2 Supervised Learning

All of the supervised learning algorithms are able to distinguish between a ransomware and a benign application based on the file system traversal, as seen in Table 1.3. Binary classification (benign vs. ransomware) is efficient in this case. However, to take one step further than the decoy folder detection, an analysis of ransomware families is carried out. For the Random Forest Classifier, 68.63 % of malicious records are correctly classified as so: a binary is no longer flagged as benign or malicious, so we can also identify the family of ransomware (e.g., Locky, Yakes). Indeed, there is a similarity between Bitman and Teslacrypt ransomware on the one hand, and Zerber and Cerber, on the other hand. The main reason behind this correlation is that they belong to the same family and most probably behave in similar ways to traverse the file system and encrypt their files. Decision Tree Classifier achieved a 61.25 % of correct ransomware classification.

Supervised Learning Algorithm	True Positive Rate	True Negative Rate	False Positive Rate	False Negative Rate	Training Time (seconds)
K nearest neighbor (n=2)	98.15	98.19	1.81	1.85	0.0039
Decision Tree	100	100	0	0	0.0095
Random Forest	100	100	0	0	0.0611

Table 1.3: Classifiers Performance Metrics.

Figure 1.7 illustrates the decision tree's rules to perform the split, thus, to classify the records. It is limited to 4 leaf nodes to be able to represent it. Indeed, a normal application does not regularly pass through those decoy folders. The majority of benign records have non-negligible values in the Prog_Files time and Windows_time. It shows the time spent by some applications such as firefox.exe or explorer.exe in those decoy folders.

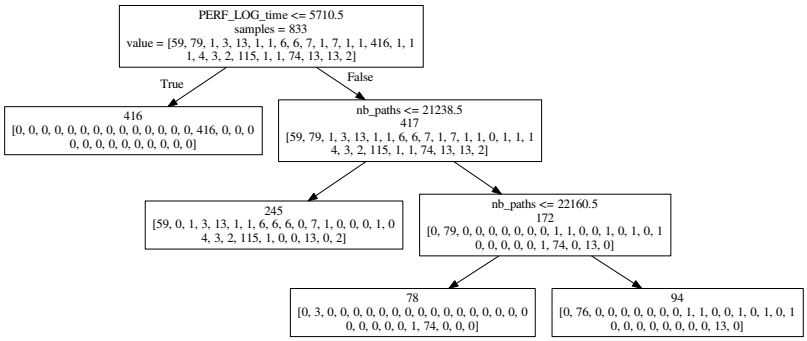

Figure 1.7: Decision Tree Classification Parameters.

The similarity between a Teslacrypt and Bitman's execution is shown through the records presented in Table 1.4.

family	Bitman	Teslacrypt	Normal
nb_paths	8199	8199	8
time_total	996315740	987726399	872039
RECYCLE_BIN_aggreg	2	2	0
RECYCLE_BIN_time	1460	3799	0
PERF_LOG_aggreg	1	1	0
PERF_LOG_time	4742522	4274485	0
PYTHON_aggreg	266	266	0
PYTHON_time	4987588	4519400	0
PROG_DATA_aggreg	188	188	5
PROG_DATA_time	131810705	132116153	772436
PROG_FILES_aggreg	0	0	1
PROG_FILES_time	0	0	455083
WINDOWS_aggreg	0	0	2
WINDOWS_time	0	0	450114
RECOVERY_aggreg	2	2	0
RECOVERY_time	131625192	131958939	0

Table 1.4: Benign and Ransom Records.

7.3 File System Traversal Velocity

Figure 1.8 illustrates the file system traversal velocity for the Xorist malware. Figure 1.9 demonstrates the Bitman malware. We observe that two implementations of the file traversal algorithm exist since there are two main curve clusters.

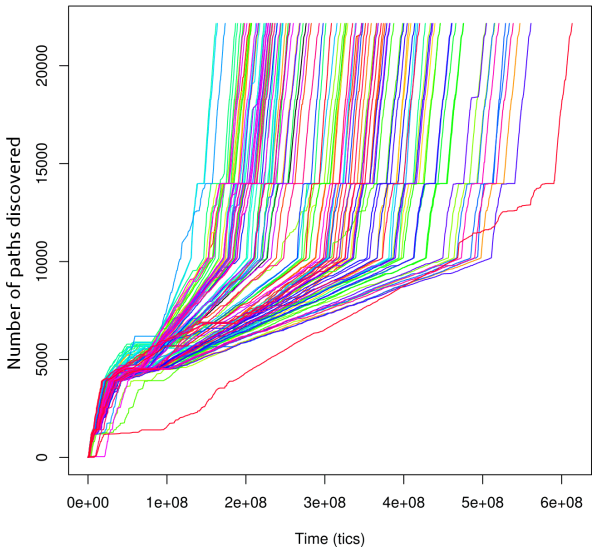

Figure 1.8: The file system's traversal velocity of Xorist samples.

The time unit considered in the graphs is the performance counter value (*i.e.*, OS internal) in units of processor ticks since the beginning of the session, which is comparable across the analyses. The speed depends on the ransomware design. The Cerber family searches files in one thread and encrypts them in another, whereas, the Xorist family uses the same thread to search and encrypt the files. Moreover, the programming or compiling choices cause some differences at runtime. To conclude, the velocity indicator provides an additional signature to distinguish between ransomware families. For other uninfected applications, no multiple file opening is shown.

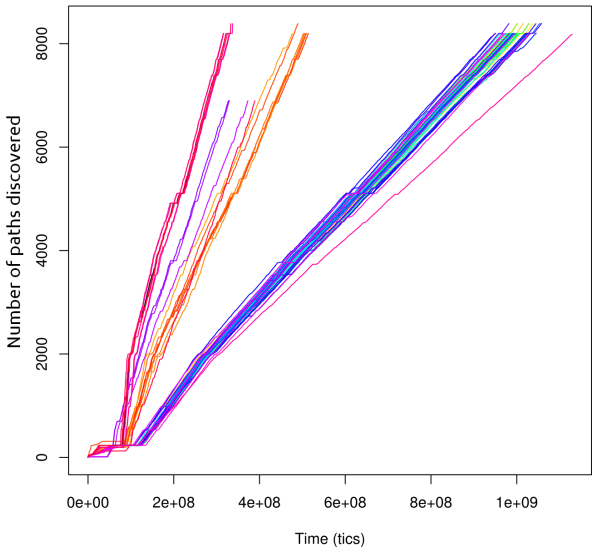

Figure 1.9: The file system's traversal velocity of Bitman samples.

7.4 Ransomware's Graph

The computation of a subset of the dataset that includes a wider variety of distinct families of pairs of graphs gives a similarity matrix, as represented in Figure 1.10.

A dozen of ransomware groups (*i.e.*, families) can be seen. Two groups represent 80% of the file systems traversal distribution. However, 10% of the samples are uncorrelated to others (*i.e.*, the blue block). The distance matrix shows that ransomware up to date have little diversity concerning the file system exploration because most of them use the Windows API for accessing the files.

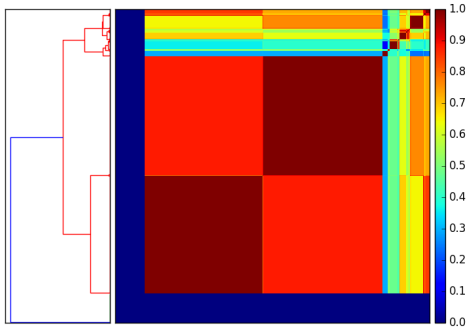

Figure 1.10: Malicious threads file system's traversal similarity matrix.

A partial view of the dendrogram (12 classified over the ransomware samples) is presented in Figure 1.11. The families that are close to each other are grouped in the same branch. The TeslaCrypt and the Bitman families are very close. It can be explained by the fact that they share the same traversal algorithm, whitelists, and blacklists. Another similarity is noticed between Cerber and Zerber ransomware. The latter can be considered as a simple variant of the first.

8 Limitations & Conclusion

Since ransomware behaves similarly in the file system traversal, more features need to be considered for family classification. In addition, any software that mimics the behavior of ransomware's traversal will

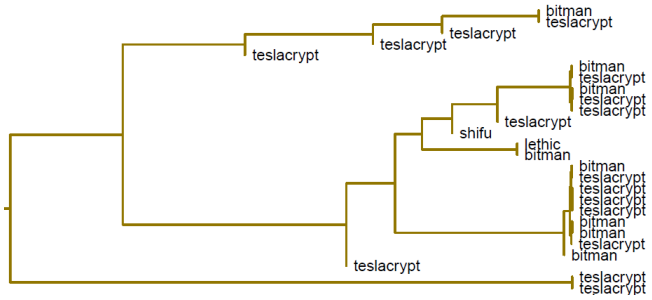

Figure 1.11: Families Graphical classification dendrogram.

be classified as malicious, so our proposal raises false positives. Another limitation occurs when no file system traversal is done before any encryption. Multithreading techniques also delay the detection of the ransomware since one thread is used for the files search while the other is immediately encrypting them. In this first countermeasure, we can detect ransomware behavior based only on monitoring *file system traversal*. We conclude that the majority of ransomware start their encryption process from the root of the hard disk. Machine learning techniques are used to get a precise classification. Based only on decoy folders, we are able to detect ransomware from various families. The work is published in ARES conference [47].

At this stage, system based countermeasures are adequately covered in the literature. Therefore, we decide to shift our focus towards analyzing network events. Our goal is to extract ransomware specific characteristics to enable its identification, if possible, at an early infection stage, limiting file loss.

2 Network Based Solution for Ransomware Detection

This Chapter presents an analysis of various ransomware families based on the collected system and network logs from a computer. Our goal is to reconstruct ransomware full activity to check if its network communication is distinguishable from benign traffic. Then, we examine if the first packet sent occurs before data's encryption to alert the administrators or afterward. We aim to define the first occurrence of the alert raised by malicious network traffic and where it takes place in a ransomware workflow.

1 Ransomware Network Traffic Dataset

1.1 Data Generation

Ransomware samples are downloaded and executed on MoM, similarly to the previous experiments carried out in Chapter 1. The duration of the experiments is two up to three minutes, which is the time required until the encryption note or encrypted files pop up on the file system. Moreover, the time constraint is also due to the encryption process involved in the ransomware infection that can also encrypt the collected information.

Wireshark and Process Monitor executables are launched on Windows OS 7 32 bits as in [83]. Each of them has an independent task for collecting the following information: Wireshark collects the information about network activity, whereas Process Monitor gathers the whole system activity (including network information).

Log formats are presented below:

- PCAP (Packet Capture) File contains data created by Wireshark consisting of the network packet data generated during a live network capture (source and destination IP address, ports, data exchanged, length of the data, ...). It can be analyzed later on for network intrusion detection purposes.

- PML (Process Monitor Log) File is created by Process Monitor and contains the log of system activities (process name, process id (PID), path, ...).

1.2 Dataset

All the methods and parsers are developed using Python and shell script. The analysis is performed on an Ubuntu 16.02 machine.

1054 ransomware are executed on Windows7 OS (table 2.1). Howbeit, 100 ransomware executables are kept for the machine learning phase since they were active, as discussed previously in Chapter 3 of Part I. A recap of ransomware families distributions is presented in Table 2.1. Even though only 100 samples are used for experiments, but machine learning is performed on packets. For example, for 12 Bitman samples, we can extract 714 network records. Whereas if we consider the network flow as shown in [89], we get only 62 malicious records to evaluate.

Family	Samples	Number of Working samples
Teslacrypt	334	11
Yakes	252	2
Shade	139	56
Cerber	95	11
Deshacop	62	2
Zerber	57	5
TorrenLocker	38	0
Bitman	27	12
Razy	26	0
Locky	24	1

Table 2.1: An overview of the active ransomware families (total of 100 active samples from 1054 tested), ranked in descending order according to their samples number.

2 Proposed methodology

This section presents the methodology for ransomware's session reconstruction. It can be implemented in a driver as future work to avoid being detected by a ransomware process.

We endeavor to thwart ransomware behavior by analyzing the network traces. The malware analysis routine consists of executing the ransomware and gathering the needed information. System logs help to reconstruct the malware session and display the timestamp of the first encryption process. Machine learning applied to network logs is used for traffic classification to distinguish between malicious and benign records. Finally, an evaluation is made to check whether malicious traffic detection occurs before the encryption process or afterward. To accomplish this task, our proposed mechanism is divided into two main parts: Data Filtering & Session Reconstruction, and Analysis & Model Development. It is thoroughly explained in the following sections. All the steps are summarized in **Ransomware Network Alert Algorithm** (RNA).

Algorithm Ransomware Network Alert
RNA

```
1:  procedure Ransomware Network Alert
2:      R_ ← {R_: Ransomware Related}
3:      def session_reconstruction(PML file, R_Hash):
4:      process_name ← {p_name:R_Hash.exe}
5:      R_pid ← {get PID / p_name=R_Hash.exe}
6:      for pid ∈ PID.PML do
7:          if Parent(pid) ∈ R_pid then
8:              R_children ← pid
9:      R_session ← {Filter(PML File) having R_pid & R_children}
10:
11:     def getR_Network_Activity(R_session, PCAP File):
12:     R_Network_Activity ← {Filter R_session}
13:     R_IP_@ ← {get R_IP @ src-dst}
14:     R_Ports ← {get R_Ports src-dst}
15:     R_Net_Act ← {Filter(Pcap File) having R_IP_@ & R_Ports}
16:
17:     Construct R_Model
18:     if evaluate(Net_Act) ∈ R_Model then
19:         return R_Alert
20:     return Benign_Activity
```

2.1 Data filtering

The Process Monitor format contains crucial information for reconstructing the malware session and activity. However, the first log file contains megabytes of data. It represents the full activity on a computer: gathered information from all running processes. Initial filtering is required to extract solely the information of the ransomware session.

In the following section, the preprocessing (filtering) of the collected data (see section 1.1) is described to gather ransomware activity.

Let $\mathbb{P}$ be the ensemble of all the processes running in Windows.
Let `p_name`, `p_pid`, `p_ppid` be respectively the name, the Process IDentifier (PID) and the Parent Process IDentifier (PPID) of a specific process p.

Ransomware executable names are associated with their MD5 or SHA-256 hash. They represent a unique identifier, that is known prior to the execution of the ransomware for testing purposes (*line 4 of Algorithm RNA*). An initial lookup is made on all the processes of the PML file that have a specific name. It should be constituted of the concatenation of the (`Ransomware_MD5Hash` or `Ransomware_SHA256`) and (`.exe`), a filename extension representing an executable file on Windows. The operator `+` denotes the concatenation operation.

```
Ransomware_name = Ransomware_MD5Hash + .exe
Ransomware_name = Ransomware_SHA256 + .exe
```

Consequently, an association of the name of the running process with the corresponding PID is achievable. It is a unique decimal number that represents this particular object (*line 5 of Algorithm RNA*). The collection of all the PIDs associated with the ransomware is achieved.

$\texttt{R\_pid} = \{\forall \texttt{p} \in \mathbb{P} \ / \ \texttt{p\_name} = \texttt{Ransomware\_name}\}$

However, any process running on Windows creates different children as threads or new processes to accomplish its tasks or parallelize the workload. In ransomware's case, one thread is created for listing the files, another one for encryption. For this reason, the tree/graph of the current processes is essential since it displays the relation among all of them (*line 20 of Algorithm RNA*).

$\texttt{R\_Children\_pid} = \{\forall \texttt{p} \in \mathbb{P} \ / \ \texttt{p\_ppid} = \texttt{R\_pid}\}$

At this stage, the identifier of the ransomware process and all the created sub-processes are at our disposal. The relation between all processes is represented by a directed graph defined as followed $G = (N, E)$ where: N is the set of nodes containing the PID, E is the set of edges, dashed arrows are representing benign processes, red arrows are representing ransomware processes.

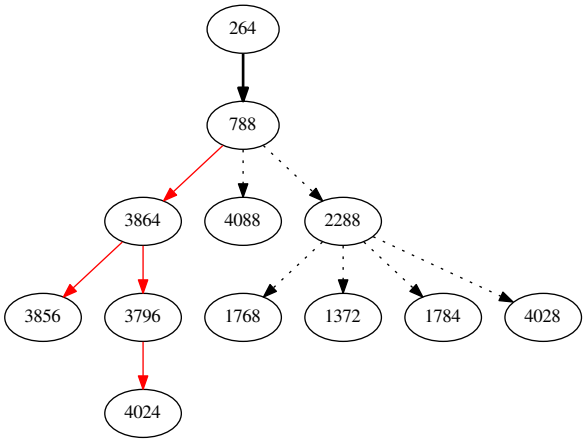

Figure 2.1: TeslaCrypt Process IDs Tree where each node contains a corresponding `pid`, dashed edges represent the benign processes whereas the red edges represent the TeslaCrypt graph.

Figure 2.1 displays a sub-tree of some subprocesses running on the machines. The red arrow marks

the beginning of TeslaCrypt's execution. TeslaCrypt malware creates many processes to accomplish its tasks, even though having a benign parent and siblings. Therefore, it is essential to build this "relation" graph (*line 16 of Algorithm RNA*).

$\texttt{R\_Activity} = \{\forall \texttt{p} \in \mathbb{P} \ / \ \texttt{p\_pid} = \texttt{R\_pid} \mid \texttt{R\_Children\_pid}\}$

Thus, initial filtering on the PML log file can be performed. It is divided into a malicious log that consists of all the actions performed by the ransomware and the second file that implies only benign records (*line 10 of Algorithm RNA*). The information gathered in the PML file is used to extract only the network communication from PCAP logs.

2.2 Ransomware Network Session Reconstruction

Since there is a gap between the data provided by the PML and PCAP file, a mapping is needed to collect exhaustive information from ransomware's network activity.

The network activity that exists in `R_Activity` acquired in the previous section is basic. It englobes only source and destination IP addresses, port numbers, and the length of the packet found in the PML File, whereas additional features can be extracted from a PCAP file such as TCP window size, checksum, header length.

We proceed by capturing the IP addresses and port numbers (*line 13 & 14 of Algorithm RNA*) used during `R_Activity` for the communication with a third party (for example the C&C), then we filter the PCAP File based on the data obtained previously (*line 15 of Algorithm RNA*).

The different features found in a PML file with the basic network elements (for instance IP addresses and ports) are presented in Table 2.2 while detailed and additional characteristics (TCP checksum, flags, windows size) can be extracted from a PCAP file are shown in Table 2.3.

Features	Record #1	Record #2
Time of Day	1/24/2019 5:46	1/24/2019 5:46
Process Name	htiyxhpnayrf.exe	htiyxhpnayrf.exe
PID	3916	3916
Operation	TCP Connect	TCP Send
Path	tivy-PC:49179 to cr1.toservers.com	tivy-PC:49179 to cr1.toservers.com
Event Class	Network	Network
Detail	Length: 0, rcvwin: 66240, seqnum: 0	Length: 896, startime: 768, endtime: 770, seqnum: 0, connid: 0

Table 2.2: PML File.

Features	Record #3
IP Src	10.1.1.9
IP Dst	198.12.157.163
TCP Srcport	49209
TCP Dstport	80
TCP Checksum	0x00006ee0
TCP Flags	0x00000002
TCP Hdr_len	32
TCP Window_size	8192
TCP Len	0
TCP Nxtseq	0

Table 2.3: PCAP File.

2.3 Supervised Machine Learning

The goal of this machine learning step is to develop a model for ransomware detection via network traffic analysis. Point anomaly represents a suspicious record at a given time t: when a specific data instance

deviates from the typical pattern. Whereas, collective anomaly represents a collection of similar events that are abnormal [122]. For example, point anomaly can be flagging Record#1 from Table 2.2 since it is not similar to benign records. Therefore, it is used for the machine learning process to flag any malicious network communication established by the ransomware. Its main advantage over collective anomaly is the early detection of ransomware presence rather than having to analyze packets to expose malicious behavior.

The supervised approach is practical since labeling the data is possible in our system. Thus, it enables the detection of other variants of ransomware based on an extrapolation of the data acquired throughout our experiments (*line 17 of Algorithm RNA*). Most of the research done in the literature on network intrusion detection via machine learning uses the following algorithms: decision tree, k-nearest neighbors, and random forest [162–164]. Therefore, they are adopted to detect ransomware behavior as a deviation from normal traffic. To perform this classification, Scikit-learn, a free software machine learning library, is used.

Our analysis addresses point anomaly subdivided in two, whether TCP or UDP protocol is used. Each packet is different from the other and presents few common features, such as IP addresses and ports. Whereas, for the collective anomaly, the conversation flow is used. Each row in the list displays the statistical values for exactly one conversation (ports, addresses, duration, and packets exchanged). This work has already been covered in the literature in [89].

Since the overall database of malware collection contains 100 active ransomware, we used the percentage split method (70/30) for each family's training and test set. It splits our dataset into random train and test subsets. The first one contains 70% of the data, while the second one 30%.

The separation between TCP and UDP training is made since the number of UDP communication outweighs the TCP ones, making our dataset unbalanced.

For network log extraction as a CSV file from the PCAP, many features provided by the Wireshark community exist. Filtering the PCAP file is possible by extracting 243 fields from the TCP protocol or 29 from the UDP protocol *(e.g., `https://www.wireshark.org/docs/dfref/t/tcp.html`)*. Nonetheless, many fields have non-existent values for all the records. Therefore, they are removed.

The features used for training UDP workflow are:
IP and Port source/destination, Protocol, UDP checksum, and length.

The features used for training TCP workflow are:
frame.len, ip.src, ip.dst, ip.proto, _ws.col.Protocol, tcp.srcport, tcp.dstport, tcp.ack, tcp.analysis.ack_rtt, tcp.analysis.acks_frame, tcp.analysis.bytes_in_flight, tcp.analysis.initial_rtt, tcp.analysis.push_bytes_sent, tcp.checksum, tcp.flags, tcp.hdr_len, tcp.len, tcp.nxtseq, tcp.window_size, tcp.window_size_scalefactor.

Data preprocessing can have a significant impact on the performance of various ML algorithms [165]. It handles, among other things, missing values and categorical variables. An intervention is needed since classification models can not handle these elements on their own. In our samples, empty values are replaced by zero, as for the IP addresses and flags, they are transformed into integers. Overall, the whole dataset consists of solely numerical values.

3 Experimental results

UDP Results

For the Cerber and Zerber samples, we achieve a 100% detection rate using any of the decision tree, random forest, or k-nearest neighbors. The difference is explicit. More than 16000 UDP packets are sent through incremental IP addresses with the same length in seconds. Additionally, the same information is being transmitted to all those different servers or zombies. The protocol used is solely UDP, very rare in a typical user environment, and is blocked in some companies. Moreover, it is comparable to a Denial of Service (DoS) attack due to the important number of contacted servers via UDP that is not common in normal behavior in just a few seconds.

The Udhisapi.dll module provides support in hosting compliant Universal Plug and Play (UPnP devices). We assume that it can be used to discover and communicate with UPnP devices across the network, such as other personal computers, printers, and mobile devices that broaden the attack vectors for ransomware.

TCP Results

The results of the other samples are presented in Tables 2.4 to 2.8.

Supervised Learning Algorithm	TPR	TNR	FPR	FNR	Training Time (seconds)
k-nearest neighbors (n=2)	99.56	98.13	1.86	043	0.004
Decision Tree	100	100	0	0	0.01
Random Forest	100	99.79	0.2	0	0.03

Table 2.4: Bitman Classifiers Performance Metrics (12 samples).

Supervised Learning Algorithm	TPR	TNR	FPR	FNR	Training Time (seconds)
k-nearest neighbors (n=2)	100	99.97	0.02	0	0.13
Decision Tree	100	100	0	0	0.16
Random Forest	100	100	0	0	0.24

Table 2.5: Cerber Classifiers Performance Metrics (11 samples).

Supervised Learning Algorithm	TPR	TNR	FPR	FNR	Training Time (seconds)
k-nearest neighbors (n=2)	100	99.99	1.4*10e-2	0	3.76
Decision Tree	100	100	0	0	1.02
Random Forest	100	100	0	0	1.57

Table 2.6: Shade Classifiers Performance Metrics (56 samples).

Supervised Learning Algorithm	TPR	TNR	FPR	FNR	Training Time (seconds)
k-nearest neighbors (n=2)	99.31	97.88	2.11	0.68	0.004
Decision Tree	99.31	99.34	0.65	0.68	0.009
Random Forest	100	100	0	0	0.035

Table 2.7: TeslaCrypt Classifiers Performance Metrics (11 samples).

Supervised Learning Algorithm	TPR	TNR	FPR	FNR	Training Time (seconds)
k-nearest neighbors (n=2)	100	100	0	0	1.59
Decision Tree	100	100	0	0	0.15
Random Forest	100	100	0	0	0.32

Table 2.8: Zerber Classifiers Performance Metrics (5 samples).

Supervised Learning Algorithm	TPR	TNR	FPR	FNR	Training Time (seconds)
k-nearest neighbors (n=2)	38.35	98.11	1.88	61.64	8.02
Decision Tree	98.46	100	0	1.53	0.54
Random Forest	95.7	100	0	4.29	0.7

Table 2.9: Zero-Day Classifiers Performance Metrics.

Decision tree provide the best results in terms of (true | false) (positive | negative) rate and training time. They spare potential overfitting problems by using random forest. As for the k-nearest neighbors, since IP addresses are huge numbers (could go up to 4 billion), they have a higher weight than TCP flags (maximum value 32).

The experiments prove that machine learning classifiers can flag ransomware network traffic for both UDP and TCP records as in signature-based detection.

A benchmark comparison is possible with the proposed work in [89]. The authors perform machine learning algorithms on protocols regardless if they were TCP or UDP based. However, we separate them since UDP records outweigh TCP ones to have a balanced dataset. In addition, raw features are used, such as described in section 2.3. It means that any record or communication can be flagged without delaying the alert mechanism that relies on having n malicious conversation flows. Decision trees lead to more accurate results (98.46% vs 97.10% in [89]).

3.1 Zero-Day Ransomware Detection

The conducted experiments are divided into two parts. Signature-based ransomware detection explained in the sections above where the training and the testing are performed on samples from a specific ransomware RA, RB, . . . , RN (see Figure 2.2, Part a).

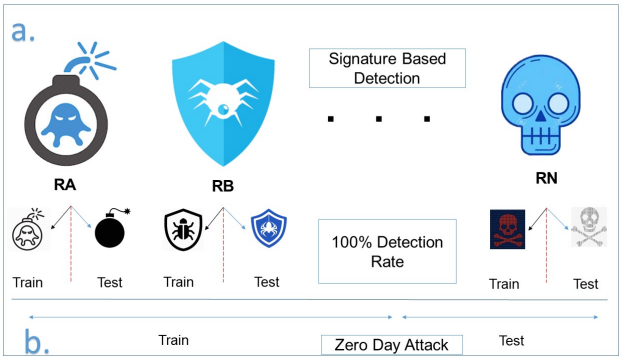

Figure 2.2: ML on Ransomware Families.

Nevertheless, to detect zero-day attacks, an administrator should test on new variants of ransomware. To implement this task, training is carried out on malware samples that appeared earlier or at the beginning of 2016. As for the tests, they are executed on different ransomware families excluded from the training set (see Figure 2.2, Part b).

Since a similarity is noticed between some Zerber and Cerber samples, in addition to TeslaCrypt and Bitman, we split our training and test set as followed:

- training set families: TeslaCrypt, Cerber, Shade (our dataset),
- test set families: Spora (15), GlobeImposter (2), Jaff (8), Matrix (3) (downloaded from *www.malware-traffic-analysis.net*).

Since test samples did not figure in the training set, we have 98.46 % as true positive rate and 100% as a true negative rate (table 2.9). They still represent a high value since the strategy of requests sent between the victim, and the C&C is shared among the majority of ransomware families.

3.2 Alert Time

Encrypted files as well as ransom notes serve as evidence that characterizes the presence of ransomware. If the detection based on analyzing network traffic occurs before the beginning of the encryption process, files losses are spared. Consequently, it is essential to recapture the time of the last packet sent and the start of ransom notes. For example, after Cerber's network communication, it creates nine different threads, and the encryption process takes place immediately after that. It leaves some nanoseconds for

the prevention mechanism to make a decision (blocking or killing the process, freezing the PC) before any file loss. Table 2.10 shows that the first alert, a network UDP send request, appeared before the ransom note #DECRYPT MY FILES#.html.

Time of Day	Operation	Path
15:45:09,81792	UDP Send	orfeas-PC:54673 → 85.93.63.255:6892
15:45:12,89509	CreateFile	C:\...\#DECRYPT MY FILES#.html
15:45:12,89919	CreateFile	C:\Py27\Lib\...\t56G_mZZIH.cerber

Table 2.10: Snippet of Cerber PML File, MD5 hash: 534da47881eba8a6d3cf676e6c89d1be.

Table 2.11 presents the percentage of samples that made a communication with the server before an obvious ransom note or encrypted file (*RansomAlert*). For Bitman and TeslaCrypt families, only 1 sample communicated with the C&C before it displays a ransom note. It means that the detection mechanism based only on network traffic is not appropriate for those families. Howbeit, network traffic detection for Shade ransomware is efficient and will spare file losses for victims.

Ransomware Family	$t_{Communication} < t_{RansomAlert}$	Percentage
Bitman	1	8.33% (1/12)
Cerber	7	100% (7/7)
Shade	55	100% (55/55)
TeslaCrypt	1	7.69% (1/11)
Yakes	0	0% (0/2)
Zerber	2	66.67% (2/3)

Table 2.11: Encryption Alert.

Since each sample provides a distinct ransom note or a specific file extension representing the ransomware, all the PML files are analyzed manually to extract the required information.

Bitman's ransom note description is presented in Figure 2.3. Examples of Bitman ransom notes names are given below:

- Recovery+ysddu.png,
- +-HELP-RECOVER-+bjkkl-+.png,
- _ReCoVeRy_+ioigp.png,
- help_recover_instructions+drl.txt,
- +-HELP-RECOVER-+wnonv-+.png.

Some Cerber samples kill the Process Monitor process several times during their execution, so the retrieved PML file is corrupted. Therefore, a difference is found between the number of active samples in Table 2.1 and in Table 2.11. To scale down any possible error, we did not consider those ten truncated samples in the Alert Analysis because the acquired data was incomplete.

3.3 Results Overview

Based on the timeline mentioned in the context displayed in Figure 1.2 Part I and on the network traffic, ransomware has evolved throughout the years and is polymorphic. Previous samples used to communicate via non-encrypted HTTP traffic (TCP requests), then other families moved to GET requests. Shade ransomware, for example, uses only TLS protocol for its communication. In addition to that, it was one of the pioneers for IPv6 communication. In 2016, UDP communication emerged. Based on the data gathered, new variants of ransomware can be detected if the divergence between new samples and existing ones are low. However, many cases are covered in our work. Attackers have to work on covert channels for exfiltrating information or keep encrypted communication similar to benign applications.

```
NOT YOUR LANGUAGE? USE https://translate.google.com
What happened to your files ?
All of your files were protected by a strong encryption with RSA4096
More information about the encryption keys using RSA4096 can be found here:
http://en.wikipedia.org/wiki/RSA_(cryptosystem)
How did this happen ?
!!! Specially for your PC was generated personal RSA4096 Key , both public and private.
!!! ALL YOUR FILES were encrypted with the public key, which has been transferred to
your computer via the Internet.
!!! Decrypting of your files is only possible with the help of the private key and decrypt program ,
which is on our Secret Server
What do I do ?
So , there are two ways you can choose: wait for a miracle and get your price doubled,
or start obtaining BITCOIN NOW! , and restore your data easy way
If You have really valuable data, you better not waste your time,
because there is no other way to get your files,
except make a payment
For more specific instructions, please visit your personal home page,
there are a few different addresses pointing to your page below:
* http://t54ndnku456ngkwsudqer.wallymac.com/68052649D4FFA17
* http://po4dbsjbneljhrlbvaueqrgveatv.bonmawp.at/68052649D4FFA17
* http://hrfgd74nfksjdcnnklnwefvdsf.materdunst.com/68052649D4FFA17
If for some reasons the addresses are not available, follow these steps
1 Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en
2 After a successful installation, run the browser
3 Type in the address bar: xlowfznrg4wf7dli.onion/68052649D4FFA17
4 Follow the instructions on the site
```

Figure 2.3: Bitman Ransome Note.

Tests are also performed on 18 samples from Cerber, Zerber, TeslaCrypt, and Bitman without any Internet connection. The encryption still took place. Nonetheless, we know that the keys are generated locally, enabling us to retrieve them via a simple hook to Windows Crypto API or is hard-coded in ransomware's executable, highly unlikely. Two identical ransomware samples are found in Bitman/TeslaCrypt and two others in Cerber/Zerber. It denotes a resemblance between those families. For example, 2d2c589941dd477acc60f6b8433c3562 MD5 hash is flagged as Bitman by 7 anti-virus companies and as TeslaCrypt by 8 other anti-virus companies [88]. They are kept for signature-based detection (no duplicate records in the same family since it appears just once) but removed from the zero-day analysis.

100% detection rate is attained. As a result, its validity needs to be checked: based on which criteria a separation is achievable.

Decision Tree (UDP Samples)

This supervised learning algorithm also provides exquisite results. Notwithstanding, the output of this separation graph generated by the decision tree is based only on the initial victim communication protocol. Therefore, it is not a reliable method for distinguishing benign traffic from malicious one based only on one characteristic.

Decision Tree (TCP Samples)

Similarities among decision trees of multiple ransomware families using TCP for communication are noticed. The separation is mainly done based on those characteristics: tcp.window_size_scalefactor , tcp.hdr_len, IP source and IP destination (Figure 2.4).

- tcp.window_size_scalefactor: All benign samples have the value of -2 or 0 for the scalefactor whereas ransomware records contains 0, 128, 256 or 1024 as values.
- tcp.hdr_len: All benign samples have 20 bytes as header length whereas ransomware records header length can be wither 20, 32, 48 or 56 bytes.
- IP source and destinations serve as blacklisted addresses.

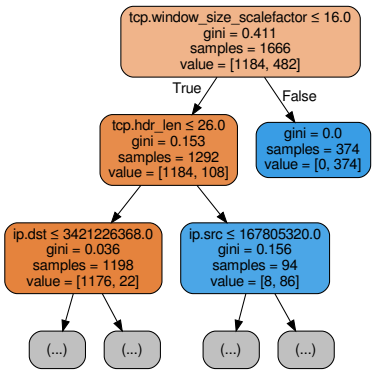

Figure 2.4: Ransom Model Graph.

4 Is There A Correlation Between System & Network Logs?

To sum up, from the system and network log analysis, we can extract the following information about ransomware.

- Ransomware's file system traversal starts at the root of Windows file system.
- Ransomware passes through specific folders and skips others to preserve the functionality of the computer. By this mean, the victim can see the ransom note and thus, proceed to the ransom payment if found necessary.
- Ransomware samples belonging to the same family can diverge in some aspects. For example, some explore the file system, then proceed with the encryption. Others parallelize their work tasks: one thread is responsible for files hunt while the other encrypts the found one immediately.
- Ransomware traffic presents specific characteristics that make a benign/malicious categorization possible. Hence, IP addresses and port numbers can be relevant indicators, thus, blacklisted to prevent or scale down ransomware propagation.
- In some cases, ransomware contacts the C&C before the encryption process.

Based on the provided information above and the information gathered from the literature, we investigate a possible correlation between those logs to have a higher probability of detecting ransomware at early stages.

To have a deeper understanding of ransomware samples, we look into the findings of the work carried out in [166] and in [167]. They offer a detailed dynamic analysis of PrincessLocker and Spora ransomware. The only information conveyed to the C&C of the attacker consists of two values: the ID of the victim and the number of files that were encrypted. There is no additional information that could be extracted, used, or combined to have a stronger defense mechanism.

Besides, having prior knowledge of a specific system event does not imply the occurrence of the network communication as seen in the ransomware generated PML files. To date and based on the

literature review and our experiments, we found no evidence expressing the explicit correlation between system and network logs. This semantic gap can be explained by no evident association amongst them, but, the combination of both types of logs provides a robust defense mechanism. This merged solution integrates multiple sequential events, as seen in section 3.7 of Chapter 2 of Part I.

5 Conclusion

In this work, we are able to detect ransomware through network traffic monitoring. We conclude that the majority of ransomware behave similarly. We found some common patterns among various families. To get a precise ransomware detection, we use machine learning techniques.

To sum up, network alerts represent a first suspicion means informing the user of the presence of a potential ransomware. However, some drawbacks exist. This first alarm can take place after the creation of encrypted files or ransom notes as we noticed in some families. In addition, few elements are needed for a classification, we have underfitting problems (Zerber samples), prone to adversarial attacks. Besides, only decision tree among the tested algorithms provided high detection rates for zero-day attacks. For all the reasons mentioned above, Network Alerts should be backed up with system data to provide a general detection mechanism, working on all types of ransomware. The work is published in FPS conference [48].

The diverse and abundant solutions developed previously cover meticulously ransomware attacks. The existent gaps found in these countermeasures can be patched by combining multiple components, just like in the random forest, to make precise predictions than any individual model would have.

Surfing the latest security trends, we bumped into an article on Bleeping Computer that discusses a malware that steals confidential military, and financial files [168]. It is somehow related to Ryuk ransomware. The sample performs a lookup on all PDF and XLS files, especially those containing words like “fraud”, “hack”, “tank”, “defence”, “military”, “checking”, “classified”, “secret”, “clandestine”, “undercover”. Any file that matches those strings is then uploaded via FTP to a remote server. We decided to explore the feasibility of such an attack in a basic user environment and provide ways to delay or stop the executable if possible. These elements are discussed in Chapter 3.

3 From Ransomware to Doxware

To compete in this arms race against security breaches, we propose an insight into plausible attacks, especially Doxware (also called leakware). The work is accomplished in collaboration with my colleague Renzo Navas, and my students Charles Berti, and Guillaume Deboisdeffre. We present a quantification model that explores the Windows file system in search of valuable data. It is based on the term frequency-inverse document frequency (TF-IDF) solution provided in the literature for information retrieval. The highest-ranked files will be then exfiltrated over the Internet to the attacker's server. Besides, we implemented honeypot files and folders to test the validity of our proposal. We conclude by presenting future perspectives in this area with the possible counter-countermeasures that can be taken by an attacker to bypass current detection mechanisms. Our approach delivers an observation of the evolution of malware throughout the last years.

1 From Data Encryption to Data Exfiltration

1.1 Where to Find Sensitive Data and How to Track It?

Google Scholar provides more than 5 million research papers regarding data sensitivity. It is not limited to a particular field but represents a common concern for a myriad of sectors (healthcare, telecom, automotive, energy). For example, mental health care is a delicate subject that could ruin a person's reputation under malicious manipulation. Therefore, attackers tend to target personal information since it intimidates the victims driving them to pay the ransom in exchange of keeping the information private. Netherlands data breach proves previous hypothesis since it came mostly from the medical sector (29%) [169].

Sensitive information depends on the equipment being used. For instance, Yang *et al.* in [170] consider that the following items represent significant data on Android OS: Unique Device ID, Location, Phone number, Contact book, SMS messages, and Calendar. These items carry a considerable advantage since each cell phone possesses them, and any application could access them via simple API calls. Taint analysis detects flows upcoming from known and predefined sources (for instance, IMEI of a cellphone) to untrusted sinks like the Internet [171]. Tracking data is, therefore, a straightforward process in Android devices having the predefined sources previously known and Internet the only untrusted sink. Taint analysis applied to applications that behold the identified sensitive information helps limiting data leakage. TaintDroid framework developed by Enck *et al.* allows users to monitor how third-party smartphone applications handle their private data in realtime [172]. It uses dynamic taint analysis to track the propagation of tainted data at different levels: instruction level, message-level between applications, and file-level. Original data can be transformed, for example, by writing its content in a pixel bitmap. However, TaintDroid issues warning reports if tainted data is leaked by an application, even if it was transformed since the taint is propagated through these side channels [173]. A similar tool developed by Sun *et al.* enables a multilevel information flow tracking by utilizing registers for taint storage, having only a 15% overhead on the CPU [174]. It presents an enhancement of TaintDroid formerly developed in terms of taint storage and resource consumption [172]. Considerable research is being conducted in this field as in [173, 175–177].

Data's value is translated by the measure taken by a company to protect it. For instance, Zhu *et al.* provide TaintEraser, a new tool that tracks sensitive user data as it flows through applications [178]. They are one of the pioneers in developing data protection from leakage on Windows OS. Their taint propagation is based on instruction and function level. They evaluate their solution on Notepad, Yahoo!Messenger and the Internet Explorer where they presented accurate results based on taint propagation. However, TaintEraser can be bypassed via data transfer in shared memory. Loginova *et al.* suggest to use cryptographic software to carry out on-the-fly encryption [179]. They state that it represents the most effective approach to overcome data leakage and to protect the information.

On Android OS, attackers know what they are looking for and where to find it, like extracting the victim's GPS location. Yet, these sensitive elements cannot be predefined on a computer level. Indeed, confidential data is only relevant to a particular end-user. For instance, it could be a project for a student or a painting for an artist. Data exists in a variety of formats and are stored in different locations for each user. We analyze in the following parts to which extent data localization is possible on a computer and if it is comparable to mobile devices, and how is the exfiltration carried out.

1.2 Data exfiltration

Data exfiltration is an attack that occurs when a malware's author carries out an unauthorized data transfer. Researchers have long been interested in this domain since it can threaten a company or individual's well-being. Giani *et al.* reveal that the bandwidth constraints depend not only on the amount of data exchanged but also on the media being used [180]. Indeed, since 2006 little has changed. Leakage methods remain the same (FTP, SSH, email...).

Data can be in one of the following states: in use (by an application), in transit (being sent over the network), or at rest (stored in a database). Countermeasures protecting data, regardless of the state, are clustered by Ulla *et al.* into three main categories [181].

- Preventive solutions prevent an attacker from stealing data. It represents a proactive measure taken by administrators to resist against exfiltration attacks.
  - Data classification techniques categorize data based on the sensitivity level. For example, data can be classified using a binary metric (sensitive/non-sensitive), or several categories can be considered, such as highly confidential, confidential, restricted, and public [182, 183].
  - Access control policies ensure that only legitimate users have access to the requested resources. It is enforced by authentication and authorization mechanisms [184, 185].
  - Encryption mechanisms protect the confidentiality of the data. Even if attackers acquire encrypted data, no information can be extracted or analyzed from the ciphertext [186, 187].
  - Cyber deception techniques mislead the attacker into considering the he/she is in hold of sensitive data or is attacking an actual server [188, 189].
- Detective techniques aim to detect the leaks. Unlike the preventive category, detective countermeasures are reactive since they can not only detect, but also, stop an attack.
  - Content inspection searches for predefines keywords, hashes, or patterns in the network traffic. A security breach is defined by a matched signature, then, the administrators are alerted [190, 191].
  - Anomaly-based detection compares known patterns of a network or a host and constructs a model to represent it. Any deviation or any resurgence of an unexpected pattern represents a potential attack (see section 5, Chapter 2, Part I).
- Investigative solutions do not help recover the leaked data. However, they identify key-elements that help protecting the current environment. It is accomplished by knowing when, how, and who is responsible for this data exfiltration.
  - Watermarking techniques tackle specifically the integrity of the data. It is used to protect copyrights and the non-repudiation of the data. Watermarking consists in adding a signature to the sensitive data to be able to track it and verify that it remains intact [192].

– Probabilistic forensics reconstructs the actions undertaken by an attacker to penetrate the system. Traces or logs are acquired from the system as well as metadata. This digital forensic includes approaches to prevent data being copied to an un-authorized USB device [193, 194].

– Counter-intelligence operation helps to track back an attack towards an attacker. However, most of the approaches proposed in the literature have not been tested [195].

There is an analogy between the countermeasures developed for exfiltration and ransomware attacks. In fact, the preventive countermeasures correspond to the delivery phase. The detective techniques correspond to the deployment and the destruction phase. Finally, the investigative solutions can be mapped to the transaction done via bitcoin.

Data are important to any entity. Therefore, we endeavor to check the possibility of sensitive data exfiltration and its repercussions on the system. We evaluate a possible Doxware attack since, by definition, it manipulates documents to leak information.

2 Context

2.1 From Ransomware To Doxware

Figure 3.1 presents ransomware and Doxware workflow. Four main stages appear in both malware (phases P1, P2, P3, and P4). The only difference resides in "valuable files hunting" followed by an exfiltration of the acquired data (phases D-P2 and D-P3). In fact, ransomware attack scope and losses are confined in a users setting: the data remains on the PC, yet, it is encrypted. With a previous backup, end-users can restore all their files. Nonetheless, in a Doxware attack, the damage is beyond repair since once the information is out to the digital world, any person has access to it. To the best of our knowledge, no previous studies were made on this specific type of malware, and it was only mentioned by researches as an advanced threat [196–199].

Similarly to ransomware, Doxware's attack vectors are mainly phishing/spam emails or unpatched security vulnerabilities on a visited website or on the victim's system [23,200]. Then, once it is infiltrated on the system, it checks if the required libraries with the appropriate versions are installed on the computer to perform its destructive intents. Afterwards, an exploration of the file system is made to search for example, for military and financial files like in [168]. Specific file types are encrypted (texts, images, and documents) using AES-256 or RSA-2048. Finally, a ransom is demanded to receive the decryption keys and to avoid sensitive information exposure. Besides, some malware authors create eBay-like auction site for stolen data [7].

Our proposed approach focuses on the file evaluation for score computation.

2.2 Data Formats Choice

Different data formats exist nowadays that are stored on a computer. They can be classified into four main categories.

1. Textual Documents represent files that contain mostly data in the form of a sequence of words or alphabetic characters. For example, contracts, agreements, company's balance sheet, medical records.

2. Pictures are designs or representations made by various means (such as painting, drawing, or photography). For instance, Magnetic Resonance Imaging (MRI), gradient descent convergence, trip pictures.

3. Videos represent a recording of a motion picture made digitally or on videotape (movies, video clip, news).

4. Some files have the combination of two or all the categories of previously mentioned types such as a PDF file (it contains paragraphs as well as images).

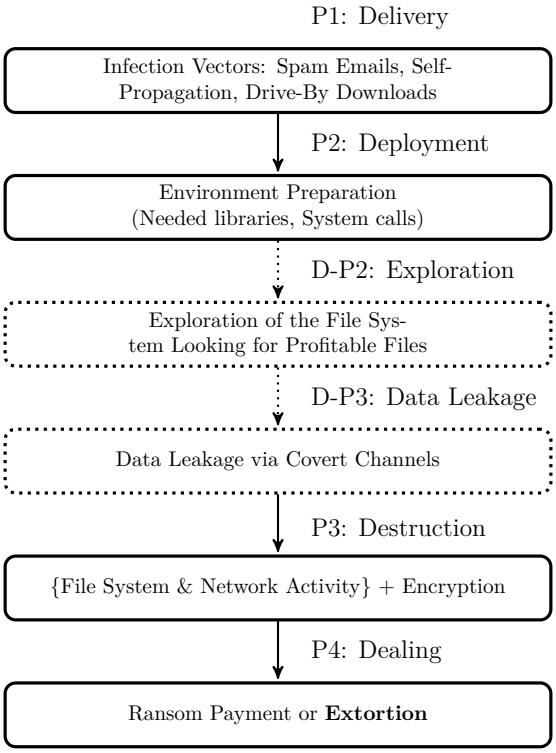

Figure 3.1: Ransomware Vs Doxware.

Nearly all processing methods in the literature for face recognition or body detection are based on machine learning algorithms [201, 202]. Some additional information is mandatory to be able to recognize bodies, clothes, poses. Many drawbacks reside in these approaches, such as their weight since they embed complex algorithms requiring considerable computation power. This means many false positives cannot be tolerated for the attacker since the transfer of large files is easily detectable. Sending a 50 Mb video that does not encompass sensitive information represents a massive loss for the attacker. For example, many packets are transmitted over the network and cannot go unnoticed. Therefore, a compromise between efficiency and stealthiness is needed. For all the reasons cited above, we develop a proof of concept in the following sections based on textual document analysis. It serves as a method for identifying files that belong to predefined topics. Contracts are chosen as a baseline for the rest of this chapter.

2.3 Natural Language Processing (NLP) and Information Retrieval (IR)

The field of NLP encompasses many topics that help to convert, to some extent, a text into a data structure. Statistics, probability, and machine learning were previously used to analyze textual documents. Recently, deep learning techniques are adopted to extract even more refined results in a shorter time due to advances in computational power and parallelization [203]. NLP is adopted in various domains. Hence, a medical NLP is used by Friedman *et al.* to extract molecular pathways from journal articles [204]. Stenetorp *et al.* rely on NLP to develop a Web-based tool for text annotation [205]. Another example worth mentioning is the application of the NLP on hate speech detection presented by Schmidt *et al.* in [206] or Al-Hassan *et al.* in [207].

Information retrieval is defined in an academic field of study as follows: "Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)" [208]. NLP techniques have been used in IR, however, since only few improvements are noticed [209], we focus on presenting the following two state-of-the art techniques used for IR.

- Bag-Of-Words (BOW) is a simple method used for object categorization. The idea relies on regrouping words by their occurrences.

Let us take as an example the following sentence: "Ransomware is an epidemic that affects the lives of individuals and the development of companies."
The output of BOW represents pairs of the words found in the sentence and their occurrences presented in Table 3.1.

Word	Number of Occurrences
ransomware, is, an, epidemic, that, affects, lives, of, individuals, and, development, of, companies	1
the	2

Table 3.1: BOW table.

Nevertheless, this technique has some well-known flaws. Some words, existent in any kind of documents ("the", "an", "always", "being") called stop words, are not representative of the document itself compared to others. Their frequency might exceed relevant elements. Hence, this technique is backed up by the TF-IDF transformation addressing the problem encountered in BOW [210, 211].

- TF-IDF has the Bag-Of-Words as a basis but with an improved layer. A corpus of files is needed because the process compares documents one to another. Better specificities of the documents can be extracted if there are many specimens as a baseline.
  - TF represents the raw count (frequency) of a given term t in a document d.

$$tf(t,d)=f_{t,d}$$

  - IDF represents the value carried by the word. IDF score is the logarithm of the number of documents divided by the number of documents that contain the word t. When the number of times a word is present in a document is significant, the value obtained in the logarithm is very close to 1, so idf_t is close to 0. The $idf_{t,D}$ coefficient highlights rare words found only in few documents. Even though not frequent enough, they are significant and are spotted by having a higher score [212, 213].

$$idf(t,D) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right)$$

    where:

    $|D|$ is the total number of the documents in the corpus

  - TF-IDF is used as a weighting factor to reflect the importance of a word in a given document or corpus. It is calculated as

$$tfidf(t,d,D) = tf_{t,d} \cdot idf_{t,D}$$

3 Content Analysis Proposal

It is possible to extract multiple keywords representing a document based on the previous observations and the literature techniques discussed in section 2.3. We make use of this information to check to which extent sensitive data can be disclosed by applying straightforward IR techniques.

3.1 Target Threat Model

The goal of the adversary is to extract sensitive data from a system represented by a personal computer in this analysis. The attacker develops a malicious application that is executed on this system, and once the required information is identified, it is sent through the network to third-party server controlled by the adversary.

We assume that the user installs the malicious application on his computer without prior knowledge of its functionalities or intents. Also, we assume that no counter-measures are in place to protect user's private data.

We focus on the evaluation algorithm that helps to determine and/or to classify data as sensitive or not. The proposed evaluation algorithm depends on some parameters of the document that the program is analyzing. It is divided into two parts.

- **Lexical Generation.** The first part is related to the attacker characterized by generating intelligent lexicons to focus on the subjects/topics he/she wants to exfiltrate. This is accomplished by relying on the TF-IDF methodology applied on a corpus of documents.
- The second part concerns the victim side, where the attacker evaluates victims documents to send them over the network.
  - **Document Content Evaluation.** It includes assigning a score to each document based on the keywords or lexicon generated previously.
  - **Metadata Evaluation.** In addition, metadata is analyzed to extract further information about the evaluated files.

To sum up, the analysis program consists of 3 modules to accomplish these tasks: **Lexical Generation**, **Document Content Evaluation**, and **Metadata Evaluation**. They are thoroughly explained in the next sections.

3.2 Chosen Corpus: Contracts

Leakware or Doxware is a vast subject and can be interpreted in many ways, whether it infects a personal or a professional machine, an individual user, or a company. Our preference from among choices is the evaluation of professional documents since they can be found on both machines. Textual documents can be saved in various types of extensions (.txt, .docx, .pdf, .rtf , .wpd, .odt...).
Existent tools for word extraction are not applicable on a PDF file containing a scanned document. A possible solution is evaluating them as images, and extracting their content with the help of an Optical Character Recognition (OCR) and Tesseract (an open-source OCR engine). Each page of the PDF document is converted into a .png image. The ratio of PDF to image can be 1/30, a memory consuming process. The program gets importantly less stealthy having a longer processing time compared to .txt. Therefore, we restrain the study domain to .txt, .docx extensions, documents being one of the most targeted files [214].

To carry out the IR process, a corpus of textual documents is required. Contracts are chosen as a topic to represent the sensitive information that is exfiltrated by the attacker. To do so, various files are downloaded from *onecle.com* and *contractsfinder.service.gov.uk*. Three types of contracts are chosen for the corpus:

- investment agreements
- marketing agreements
- partnership agreements

The terms “investment”, “marketing”, and “partnership” are explicitly marked in the documents. Additional documents are gathered from Google Scholar, online courses, and forums.

The same procedure can be carried out on other topics that convey as well critical information like medical records, biometric data, and political opinions. Thus, the database that contains the pairs of keywords and topics can be extended.

3.3 Lexical Generation

On the attacker’s machine, a pre-processing is made for lexicons generation of any requested subject. The topic represents the files that the attacker is searching for on the victim’s machine.

Initially, TF-IDF transformation is applied to the union of documents in the corpus. The top -n (n is a given integer that represents the wanted number of significant words) results represent the words having the highest score for each document.
Table 3.2 and Table 3.3 present the ten (n=10) words having the highest TF-IDF score.

Word	TF-IDF Score for Doc.1
company	0.3980
section	0.3771
agreement	0.3324
1look	0.2689
purchaser	0.240
shares	0.2290
shall	0.2220
date	0.1773
material	0.1550
closing	0.1508

Table 3.2: TF-IDF Scores for Document 1.

Word	TF-IDF Score for Doc.2
company	0.5583
buyer	0.4851
shall	0.2415
agreement	0.2166
section	0.2137
acquisition	0.1297
stock	0.1274
securities	0.1106
voting	0.1100
proposal	0.1094

Table 3.3: TF-IDF Scores for Document 2.

The next step relies on creating a function that associates each word in the lexicon to an importance "score". Let w_i be a word that represents the target subject. Let n be the number of words taken into consideration by a document (top n words with highest TF-IDF score). The word w_i has a $p_{i,j}$ position in the document j, the corpus contains N documents. Its value is built as following.

$$Sc(w_i, j) = \frac{n - p_{i,j}}{n}$$

As a result, the total score Sc_i is:

$$Sc_i = \frac{1}{N} \cdot \sum_{j=1}^{n} Sc(w_i, j) = \frac{1}{N} \cdot \sum_{j=1}^{n} \frac{n - p_{i,j}}{n}$$

Sc_i is divided by n for normalization purposes so that any word can have a maximum score of 1. For example, the word "agreement" is the 3$^{\text{d}}$ keyword having the highest TF-IDF score in the first document, however, in the second document it is ranked as the 4$^{\text{th}}$ most important word.

$$Sc(agreement, Doc.1) = \frac{10 - 2}{10} = 0.8$$

$$Sc(agreement, Doc.2) = \frac{10 - 3}{10} = 0.7$$

The final score is divided by the number of documents considered to maintain the values between 0 and 1.

$$Sc(agreement, Docs) = \frac{0.8 + 0.7}{2} = 0.75$$

A part of the investment agreement lexicon produced for those two documents is presented in Table 3.4.

At the end of this step, three lexicons are generated, each representing "investment", "marketing", or "partnership" agreements. It is created by applying the total score on the N documents of the chosen corpus.

3.4 Document Content Evaluation

The lexicons are already embedded in the malware source code on the victim's side. They are used to process a content score which is combined with a metadata score for a final evaluation score.

At first, a dictionary containing every word of the document with its number of occurrences is extracted. The initial value of the content score is 0.

Let CS be this content score, Sc_i the score of the word i of the lexicon being studied created previously, n the number of words in the lexicon and occ_i the number of occurrences of the word i in the document analyzed.

$$CS = \sum_{i=1}^{n} Sc_i \cdot occ_i$$

Word	Score
company	1
section	0.75
agreement	0.75
1look	0.35
purchaser	0.3
shares	0.25
shall	0.6
acquisition	0.25
buyer	0.45
stock	0.2
securities	0.15

Table 3.4: Score of Words.

Let us consider a document composed of the following sentence: "Party B promises to commence the relevant applications under the Project within 15 days following the execution of this agreement." The content evaluation is presented in Table 3.5 and in the following equation.

Word	Number of Occurrences	Lexicon Score
party	1	0
promises	1	0
commence	1	0
relevant	1	0
applications	1	0
the	3	0
following	1	0
execution	1	0
agreement	1	0.75

Table 3.5: Content Evaluation.

$$\begin{aligned} CS &= 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 + 3 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0.75 \\ &= 0.75 \end{aligned}$$

The content score is calculated for each of the three previously generated lexicons. The highest score is retained, followed by a division by the size of the document. This division helps to maintain the same probability to have important documents regardless of the size if they carry the same information.

$$CS_{\text{final}} = \frac{1}{document\_size} \cdot \max\left(CS_{investment}, CS_{marketing}, CS_{partnership}\right)$$

We select randomly 36 documents that do not contain any information about contracts such as medical records, statistics, state-of-the-art of ransomware, etc. and we compare the scores to the 36 contracts in our corpus. The minimum (Min), the maximum (Max), the average (Avg), and the standard deviation (σ) of the obtained scores are presented in Table 3.6.

Statistics	Random Non Decoy Files	Important Files
Min	0.503	1.31
Max	5.66	7.698
Avg	1.653	4.29
σ	2.045	1.359

Table 3.6: Statistics of the Scores of Random and Important Files.

3.5 Metadata Analysis

Metadata is "a set of data that describes and gives information about other data". This information is stored in various file types and its analysis describes the considered document.

Fifteen metadata can be accessed and extracted from a Word document. Those are author, category, comments, content status, created, identifier, keywords, language, last modified by, last printed, modified, revision, subject, title, and version. These details are useful for an attacker to determine if the analyzed file is important to the user. It is accomplished by checking the date of creation and modification. In fact, old files are of no interest for exfiltration whereas recent files used by end-users carry relevant information. In addition, the multiple revision of a file indicates that the user is manipulating this document, therefore, it contains pertinent information.

Most of the cited metadata analyzed are not filled in (the content is null). Some metadata have to be manually annotated like keywords to support searching and indexing, as well as the comments part to describe the content of the resource. Therefore, the only features kept and the most relevant ones are: the number of revisions, created, and last printed.

- If more than 1 revision is made, the metadata score is incremented by 5.
- If the document is created in 2019 (the timeline where the experiments were made), the metadata score is incremented by 1.
- Additionally, if the document is printed in 2019, the metadata score is incremented by 2.

3.6 Proposal Summary

This section summarizes the previous steps taken to achieve a complete scan of the file system of the victims computer in search if valuable files, more specifically, contracts. It is portrayed by the algorithm Valuable File Hunting (VFHA).

1. Initially, the lexicons are generated based on the contract topic (line 2 in VFHA).
2. Then, the parsing of the target file system is made searchings for .txt and .docx extensions (line 9 in VFHA).
3. The file score is calculated as following.
   (a) The content of .txt file is extracted, and vocabulary analyzed. Each word is compared with a lexicon previously created that contains recurrent and relevant words in a contract based document.
   (b) The same procedure is done for the .docx files. However, an additional step is made for the metadata analysis. The significant metadata are the number of revisions, creation date, and the date when it was last printed. They are added to the total sum representing the value of a document (line 3 and 9 in VFHA).
4. "Summarize" step: Each document has a total score that has been assigned, so the list of tuples (path, score) is sorted according to the value obtained, where the attacker chooses which ones he/she wants to extract. For instance the first 50 files (line 23 and 24 in VFHA).

4 Proposal Analysis

4.1 Created Lexicons

The chart presented in Figure 3.2 shows the most common words in a contract document. They are: "shall", "partnership", "agreement" and also "section". Indeed, the corpus gathered is previously identified as a contract type document, which is an advantage allowing us to perform a relatively simple algorithm to determine whether a document belongs to this category or not, although law documents also acquire an important score and may also be as valuable as a contract.

Algorithm Valuable File Hunting VFH

```
procedure ALGORITHM 1
    Topic_Lexicon ← { lexicon_generator (Corpus having the same Topic: Contract) }
    def analyse_content(File f):
    for word ∈ f do
        if word ∈ Topic_Lexicon then
            f_score += score(word) * number_occurrences
    return f_score/len(f)

    def analyse_metadata(File f):
    if f.core_properties.revision > 1 then
        f_metadata_score += 5
    else if f.created == "2019" then
        f_metadata_score += 1
    else if f.lastprinted == "2019" then
        f_metadata_score += 2
    return (f_metadata_score)

    Parse the File System
    if FileExtension ∈ .txt or .docx then
        FileList ← {Analyse MetaData and Content}
    else
        Continue;
    Sort FileList by highest_Score
    Send n first valuable files to the attacker's server (future work)
```

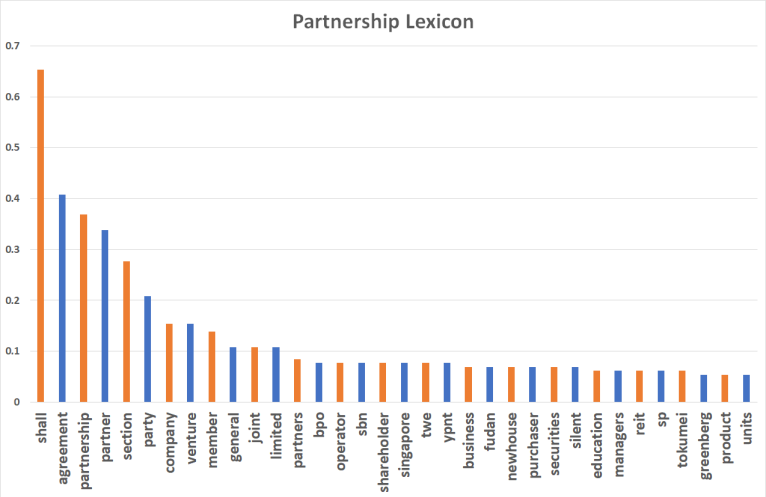

Figure 3.2: An Example of the Lexicon with the associated Scores.

4.2 Test Bench Results

A Windows 7 Virtual Machine is created for the proof of concept. It holds 50 noise documents and 10 contracts are added. After running the algorithm, 15 files are selected. Among those, there were six false positives, where 5 are the noise files and 1 is the Windows configuration file.

For example, a Python "README.txt" note is the following: "*This directory exists so that 3rd* ***party*** *packages can be installed here. Read the source for site.py for more details.*". It contains a single occurrence of the keyword **party**. However, the final score depends on the file size. Since it is only 121 bytes, relatively small compared to other texts on the file system, therefore, it is associated with a high

score.

The noise documents are the following:

- Storm.docx represents a documentation of Storm a free and open source distributed real-time computation system.
- Food-processing.docx is a review of the efforts and inventions in field of emerging food processing technologies since their inception to present day.
- TF-IDF.docx is an article explaining the TF-IDF formula and its advantages.
- Csquotes.docx provides explanation about the csquotes package used in Latex.
- Localization-bieber.docx represents the special string used in Latex.

Document	Keywords
Storm.docx	“section”, “application”, “online”, and “programs”
Food-processing.docx	“company”, “material”, “date”, and “investment”
TF-IDF.docx	“section”, “documents”, “management”, and “limited”
Csquotes.docx	“section”, “closing”, “boundary”, and “units”
Localization-bieber.docx	“company”, “article”, “series”, and “advance”

Table 3.7: Retrieved Keywords from the Noise Documents.

Specific Windows file system path can be removed from the file traversal, in addition to “README.txt” related documents to gain better results. A limitation of extracting keywords from documents using solely TF-IDF is the absence of the semantic context. As seen previously, a “README” document or documentation can contain some specific keywords like “party”, however, is not relevant to contracts. Table 3.7 shows some selected keywords contained in the noise documents. Even though different contexts and semantics are considered (food processing or latex explanation), they still carry some information related to contracts. Therefore, the use of the sequence of n-keywords for a given topic provides an accurate context of the document being analyzed, reducing false positives.
Besides, since each word is considered independently, multiple scores can be attributed to the same root of the keyword. For example, “parties” score is 0.1062 whereas “party” score is 0.2804. It is convenient to reduce inflected (or derived) words to their root form and remove inflectional endings to assign a unique score to the root of the word. Those two concepts are defined as Stemming and Lemmatization [215–217].

5 Protecting User Assets

Protection against malware attacks, especially zero-days, is a challenge for all researchers. Residual risk remains: de facto, despite various countermeasures employed by a party, an attacker can always find a way to penetrate the system (he/she still risks to be detected). If committed, anyone can reach their malicious intent. However, our goal is to complicate the intrusion task and to identify it if possible. Users should know the existent vulnerabilities to see what patches can be used to circumvent malevolent attacks. Some countermeasure can be deployed by users to protect their data from being exfiltrated.

1. Honeypot Folders: They can be created in any environment, regardless of the operating system used. Since Doxware traverses the whole file system looking for assets, any process or thread that passes through this lure folder can be immediately flagged then stopped. A drawback would be malware’s multi-threading techniques, it can still be exposed but after a certain epsilon time.
2. Data Tainting: Sensitive data in a computer is extremely private and depends on the end-users, unlike Android OS (IMEI, GPS location,... existent on all mobiles). Therefore, a general protection model is impossible to develop in real life. However, each individual can add a layer, a taint, on his preferred/sensitive information. Thus, each exfiltration attempt over the network can be detected. Nonetheless, the explosion of tainted data may slow down the system.

3. Data Encryption: It remains a robust way adopted by the global community. Indeed, brute-forcing the encryption key takes decades. Even though an attacker acquired the encrypted files, he/she cannot menace the victims or blackmail them since no access to the decrypted data is possible.

We investigate the Honeypot Folders technique in the particular case of a Doxware attack to leak important files (contracts).

6 Honeypot in the case of a Doxware attack

In this section, we propose a honeypot-based countermeasure against Doxware attacks. First, a recapitulate of recent work done in the literature to stop ransomware using honeypot techniques is presented (previously discussed in section 3.3, Chapter 2, Part I). Then, a combination of honeypot folders and files to detect and mitigate file exfiltration (Doxware) attacks is proposed. Finally, an evaluation of the implemented proposal is done having further discussion.

6.1 Honeypot Key Elements

- An essential aspect to take into consideration while generating decoys is the location of the honeypot files and folders. After performing a static analysis of 11 ransomware samples, the authors in [96] conclude that they traverse the file system in the following order:
  1. C:/User/USER NAME/Documents
  2. C:/
  3. C:/User/USER NAME/Desktop
  4. C:/User/USER NAME/Favorites
- Since, the traverse can be in order or reverse order of Windows-1252, folders names should be created accordingly to increase the chances of being accessed first. Consequently, attacks can be stopped rapidly after their initial execution. This information is useful since Doxware represents ransomware upgraded version (section 6.2.1).
- Two types of decoy files exist Low and High Interaction. Low interaction decoy files contain arbitrary data to detect ransomware's actions on the documents, whereas high interaction decoy files contain false information to confuse the attacker and lead him to further deception mechanisms implemented on the system [98]. The latter is used in the use case since Doxware tackles the content of files. The generated decoy content relies on lexicons frequency distribution (section 6.2.2).

6.2 Proposal Overview

Our honeypot-based proposal relies on the generation of decoy folders and files. The procedure for decoy folder name generation is described in section 6.2.1. The process for decoy file content generation is described in section 6.2.2. We implement and evaluate our proposal, and offer some discussion in Section 6.3.

6.2.1 Decoy Folder Name Generation

As stated in section 6.1, three crucial elements need to be considered while developing and deploying honeypot folders: their name, location, and content.
Applications, including ransomware, carry out file operations by calling two Windows APIs, *FindFirstFile* and *FindNextFile* [218]. Thus, the following decoy name generation methodology is used to be the first folder selected in the file system traversal.

1. All the repositories in a specific path are sorted by ascending order. Then, the first and last folder are chosen in the list.

2. To create the corresponding honeypot folder, the first character in the chosen folder is selected, and represented in hexadecimal. Then, a subtraction or addition is performed to embody the first and the last folders that are traversed by malicious processes in the case of a Doxware attack.

3. The string is appended to a random word extracted from "The Brown Corpus" [219]. The first character allowed by Windows-1052 is chosen for naming repositories that is the space character (0x20) and the last one, which is the letter Å" (0x9e).

Figure 3.3 displays a list of the names of all files in the current working directory "Desktop". The first folder is "2020" and the last one "Work-Documents". They are selected as a baseline for generating decoy folders names.

$$\begin{aligned} FirstGeneratedDecoyName &= concat(␣, ␣, (hex(2) - 1), randomString) \\ &= concat(␣, ␣, 1, general) \\ &= ␣␣1general \end{aligned}$$

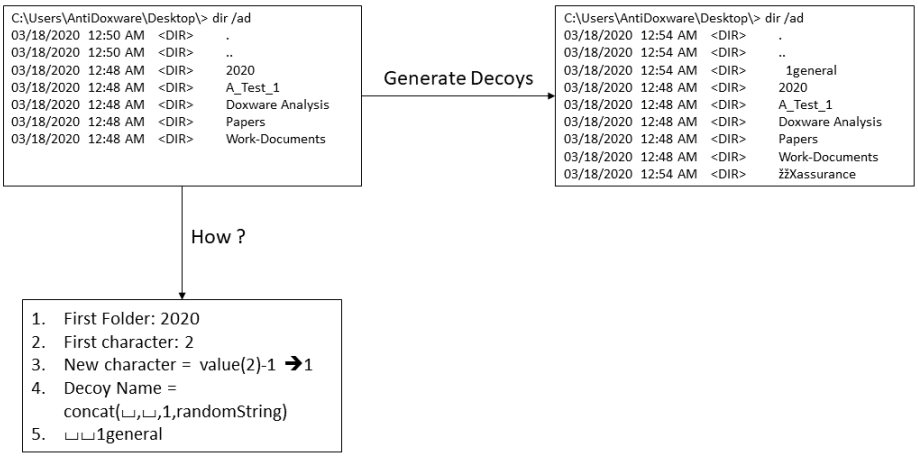

Figure 3.3: Decoy Folders Generation.

6.2.2 Decoy File Content Generation

Honeypot (or Decoy) Files are used as a proactive countermeasure against the Data exfiltration attack (i.e., Doxware malware) proposed in this section. It serves as a complementary measure to the Honeypot Folders detailed in 6.2.1.

Rationale. Before any attack, *honeypot files* that semantically resemble relevant, targeted documents are generated and stored. Those files contain no meaningful information. However, an attacker as advanced as the one of section 3 (i.e., using the VFH Algorithm) will flag those files as valuable and exfiltrate them. Hence, data with real sensitive information are less likely to be exfiltrated, as the attacker is not prioritizing them.

Honeypot Files Generation. Our proposal assumes that the VFH Algorithm is partially known. The file generation is based on the construction of a *lexicon* (section 3.3) that corresponds to the subject of the files we want to protect. A *lexicon* consist of n tuples $(word_i, weight_i)$ $i \in \{1, ..., n\}$. A $weight_i$ has a value between 0 and 1, but the sum of all weights is not 1. We propose to normalize them so this sum equals 1. This normalized weight p_i now can be interpreted as the relative frequency of $word_i$ in the

given *lexicon*. With p_i the probability mass function (pmf) of *word* taken as a discrete random variable W is constructed. Finally, a Honeypot File *HF* consists of a succession of N independent and identically distributed *word* random variables $HF = \{W_1, W_2, ..., W_N\}$.

Note. *HFs* constructed this way do not guarantee the maximum value according to the VFH Algorithm. The maximum value is obtained by the repetition of the single word with the highest weight. However, *HFs* generated with our proposal (i.e., independent and lexicon-based distributed) is more diverse and semantically complex, providing a better resemblance to reality than a single-word-repetition approach.

The decoy generation proposal is resumed in Algorithm Decoy Folders Creation (DFC).

Algorithm Decoy Folders Creation DFC

```
1: procedure ALGORITHM 1
2:     def Create_Decoy(Path path):
3:     folders ← {sorted_Windows_1252_Order(GetFolders)}
4:     GenerateDecoyNames(folders) # Section 6.2.1
5:     for f ∈ DecoyFolders do # We generate decoy files inside the folders
6:         GenerateDecoyFiles() # Section 6.2.2
7:     return DecoyFoldersPath
8:
9:     Monitor Decoy Folders
10:    if changeOccured (DecoyFolders  and / or DecoyFiles) then
11:        ALERT!!!
12:    else
13:        Continue;
```

6.3 Evaluation and Discussion

Implementation and Evaluation. We implemented a proof of concept of our proposal in Python 3 that generates Decoy Folder and Files as specified previously.

Initially, 500 decoy files of different length containing from 50 to 8000 words are generated. The score of each of the generated files is calculated as described in section 3.4. The minimum (Min), the maximum (Max), the average (Avg), and the standard deviation (σ) of the obtained scores are presented in Table 3.8.

Statistics	50-Words	200-Words	500-Words	1000-words	2000-Words	4000-Words	8000-Words
Min	13.316	15.734	16.715	17.438	17.787	17.9	18.259
Max	36.626	32.336	31.465	30.106	29.482	29.259	29.026
Avg	23.692	23.357	23.071	23.031	23.035	22.979	22.999
σ	4.717	4.133	3.967	3.963	3.935	3.892	3.888

Table 3.8: Statistics of the Scores of Different Decoy Files.

Regardless of the size and the number of words generated, the score of a specific file fluctuate in general around the value 23. If the number of the words per file increases, the minimum value increases too since the document contains more and more essential lexicons that represent an important file. The maximum value decreases since it takes into consideration the size of the data.

Discussion. The combination of having distributed decoy folders in different paths and a rich, readable content provides protection in two levels: the first one helps to flag any suspicious process passing through those decoys (it is not restricted to Doxware attacks). The content helps to mislead the attackers into believing that they possess valuable data.
If the attackers perform a recursive lookup in alphabetical order for specific files, they are instantly blocked since they passed through decoys. The Watchdog observer raises an alert immediately. Considering the attackers were able to bypass decoy folders, the data they would have collected revolves

mainly about keywords extracted from the contracts lexicon, which do not represent information of value for any given individual or company.

7 Conclusion

We discussed the potential danger of sensitive data localization and quantification that can be carried out by Doxware malware. Windows OS is the target system throughout the experiments. A proof of concept is developed based on contract topic and passwords hunt. State-of-the-art methods are used, such as TF-IDF and Bag of Words, in addition to a document's metadata. The associated score of each document is calculated then normalized. Few samples of files regarding the same topic are needed to identify new target topics. The work is published in CRiSIS conference [220], extended in a journal published in [221] . New options can be added as building bricks such as PDF and Images analysis, which will strengthen the offensive invasion from the attacker's point of view. Reducing false positive rate can be done by eliminating the Windows system path and choosing randomly N last visited files in Windows' Quick Access. Threats arising from this cyber warfare are exponential.

The ongoing work examines the strategies of various commercial anti-ransomware tools when they generate decoy files and is detailed in the following section. Additionally, future work is presented concerning Windows-based ransomware countermeasures as well as Android OS. We present the shortcomings of the latter.

4 Honeypot Ransomware Detection

Honeypot techniques have an undeniable potential in protecting users' data, yet, the generation of these repositories is not a trivial task. Re-designing decoys generation of the deception-based techniques improves the protection of the data of users, as mentioned in [99].

This Chapter presents the work in progress regarding the limitations of honeypot techniques. We evaluate the effectiveness of the decoy strategies of three commercial anti-ransomware tools. We describe their workflow process and their limitation that helps to construct robust schemes of honeypot files. Then, we analyze the features of the Master File Table (MFT) for a possible classification between decoy files and benign user files.

1 Ransomware Detection Tools

Different experiments are carried out to evaluate the configuration of the generated decoys. The configuration includes but is not limited to the number of files, their types, their location and the name assigned to the developed decoys. Then, a straightforward technique is presented to bypass the commercial tools presenting their limitation.

1.1 Environment Setup

Three baseline Windows 10 virtual machines are created that have the same configuration and the same tree of the file system presented in Figure 4.1.

Each folder displayed in the tree below contains a collection of PDF and JPG files. The PDF corpus is composed of articles and papers downloaded from Google Scholar, online forums, online courses, open-source books, contracts collected over the past two years. The images are downloaded by scrawling the WEB using keywords (top Google search queries in the US for 2020 and elements related to computer security) using Google images download [222].

1.2 Anti-Ransomware Tools

1.2.1 Padvish AntiCrypto 1.5.155.1123

It is a free tool developed by Amnpardaz Software Company [223]. It detects all types of ransomware since it relies on the behavioral aspect of the ransomware that is the encryption. Based on the AV-Test performed on this product, a 100% detection rate was achieved for real-world ransomware [224]. However, 50% of the simulated attacks had at least one file encrypted.

Padvish AntiCrypto proceeds by creating the following folders:

1. C:\!!AntiCrypto!!
2. C:\Users\decoy\Desktop\!!AntiCrypto!!
3. C:\Users\decoy\Documents\!!AntiCrypto!!

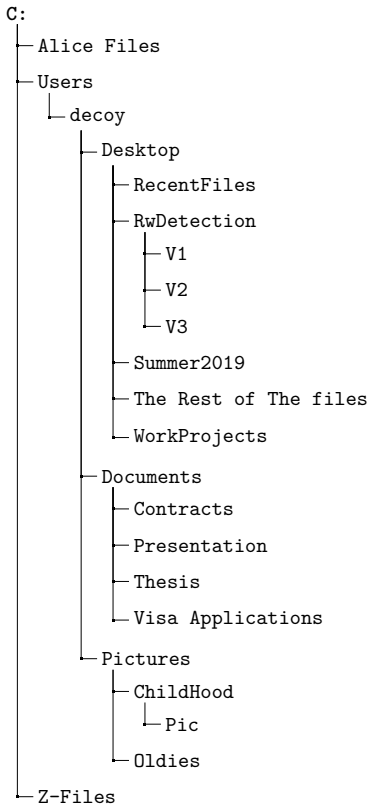

Figure 4.1: Tree of Windows' baseline file system.

Path	Permission	#C_Files	#PDF	#Doc	#Txt	#Js	#Jpg
1-3	Not Hidden	20	4	4	4	4	4

Table 4.1: Padvish Decoy Files Characteristics.

In total, 60 files are created, 20 in each of the presented directories above (Table 4.1). Each record has a unique name composed of 5 characters (capital letters).

When a recursive lookup listing the documents available on the file system is made, Padvish Crypto does not raise any alert. Nonetheless, if the *encrypt function* is called from the cryptography library of Python, AntiCrypto stops the process. So Padvish relies on the combination of decoy files and hooking to Python's cryptography library to stop ransomware. However, it does not detect the encryption made by Python binding to the Networking and Cryptography (NaCl) library.

1.2.2 CyberReason RansomFree 2.4.2.0

It is developed by CyberReason and relies in part on decoy folders/files to detect ransomware. However, it is not available since November 2018.

It proceeds by creating the following folders:

1. C:\Akdocuments
2. C:\Users\Ak6opr
3. C:\Users\Q9dsm
4. C:\Users\decoy\Documents\Bsettingsettings62
5. C:\Users\decoy\Documents\Wstores129
6. C:\Users\decoy\Desktop\0K, this directory is for Ransomware detection (just leave it here)

Path	Permission	#C_Files	#JPG	#PEM	#DOC/X	#RTF	#XLS	#MDB	#SQL	#TXT
1	Not Hidden	10	1	1	2	1	2	1	1	1
2-6	Hidden	10	1	1	2	1	2	1	1	1

Table 4.2: RansomFree 2.4.2.0 Decoy Files Characteristics.

In total, 60 files are created, 10 in each of the presented directories above (Table 4.2). Even though the same configuration is applied to all of the directories, different files are created in each one of them.

During the tests made having the countermeasure in place, it did not detect/stop any encryption on the file system.

1.2.3 AntiRansom V3

It is developed by Yago Jesus and is based on honeypot folders to detect ransomware similarly to RansomFree 2.4.2.0.

It proceeds by creating the following folder:

1. C:\Users\decoy\40JdLhZ4jP

Path	Permission	#C_Files	#PDF	#XLS
1	Hidden	37181	17031	20150

Table 4.3: AntiRansom V3 Decoy Files Characteristics.

Table 4.4 compares the three ransomware detection mechanisms.

Characteristics	Detection Tools		
	AntiCrypto	RansomFree	AntiRansom
The directories/files are created in different locations?	✓	✓	✗
The directories created start with a special character?	✓	✗	✗
Distinct repository configuration is put in place?	✗	✗	✗
The created files have comprehensible names?	✗	✓	✗
Different file formats are considered?	✓	✓	✓
The files are unusable (damaged) files?	✓	✓	✓
More than 100 files are created?	✗	✗	✓
An alert is raised when honeypot files accessed?	✗	✗	✗

Table 4.4: Ransomware Detection Mechanisms Overview.

1.3 Bypassing Decoy Folders

To bypass the decoy folders presented in Section 1.1, skipping the first and the last directory in each path is sufficient, as presented in the following algorithm Bypassing Decoys. None of the previous tools stops the encryption process of the script, making the files unavailable. Ransomware or any other potential malware evade the honeypot mechanisms implemented.

Algorithm Bypassing Decoys BD

```
1: procedure ALGORITHM 1
2:     def Encrypt_Directory (Path path):
3:     folders ← {sorted_Directories_In_Alphabetical_Order}
4:     int skipFirstNdirectories=SFD
5:     int kipLastNdirectories=SLD
6:     for i in range(SFD, len(folders)-SLD): do
7:         EncryptUserFiles()
```

2 Classification Decoy/Non Decoy

2.1 MFT

The Master File Table (MFT) is a database that stores the information about all the files and directories on a New Technology File System (NTFS) volume. Each user or system file is mapped to one or multiple records on the MFT and can contain one or more attributes. Parsing an MFT helps to retrieve information about file attributes (such as read-only and archive), time stamps (such as file creation and last modified), and the path on the file system.

Previous analysis carried out on the MFT helped detecting ransomware attacks. Kharraz *et al.* propose monitoring the changes in the MFT since data destruction is done by a ransomware, but it is represented by an unallocated memory space rather than overwriting it with random information enabling victims files recovery [110, 225].

Petya ransomware authors proceed with the encryption of the MFT to minimize the chance of recovering the files [22,226]. This work analyzes the features of an MFT for a possible decoy file identification.

2.2 Objective

The goal of this initial analysis is to check the possibility of efficiently detecting honeypot folders/files on a Windows computer. Machine learning is used on the features extracted from the Master File Table (MFT) via the execution of the Mft2Csv application [227] on the three virtual machines. The holdout method is used to evaluate different supervised ML models. Each entry in the MFT file contains 124 features. They describe the files' attributes (creation, modification, and last accessed time), permission (hidden or not), the path on the file system, its offset from the beginning of the MFT file, etc.

Some features do not contain valuable information (NaN or the same value for all the records) or fluctuate between 2 values (0 or 1), therefore, they are deleted. Then as well, redundant columns having the same value for all the records are deleted.

2.3 Supervised Learning Results

Two test bench are considered in this section.

- The first test bench consists in adding the user baseline files to the system. Afterwards, the commercial tools are deployed on a separate virtual machine. Once all the decoys are generated, the Mft2Csv application is executed.
- The second test bench is an extension of the first one, where additional user files embedded in a directory are added.

2.3.1 Test Bench Number 1

Supervised Learning Algorithm	True Positive Rate	True Negative Rate	False Positive Rate	False Negative Rate
K nearest neighbor (n=2)	100	100	0	0
Decision Tree	100	100	0	0
Random Forest	100	100	0	0

Table 4.5: AntiCrypto Classifiers Performance Metrics

The testbed 1 consists in installing the ransomware countermeasure software after deploying the user files. There is a 100% distinction between the generated decoy file and a normal one based on the two features using decision trees. Those features are:

1. ***SI_USN:*** $STANDARD_INFORMATION of Update Sequence Number (USN is a 64-bit number in Microsoft that increases whenever an object is changed).
2. ***SI_ATime:*** $STANDARD_INFORMATION of the File Modified Time.

Since all decoy files are created at the same time, they are placed one after the other in the log file, which is not the case for the user files distributed over time. In addition, the update sequence number is incremented since changes (file creation) occured in the file system (Table 4.6).

SI_USN Feature	User Files	AntiCrypto Honey Files
Min	11828560	20673776
Max	19396568	20756112
Avg	16267941.84	20725948.57
σ	2205923.41	35456.22

Table 4.6: SI_USN variations for different file types.

Another important feature that enables the classification is SI_ATime. Considering the software is installed after the baseline configuration of the files on the file system, the creation and modification time associated to those records are higher than the user files.

2.3.2 Test Bench Number 2

The second test bench relies on having the baseline system, and adding the honeypot files. Eventually, the end-user downloads, creates, and writes new files.

A new folder *05-04-2020* is created that contains the same content as *RwDetection*, with modified sub-folders names.

Each set contains 50% of the records from previously generated folders (Figure 4.1), and 50% of the *05-04-2020* folder.

After each restart, the tools presented above update the folders by maintaining the same configuration and creating new files. Hence, their timestamp is modified and corresponds to the boot time. Thus, in the second testbed scenario, the 100% classification is based on the creation and modification time of the files/folders (SI_ATime and SI_Mtime features).

Discussion

Decoy elements act as bait alerting users if they are modified. So they retain some specific characteristics throughout their deployment and post-production phase. These features are the creation and modification time, or the same last accessed or modification time. Therefore, relying solely on the SI_USN feature to detect decoy components is not an ergonomic solution since each system has its sequence of numbers. It is impossible to know a priori these numbers. File categorization is possible by inspecting the creation time and last modification time of each one of them to avoid skipping the first and last folder of each directory. If these timestamps are identical, then a decoy file is detected. Using Algorithm Bypassing Decoys V2, we could encrypt all the files without any alert risen from the software presented previously.

Algorithm Bypassing Decoys V2 BD2

```
1:  procedure ALGORITHM 2
2:      def Encrypt_Directory (Path path):
3:      ctime = creation_time
4:      atime = last_accessed_time
5:      mtime = last_modified_time
6:      folders ← { sorted_Directories_In_Alphabetical_Order }
7:      for f in path  do
8:          if  f.getctime() == f.getmtime() then
9:              pass
10:         else if f.getatime() == f.getmtime() then
11:             pass
12:         else
13:             EncryptUserFiles()
```

3 Decoy Files Recommendations

- Generally, ransomware accesses various directories to establish a proper environment for the encryption. After analyzing the logs of 90 ransomware samples found in [48] , ReadFile events occur in the following folders : C:\$Mft, C:\ProgramData\, C:\Python27\, C:\Recovery\, C:\$Recycle.Bin\. Therefore, decoy files can be placed in those folders, and if they are accessed in a short period, a potential threat is discovered. This setup should be integrated with the generation of decoys in the most used directories (Desktop, Documents, Pictures, Downloads).

- It is crucial to generate more than two decoy folders in each directory, that represent the first and the last folder traversed since they can be easily bypassed. The name of these decoys should be created in a way that increases their probability of being accessed first. Ideally, one in the beginning and at the end of each directory should be produced, combined with randomly distributed folders.

- Some ransomware authors encrypt files having a minimum size (for example, 20kb). Therefore, decoy files should have this minimum size as a baseline with variations. Having more than 100 files in each decoy folder is not mandatory since the file monitor will flag any modification attempt within a predefined threshold, thus blocking the malicious threads and processes. Hence, if a process modifies two files of the decoys generated, it will not be stopped (threshold=2 files). However, if more than two elements are altered, the process will be terminated.

- Timestamps of decoys should be modified, especially the creation, the last accessed, and modified time. By doing that, the generated files blend in seamlessly in the file system.

4 Conclusion

Needless to say that the deployment of a honeyfile structure protects to some extent end-users from any attacks, in our case, ransomware. Besides their location on the file system, it is essential to generate decoy folders with adequate names. However, there is no guarantee that the decoy will be traversed at the beginning of the infection process. Distributed repositories should be placed in the file system containing n randomly generated files to maximize the chances of succeeding using honeypot methodology. The main limitation of the tools available is their predictability in generating files in order and reverse order of Windows-1252 (character encoding of the Latin alphabet). Thus, being bypassed by spiking the first and last folder.

Honeypot techniques do not represent an ultimate countermeasure for ransomware or any other type of malware or attacks. They should be coupled with key elements specific to ransomware to flag malicious behavior (encryption, the number of Input/Output requests, the usage of cryptographic libraries, blacklisted IP addresses).

A dynamic and personalized decoy folder generator can be considered as a prototype. It provides a considerable advantage over standard generators that are easily bypassed. Our proposal is divided as follows:

- The initial phase consists of automatically analyzing the considered file system. It includes checking the number of files distributed in each directory, the file types considered (.pptx, .jpg, .txt), the files attributes (last created, modified, accessed, name, size). Based on these acquired elements, a graph representing the studied environment is constructed. It defines the location of these decoys, how folders are nested, and the distribution of the files with the adjusted attributes.

- The generated graph serves as an input to the moving target defense technique. Thus, the complexity of the attack increases in an ever-changing environment. The new files incorporated into the file system are crafted to have a higher probability of being modified first. Introducing the randomness aspect into our generator algorithm increases the chances of detecting ransomware at early stages, even if multithreading or a delayed attack strategy is in place.

5 Future Perspectives and Conclusion

1 Future Perspectives

The anti-ransomware ecosystem constantly follows the rules of cat-and-mouse, therefore, ongoing research is necessary to keep track of the metamorphic and polymorphic behavior. We present in the section, two different types of perspectives. The first one is related to Windows Ransomware workflow, and the second one tackles Android OS.

1.1 Windows-Based Ransomware Countermeasures Roadmap

A thorough analysis of the existing work in the literature mapped by inspecting actual gaps and providing research perspectives in this area are essential for the establishment of successful countermeasures. Therefore, we complement our final step of the thesis by presenting future challenges and research propositions to offer an exhaustive surface of possible ransomware countermeasures. They are associated to the four phases of the ransomware attack as discussed in Chapter 2 of Part I.

1.1.1 P1: Delivery

The aftermath of an attack depends on the existing mechanisms installed before any infection to protect the assets of a company. The delivery phase is responsible for the subsequent file losses during a ransomware attack. Research work is limited at this stage since any action taken by a user will trigger ransomware execution. Since phishing emails remain at the core of most malware attacks [20], further research on email classification will reduce the number of infected people significantly [228, 229]. Regular backups have to be put in place, stored on a different device, not accessible by ransomware, and disconnected from the internet. Signature-based detection has a limitation in detecting zero-day attacks. Obfuscated or packed malware should be dealt with instead of omitting samples from the analysis like in [67, 70] or classifying them directly as malicious software [230]. For example, in [231], the authors propose the version identification of packed malware.

1.1.2 P2: Deployment

An extensive work targeting the detection of suspicious API calls is achieved. API calls are analyzed from several perspectives. Their frequency, sequence, and total number are among multiple features taken into account as input variables for ML algorithms. At this stage, a discrepancy between datasets is noticeable since different samples have been analyzed throughout the experiments. It would be valuable for the research community to share the dataset produced by the authors to have later on a benchmark to validate or not future developed solutions. Moreover, a refined multiclass classification of ransomware samples can be studied to enhance actual binary classification (trustworthy application versus ransomware). Finally, the extracted specific ransomware patterns can be compared to other types of malware to highlight the divergence or the similarity. This clusterization facilitates applying an active response to each type of attack, containing the threat shortly after the intrusion.

1.1.3 P3: Destruction

The destruction phase represents a race against the clock. Indeed, attackers are encrypting files massively while real-time deployed countermeasures are analyzing the environment changes to block potential threats.
Some of the developed techniques do not propose an evaluation of the resources consumed during the real-time analysis (overhead) nor the performance impacts on the system. These kinds of evaluations should be addressed in future studies.
An important number of ransomware samples seen in the analyzed papers did not launch the malicious intent defined by the encryption process. A further investigation would indicate if the creation of a realistic environment (saved credentials, browser history, and realistic decoy files including images and documents [111]) is sufficient to trigger the ransomware attacks.
An online repository can be made containing the list of the blacklisted IP addresses of the C&C servers to prevent ransomware spreading. It addition, it can be updated regularly by trusted users who have predefined roles. This part englobes a web crawler that surfs malware databases downloading the latest ransomware samples, executing them in a confined or bare-metal environment, and extracting the required data to update the existing catalog of ransomware specifications.
Many statistical tools have been proposed in the literature identifying the randomness aspect of data (Chi-squared test, Kullback-Liebler divergence, Shannon's entropy), thus, capable of detecting encryption. However, Shannon's entropy is not efficient in differentiating between an image, a zipped file, or an encrypted one raising the false-positive rates. Besides, the Chi-squared test can be bypassed by bytes permutation, as shown in [101]. Consequently, future studies can evaluate the impact of different encryption schemes on the randomness of data. Consequently, an accurate classification of ransomware and a trustworthy application is achievable based on the most suitable statistical tool (including machine learning), or, if there are no, newly developed algorithms.

1.1.4 P4: Dealing

At this stage, the attacker has completed his/her tasks. Encrypted files represent a mass casualty. Few countermeasures mentioned above (for example, a possible key retrieval) are capable of recovering the data if there is no backup in place. Therefore, analyzing the initial infection phase is fundamental to prevent information loss. An essential aspect of ransomware workflow is the ransom note that displays the instructions for the victims to retrieve the decryption keys. To date and to the best of our knowledge, there are no papers published discussing ransomware detection or tracking via processing the text available in the ransom note beside the ongoing work in [232]. Blockchain is the principal technology chosen by cyberattackers since they can distribute their attacks or perform transaction pseudo-anonymously with enhanced privacy discussed in [233]. Semi-autonomous blockchain-based ransomware is developed by Delgado-Mohatar *et al.* presenting alarming facts about the usage of smart contracts, among others, if criminals will have a tight grip on this technology [234]. The automation helps to process payments and release decryption keys without the need of human operators. Further research work should endeavor to propose "patches" for possible blockchain breaches.
Lastly, covering the interactions between researchers and law enforcement agencies and the economic part would provide all the elements to constitute and end-to-end ransomware solution. The RAMSES project brings a positive initiative for ransomware tracking [235].

1.2 Mobile Ransomware

Mobile ransomware presents a major concern for end-users since they rely on their devices to accomplish their daily tasks. Currently, a mobile device is equivalent to a database containing massive sensitive information, including contacts, emails, pictures, passwords, and credit cards. Therefore, it is essential to tackle mobile ransomware defense mechanisms found in the literature. We divided them based on the mechanisms used to detect ransomware (sections 1.2.1 and 1.2.2). The proposed solutions are developed on the Android OS. Finally, we present the shortcomings of mobile ransomware in section 1.2.3.

1.2.1 API Calls

Maiorca *et al.* statically analyze the Dalvik bytecode to extract API packages found. Using API packages reduces the number of features needed for classification [236]. APIs contained in the invoke-type instructions are checked to see if they belong to system packages. The occurrences of each system API package in the Android app are counted. They serve as an input vector for the random forest learning algorithm. The mobile ransomware samples are gathered from HelDroid (public database) and VirusTotal (private database since the ransomware samples are not publicly available). R-PackDroid can be reliably used to discriminate between applications (benign, malware, and ransomware) based solely on the system APIs. However, since static analysis is used, it is not immune to dynamically loaded libraries or encrypted classes found in the executables. Besides, the authors did not mention the type of ransomware considered during the experiments (scareware, crypto or locking ransomware).

Similarly, Abdullah *et al.* collect system calls for each executed Android application [237]. VirusTotal is used to download malicious applications. The testing dataset of benign applications is downloaded from Google Play Store. A vector containing a set of features represents each application. Existing features are represented by the integer "1" in the vector whereas missing ones by "0". Fifty-two system calls are selected for the training phase (getpid, chmod, read, bind, gettimeofday, etc.). Random forest, J48, and naïve Bayes are used as classification algorithms. They successfully detect ransomware instances.

1.2.2 Multiple IOC (indicators of compromise)

DNA-Droid developed in [238] includes static and dynamic analysis needed for the detection module. The prototype is based on a combination of three components. Text classification is used for extracting extortion strings to detect ransomware. Based on the strings and sentences extracted, the APK content is categorized by topic (encrypt 20%, lock 40%, money 20%, porn 5%, and threat 15%). The image classification module detects logos used by the attackers to lure the victim into paying the ransom. The logos consist of well-known brands or agencies (Google, IKEA, Department of Justice). The API calls and permissions module extracts the list of permissions from the AndroidManifest.xml, and by decompiling the APK, it obtains API methods used in the app. Deep Auto Encoder is used to provide the score of the app's maliciousness: a value between 0 (benign) and 1(malicious). Then if an application is marked as malicious, a dynamic analysis is performed to validate or reject the previous decision. It is based on the sequence of API calls collected during the execution of the application. Multiple sequence alignment is used to distinguish between various ransomware families based on the collected APIs. Ransomware samples are collected from R-PackDroid, HellDroid, and Contagio. The benign dataset is composed of goodware samples downloaded from the Google Play store. The authors successfully detect ransomware, especially in the second round (dynamic analysis) during the first five minutes. Nonetheless, the number of lost/encrypted files is not mentioned.

Andronio *et al.* focus on analyzing Android APK files for classifying samples as scareware, locking, or encrypting ransomware [14]. Natural language processing (NLP) supervised classifier is used to detect threatening sentences. The dataset consists of the strings extracted from disassembling ransomware binaries. Then, FlowDroid, a static taint analysis, is used to detect unsolicited file-encryption operations (reading external storage and encrypting functions). Attackers can lock the mobile by calling the lockNow() function with administrator privileges. Another method consists of creating an Immortal Activity or dialog, disabling the home, back, and close functionality. By inspecting the source code of an executable, this information can be flagged. Threatening text should be found as well as a locking and/or an encrypting activity to detect ransomware. A diverse set of datasets is used during the experiments. They include AndRadar, AndroTotal, MalGenome, and VirusTotal datasets. The language barrier is a limitation as described by the authors since they search for threatening text in English. Another limitation is the evasion mechanism that does not detect the dynamically loaded libraries required to carry out the attacks.

Ferrante *et al.* present mobile ransomware detection mechanisms based on the combination of static (code inspection without running the malware) and dynamic approach (analyzing the behavior of the malware) [239]. They state that there is limited work in the literature addressing mobile ransomware. The authors extract pairs (2-grams) of opcodes found in trusted and malicious applications. They perform a feature selection to reduce the number of pairs from 10^{12} to 50 2-grams. Then, supervised machine learning (J48, NaiveBayes, and Logistic Regression) is applied to the 2-grams reduced feature

vector to classify malicious/trusted applications. To propose a runtime detection, Ferrante *et al.* use a lightweight method to monitor memory, CPU and network usage, and statistics on system calls. A sliding window is adopted to define the scope of ransomware/trusted application in the runtime behavior. Benign applications are downloaded from Google Play Store, whereas ransomware is taken from a free database HelDroid. The combination of both static and dynamic analysis provides full coverage against Android ransomware. However, their set of malicious applications is not limited to crypto-ransomware but contains also locking ransomware and ransomware using scare tactics.

Articles	Type	Approaches		Tested	Detection/Protection Mechanism
		Static	Dynamic	Solution	
[236]	API Calls	✓	X	✓	Random forest applied to the occurrences of system API packages in the Android apps to classify the executables as ransomware, malware, or trusted.
[237]	API Calls	X	✓	✓	Random forest, J48, and naïve bayes are applied to fifty-two collected system calls to detect ransomware samples.
[238]	Multiple IOC	✓	✓	✓	Applying ML on static (threatening texts, logos and API methods found in the APK) and dynamic features (sequence of API calls) to detect Android ransomware.
[14]	Multiple IOC	✓	✓	✓	Applying NLP to extract threatening texts and tracking encrypting functions and locking heuristics to detect Android ransomware.
[239]	Multiple IOC	✓	✓	✓	Applying ML on op-codes (static approach) and memory, CPU and network usage, and statistics on system calls (dynamic approach) for ransomware detection

Table 5.1: Mobile Ransomware Detection Mechanisms.

1.2.3 Discussion

Ransomware affects mobile devices too. Being a different type of operating system, we investigate the current state of the art in mobile ransomware to check if there are any similarities in the methodology used for ransomware detection. Most of the countermeasures proposed in the literature, including R-PackDroid, HELDROID, and DNA-Droid, analyze different ransomware types (scareware, encrypting, and locking ransomware) and are not limited to crypto-ransomware. Therefore, we do not have an exact percentage of mobile crypto-ransomware in the wild. HELDROID relies on taint analysis to detect encryption activity in Android devices using FlowDroid. Many techniques have been proposed to this end, especially monitoring/tracking user activity or sensitive information like Scandroid, TaintDroid [172, 240] and others [241–243]. However, current literature is not abundant with mobile ransomware journals or papers, as stated in [236, 238]. Even in the latest survey released in 2020 about ransomware in Windows and Android platforms, only six papers were mentioned related to mobile ransomware [244]. More general malware-oriented papers are reviewed. Most surveys classify mobile ransomware detection based on static, dynamic, or hybrid analysis [237, 239, 245, 246]). Model checking is also used to detect Android ransomware [247]. There is still no extensive coverage of mobile ransomware, as stated in Kaspersky's report released in 2016 and 2017. The most dangerous mobile ransomware examples (Fusob or Small) did not encrypt users' files but blocked access to the device. Besides, Kaspersky Lab experts do not believe that crypto-ransomware for mobile will undergo any noticeable development in the future due to the security features implemented recently into the Android OS, which limits the ability of third-party apps to get unlimited access to users' files [248, 249]. Therefore, we decided to focus solely on Windows ransomware in the current thesis. We will propose another classification scheme for Android ransomware

once the literature will be abundant with the proposed solutions.

2 Conclusion

This thesis focuses on Windows-based ransomware that resurged in the second decade of the 20th century. We divided it into two parts. The first one introduces ransomware as well as its workflow and the timeline of recent attacks. We break down the ransomware intrusion into four phases Delivery, Deployment, Destruction, and Dealing. Then, we associate the evaluated countermeasures developed in the literature to each of those presented phases. This classification provides the scope of the possible solutions as well as their limitations for future contributions in this field. The second part consists of three contributions. The system based contribution analyzes the file system traversal of the threads to provide a refined granularity for accurate ransomware detection. It is complemented by a network traffic examination between the PC of the victim and the server of the attacker to spot specific ransomware related malicious records. Finally, we evaluate a modified version of ransomware, called Doxware, by generating important lexicons regarding a subject and checking to which extent sensitive documents can be exfiltrated via text processing algorithms.

Let's come back to our main question on Windows-Based Ransomware: *What is Next?*

Even though ransomware has been studied thoroughly, we can not assert that this research area is "complete". In fact, new malware and ransomware emerge continually making the developed countermeasures inadequate to detect those new properties.

Numerous techniques exist to create a polymorphic version of an existent code. The authors in [61] cited NOPs operation and code obfuscation, multi-staged attack, polymorphic blending, conversion to metamorphic code, and sandbox evasion. By using these techniques, the future polymorphic and metamorphic ransomware can become untraceable and undetectable.

Genc *et al.* delved into analyzing current solutions' weaknesses and projected themselves in the future for an anticipation of potential ransomware attacks [101]. Indeed, the authors presented seven evasion techniques based on the most robust countermeasure employed in the cyberwar against ransomware. For key creation that bypasses the use pseudo-random number generator, they can be created directly from the file itself using Convergent Encryption, and memory dump will retrieve only the key being used at this specific momentum. Another counter-counter measure of file identification is skipping the first 5120 bytes of any given document, thus preventing the alteration of the magic bytes and the triggering of the alert. To evade statistical tools used to calculate the entropy or any other characteristics of the files, a simple permutation of the bytes will leave the score intact. Nevertheless, using reverse engineering techniques applied to the binary, researchers can obtain the permutation algorithm used.

Sechel assessed the effectiveness of AVs in ransomware's pandemic to disclose the code obfuscation [28]. He mentioned that advanced stealthiness techniques are rarely applied during the different phases of infection since the user will eventually know that his system was under attack. To perform evasion techniques, ransomware authors relied on packers and cryptor (compressing and encrypting the executables). The overall detection rate by VirusTotal for 11 different crypto-ransomware is 83.72%. Then, he obfuscated his source code using Themida that enables several protection features (memory guard, resource encryption, monitor blockers, exiting silently when debugger detected). The initial detection rate was 32.58%, jumped to 44.95% the following day due to an update of the VirusTotal database. Cuckoo sandbox was able to correctly identify four samples as ransomware, while the rest were malicious software based on network communication and code injection. There is no doubt that these techniques can hinder detection and/or classification by various AVs.

Besides computers, ransomware attack surface is increasing. Different types of equipments are targeted. IoT devices are known to have some drawbacks due to the limited resources provided in the hardware, the defective architecture since the point of failure affects the network [250]. Also, a ransomware attack is successfully carried out on a Canon DSLR camera [251]. The list increases with ransomware impact on SCADA systems [252]. Ransomware authors are improving their techniques and adjusting them based on the countermeasures developed in the literature. These attacks remain the number one concern for industries since economic incentive is on the line.

Ransomware attacks are evolving, and their authors find multiple ways to bypass some of the current detection mechanisms. We conclude that one specific applied countermeasure whether it is located on the system or network level is not sufficient to prevent any file loss. Some files will be eventually encrypted before an alert has risen, or the proposed solutions have some limitations that could be easily bypassed. Therefore, the ultimate solution does not exist. Researchers must combine different aspects of the ransomware behavior for building a safe countermeasure.

Appendices

A Extended French Résumé

Les cyber-attaquants ont l'habitude de monter des attaques contre les hôpitaux ; par exemple, l'attaque menée sur le centre de cardiologie d'Acadiana en avril 2017. Les attaques SamSam à la fin de l'année 2018 ont ciblé un large éventail de secteurs, mais le secteur de la santé a été de loin le plus touché. De même, la résurgence de Ryuk en 2019 a forcé les prestataires de services de santé australiens à arrêter leurs services et systèmes. Les attaques ont pour objectif un gain monétaire. Cependant, aujourd'hui, des millions de vies sont en jeu. Outre les hôpitaux, l'attaque de WannaCry fin 2017 a touché de nombreuses entreprises telles que FedEx, Nissan, des compagnies de chemin de fer en Allemagne et plus de 200 000 ordinateurs dans le monde entier. Pour toutes les raisons citées ci-dessus, nous nous sommes concentrés sur les rançongiciels, et nous les avons considérés comme un type de logiciel malveillant qui devrait être surveillé.

Cette thèse est divisée en deux parties. La revue de la littérature est présentée dans la partie I. Un examen approfondi du flux de travail des logiciels de rançon est proposé. Il comprend les vecteurs d'infection, les mécanismes de détection actuels utilisés pour protéger le système contre ces attaques, et enfin, les contre-mesures prises par les attaquants qui leur permettent de contourner les mécanismes de détection employés par les administrateurs.

1 Revue de la Littérature

Un rançongiciel est un logiciel malveillant qui retient les données des victimes en otage et procède à la libération si le paiement de la rançon est effectué à temps. De nos jours, deux types de rançongiciels peuvent infecter un ordinateur : un rançongiciel qui verrouille l'accès à l'ordinateur ou un rançongiciel de chiffrement. Le premier ne modifie pas les données mais bloque l'accès d'une personne à son ordinateur personnel. Ainsi, il est possible de récupérer des données, tandis que le rançongiciel de chiffrement chiffre des fichiers spécifiques sur le système.

La description précise du comportement d'un logiciel malveillant et la définition de ses caractéristiques nécessitent une analyse statique et dynamique. Même si chaque rançongiciel possède ses propres caractéristiques (les échantillons appartenant à la même famille peuvent aussi légèrement diverger), les étapes globales effectuées par tout logiciel de rançon sont similaires. De multiples variantes de l'attaque par rançon ont été présentées dans la littérature ; elles consistent de 4, 5 ou 6 phases [6, 19–21]. La majorité des exécutables peuvent être regroupés comme suit : une attaque multi-phases comprenant 4 phases (Pi, où i est le numéro de la phase): la distribution (P1), le déploiement (P2), la destruction (P3) et la transaction (P4). La restauration des données décrite par (P5) dépend de l'action de la victime après l'infection. Elle représente les conséquences d'une attaque. Certains utilisateurs refusent de payer la rançon pour des raisons éthiques. L'attaque est schématisée dans la Figure A.1.

P1: Distribution. Dans un premier temps, le rançongiciel recherche une vulnérabilité et s'appuie sur tous les mécanismes disponibles pour pénétrer le système cible. Zimba *et al.* présentent différents moyens d'infection par le logiciel rançon (emails spam, macros, porte dérobée, vulnérabilité "zero-day") [22, 23].

P2: Déploiement. Une fois que le logiciel malveillant s'infiltre dans le système, il charge toutes les librairies nécessaires pour réaliser sa charge utile.

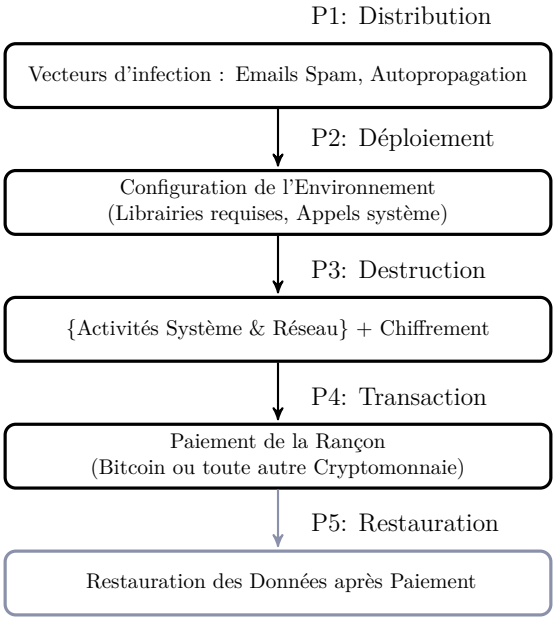

Figure A.1: Mode Operatoire du Rançongiciel.

Destruction. Ensuite, l'interrogation des divers volumes sur la machine cible par ordre alphabétique commence. Différentes extensions de fichiers sont visées : .xls, .jpg, .pdf. Certains dossiers peuvent être omis de la recherche, tels que ProgramFiles et Windows [24]. Après la recherche, le rançongiciel tente de communiquer avec le système de commande et de contrôle (C&C) pour recevoir des informations (clés de chiffrement). Le processus de chiffrement commence par des appels système à l'algorithme de chiffrement AES.

Auparavant, les attaquants optaient pour un chiffrement symétrique (AES standard). Cependant, grâce à la rétro-ingénierie, les chercheurs sont en mesure de fournir des outils de déchiffrerement pour les fichiers chiffrés puisque les auteurs de rançongiciel ont utilisé un chiffrement faible [25–27]. Par la suite, les attaquants se sont appuyés sur la combinaison de chiffrement symétrique et asymétrique pour concevoir un logiciel malveillant invincible. Chaque clé symétrique est générée localement sur la machine cible visée et permet de chiffrer un fichier spécifique ou plusieurs documents (selon l'algorithme mis en œuvre). Ensuite, la clé symétrique est chiffrée avec la clé publique de l'attaquant et ajoutée au fichier ciblé [28]. Ce schéma est connu sous le nom de cryptographie hybride.

Il existe trois types de rançongiciel ou classes de chiffrement [29, 30] :

- **Classe A** représente l'ensemble de rançongiciel qui effectuent le chiffrement sur place ; ouvreture du fichier, lecture du contenu, chiffrement puis fermeture du fichier avec possibilité de le renommer.
- **Classe B** est une autre extension où le fichier est déplacé vers un autre répertoire avant d'effectuer le chiffrement et remis dans le répertoire d'origine une fois la tâche accomplie.
- **Classe C** ouvre le fichier original, puis en crée un autre pour écrire les données chiffrées. Le fichier d'origine est supprimé.

Transaction. Enfin, la note de rançon est affichée à l'utilisateur en lui indiquant les étapes nécessaires pour récupérer les fichiers verrouillés. La plupart du temps, les auteurs de rançongiciel affichent la note de rançon à la fin de la phase d'infection car ils ne veulent pas être détectés/arrêtés pendant le processus de destruction. Habituellement, les notes de rançon sont rédigées dans la même langue que celle du PC configuré par les victimes. Elles peuvent avoir plusieurs formats comme des images, des textes et le langage de balisage hypertexte (HTML). Les notes de rançon expliquent aux utilisateurs l'attaque qui s'est produite et la procédure requise pour obtenir les clés de déchiffrement.

Restauration. À ce stade, l'attaquant a réussi à accomplir l'attaque. Le paiement ou non de la rançon reste un débat à ce jour, cependant, les chercheurs et les agences de sécurité recommandent fortement de ne pas payer la rançon [34, 35]. En fait, si la somme requise est payée, les utilisateurs soutiennent le modèle économique des criminels et sont donc partiellement responsables de l'augmentation du nombre de personnes infectées. Certaines organisations possèdent des données sensibles, et comme elles ne disposent pas de sauvegardes adéquates, elles procèdent au paiement de la rançon. Néanmoins, rien ne garantit que les attaquants leur rendront les outils (outil et clés de déchiffrement) pour restaurer les données. Le projet No More Ransom aide les victimes de rançongiciel à récupérer l'accès à leurs fichiers [36].

Une analyse approfondie permet de comprendre le fonctionnement de ces programmes malveillants et leurs future cibles. La chronologie des attaques par rançongiciel est présentée dans la Figure A.2. Elle donne un aperçu des diverses familles de rançongiciel, les dates de diffusion ainsi que leurs systèmes d'exploitation cibles. Elle représente la majorité des échantillons examinés et analysés dans la littérature.

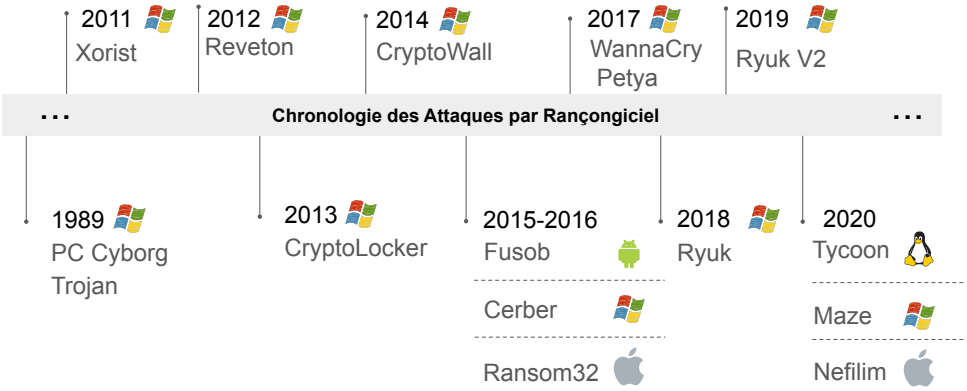

Figure A.2: Chronologie des Attaques par Rançongiciel.

Cet aperçu présente deux faits: les attaques par rançongiciel ciblent spécifiquement Windows car c'est l'un des systèmes d'exploitation le plus utilisé et les plus ergonomique. En outre, des fonctionnalités avancées sont continuellement ajoutées à la charge utile du logiciel de rançon, modifiant son comportement. Cependant, on a récemment remarqué un passage aux appareils IoT, SCADA et Android [11–14]. Nous présentons par la suite les contre-mesures de rançongiciel qui réduisent le succès de cette activité. Elles sont regroupées en fonction de la méthodologie utilisée et leur appartenance aux différentes phases de la taxonomie de l'attaque. Une classification de techniques développées dans la littérature est présentée dans la partie suivante. À notre connaissance, cet aspect n'était pas couvert auparavant dans les domaines de recherche.

P1: Distribution. Les vecteurs d'infection sont généralement difficiles à tracer car les logiciels malveillants ne sont pas analysés à la volée. Les échantillons de logiciels malveillants sont téléchargés à partir de bases de données publiques et exécutés dans un environnement "Sandbox" ou "bare-metal". L'un des meilleurs mécanismes de défense à ce stade consiste à la sensibilisation aux cyber-menaces liées au rançongiciel, par exemple en supprimant immédiatement un courrier électronique suspect. La partie sensibilisation comprend les meilleures pratiques à prendre en considération comme mesure de protection sur un ordinateur. Elle est illustrée par un système à jour comprenant les derniers correctifs et des sauvegardes fréquentes [55, 56]. La sauvegarde des données est détaillée ensuite qui est complétée par une mesure de contrôle d'accès mise en place pour protéger des dossiers spécifiques sur le système de fichiers. Pour permettre un point de restauration, le renommage des exécutables Volume Shadow Copies dans Windows est présenté. Enfin, la détection par signature qui permet de dépister les logiciels

malveillants sans qu'ils ne soient exécutés est décrite.

P2: Déploiement. L'étape suivante pour limiter l'impact des attaques par rançongiciel repose sur l'inspection des appels API. Ils montrent l'interaction entre le logiciel malveillant et l'ordinateur de la victime. De nombreux attaquants s'appuient sur les services fournis par l'API cryptographique de Microsoft pour compléter l'exécution de leur charge utile, tels que le générateur de nombres aléatoires, le chiffrement AES. L'écriture d'un code spécifique est susceptible de contenir des erreurs. Ainsi, les attaquants préfèrent utiliser des services intégrés pour accomplir leurs tâches. Par conséquent, les chercheurs ont analysé les appels API, y compris leurs modèles et leur fréquence, pour classer les processus. La surveillance des événements Windows permet d'extraire des modèles pour décrire le comportement habituel de tout utilisateur par rapport à un rançongiciel.

P3: Destruction. La phase de destruction est caractérisée par le processus de chiffrement qui affecte un nombre important de fichiers des utilisateurs. Dans un premier temps, la communication malveillante avec le C&C de l'attaquant est signalée qui représente un élément critique de l'attaque de rançongiciel. Ensuite, les contre-mesures du pot de miel sont développées pour détecter le rançongiciels qui parcours le système de fichiers pour collecter des extensions de fichiers spécifiques (.doc, .xls, .txt, .jpg). "Moving target defense" présente une technique de défense qui modifie régulièrement les extensions de fichiers en omettant les types de fichiers recherchés par le logiciel de rançon. Des opérations massives, notamment l'ouverture, la lecture et l'écriture, illustrent la phase de chiffrement. Les informations chiffrées ont une entropie plus élevée[1]. Les outils statistiques adoptés dans la littérature qui distinguent un texte non chiffré d'un texte chiffré sont aussi examinés. Certains auteurs combinent plusieurs indicateurs de compromission pour détecter les comportements malveillants. Enfin, si aucune solution temps réel ne peut arrêter le processus de chiffrement, il n'y a aucune garantie sur la possibilite de restaurer les clés de chiffrement utilisés par l'attaquant.

P4: Transaction. La dernière étape de l'attaque du rançongiciel consiste en un échange entre l'attaquant et la victime. C'est la phase la plus critique de l'intrusion. En effet, le cyber-attaquant affiche une note de rançon indiquant les étapes que l'utilisateur doit suivre pour le paiement afin de recevoir les clés de déchiffrement. Les papiers de l'état de l'art se penchent sur l'extraction et le regroupement des adresses de rançongiciel trouvés dans la blockchain. L'objectif est de surveiller le flux de bitcoins, de visualiser les transactions et de fournir une estimation des personnes infectées.

Un travail approfondi est réalisé dans la littérature couvrant tous les aspects d'une attaque de rançongiciels depuis la livraison jusqu'à la phase de distribution. Le blocage d'un rançongiciel pendant la phase de livraison n'est pas une tâche triviale. L'utilisateur est tenu responsable (en partie) à ce stade de l'exécution de l'intention malveillante de l'attaquant, même si elle a été réalisée sans le savoir ou sans le vouloir. Par conséquent, la sensibilisation réduit le risque potentiel d'être infecté par un logiciel malveillant encore inconnu. Les appels API font l'objet d'une étude approfondie lors de la phase de déploiement. Ils sont analysés en profondeur sous toutes leurs formes, y compris leur fréquence et la séquence de n-grammes implémentée dans le code du logiciel malveillant. Ainsi, nous nous sommes concentrés sur l'étape de destruction afin de fournir un prototype pour la détection des logiciels malveillants dès qu'il sont installés et qu'ils ont mis en place l'environnement requis sur la machine de la victime. Aucune étude préalable n'a été menée sur la traversée du logiciel malveillant ; par conséquent, nous présentons nos observations dans la deuxième partie de la thèse. En outre, nous analysons la communication réseau et les notes de rançon pour vérifier la chronologie des événements liés au logiciel de rançon. Enfin, nous présentons la présence d'une nouvelle menace Doxware qui pourrait rendre les données des victimes indisponibles pour eux mais ainsi partagées avec des tiers (par exemple, des concurrents).

Nous concluons la partie de revue en présentant la plateforme d'analyse dynamique que nous avons choisie, et nous donnons un aperçu des échantillons de rançongiciel acquis dans les bases de données publiques.

[1] Les échantillons récents de la famille Xorist chiffrent les fichiers en maintenant une entropie inaltérée après le chiffrement.

Dans les contre-mesures développées précédemment, les binaires de rançon sont principalement exécutés sur des machines virtuelles. En effet, les machines virtuelles sont choisies pour leur simplicité permettant l'exécution à grande échelle de nombreux échantillons. L'analyse dynamique automatisée est l'approche adoptée par les chercheurs. Il est impossible d'inspecter manuellement tous les exécutables trouvés dans les bases de données open-source. Néanmoins, de nombreux échantillons malveillants vérifient la présence de tels environnements et tentent d'échapper à la détection. Les auteurs de logiciels malveillants utilisent des techniques d'empreintes digitales pour détecter la présence d'un environnement contrôlé. La prise d'empreintes digitales consiste à vérifier des caractéristiques spécifiques connues à l'avance par les attaquants. Elle comprend les clés de registre, les processus en arrière-plan, les hook de fonction ou les adresses IP. Cependant, de nombreux échantillons testés ne fonctionnent pas correctement : aucun chiffrement n'est enregistré. En effet, certains échantillons de rançongiciel arrêtent leur exécution s'ils sont exécutés sur des machines virtuelles. Une approche pour détecter les techniques d'évasion des logiciels malveillants consiste à exécuter les échantillons sur différentes plateformes et à observer les changements de comportement. Nous expliquons nos choix pour l'adoption de la plate-forme bare-metal.

Bare-Metal (BM). Il s'agit d'un système d'analyse des logiciels malveillants qui ne peut pas être distingué d'un véritable hôte [145]. Le système d'exploitation fonctionne sur du matériel réel. Les particularités d'une plate-forme BM sont l'absence de virtualisation et la restauration efficace de l'environnement [137]. À ce jour, la plate-forme BM est le meilleur outil disponible pour l'étude des logiciels malveillants. Cependant, certaines difficultés sont apparues, par exemple, la technique de restauration du système qui peut nécessiter un redémarrage après chaque exécution est consommatrice de temps et de ressources. Elle n'est pas extensible puisque chaque échantillon est exécuté sur une machine distincte. Certaines plate-formes d'analyse bare-metal ont été proposées dans la littérature qui traitent des méthodes d'évasion des logiciels malveillants [137, 146, 147].

Le pourcentage d'échantillons Reveton qui utilisent des techniques anti-débugging est de 74,8%, et les techniques anti-machine virtuelle de 62,8% acquises lors des enquêtes menées par Chen *et al.* [148]. Cette expérience montre que plus de 50% des échantillons de rançon ne chiffrent pas les fichiers ou ne réalisent pas leur intention malveillante s'ils sont exécutés sur une machine virtuelle. Nous avons choisi de réaliser nos expériences en utilisant une plateforme bare-metal précédemment développée par Palisse *et al.* appelée MoM [17].

Dans un premier temps, nous testons le statut de chaque échantillon de rançongiciel, en vérifiant s'il effectue le processus de chiffrement ou non. Après avoir exécuté l'échantillon, une fonction de hachage vérifie l'intégrité des fichiers de l'utilisateur. Si elle ne correspond pas à la valeur de référence, l'échantillon est considéré comme actif. Ensuite, pour tous les échantillons actifs, nous définissons les actions qui doivent être prises par le système pour collecter les journaux du réseau et du système (par exemple, lancer Wireshark, exécuter le pilote du noyau).

Les noms des familles de rançongiciel les plus récents sont recueillis sur des forums en ligne, dans des bases de données de logiciels malveillants récemment mises à jour et dans des articles scientifiques. Ensuite, un logiciel de balayage (crawler) télécharge les échantillons de rançon à partir de deux bases de données : Virus Share [150] et Malwaredb.Malekal [151] (actuellement en panne). Enfin, il est exécuté sur des machines Windows 7 32 bits pendant une durée de 2 à 3 minutes. Un dump correspondant àu comportement du logiciel est enregistré pour une analyse post mortem plus approfondie. Des machines parallèles sont utilisées pour effectuer les tests. La différence entre les échantillons de rançon présente dans les expériences tout au long de la thèse est dû en partie aux familles de rançon inactives après un temps epsilon de leur diffusion.

Par exemple, dans notre dernière campagne lancée en janvier-février 2019, 1054 rançongiciels ont été exécutés sur le système d'exploitation Windows 7. **Cependant, 100 exécutables sont conservés pour la phase d'analyse puisqu'ils sont actifs** (chiffrement des fichiers de la victime). Une augmentation des échantillons inactifs est constatée à travers les expériences menées par les chercheurs. Dans le meilleur des cas, 76,8% des échantillons sont inactifs ([152]). En moyenne, 82,67% des binaires sont inactifs.

Les raisons de l'**inactivité** d'un exécutable de rançongiciel donné peuvent être l'une des suivantes.

- Le C&C ou l'adresse IP/nom de domaine externe est en panne.
- Environnement de travail inadéquat (DLLs manquantes, version Windows non adaptée).
- Le Ransomware évite d'être exécuté sur des environnements particuliers (par exemple, GandCrab évite d'infecter les pays syriens).
- Le Ransomware suspecte d'être surveillée/analysée (VM ou outils de débogage).
- L'empreinte de l'environnement de test est déjà enregistrée dans le serveur de l'attaquant.

L'étude d'un nombre considérable d'échantillons de logiciels malveillants permet de connaître davantage leur comportement. Ainsi, l'analyse dynamique permet d'établir le déroulement de tout processus exécuté. L'analyse dynamique contribue à fournir une solution ciblée pour les menaces de logiciels de rançon afin de les détecter aux premiers stades. Laplate-forme bare-metal a la plus grande probabilité de réaliser cette analyse exhaustive ; c'est la raison pour laquelle elle a été choisie dans le cadre de nos expériences. Une enquête plus approfondie est nécessaire pour indiquer les éléments appropriés qui soutiennent la création d'un environnement réaliste. Ces éléments sont examinés dans les travaux futurs, ainsi que la raison de l'inactivité de plus de 80% des exécutables. Nous procédons en présentant dans la deuxième partie de la thèse, les contre-mesures développées dans la phase de destruction. La première technique est basée au niveau du système, la seconde aborde la partie réseau, enfin, nous présentons les menaces plausibles rencontrées par une attaque Doxware.

2 Contributions

La partie II est constituée des contributions réparties en trois chapitres.

Tout d'abord, une technique de détection de rançongiciel qui sert de système de détection d'intrusion (IDS) est présentée. Elle est basée sur une exploration du système de fichiers avant que l'attaque (chiffrement) n'ait lieu. En effet, le but de la charge utile du rançongiciel est d'explorer le système de fichiers pour trouver les fichiers à chiffrer. Cette recherche se fait depuis la racine du système de fichiers, ou directement depuis le dossier de l'utilisateur, avec une recherche en profondeur ou en largeur. Comme pour la phase d'exploration, les "threads" qui traversent le système de fichiers se comportent de manière similaire et prévisible, ce qui permet une détection rapide du rançongiciel. Certains répertoires et fichiers sont rarement visités par l'utilisateur ou par un des outils habituels du système et peuvent donc être considérés comme un piège. Si un logiciel manipule ces fichiers, il peut indiquer un accès illégal. Ils sont appelés "dossiers de leurre". Les rançongiciels n'attaquent pas les fichiers qui permettent le bon fonctionnement de la machine. En effet, l'utilisateur doit pouvoir l'utiliser pour payer la rançon. La solution proposée consiste à vérifier si un "thread" passe dans les dossiers de leurre. Si c'est le cas, il les marque et incrémente ensuite le compteur de dossiers de leurre. Si un seuil est atteint, le "thread" est reconnu comme malveillant. La liste des dossiers de leurres qui sont utilisés dans nos expériences est présentée dans la Figure A.3.

Recycle_bin ; (C:\$Recycle.Bin)
Python ; (C:\Python)
Perf_log ; (C:\PerfLogs)
Prog_data ; (C:\Prog_data)
Prog_files ; (C:\Prog_files)

Figure A.3: La liste des dossiers de leurres utilisés pour le calcul du score.

Ensuite, une amélioration de la classification précédente est faite en utilisant une approche supervisée. En plus de l'accès aux dossiers de leurres mentionnés précédemment, d'autres caractéristiques sont prises en considération pour effectuer un apprentissage supervisé sur les informations collectées. Les caractéristiques utilisées pour former les classificateurs sont les suivantes: le nombre total de répertoires explorés, la durée de traversée du système de fichiers, le nombre de sous-dossiers explorés dans le dossier de leurre actuel et finalement, le temps passé dans le premier sous-dossier exploré dans le dossier de leurre actuel. Les paramètres utilisés pour l'apprentissage des classificateurs sont présentés dans le tableau A.1.

Répertoires Explorés	Nombre total de répertoires explorés
Durée de Traversée	Durée de traversée du système de fichiers
Sous-dossiers Explorés	Le nombre de sous-dossiers explorés dans le dossier de leurre actuel (Dossiers de leurres actualisés: Recycle_bin, Perf_log, Windows, Python, Prog_data, Prog_files, Recovery)
Temps Passé	Le temps passé dans le premier sous-dossier exploré dans le dossier de leurre actuel

Table A.1: La liste des paramètres utilisés pour l'entraînement des classificateurs.

Le parcours du système de fichiers de chaque rançongiciel peut être comparé à celui des autres pour savoir s'ils appartiennent à la même famille ou s'ils partagent un code concernant l'exploration du chemin. Dans un premier temps, un graphe orienté des dossiers explorés pour chaque échantillon est construit. Ensuite, la matrice de similarité correspondant à l'ensemble de données du logiciel de rançon est calculée. Enfin, une classification des échantillons est effectuée sur la base de la matrice de similarité en utilisant une technique de regroupement hiérarchique, qui donne un dendrogramme.

Un graphe est défini comme le tuple $G = (V, E, \mu, \nu)$ où:

- V l'ensemble des nœuds
- E l'ensemble des arêtes
- η l'ensemble des nœuds couverts par un rançongiciel
- θ l'ensemble des nœ couplés avec le nombre de sous-dossiers regroupés
- $\mu : V \rightarrow \eta$
- $\nu : E \rightarrow \theta$

étant donné deux graphes $G_1(\eta_1 ; E_1)$ et $G_2(\eta_2 ; E_2)$ ayant le même nombre ou un nombre différent de nœuds et d'arêtes, la similarité des graphes détermine le degré de similarité (un nombre réel entre 0 et 1) entre ces deux graphes. Elle compte le nombre d'arêtes qui ont les mêmes source et destination dans les deux graphes. G_1 et G_2 sont considérés comme similaires si le nombre réel retourné par l'algorithme dédié est proche de 1, ou de tout autre seuil prédéfini par l'utilisateur (par exemple, 0,96). Pour calculer la matrice de similarité entre les différents graphes, on a recours à Graph-tool [161]. Pour classifier les échantillons, on utilise un regroupement hiérarchique non supervisé sur cette matrice de similarité. Les feuilles de l'arbre représentent le nom des familles des divers échantillons.

Trois des 694 logiciels de rançon ne passent que par les dossiers Perf_log et Python. Ils appartiennent aux familles Crpytxxx (1 échantillon) et Gpcode (2 échantillons). 99,56% des threads malveillants sont correctement détectés. Cela met en évidence le fait que la majorité de la collection de rançongiciel commence l'exploration du système de fichiers à partir de la racine du disque dur.
Aucun des fils bénins collectés ne passe par au moins 3 des dossiers de leurres sélectionnés. Huit applications bénignes visitent les dossiers Prog_data et Prog_files. Il s'agit de SearchProtocol (deux fois), explorer.exe, svchost.exe (deux fois), avp.exe, klnagent.exe et vapm.exe.

Tous les algorithmes d'apprentissage supervisé sont capables de faire la distinction entre un logiciel rançon et une application bienveillante en fonction de la traversée du système de fichiers. La classification binaire (bénin contre rançongiciel) est efficace dans ce cas.

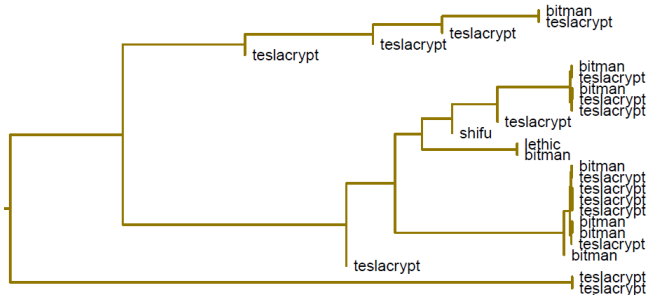

Figure A.4: Classification des familles via un dendrogramme.

Les familles qui sont proches les unes des autres sont regroupées dans une même branche. Les familles TeslaCrypt et Bitman sont semblables (Figure A.4). Cela peut s'expliquer par le fait qu'elles partagent le même algorithme de traversée. Une autre similitude est constatée entre les logiciels de rançon Cerber et Zerber. Ce dernier peut être considéré comme une simple variante du premier.

La deuxième contribution permet de repérer la même traçabilité des logiciels de rançon mais basée sur l'analyse du réseau. Un mécanisme de filtrage des données dans les fichiers PML et PCAP permet de reconstruire la session du rançongiciel. Ensuite, des modèles appropriés sont créés par apprentissage supervisé sur les flux réseau. Enfin, les notes de rançon et les fichiers chiffrés sont évalués pour vérifier si la détection s'est produite à un moment t inférieur au début du chiffrement.

Les noms des exécutables des rançongiciels sont associés à leur signature MD5 ou SHA-256. Ils représentent un identifiant unique, qui est connu avant l'exécution du logiciel rançon. Une première

recherche est effectuée sur tous les processus lancés qui ont un nom spécifique. Le nom doit être constitué de la concaténation de (`Ransomware_MD5Hash` ou `Ransomware_SHA256`) et (`.exe`).

Par conséquent, il est possible d'associer le nom du processus en cours avec l'identificateur de processus (PID) correspondant. Il s'agit d'un nombre décimal unique qui représente cet objet particulier. La collecte de tous les PIDs associés au logiciel de rançon est alors réalisée. Tout processus s'exécutant sous Windows crée de nouveaux processus pour accomplir ses tâches ou pour paralléliser la charge de travail. Dans le cas du rançongiciel, un "thread" est créé pour lister des fichiers, un autre pour le chiffrement. Pour cette raison, le graphe des processus en cours est essentiel puisqu'il affiche la relation entre tous ces processus.

Ainsi, un premier filtrage du fichier PML peut être effectué. Il est divisé en un journal malveillant qui comprend toutes les actions effectuées par le logiciel de rançon et un second fichier qui n'implique que des enregistrements bénins. Les informations recueillies dans le fichier PML sont utilisées pour extraire uniquement la communication réseau des journaux du PCAP.

L'activité de réseau acquise précédemment dans la session du rançongiciel est basique. Elle n'englobe que les adresses IP source et destination, les numéros de port et la longueur du paquet trouvé dans le fichier PML, tandis que des caractéristiques supplémentaires peuvent être extraites d'un fichier PCAP telles que la taille de la fenêtre TCP, la longueur de l'en-tête. Nous procédons en capturant les adresses IP et les numéros de port utilisés par le rançongiciel, puis nous filtrons le fichier PCAP en fonction des données obtenues précédemment.

L'apprentissage supervisé permet de créer différents modèles pour détecter le trafic réseau suspicieux. L'analyse est subdivisée en deux parties en fonction du protocole utilisé (TCP ou UDP).

Finalement, les fichiers chiffrés ainsi que les notes de rançon servent de preuves qui caractérisent la présence d'un logiciel de rançon. Si la détection basée sur l'analyse du trafic réseau a lieu avant le début du processus de chiffrement, les pertes de fichiers sont épargnées. Par conséquent, il est essentiel de retrouver l'heure du dernier paquet envoyé et du début des notes de rançon. Cela laisse quelques nanosecondes au mécanisme de prévention pour prendre une décision (bloquer le processus ou geler le PC) avant toute perte de fichier.

L'arbre de décision fournit les meilleurs résultats en termes de (vrai | faux) (positif | négatif) et temps d'apprentisage. Il permet d'éviter les problèmes potentiels d'overfitting en utilisant une forêt d'arbres décisionnels.

Les logiciels de rançon ont évolué au fil des ans et sont polymorphes. Les échantillons précédents communiquaient via un trafic HTTP non chiffré (requêtes TCP), puis d'autres familles sont passées aux requêtes GET. Shade ransomware, par exemple, utilise uniquement le protocole TLS pour sa communication. En outre, il a été l'un des pionniers de la communication IPv6. En 2016, la communication UDP a vu le jour. Sur la base des données recueillies, de nouvelles variantes de rançongiciel peuvent être détectées si la divergence entre les nouveaux échantillons et les échantillons existants est faible. Cependant, de nombreux cas sont couverts par nos travaux.

Des tests sont également effectués sur 18 échantillons de Cerber, Zerber, TeslaCrypt et Bitman sans aucune connexion Internet. Le chiffrement a quand même eu lieu. Néanmoins, les clés sont générées localement, ce qui nous permet de les récupérer par un simple hook à la Cypto API de Windows ou sont codées en dur dans l'exécutable de ransomware, ce qui est très peu probable. Deux échantillons identiques de ransomware se trouvent dans Bitman/TeslaCrypt et deux autres dans Cerber/Zerber. Cela dénote une ressemblance entre ces familles. Par exemple, le MD5 2d2c589941dd477acc60f6b8433c3562 est signalé comme Bitman par 7 antivirus et comme TeslaCrypt par 8 autres [88]. Ils sont conservés pour la détection basée sur la signature (pas d'enregistrements en double dans la même famille car elle n'apparaît qu'une seule fois) mais sont retirés de l'analyse "zero day".

Comme chaque échantillon fournit une note de rançon distincte ou une extension de fichier spécifique représentant le logiciel de rançon, tous les fichiers PML sont analysés manuellement pour extraire les informations requises.

Des exemples de noms de notes de rançon de Bitman sont donnés ci-dessous :

- Recovery+ysddu.png,
- +-HELP-RECOVER-+bjkkl-+.png,

- _ReCoVeRy_+ioigp.png,
- help_recover_instructions+drl.txt,
- +-HELP-RECOVER-+wnonv-+.png.

Enfin, nous proposons un aperçu des attaques plausibles, en particulier le Doxware (appelé aussi leakware). Un modèle de quantification qui explore le système de fichiers Windows à la recherche de données importantes est présenté. Notre approche fournit une observation de l'évolution des logiciels malveillants au cours des dernières années.

L'algorithme d'évaluation construit dépend de certains paramètres du document que le programme analyse. Il est divisé en deux parties. La première partie est liée à l'attaquant et est caractérisée par la génération de lexiques intelligents pour se concentrer sur les sujets que l'attaquant veut exfiltrer. La deuxième partie concerne les victimes, où l'attaquant évalue les documents pour les envoyer sur le réseau. Le programme d'analyse se compose de 3 modules pour accomplir ces tâches : Génération lexicale, évaluation du contenu des documents, et évaluation des métadonnées.

Dans un premier temps, la transformation TF-IDF est appliquée à l'union des documents dans le corpus pour créér le lexique représentent ces documents. L'étape suivante repose sur la création d'une fonction qui associe chaque mot du lexique à un "score". Les lexiques sont déjà intégrés dans le code source du logiciel malveillant du côté de la victime. Ils sont utilisés pour traiter le contenu des dossiers analysés qui est combinée à une score de métadonnées pour obtenir l'évaluation finale. C'est ainsi que l'attaquant peut récupérer les documents les plus importants qui se trouvent sur l'ordinateur de la victime.

Ensuite, nous proposons une contre-mesure basée sur le pot de miel contre les attaques de doxware. Une combinaison de dossiers et de fichiers de pot de miel permet de détecter et d'atténuer les attaques par exfiltration de fichiers.

Generation des Éléments de Leurre:

- Généralement, le logiciel de rançon accède à divers répertoires pour établir un environnement approprié pour le chiffrement. Après l'analyse des journaux des 90 échantillons de rançongiciel trouvés dans [48], les événements ReadFile se produisent dans les dossiers suivants : C:\$Mft, C:\ProgramData\, C:\Python27\, C:\Recovery\, C:\$Recycle.Bin\. Par conséquent, des fichiers leurres peuvent être placés dans ces dossiers, et si on y accède dans un court laps de temps, une menace potentielle est découverte. Cette configuration doit être intégrée à la génération des leurres dans les répertoires les plus utilisés (Bureau, Documents, Images, Téléchargements).

- Il est crucial de générer plus de deux dossiers leurres dans chaque répertoire, qui représentent le premier et le dernier dossier parcouru puisqu'ils peuvent être facilement contournés. Le nom de ces leurres doit être créé de manière à augmenter la probabilité qu'ils soient consultés en premier. Idéalement, il faudrait en produire un au début et à la fin de chaque répertoire, combiné à des dossiers distribués de manière aléatoire.

- Certains auteurs de ransomware chiffrent des fichiers ayant une taille minimale (par exemple, 20kb). Par conséquent, les fichiers leurres doivent avoir cette taille minimale comme référence avec des variations. Il n'est pas obligatoire d'avoir plus de 100 fichiers dans chaque dossier leurre, car le moniteur de fichiers signalera toute tentative de modification dans un seuil prédéfini, bloquant ainsi les fils et processus malveillants. Ainsi, si un processus modifie deux fichiers des leurres générés, il ne sera pas arrêté (seuil=2 fichiers). Cependant, si plus de deux éléments sont modifiés, le processus sera arrêté.

- L'horodatage des leurres doit être modifié, en particulier la création, le dernier accès et l'heure de modification. Ainsi, les fichiers générés se dissimulent de manière transparente dans le système de fichiers.

Cette thèse porte sur les logiciels de rançon basés sur Windows qui ont resurgi dans la deuxième décennie du XXème siècle. Même si les logiciels de rançon ont fait l'objet d'une étude approfondie, nous ne pouvons pas affirmer que ce domaine de recherche est "complet". En fait, de nouveaux logiciels

malveillants et de nouveaux logiciels de rançon apparaissent continuellement, ce qui rend les contre-mesures développées inadéquates pour détecter ces nouvelles propriétés.

Outre les ordinateurs, les attaques par des logiciels de rançon sont de plus en plus fréquentes. Différents types d'équipements sont visés. Les dispositifs d'IoT sont connus pour présenter certains inconvénients, en raison des ressources limitées fournies par le matériel et de leur architecture ayant un point de défaillance unique, deviennent de plus en plus ciblés [250]. De plus, une attaque par rançon est menée avec succès sur un appareil photo numérique Canon [251]. La liste se multiplie avec l'impact des rançongiciel sur les systèmes SCADA [252]. Les auteurs de ces logiciels malveillants améliorent leurs techniques et les adaptent en fonction des contre-mesures développées dans la littérature. Ces attaques restent la préoccupation primaire des industries puisque la réputation et la perte matérielle sont en jeu.

Nous concluons qu'une contre-mesure spécifique appliquée, qu'elle soit située au niveau du système ou du réseau, ne suffit pas à prévenir toute perte de fichiers. Certains fichiers seront finalement chiffrés avant qu'une alerte ne soit déclenchée, ou les solutions proposées présentent certaines limitations qui pourraient être facilement contournées. Par conséquent, une solution ultime n'existe pas. Les chercheurs doivent combiner différents aspects du comportement du logiciel de rançon pour élaborer une contre-mesure adéquate.

Bibliography

[1] Eduardo Berrueta, Daniel Morato, Eduardo Magaña, and Mikel Izal. A survey on detection techniques for cryptographic ransomware. *IEEE Access*, 7:144925–144944, 2019.

[2] Akashdeep Bhardwaj, Vinay Avasthi, Hanumat Sastry, and GVB Subrahmanyam. Ransomware digital extortion: a rising new age threat. *Indian Journal of Science and Technology*, 9(14):1–5, 2016.

[3] HIPAA Journal. Cardiology Center of Acadiana Ransomware Attack Impacts 9,700 Patients. hipaajournal.com, April 2017.

[4] ISTR. Symantec annual report, 2017.

[5] Malwarebytes. 2019 state of malware. `https://resources.malwarebytes.com/resource/2019-state-malware-malwarebytes-labs-report/`, 2019.

[6] Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C Snoeren, and Damon McCoy. Tracking ransomware end-to-end. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 618–631. IEEE, 2018.

[7] Lawrence Abrams. Revil ransomware creates ebay-like auction site for stolen data. `https://www.bleepingcomputer.com/news/security/revil-ransomware-creates-ebay-like-auction-site-for-stolen-data/`, 2020.

[8] Miss Harshada U Salvi and Mr Ravindra V Kerkar. Ransomware: A cyber extortion. *ASIAN JOURNAL FOR CONVERGENCE IN TECHNOLOGY (AJCT)*, 2, 2016.

[9] Ibrar Yaqoob, Ejaz Ahmed, Muhammad Habib ur Rehman, Abdelmuttlib Ibrahim Abdalla Ahmed, Mohammed Ali Al-garadi, Muhammad Imran, and Mohsen Guizani. The rise of ransomware and emerging security challenges in the internet of things. *Computer Networks*, 129:444–458, 2017.

[10] Pavol Zavarsky, Dale Lindskog, et al. Experimental analysis of ransomware on windows and android platforms: Evolution and characterization. *Procedia Computer Science*, 94:465–472, 2016.

[11] Amin Azmoodeh, Ali Dehghantanha, Mauro Conti, and Kim-Kwang Raymond Choo. Detecting crypto-ransomware in iot networks based on energy consumption footprint. *Journal of Ambient Intelligence and Humanized Computing*, 9(4):1141–1152, 2018.

[12] Syed Rameem Zahra and Mohammad Ahsan Chishti. Ransomware and internet of things: A new security nightmare. In *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 551–555. IEEE, 2019.

[13] Aaron Zimba, Zhaoshun Wang, and Hongsong Chen. Multi-stage crypto ransomware attacks: A new emerging cyber threat to critical infrastructure and industrial control systems. *Ict Express*, 4(1):14–18, 2018.

[14] Nicoló Andronio, Stefano Zanero, and Federico Maggi. Heldroid: Dissecting and detecting mobile ransomware. In *International Symposium on Recent Advances in Intrusion Detection*, pages 382–404. Springer, 2015.

[15] Mike Bond and George Danezis. A pact with the devil. In *Proceedings of the 2006 workshop on New security paradigms*, pages 77–82. ACM, 2006.

[16] Lawrence Abrams. Ransomware gangs team up to form extortion cartel, 2020.

[17] Aurélien Palisse. *Analyse et détection de logiciels de rançon.* PhD thesis, Rennes 1, 2019.

[18] Aditya Tandon and Anand Nayyar. A comprehensive survey on ransomware attack: A growing havoc cyberthreat. In *Data Management, Analytics and Innovation*, pages 403–420. Springer, 2019.

[19] Allan Liska and Timothy Gallo. *Ransomware: Defending against digital extortion.* " O'Reilly Media, Inc.", 2016.

[20] Aaron Zimba. Malware-free intrusion: a novel approach to ransomware infection vectors. *International Journal of Computer Science and Information Security*, 15(2):317, 2017.

[21] KAO Da-Yu, Shou-Ching HSIAO, and TSO Raylin. Analyzing wannacry ransomware considering the weapons and exploits. In *2019 21st International Conference on Advanced Communication Technology (ICACT)*, pages 1098–1107. IEEE, 2019.

[22] Daniele Sgandurra, Luis Muñoz-González, Rabih Mohsen, and Emil C Lupu. Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. *arXiv preprint arXiv:1609.03020*, 2016.

[23] Aaron Zimba, Luckson Simukonda, and Mumbi Chishimba. Demystifying ransomware attacks: Reverse engineering and dynamic malware analysis of wannacry for network and information security. *Zambia ICT Journal*, 1(1):35–40, 2017.

[24] D Paul Joseph and Jasmine Norman. A review and analysis of ransomware using memory forensics and its tools. In *Smart Intelligent Computing and Applications*, pages 505–514. Springer, 2020.

[25] Eugene Kolodenker, William Koch, Gianluca Stringhini, and Manuel Egele. Paybreak: defense against cryptographic ransomware. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 599–611. ACM, 2017.

[26] Aaron Zimba, Zhaoshun Wang, and Hongsong Chen. Reasoning crypto ransomware infection vectors with bayesian networks. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 149–151. IEEE, 2017.

[27] Saiyed Kashif Shaukat and Vinay J Ribeiro. Ransomwall: A layered defense system against cryptographic ransomware attacks using machine learning. In *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, pages 356–363. IEEE, 2018.

[28] Sergiu Sechel. A comparative assessment of obfuscated ransomware detection methods. *Informatica Economica*, 23(2):45–62, 2019.

[29] Nolen Scaife, Henry Carter, Patrick Traynor, and Kevin RB Butler. Cryptolock (and drop it): stopping ransomware attacks on user data. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 303–312. IEEE, 2016.

[30] Julian Wolf. Ransomware detection. `https://julian-wolf.eu/work/ransowaredetection/`, 2017.

[31] Harlan Carvey. The windows registry as a forensic resource. *Digital Investigation*, 2(3):201–205, 2005.

[32] Jennifer Mankin. *Classification of malware persistence mechanisms using low-artifact disk instrumentation.* PhD thesis, Citeseer, 2013.

[33] Maxat Akbanov, Vassilios G Vassilakis, and Michael D Logothetis. Wannacry ransomware: Analysis of infection, persistence, recovery prevention and propagation mechanisms. *Journal of Telecommunications & Information Technology*, (1), 2019.

[34] Cath Everett. Ransomware: to pay or not to pay? *Computer Fraud & Security*, 2016(4):8–12, 2016.

[35] Edward Cartwright, Julio Hernandez Castro, and Anna Cartwright. To pay or not: game theoretic models of ransomware. *Journal of Cybersecurity*, 5(1), 2019.

[36] Law enforcement and IT Security companies. No more ransom! `https://www.nomoreransom.org/`, 2016.

[37] Lawrence Abrams. Teslacrypt shuts down and releases master decryption key. `https://www.bleepingcomputer.com/news/security/teslacrypt-shuts-down-and-releases-master-decryption-key/`, 2016.

[38] Kaspersky. Ransomware is kaspersky lab's story of the year 2016. `https://www.kaspersky.com/about/press-releases/2016_attacks-on-business-now-equal-one-every-40-seconds`, 2016.

[39] James Wyke and Anand Ajjan. The current state of ransomware. *SOPHOS. A SophosLabs Technical Paper*, 2015.

[40] Chih-Yuan Yang and Ravi Sahita. Towards a resilient machine learning classifier–a case study of ransomware detection. *arXiv preprint arXiv:2003.06428*, 2020.

[41] Sharma Divya Mukesh. An analysis technique to detect ransomware threat. In *2018 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–5. IEEE, 2018.

[42] John MacRae and Virginia NL Franqueira. On locky ransomware, al capone and brexit. In *International Conference on Digital Forensics and Cyber Crime*, pages 33–45. Springer, 2017.

[43] Ahmad O Almashhadani, Mustafa Kaiiali, Sakir Sezer, and Philip O'Kane. A multi-classifier network-based crypto ransomware detection system: A case study of locky ransomware. *IEEE Access*, 7:47053–47067, 2019.

[44] Ade Kurniawan and Imam Riadi. Detection and analysis cerber ransomware based on network forensics behavior. *International Journal of Network Security*, 20(5):836–843, 2018.

[45] Avast. Cerber ransomware: Everything you need to know. `https://www.avast.com/c-cerber`, 2020.

[46] Jan Stiborek, Tomáš Pevnỳ, and Martin Rehák. Probabilistic analysis of dynamic malware traces. *Computers & Security*, 74:221–239, 2018.

[47] Routa Moussaileb, Benjamin Bouget, Aurélien Palisse, Hélène Le Bouder, Nora Cuppens, and Jean-Louis Lanet. Ransomware's early mitigation mechanisms. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, page 2. ACM, 2018.

[48] Routa Moussaileb, Nora Cuppens, Jean-Louis Lanet, and Hélène Le Bouder. Ransomware network traffic analysis for pre-encryption alert. In *International Symposium on Foundations and Practice of Security*, pages 20–38. Springer, 2019.

[49] Anthony Arrott, Arun Lakhotia, Ferenc Leitold, and Charles LeDoux. Cluster analysis for deobfuscation of malware variants during ransomware attacks. In *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, pages 1–9. IEEE, 2018.

[50] CERT-MU. The petya cyberattack. `http://cert-mu.govmu.org/`, 2017.

[51] MS17-010 Vulnerability. https://docs.microsoft.com/en-us/security-updates/securitybulletins/2017/ms17-010, 2017.

[52] Nina-Birte Schirrmacher, Jan Ondrus, and Ter Chian Felix Tan. Towards a response to ransomware: Examining digital capabilities of the wannacry attack. In *PACIS*, page 210, 2018.

[53] Sana Aurangzeb, Muhammad Aleem, Muhammad Azhar Iqbal, and Muhammad Arshad Islam. Ransomware: A survey and trends. *Journal of Information Assurance & Security*, 6(2), 2017.

[54] Kul Prasad Subedi, Daya Ram Budhathoki, and Dipankar Dasgupta. Forensic analysis of ransomware families using static and dynamic analysis. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 180–185. IEEE, 2018.

[55] Jordan W Han, Ong J Hoe, Joseph S Wing, and Sarfraz N Brohi. A conceptual security approach with awareness strategy and implementation policy to eliminate ransomware. In *Proceedings of the 2017 International Conference on Computer Science and Artificial Intelligence*, pages 222–226. ACM, 2017.

[56] Shaunak Sanjay Ganorkar and Kamalanathan Kandasamy. Understanding and defending cryptoransomware. *ARPN Journal of Engineering and Applied Sciences*, 12(12):3920–3925, 2017.

[57] Kanwalinderjit K Gagneja. Knowing the ransomware and building defense against it-specific to healthcare institutes. In *2017 Third International Conference on Mobile and Secure Services (MobiSecServ)*, pages 1–5. IEEE, 2017.

[58] P Raunak and P Krishnan. Network detection of ransomware delivered by exploit kit. *ARPN Journal of Engineering and Applied Sciences*, 12:3885–3889, 06 2017.

[59] P. L. Gallegos-Segovia, J. F. Bravo-Torres, V. M. Larios-Rosillo, P. E. Vintimilla-Tapia, I. F. Yuquilima-Albarado, and J. D. Jara-Saltos. Social engineering as an attack vector for ransomware. In *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pages 1–6, Oct 2017.

[60] Vadim Kotov and Fabio Massacci. Anatomy of exploit kits. In *International symposium on engineering secure software and systems*, pages 181–196. Springer, 2013.

[61] Navneet Kaur Popli and Anup Girdhar. Behavioural analysis of recent ransomwares and prediction of future attacks by polymorphic and metamorphic ransomware. In *Computational Intelligence: Theories, Applications and Future Directions-Volume II*, pages 65–80. Springer, 2019.

[62] Jason Castiglione and Dusko Pavlovic. Dynamic distributed secure storage against ransomware. *IEEE Transactions on Computational Social Systems*, 2019.

[63] Muhammet Baykara and Baran Sekin. A novel approach to ransomware: Designing a safe zone system. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pages 1–5. IEEE, 2018.

[64] Anjali Kumari, Md Zakirul Alam Bhuiyan, Jigyasa Namdeo, Shipra Kanaujia, Ruhul Amin, and Satyanarayana Vollala. Ransomware attack protection: A cryptographic approach. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, pages 15–25. Springer, 2019.

[65] Mattias Weckstén, Jan Frick, Andreas Sjöström, and Eric Järpe. A novel method for recovery from crypto ransomware infections. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pages 1354–1358. IEEE, 2016.

[66] Daniel Nieuwenhuizen. A behavioural-based approach to ransomware detection. *Whitepaper. MWR Labs Whitepaper*, 2017.

[67] May Medhat, Samir Gaber, and Nashwa Abdelbaki. A new static-based framework for ransomware detection. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 710–715. IEEE, 2018.

[68] Yara rules. `https://yara.readthedocs.io/en/latest/`, 2014.

[69] Baoli Li and Liping Han. Distance weighted cosine similarity measure for text classification. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 611–618. Springer, 2013.

[70] Deepti Vidyarthi, CRS Kumar, Subrata Rakshit, and Shailesh Chansarkar. Static malware analysis to identify ransomware properties. *International Journal of Computer Science Issues (IJCSI)*, 16(3):10–17, 2019.

[71] Nitin Naik, Paul Jenkins, Nick Savage, and Longzhi Yang. Cyberthreat hunting-part 1: Triaging ransomware using fuzzy hashing, import hashing and yara rules. 2019.

[72] Nitin Naik, Paul Jenkins, Nick Savage, and Longzhi Yang. Cyberthreat hunting-part 2: Tracking ransomware threat actors using fuzzy hashing and fuzzy c-means clustering. 2019.

[73] Zhi-Guo Chen, Ho-Seok Kang, Shang-Nan Yin, and Sung-Ryul Kim. Automatic ransomware detection and analysis based on dynamic api calls flow graph. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, pages 196–201. ACM, 2017.

[74] Sumith Maniath, Aravind Ashok, Prabaharan Poornachandran, VG Sujadevi, AU Prem Sankar, and Srinath Jan. Deep learning lstm based ransomware detection. In *2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE)*, pages 442–446. IEEE, 2017.

[75] Yuki Takeuchi, Kazuya Sakai, and Satoshi Fukumoto. Detecting ransomware using support vector machines. In *Proceedings of the 47th International Conference on Parallel Processing Companion*, page 1. ACM, 2018.

[76] R Vinayakumar, KP Soman, KK Senthil Velan, and Shaunak Ganorkar. Evaluating shallow and deep networks for ransomware detection and classification. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 259–265. IEEE, 2017.

[77] Nikolai Hampton, Zubair Baig, and Sherali Zeadally. Ransomware behavioural analysis on windows platforms. *Journal of information security and applications*, 40:44–51, 2018.

[78] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, and Syed Zainuddin Mohd Shaid. A 0-day aware crypto-ransomware early behavioral detection framework. In *International Conference of Reliable Information and Communication Technology*, pages 758–766. Springer, 2017.

[79] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, Yuli Adam Prasetyo, Syed Zainudeen Mohd Shaid, and Asmawi Fadillah Mohd Ariffin. Zero-day aware decision fusion-based model for crypto-ransomware early detection. *International Journal of Integrated Engineering*, 10(6), 2018.

[80] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, and Syed Zainudeen Mohd Shaid. Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. *Future Generation Computer Systems*, 2019.

[81] Aurélien Palisse, Hélène Le Bouder, Jean-Louis Lanet, Colas Le Guernic, and Axel Legay. Ransomware and the legacy crypto api. In *International Conference on Risks and Security of Internet and Systems*, pages 11–28. Springer, 2016.

[82] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

[83] Sajad Homayoun, Ali Dehghantanha, Marzieh Ahmadzadeh, Sattar Hashemi, and Raouf Khayami. Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence. *IEEE transactions on emerging topics in computing*, 2017.

[84] ZiHan Wang, Xu Wu, ChaoGe Liu, QiXu Liu, and JiaLai Zhang. Ransomtracer: exploiting cyber deception for ransomware tracing. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pages 227–234. IEEE, 2018.

[85] Mohammad Mehdi Ahmadian, Hamid Reza Shahriari, and Seyed Mohammad Ghaffarian. Connection-monitor & connection-breaker: A novel approach for prevention and detection of high survivable ransomwares. In *2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, pages 79–84. IEEE, 2015.

[86] Aditya K Sood and Sherali Zeadally. A taxonomy of domain-generation algorithms. *IEEE Security & Privacy*, 14(4):46–53, 2016.

[87] Aragorn Tseng, Y Chen, Y Kao, and T Lin. Deep learning for ransomware detection. *IEICE Tech. Rep.*, 116(282):87–92, 2016.

[88] Virus Total. Virustotal-free online virus, malware and url scanner. *Online: https://www. virustotal. com/en*, 2012.

[89] Omar MK Alhawi, James Baldwin, and Ali Dehghantanha. Leveraging machine learning techniques for windows ransomware network traffic detection. In *Cyber Threat Intelligence*, pages 93–106. Springer, 2018.

[90] Karim Ganame, Marc André Allaire, Ghassen Zagdene, and Oussama Boudar. Network behavioral analysis for zero-day malware detection–a case study. In *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, pages 169–181. Springer, 2017.

[91] Greg Cusack, Oliver Michel, and Eric Keller. Machine learning-based detection of ransomware using sdn. In *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pages 1–6. ACM, 2018.

[92] Krzysztof Cabaj, Marcin Gregorczyk, and Wojciech Mazurczyk. Software-defined networking-based crypto ransomware detection using http traffic characteristics. *Computers & Electrical Engineering*, 66:353–368, 2018.

[93] Maxat Akbanov, Vassilios G Vassilakis, and Michael D Logothetis. Ransomware detection and mitigation using software-defined networking: The case of wannacry. *Computers & Electrical Engineering*, 76:111–121, 2019.

[94] Krzysztof Cabaj, Piotr Gawkowski, Konrad Grochowski, and Dawid Osojca. Network activity analysis of cryptowall ransomware. *Przeglad Elektrotechniczny*, 91(11):201–204, 2015.

[95] Iyad Kuwatly, Malek Sraj, Zaid Al Masri, and Hassan Artail. A dynamic honeypot design for intrusion detection. In *The IEEE/ACS International Conference onPervasive Services, 2004. ICPS 2004. Proceedings.*, pages 95–104. IEEE, 2004.

[96] Jeonghwan Lee, Jinwoo Lee, and Jiman Hong. How to make efficient decoy files for ransomware detection? In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, pages 208–212. ACM, 2017.

[97] Chris Moore. Detecting ransomware with honeypot techniques. In *2016 Cybersecurity and Cyberforensics Conference (CCC)*, pages 77–81. IEEE, 2016.

[98] Ahmed El-Kosairy and Marianne A Azer. Intrusion and ransomware detection system. In *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*, pages 1–7. IEEE, 2018.

[99] Ziya Alper Genç, Gabriele Lenzini, and Daniele Sgandurra. On deception-based protection against cryptographic ransomware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 219–239. Springer, 2019.

[100] Suhyeon Lee, Huy Kang Kim, and Kyounggon Kim. Ransomware protection using the moving target defense perspective. *Computers & Electrical Engineering*, 78:288–299, 2019.

[101] Ziya Alper Genç, Gabriele Lenzini, and Peter YA Ryan. Next generation cryptographic ransomware. In *Nordic Conference on Secure IT Systems*, pages 385–401. Springer, 2018.

[102] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. {UNVEIL}: A large-scale, automated approach to detecting ransomware. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 757–772, 2016.

[103] Aurélien Palisse, Antoine Durand, Hélène Le Bouder, Colas Le Guernic, and Jean-Louis Lanet. Data aware defense (dad): towards a generic and practical ransomware countermeasure. In *Nordic Conference on Secure IT Systems*, pages 192–208. Springer, 2017.

[104] Faustin Mbol, Jean-Marc Robert, and Alireza Sadighian. An efficient approach to detect torrentlocker ransomware in computer systems. In *International Conference on Cryptology and Network Security*, pages 532–541. Springer, 2016.

[105] Toshima Singh Rajput. Evolving threat agents: Ransomware and their variants. *International Journal of Computer Applications*, 164(7):28–34, 2017.

[106] Kyungroul Lee, Sun-Young Lee, and Kangbin Yim. Machine learning based file entropy analysis for ransomware detection in backup systems. *IEEE Access*, 2019.

[107] Rakshit Agrawal, Jack W Stokes, Karthik Selvaraj, and Mady Marinescu. Attention in recurrent neural networks for ransomware detection. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3222–3226. IEEE, 2019.

[108] Manaar Alam, Sarani Bhattacharya, Debdeep Mukhopadhyay, and Anupam Chattopadhyay. Rapper: Ransomware prevention via performance counters. *arXiv preprint arXiv:1802.03909*, 2018.

[109] Christopher Chew and Vimal Kumar. Behaviour based ransomware detection. In *Proceedings of 34th International Conference*, volume 58, pages 127–136, 2019.

[110] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 3–24. Springer, 2015.

[111] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. Shieldfs: a self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 336–347, 2016.

[112] Manish Shukla, Sutapa Mondal, and Sachin Lodha. Poster: Locally virtualized environment for mitigating ransomware threat. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1784–1786. ACM, 2016.

[113] Kyungroul Lee, Kangbin Yim, and Jung Taek Seo. Ransomware prevention technique using key backup. *Concurrency and Computation: Practice and Experience*, 30(3):e4337, 2018.

[114] Nilesh Kakade, Mayur Shinde, Akshay Gawali, and Ajinkya Bhoite. Java based honeypot: Intrusion detection system. 2018.

[115] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. Bitiodine: Extracting intelligence from the bitcoin network. In *International Conference on Financial Cryptography and Data Security*, pages 457–468. Springer, 2014.

[116] F secure Labs. Cryptolocker: Pac-man fever. `https://archive.f-secure.com/weblog/archives/00002642.html`, 2013.

[117] Reddit. Disturbing bitcoin virus: Encrypts (instead of deleting) victims files, then demands transaction id to decrypt proving they made a 2btc payment to attacker... quickbt received 2 separate calls about this just yesterday.. `https://www.reddit.com/r/Bitcoin/comments/1o53hl/disturbing_bitcoin_virus_encrypts_instead_of/`, 2013.

[118] Hiroki Kuzuno and Christian Karam. Blockchain explorer: An analytical process and investigation environment for bitcoin. In *2017 APWG Symposium on Electronic Crime Research (eCrime)*, pages 9–16. IEEE, 2017.

[119] Mikkel Alexander Harlev, Haohua Sun Yin, Klaus Christian Langenheldt, Raghava Mukkamala, and Ravi Vatrapu. Breaking bad: De-anonymising entity types on the bitcoin blockchain using supervised machine learning. In *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[120] Building trust in blockchains. `https://www.chainalysis.com/`, 2014.

[121] Cuneyt Gurcan Akcora, Yitao Li, Yulia R Gel, and Murat Kantarcioglu. Bitcoinheist: Topological data analysis for ransomware detection on the bitcoin blockchain. *arXiv preprint arXiv:1906.07852*, 2019.

[122] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[123] Maheshkumar Sabhnani and Gürsel Serpen. Application of machine learning algorithms to kdd intrusion detection dataset within misuse detection context. In *MLMTA*, pages 209–215, 2003.

[124] Adetunmbi A Olusola, Adeola S Oladele, and Daramola O Abosede. Analysis of kdd'99 intrusion detection dataset for selection of relevance features. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, pages 20–22. Citeseer, 2010.

[125] Reda M Elbasiony, Elsayed A Sallam, Tarek E Eltobely, and Mahmoud M Fahmy. A hybrid network intrusion detection framework based on random forests and weighted k-means. *Ain Shams Engineering Journal*, 4:753–762, 2013.

[126] H Günes Kayacik, A Nur Zincir-Heywood, and Malcolm I Heywood. Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets. In *Proceedings of the third annual conference on privacy, security and trust*, 2005.

[127] Yinhui Li, Jingbo Xia, Silan Zhang, Jiakai Yan, Xiaochuan Ai, and Kuobin Dai. An efficient intrusion detection system based on support vector machines and gradually feature removal method. *Expert Systems with Applications*, 39(1):424–430, 2012.

[128] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, CISDA'09, pages 53–58, Piscataway, NJ, USA, 2009. IEEE Press.

[129] L Dhanabal and SP Shantharajah. A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. *International Journal of Advanced Research in Computer and Communication Engineering*, 4(6):446–452, 2015.

[130] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS)*, BICT'15, pages 21–26, ICST, Brussels, Belgium, Belgium, 2016. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[131] Saurabh Mukherjee and Neelam Sharma. Intrusion detection using naive bayes classifier with feature reduction. *Procedia Technology*, 4:119 – 128, 2012. 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012.

[132] Marek Małowidzki, P Berezinski, and Michał Mazur. Network intrusion detection: Half a kingdom for a good dataset. In *Proceedings of NATO STO SAS-139 Workshop, Portugal*, 2015.

[133] Markus Ring, Sarah Wunderlich, Deniz Scheuring, Dieter Landes, and Andreas Hotho. A survey of network-based intrusion detection data sets. *Computers & Security*, 2019.

[134] Stratosphere IPS. `https://www.stratosphereips.org/l`.

[135] Robert P Goldberg. Survey of virtual machine research. *Computer*, 7(6):34–45, 1974.

[136] Markus Wagner, Fabian Fischer, Robert Luh, Andrea Haberson, Alexander Rind, Daniel A Keim, and Wolfgang Aigner. A survey of visualization systems for malware analysis. In *Eurographics Conference on Visualization (EuroVis)*, pages 105–125, 2015.

[137] Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. Barebox: efficient malware analysis on bare-metal. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 403–412, 2011.

[138] Tavis Ormandy. An empirical study into the security exposure to hosts of hostile virtualized environments, 2007.

[139] Abdul Ali. Virtual machine escapes. `https://www.researchgate.net/publication/255791810_virtual_machine_escapes`, 2013.

[140] Claudio Guarnieri. Cuckoo sandbox. `https://cuckoosandbox.org/`, 2010.

[141] ORACLE. Virtualbox. `https://www.virtualbox.org/`, 2007.

[142] Vmware. `https://www.vmware.com/`, 1998.

[143] Alexei Bulazel and Bülent Yener. A survey on automated dynamic malware analysis evasion and counter-evasion: Pc, mobile, and web. In *Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium*, pages 1–21, 2017.

[144] Anchal Sancheti. Resolving security issues in the virtual machine file system. *International Journal of Engineering Research and General Science*, 3(5):366–370, 2015.

[145] Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. Barecloud: bare-metal analysis-based evasive malware detection. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 287–301, 2014.

[146] Carsten Willems, Ralf Hund, Andreas Fobian, Dennis Felsch, Thorsten Holz, and Amit Vasudevan. Down to the bare metal: Using processor features for binary analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 189–198, 2012.

[147] Adam Wallred. Nvmtrace. `https://github.com/adamwallred/nvmtrace`, 2014.

[148] Ping Chen, Christophe Huygens, Lieven Desmet, and Wouter Joosen. Advanced or not? a comparative study of the use of anti-debugging and anti-vm techniques in generic and targeted malware. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 323–336. Springer, 2016.

[149] `https://digitalcorpora.org/`, 2011.

[150] `https://virusshare.com/`, 2012.

[151] `http://malwaredb.malekal.com`, 2012.

[152] Ziya Alper Genç, Gabriele Lenzini, and Peter YA Ryan. No random, no ransom: a key to stop cryptographic ransomware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 234–255. Springer, 2018.

[153] Jian Wu, Pradeep Teregowda, Juan Pablo Fernández Ramírez, Prasenjit Mitra, Shuyi Zheng, and C Lee Giles. The evolution of a crawling strategy for an academic document search engine: whitelists and blacklists. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 340–343, 2012.

[154] Rohan Durve and Ahmed Bouridane. Windows 10 security hardening using device guard whitelisting and applocker blacklisting. In *2017 Seventh International Conference on Emerging Security Technologies (EST)*, pages 56–61. IEEE, 2017.

[155] Python. `https://www.python.org/`.

[156] Scikit-learn. http://scikit-learn.org/.

[157] Daniel T Larose and Chantal D Larose. k-nearest neighbor algorithm. *Discovering Knowledge in Data: An Introduction to Data Mining, Second Edition*, pages 149–164, 2005.

[158] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[159] Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[160] Danai Koutra, Ankur Parikh, Aaditya Ramdas, and Jing Xiang. Algorithms for graph similarity and subgraph matching. In *Proc. Ecol. Inference Conf.*, 2011.

[161] Tiago de Paula Peixoto. Graph-tool: Efficent network analysis with Python. graph-tool.skewed.de.

[162] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.

[163] S Revathi and A Malathi. A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection. *International Journal of Engineering Research & Technology (IJERT)*, 2(12):1848–1853, 2013.

[164] Amuthan Prabakar Muniyandi, R Rajeswari, and R Rajaram. Network anomaly detection by cascading k-means clustering and c4. 5 decision tree algorithm. *Procedia Engineering*, 30:174–182, 2012.

[165] SB Kotsiantis, Dimitris Kanellopoulos, and PE Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006.

[166] Yassine Lemmou and El Mamoun Souidi. An overview on spora ransomware. In *International Symposium on Security in Computing and Communication*, pages 259–275. Springer, 2017.

[167] Y. Lemmou and E. M. Souidi. Princesslocker analysis. In *2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security)*, pages 1–10, June 2017.

[168] Ryuk Related Malware Steals Confidential Military, Financial Files. `https://www.bleepingcomputer.com/news/security/ryuk-related-malware-steals-confidential-military-financial-files/?fbclid=IwAR09GDZRRVV7C_QHDzNgf1SqfMWjdGGsfjFhg2pDRJTXp9W9mKhej9cGk-A`, 2019.

[169] Data Protection and Privacy across sectors and borders . `https://bit.ly/2D2r77M`.

[170] Zhemin Yang and Min Yang. Leakminer: Detect information leakage on android with static taint analysis. In *2012 Third World Congress on Software Engineering*, pages 101–104. IEEE, 2012.

[171] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *Acm Sigplan Notices*, 49(6):259–269, 2014.

[172] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):1–29, 2014.

[173] Mariem Graa, Nora Cuppens-Boulahia, Frédéric Cuppens, Jean-Louis Lanet, and Routa Moussaileb. Detection of side channel attacks based on data tainting in android systems. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 205–218. Springer, 2017.

[174] Mingshen Sun, Tao Wei, and John Lui. Taintart: A practical multi-level information-flow tracking system for android runtime. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 331–342. ACM, 2016.

[175] Yu Feng, Saswat Anand, Isil Dillig, and Alex Aiken. Apposcopy: Semantics-based detection of android malware through static analysis. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 576–587. ACM, 2014.

[176] Lei Xue, Chenxiong Qian, Hao Zhou, Xiapu Luo, Yajin Zhou, Yuru Shao, and Alvin TS Chan. Ndroid: Toward tracking information flows across multiple android contexts. *IEEE Transactions on Information Forensics and Security*, 14(3):814–828, 2019.

[177] Lok Kwong Yan and Heng Yin. Droidscope: Seamlessly reconstructing the {OS} and dalvik semantic views for dynamic android malware analysis. In *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*, pages 569–584, 2012.

[178] David Yu Zhu, Jaeyeon Jung, Dawn Song, Tadayoshi Kohno, and David Wetherall. Tainteraser: Protecting sensitive data leaks using application-level taint tracking. *ACM SIGOPS Operating Systems Review*, 45(1):142–154, 2011.

[179] Natalia Loginova, Elena Trofimenko, Olexander Zadereyko, and Rashid Chanyshev. Program-technical aspects of encryption protection of users' data. In *2016 13th international conference on modern problems of radio engineering, telecommunications and computer science (TCSET)*, pages 443–445. IEEE, 2016.

[180] Annarita Giani, Vincent H Berk, and George V Cybenko. Data exfiltration and covert channels. In *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V*, volume 6201, page 620103. International Society for Optics and Photonics, 2006.

[181] Faheem Ullah, Matthew Edwards, Rajiv Ramdhany, Ruzanna Chitchyan, M Ali Babar, and Awais Rashid. Data exfiltration: A review of external attack vectors and countermeasures. *Journal of Network and Computer Applications*, 101:18–54, 2018.

[182] Hao Li, Zewu Peng, Xinyao Feng, and Hongxia Ma. Leakage prevention method for unstructured data based on classification. In *International Conference on Applications and Techniques in Information Security*, pages 337–343. Springer, 2015.

[183] Sultan Alneyadi, Elankayer Sithirasenan, and Vallipuram Muthukkumarasamy. Detecting data semantic: a data leakage prevention approach. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 910–917. IEEE, 2015.

[184] Anna Cinzia Squicciarini, Giuseppe Petracca, and Elisa Bertino. Adaptive data protection in distributed systems. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 365–376, 2013.

[185] Jignesh C Doshi and Bhushan Trivedi. Hybrid intelligent access control framework to protect data privacy and theft. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1766–1770. IEEE, 2015.

[186] Xiaosong Zhang, Fei Liu, Ting Chen, and Hua Li. Research and application of the transparent data encpryption in intranet data leakage prevention. In *2009 International Conference on Computational Intelligence and Security*, volume 2, pages 376–379. IEEE, 2009.

[187] Matthew Burnside and Angelos D Keromytis. F3ildcrypt: End-to-end protection of sensitive information in web services. In *International Conference on Information Security*, pages 491–506. Springer, 2009.

[188] Salvatore J Stolfo, Malek Ben Salem, and Angelos D Keromytis. Fog computing: Mitigating insider data theft attacks in the cloud. In *2012 IEEE symposium on security and privacy workshops*, pages 125–128. IEEE, 2012.

[189] Stefan Thaler, Jerry den Hartog, and Milan Petkovic. Towards creating believable decoy project folders for detecting data theft. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 161–169. Springer, 2016.

[190] Isredza Rahmi A Hamid and Jemal H Abawajy. An approach for profiling phishing activities. *Computers & Security*, 45:27–41, 2014.

[191] Christopher Francis-Christie and Dan Lo. A combination of active and passive video steganalysis to fight sensitive data exfiltration through online video. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 371–376. IEEE, 2016.

[192] Swathi Melkundi and Chaitali Chandankhede. A robust technique for relational database watermarking and verification. In *2015 International Conference on Communication, Information & Computing Technology (ICCICT)*, pages 1–7. IEEE, 2015.

[193] Panagiotis Papadimitriou and Hector Garcia-Molina. A model for data leakage detection. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1307–1310. IEEE, 2009.

[194] Theodoros Kavallaris and Vasilios Katos. On the detection of pod slurping attacks. *computers & security*, 29(6):680–685, 2010.

[195] Yuta Ikegami and Toshihiro Yamauchi. Attacker investigation system triggered by information leakage. In *2015 IIAI 4th International Congress on Advanced Applied Informatics*, pages 24–27. IEEE, 2015.

[196] Chris Ensey. Ransomware has evolved, and its name is doxware. *DARKReading. nformationWeek Business Technology Network*, 2017.

[197] Stefan Lueders et al. Computer security: Enter the next level: Doxware. 2017.

[198] Ibrahim Nadir and Taimur Bakhshi. Contemporary cybercrime: A taxonomy of ransomware threats & mitigation techniques. In *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pages 1–7. IEEE, 2018.

[199] James A Sherer, Melinda L McLellan, Emily R Fedeles, and Nichole L Sterling. Ransonware-practical and legal considerations for confronting the new economic engine of the dark web. *Rich. JL & Tech.*, 23:1, 2016.

[200] What is Doxware? `https://www.securemac.com/blog/what-is-doxware`, 2020.

[201] Marian Stewart Bartlett, Gwen Littlewort, Mark Frank, Claudia Lainscsek, Ian Fasel, and Javier Movellan. Recognizing facial expression: machine learning and application to spontaneous behavior. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 568–573. IEEE, 2005.

[202] Xudong Sun, Pengcheng Wu, and Steven CH Hoi. Face detection using deep learning: An improved faster rcnn approach. *Neurocomputing*, 299:42–50, 2018.

[203] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[204] Carol Friedman, Pauline Kra, Hong Yu, Michael Krauthammer, and Andrey Rzhetsky. Genies: a natural-language processing system for the extraction of molecular pathways from journal articles. In *ISMB (supplement of bioinformatics)*, pages 74–82, 2001.

[205] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, 2012.

[206] Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International workshop on natural language processing for social media*, pages 1–10, 2017.

[207] Areej Al-Hassan and Hmood Al-Dossari. Detection of hate speech in social networks: a survey on multilingual corpus. In *6th International Conference on Computer Science and Information Technology*, 2019.

[208] UP Cambridge. Introduction to information retrieval. 2009.

[209] Thorsten Brants. Natural language processing in information retrieval. In *CLIN*, 2003.

[210] Hanna M Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984. ACM, 2006.

[211] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.

[212] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

[213] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):13, 2008.

[214] Daniel Gonzalez and Thaier Hayajneh. Detection and prevention of crypto-ransomware. In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pages 472–478. IEEE, 2017.

[215] Juan-Manuel Torres-Moreno. Beyond stemming and lemmatization: Ultra-stemming to improve automatic text summarization. *arXiv preprint arXiv:1209.3126*, 2012.

[216] Vimala Balakrishnan and Ethel Lloyd-Yemoh. Stemming and lemmatization: a comparison of retrieval performances. 2014.

[217] Garrett Nicolai and Grzegorz Kondrak. Leveraging inflection tables for stemming and lemmatization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1138–1147, 2016.

[218] Yun Feng, Chaoge Liu, and Baoxu Liu. Poster: A new approach to detecting ransomware with deception. In *38th IEEE Symposium on Security and Privacy*, 2017.

[219] Accessing text corpora and lexical resources, 2012.

[220] Routa Moussaileb, Charles Berti, Guillaume Deboisdeffre, Nora Cuppens, and Jean-Louis Lanet. Watch out! doxware on the way... In *International Conference on Risks and Security of Internet and Systems*, pages 279–292. Springer, 2019.

[221] Routa Moussaileb, Renzo E. Navas, and Nora Cuppens. Watch out! doxware on the way... *Journal of Information Security and Applications*, 55:102668, 2020.

[222] Google Images Download. `https://github.com/hardikvasa/google-images-download`, 2018.

[223] Padvish. `https://padvish.com/en-us/Main`, 2020.

[224] AV-Test Report. `https://www.av-test.org/fileadmin/pdf/reports/AV-TEST_Padvish_Ransomware_Test_October_2017.pdf`, 2017.

[225] Jaan Priisalu. Detection of ransomware on windows operating systems. `https://digikogu.taltech.ee/testimine/et/Download/1b254ffd-7941-49b8-b870-a6fd4ec6a6a6/LunavaratuvastamineWindowsoperatsioonissteemid.pdf`, 2016.

[226] Sharifah Yaqoub A Fayi. What petya/notpetya ransomware is and what its remidiations are. In *Information Technology-New Generations*, pages 93–100. Springer, 2018.

[227] Master File Table ($MFT) to a csv. `https://github.com/jschicht/Mft2Csv/wiki/Mft2Csv`, 2014.

[228] Ahmed Alghoul, Sara Al Ajrami, Ghada Al Jarousha, Ghayda Harb, and Samy S Abu-Naser. Email classification using artificial neural network. 2018.

[229] Nadjate Saidani, Kamel Adi, and Mohand Said Allili. A semantic-based classification approach for an enhanced spam detection. *Computers & Security*, page 101716, 2020.

[230] Edgar O Osaghae. Classifying packed programs as malicious software detected. *International Journal of Information Technology and Electrical Engineering*, 5:22–25, 2016.

[231] Leo Hyun Park, Jungbeen Yu, Hong-Koo Kang, Taejin Lee, and Taekyoung Kwon. Birds of a feature: Intrafamily clustering for version identification of packed malware. *IEEE Systems Journal*, 2020.

[232] Yassine Lemmou. Ransom note files. `https://github.com/lemmou/RansomNoteFiles`, 2019.

[233] Christos Karapapas, Iakovos Pittaras, Nikos Fotiou, and George C Polyzos. Ransomware as a service using smart contracts and ipfs. *arXiv preprint arXiv:2003.04426*, 2020.

[234] Oscar Delgado-Mohatar, José María Sierra-Cámara, and Eloy Anguiano. Blockchain-based semi-autonomous ransomware. *Future Generation Computer Systems*, 2020.

[235] RAMSES. Ramses. `https://ramses2020.eu/`, 2017.

[236] Davide Maiorca, Francesco Mercaldo, Giorgio Giacinto, Corrado Aaron Visaggio, and Fabio Martinelli. R-packdroid: Api package-based characterization and detection of mobile ransomware. In *Proceedings of the symposium on applied computing*, pages 1718–1723, 2017.

[237] Zubaile Abdullah, Farah Waheeda Muhadi, Madihah Mohd Saudi, Isredza Rahmi A Hamid, and Cik Feresa Mohd Foozy. Android ransomware detection based on dynamic obtained features. In *International Conference on Soft Computing and Data Mining*, pages 121–129. Springer, 2020.

[238] Amirhossein Gharib and Ali Ghorbani. Dna-droid: A real-time android ransomware detection framework. In *International Conference on Network and System Security*, pages 184–198. Springer, 2017.

[239] Alberto Ferrante, Miroslaw Malek, Fabio Martinelli, Francesco Mercaldo, and Jelena Milosevic. Extinguishing ransomware-a hybrid approach to android ransomware detection. In *International Symposium on Foundations and Practice of Security*, pages 242–258. Springer, 2017.

[240] Adam P Fuchs, Avik Chaudhuri, and Jeffrey S Foster. Scandroid: Automated security certification of android. Technical report, 2009.

[241] Golam Sarwar, Olivier Mehani, Roksana Boreli, and Mohamed Ali Kaafar. On the effectiveness of dynamic taint analysis for protecting against private information leaks on android-based devices. In *SECRYPT*, volume 96435, 2013.

[242] Wei Huang, Yao Dong, Ana Milanova, and Julian Dolby. Scalable and precise taint analysis for android. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis*, pages 106–117, 2015.

[243] Jie Zhang, Cong Tian, and Zhenhua Duan. Fastdroid: efficient taint analysis for android applications. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 236–237. IEEE, 2019.

[244] Abdulrahman Alzahrani, Ali Alshehri, Hani Alshahrani, and Huirong Fu. Ransomware in windows and android platforms. *arXiv preprint arXiv:2005.05571*, 2020.

[245] Usama Desai. A survey on android ransomware and its detection methods. 2019.

[246] Jithin Chandra Mohan and Renuka Kumar. On the efficacy of android ransomware detection techniques: A survey. *International Journal of Pure and Applied Mathematics*, 115(8):115–120, 2017.

[247] Mercaldo, Francesco and Nardone, Vittoria and Santone, Antonella and Visaggio, Corrado Aaron. Ransomware steals your phone. formal methods rescue it. In *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*, pages 212–221. Springer, 2016.

[248] Kaspersky Lab. Ksn report: Ransomware in 2014-2016. `https://s3-eu-west-1.amazonaws.com/khub-media/wp-content/uploads/sites/43/2018/03/07190822/KSN_Report_Ransomware_2014-2016_final_ENG.pdf`, 2016.

[249] Kaspersky Lab. Ksn report: Ransomware in 2016-2017. `https://go.kaspersky.com/rs/802-IJN-240/images/KSN_Report_Ransomware_2016-2017_ENG.PDF`, 2017.

[250] Vikas Hassija, Vinay Chamola, Vikas Saxena, Divyansh Jain, Pranav Goyal, and Biplab Sikdar. A survey on iot security: application areas, security threats, and solution architectures. *IEEE Access*, 7:82721–82743, 2019.

[251] Eyal Itkin. Say cheese: Ransomware-ing a dslr camera, 2019.

[252] Jaime Ibarra, Usman Javed Butt, Anh Do, Hamid Jahankhani, and Arshad Jamal. Ransomware impact to scada systems and its scope to critical infrastructure. In *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3)*, pages 1–12. IEEE, 2019.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Analyse de Logs pour les Besoins de Détection de Logiciels Malveillants

Mot clés : Rançongiciel, Logs Système & Réseaux, Détection, Doxware

Résumé : Les rançongiciels demeurent la menace informatique principale pour les particuliers, les entreprises et les gouvernements. Les conséquences de ces attaques peuvent causer des pertes irréversibles si les exigences des attaquants ne sont pas satisfaites à temps. Cette thèse cible les rançongiciels Windows. Ils affectent les données des utilisateurs sauvegardées sur les ordinateurs ainsi que de nombreux services publics. Quatre étapes de l'attaque des rançongiciels sont définies : infection, déploiement, destruction et transaction. Les contre-mesures sont regroupées selon les techniques utilisées et attribuées à chaque phase de l'attaque. Cette thèse présente trois contributions. Le premier mécanisme de détection est situé dans la couche du système de fichiers. Il est basé sur la traversée du système qui permet d'exposer les comportements malveillants. Cette thèse propose également une analyse du trafic réseau. Les échantillons sont collectés pour une détection au niveau des paquets. Une étude des notes de rançon est faite pour situer la contre-mesure réseau dans l'étape appropriée de l'intrusion. La dernière contribution donne un aperçu des attaques, particulièrement des Doxware. Un modèle de quantification qui explore le système de fichiers Windows à la recherche de données importantes est présenté et complémenté par les pots de miels pour protéger les fichiers sensibles. Enfin, cette thèse offre des perspectives permettant d'établir un meilleur plan d'action pour les chercheurs.

Title: Log Analysis for Malicious Software Detection

Keywords: Ransomware, System & Network Logs, Detection, Doxware

Abstract: Ransomware remains the number one cyberthreat for individuals, enterprises, and governments. Malware's aftermath can cause irreversible casualties if the requirements of the attackers are not met in time. This thesis targets Windows ransomware. It affects users' data and undermines many public services. Four stages of this malware attack are defined: delivery, deployment, destruction, and dealing. The corresponding countermeasures are assigned to each phase of the attack and clustered according to the techniques used. This thesis presents three contributions. The first detection mechanism is located in the file system layer. It is based on the system traversal that is sufficient to highlight the malicious behavior. This thesis proposes also an analysis of the network traffic. It is generated by collected ransomware samples to perform a packet-level detection. A study of the ransom notes is made to define where it takes place in a ransomware workflow. The last contribution provides an insight into plausible attacks, especially Doxware. A quantification model that explores the Windows file system in search of valuable data is presented. It is based on the term frequency-inverse document frequency solution provided in the literature for information retrieval. Honeypot techniques are also used to protect the sensitive files of the users. Finally, this thesis provides future perspectives granting a better roadmap for researchers.