A Survey On Windows-Based Ransomware Taxonomy And Detection Mechanisms: Case Closed?

Routa Moussaileb, Nora Boulahia Cuppens, Jean-Louis Lanet, Hélène Le Bouder

A Survey On Windows-Based Ransomware Taxonomy And Detection Mechanisms: Case Closed?

ROUTA MOUSSAILEB, IMT Atlantique and Inria, France
NORA CUPPENS, IMT Atlantique and Polytechnique Montréal, France & Canada
JEAN-LOUIS LANET, LHS-PEC, Inria, France
HÉLÈNE LE BOUDER, IMT Atlantique, France

Ransomware remains an alarming threat in the 21st century. It has evolved from being a simple scare tactic into a complex malware capable of evasion. Formerly, end-users were targeted via mass infection campaigns. Nevertheless, in recent years, the attackers have focused on targeted attacks since the latter are profitable and can induce severe damage. A vast number of detection mechanisms have been proposed in the literature. We provide a systematic review of ransomware countermeasures starting from its deployment on the victim machine until the ransom payment via cryptocurrency. We define four stages of this malware attack: Delivery, Deployment, Destruction, and Dealing. Then, we assign the corresponding countermeasures for each phase of the attack and cluster them by the techniques used. Finally, we propose a roadmap for researchers to fill the gaps found in the literature in ransomware's battle.

CCS Concepts: • **Security and privacy → Malware and its mitigation**;

Additional Key Words and Phrases: Ransomware, Malware, System Security

1 INTRODUCTION

Malware remains a recurring threat that affected people decades ago, and it still is in an exponential rise. The tradeoff prevails in what researchers and security analysts try to protect and attackers that always find a way around. Everything revolves around the pyramid of confidentiality, integrity, and availability (CIA). Monetary gain was an initial catalyst for cyber attackers in addition to showing their expertise in the domain. Sixteen million USD are traced back to ransomware payment via Bitcoin during a period of two years [73]. Some of the essential malware families are worms, virus, Trojan horse, logic bombs, and ransomware. This paper focuses on ransomware, our main research study.

We will have an in-depth look at ransomware's workflow. It includes the initial infection means, the current detection mechanisms employed to protect the system from such prominent attacks and finally the measures taken by attackers to bypass these solutions.

Ransomware activity has been in decline since 2018. Symantec reported a drop of 20% in their latest Internet Security Threat Report published in February 2019 [111]. However, a shift of the target is noticed where attackers are turning their focus from regular consumers to businesses. De facto, patches are provided to end-users, so ransomware will not penetrate systems using exploit kits (EKs). EKs are "tools used by cybercriminals to perform drive-by-download attacks " [38]. A drive-by download attack represents the unintended download of a computer software from the Internet. In addition, extortion is much more fruitful when applied to businesses. Indeed, they have the required funds to pay a ransom, and they have their reputation to maintain. 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 [87].

Authors' addresses: Routa Moussaileb, routa.moussaileb@imt-atlantique.fr, ruta.moussaileb@irisa.fr, IMT Atlantique and Inria, Cesson-Sévigné & Rennes, France, 35000; Nora Cuppens, nora.cuppens@imt-atlantique.fr, nora.boulahia-cuppens@polymtl.ca, IMT Atlantique and Polytechnique Montréal, Cesson-Sévigné & Montréal, France & Canada, 35510, H3T 1J4; Jean-Louis Lanet, jean-louis.lanet@inria.fr, LHS-PEC, Inria, Rennes, France, 35000; Hélène Le Bouder, IMT Atlantique, Cesson-Sévigné, France, 35510.

Besides, another propagation methodology revolves around a compelling concept, especially for Script kiddies or unexperienced attackers, Ransomware As A Service (RaaS) [17]. It provides all the necessary tools for cybercriminals to carry out their attacks without having prior knowledge of the end-to-end system nor the infrastructure of the victim's machine.

Funding is given to researchers to pursue their work on malware detection analysis to eradicate or decrease this ransomware in the wild [101].

Few surveys exist in the literature that tackle Windows ransomware. Current white papers propose a detailed ransomware anatomy based on the static and dynamic analysis performed [20, 22, 23]. Our survey provides a synthesis of ransomware detection techniques developed by the research community since 2014. We follow the guidelines for performing systematic literature reviews in software engineering described in [37]. We reviewed thoroughly papers collected from the scientific research database, including but not limited to Google Scholar, ScienceDirect, IEEE Xplore. The keywords used for this search are "ransomware, detection, countermeasure, analysis, Windows, Android", and some examples of ransomware samples like Locky, WannaCry. Besides databases, forums were also queried since they provide the latest insight in security, such as Bleeping Computer. Linux based countermeasures are omitted from this survey. We extract the methodology developed by authors in each paper and present its limitations if there are any. Common techniques are clustered to present the available solutions developed in the literature for a specific attack phase. Tested prototypes are specially taken into consideration since they give an accurate representation of the feasibility of the solution. We delve into presenting the mapping between ransomware attack phases and countermeasures. We bring to the table a classification of those elements based on ransomware's workflow. To the best of our knowledge, this aspect was not previously covered in research areas. This survey serves as an entry point to ransomware domain, including its lifecycle and the actions undertaken to thwart it. Even though this survey focuses on Windows ransomware, we address mobile-based ransomware, specifically on the Android operating system, and present our findings and the shortcoming noticed in the literature.

The paper is structured as follows: Ransomware's workflow and description are presented in Section 2. Ransomware state of the art detection mechanisms are categorized in Section 3. Polymorphic ransomware is presented in Section 4. Ransomware static and dynamic analysis are presented in Section 5. Mobile ransomware is discussed in Section 6. Finally, the conclusion is drawn in Section 7.

2 CONTEXT

We present a detailed anatomy of ransomware in Section 2.1. We describe it as a malicious software capable of damaging the files of the victims. Then, we give an overview of the different types of ransomware in the wild and the encryption scheme used. We proceed by dividing ransomware attack into four stages based on the information gathered from the literature review and our experiments carried out in the last five years. Ransomware timeline is shown in Section **??**, 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, providing a potential roadmap for future research work (see Section 3).

2.1 Ransomware's Workflow

Ransomware is a malicious software that holds the victims' data hostage and proceeds with the release if the ransom payment was made in time. Two types of ransomware can infect a computer nowadays: a locking or a crypto ransomware. Locking ransomware does not alter your data but blocks a person's access to his/her personal computer. Thus, potential

data retrieval is achievable, whereas crypto-ransomware encrypts specific files from your file system, making recovery impossible if no appropriate patches were installed on the target computer before any infection [113].

Crypto-ransomware is also divided into subparts. It depends on the encryption type used (symmetric and/or asymmetric encryption) and how it is performed. The encryption mechanism represents the malicious intent of ransomware's payload. In fact, 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.

Outlining accurately any malware behavior and defining its characteristics require both static and dynamic analysis. Even though samples belonging to the same ransomware family might slightly diverge, the overall steps performed by any given sample are similar. Multiple variations of ransomware attacks have been presented in the literature; they consist of 4, 5, or 6 phases [60, 73, 86, 123]. The majority of 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 attack is summarized in Fig. 1.

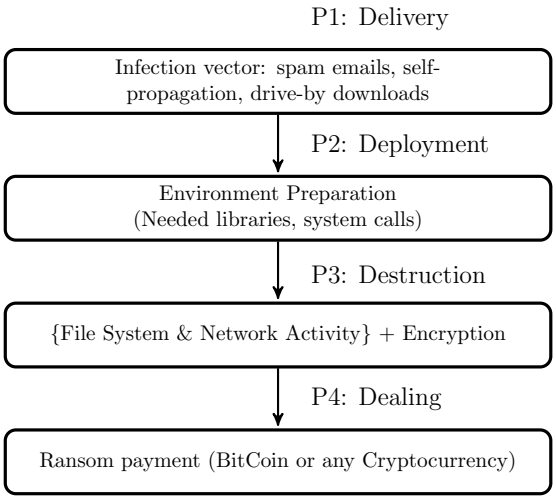

Fig. 1. Ransomware's Workflow.

Delivery: Initially, 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) [106, 110, 124].

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 to verify if they are being monitored or not.

Destruction: Then, the querying of volumes on the target machine by alphabetical order begins. Different file extensions are targeted: .xls, .jpg, .pdf. Some folders are omitted from the search, such as ProgramData and Windows [74]. After the search, 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 256 or embedded AES encrypting algorithms.

Formerly, attackers opted for symmetric encryption (standard AES). However, through reverse engineering, researchers were able to provide decryption tools for the encrypted files [78, 107, 125]. 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, this symmetric key is encrypted with the attacker's public key [105]. This scheme is known as hybrid cryptography: a masterpiece for culprits.

Three types of crypto ransomware or classes exist [104, 120]: **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 enabling the execution of ransomware every time the computer is restarted, ensuring its persistence [6, 88].

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 phase since they do not want to be detected/stopped during their destruction process.

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.

A detailed table containing ransomware families and their characteristics regarding the encryption mechanisms used, the infection process, and the infected platform type is presented in [22, 23, 110].

This overview presents two facts: ransomware attacks target specifically Windows since it is one of the most used and ergonomic OS worldwide. However, a shift to IoT, SCADA, and Android devices is recently noticed [14, 15, 121, 126].

3 RANSOMWARE DETECTION MECHANISMS

The literature is abundant with different ransomware detection mechanisms. A broad spectrum of targeted systems in the research area exists; however, our focus will be on our domain of expertise: Windows OS.

One of the main advantages of this systematic literature review is to draw a detailed anatomy of ransomware attacks since their installation on the victims' PCs until the payment is completed. It is followed by a description of 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 [53] to retrieve the encrypted files.

The following sections represent the defense mechanism taxonomy clustered corresponding to each phase of ransomware's workflow, as presented in Fig. 1. We gather the existent solutions that cover a specific phase of the intrusion and the methods deployed. The countermeasures corresponding to each phase presented in Fig. 2 are thoroughly explained in the following sections.

3.1 P1: Delivery

Infection vectors are usually hard to trace since researchers execute malware in a Sandbox or bare metal environment. One of the best defense mechanisms at the delivery stage is to raise awareness about ransomware cyber threats, for

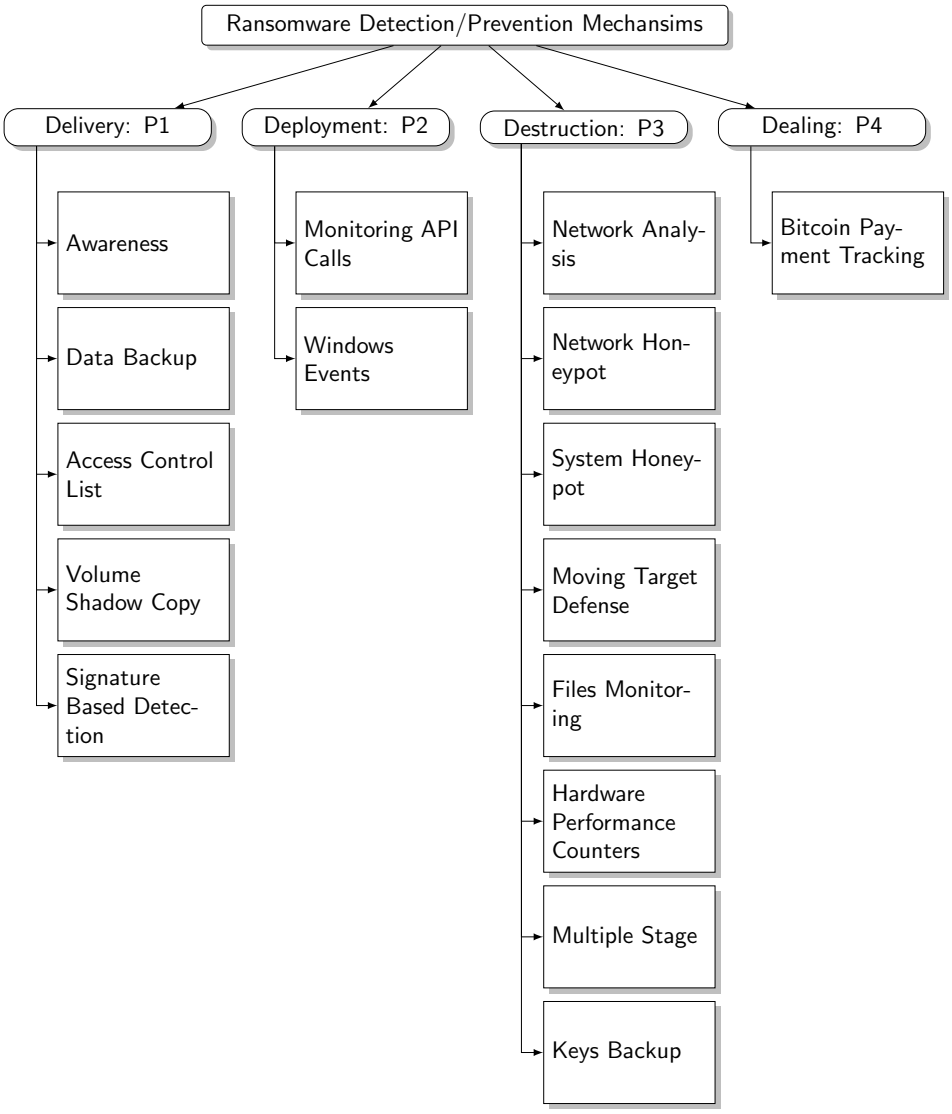

Fig. 2. Ransomware Detection Mechanisms Summary.

example, by deleting a suspicious email immediately or filtering spam ones. These aspects corresponding to the initial phase of ransomware attacks are developed in this part. Table 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 3.1.1). It is portrayed by a safe web browsing, having an up-to-date system including the recent patches and frequent backups [66, 70]. A proposition of an incidence response is portrayed in [63] that could involve using software-defined networking [103]. Data backup is further explained in Section 3.1.2 that is complemented by an access

control list measure in place to protect specific assets on the file system (Section 3.1.3). To enable a clear restoration point, renaming the Volume Shadow copies executable in Windows is presented in Section 3.1.4. Finally, the signature-based detection that helps to flag malicious software without their execution on any system is described in Section 3.1.5.

3.1.1 Awareness. Han *et al.* develop a proof of concept to check whether an individual's requested website is SSL certified or not [70]. 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 ransomware attacks [66]. 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).

The authors in [64] 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 ransomware attacks 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. Spam email 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 [123]. He described ransomware's attack model 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 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 [63, 103], attackers use various techniques like drive-by downloads, malvertising to penetrate the target system.

The authors in [100] 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 ransomware threat rather than developing specific countermeasures or detecting it.

3.1.2 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 [56]. 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. Their solution 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 will 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 [16]. 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 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 loss will affect the end-user or the company.

3.1.3 Access Control List. Kumari *et al.* propose a locking file mechanism to prevent any alteration of the data [79]. 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.

3.1.4 Microsoft Volume Shadow Copy. 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 [119]. The proposed idea consists of renaming the Volume Shadow copies (VSS) executable in Windows, so a given ransomware sample 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.

3.1.5 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 [97]. 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 [91]. 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 [2]. 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 [110]. 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 [85]. Their implemented CRSTATIC tool can detect crypto-ransomware without executing the sample, based on the features provided above.

The authors in [116] 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 [95] 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 [96]. 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.

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

Table 1. Ransomware Detection Mechanisms Overview for the Delivery Phase P1.

3.2 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 3.2.1). Monitoring carefully Windows events helps to extract patterns to describe the habitual behavior of any user accurately compared to ransomware (Section 3.2.2). These methods are summarized in Table 2.

3.2.1 API Calls. Chen *et al.* monitor the API calls made by ransomware to generate API calls flow graphs (CFG) [57]. It is a proactive solution that provides an early stage detection while a ransomware sample 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 [88]. 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 [112]. 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 [69, 117]. 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) [69].

Al-Rimy *et al.* propose a 0-Day aware crypto-ransomware behavioral detection framework [9]. 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 [8]. 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 the 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 ransomware attack by using two novel techniques incremental bagging (iBagging) and enhanced semi-random subspace selection (ESRS) [10]. 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 [99]. 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 ransomware's infection process. The authors use bare-metal hosts during the experiments.

3.2.2 Windows Events. The initial stage of ransomware delivery is similar to reconnaissance for Advanced Persistent Threats (APT) [24]. Both malware rely on social engineering techniques to perform the attackers' required task (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 planted on the machine proceeds with a sequence of specific events that is explored by Homayoun *et al.* [72]. 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 ransomware (for example, ransomware applications conduct a more extensive 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 Solution	Detection/Protection Mechanism
		Static	Dynamic		
[8, 10, 57, 88, 112, 117]	API Calls	-	✓	✓	API Calls sequence &/or frequency features used to detect ransomware via applying ML algorithms
[99]	API Calls	-	✓	✓	Intercepting calls made to MS-CAPI
[72]	Windows Events	-	✓	✓	Maximal frequent patterns extracted from (registry, DLL, transition file to DLL) events then ML applied to detect ransomware

Table 2. Ransomware Detection Mechanisms for the Deployment Phase P2.

3.3 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 ransomware attacks (Section 3.3.1 and Section 3.3.2). Then, the honeypot countermeasures are developed in Section 3.3.3 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 ransomware search is presented in Section 3.3.4. Massive operations, including open, read, and write, portray the encryption phase. Encrypted information will have higher entropy. The statistical tools adopted in the literature that distinguishes a non-encrypted text from an encrypted one are discussed in Section 3.3.5. This step consumes resources; therefore, the hardware events can depict the ongoing ransomware attack (Section 3.3.6). Some authors combine multiple indicators of compromise to detect malicious behavior (Section 3.3.7). Finally, if no real-time solution can stop the encryption process, the restoration of keys can save user files (Section 3.3.8). The methods presented in the destruction phase are summarized in Table 3.

3.3.1 Network Traffic Analysis. Wang *et al.* propose a mechanism for remote desktop protocol tracing and tracking down [118]. 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 [4] 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 [109]). 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 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 [115]. 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 [114] using Weka [12]. 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 made up to 40 000 different UDP requests while TeslaCrypt 05330fff36ad3e359be8bb2b33f09436 only 2 TCP requests [94]. 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 [65] 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 refers to DGA algorithms to detect the DNS requests generated by ransomware. 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 [13]. 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 [59]. 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 [55] 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, 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 [5]. 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 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. In 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.

3.3.2 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 [54]. 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.

3.3.3 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.

Proceeding with possible detection techniques, Moussaileb *et al.* introduce a graph-based mechanism to detect malicious threads in the system [93]. Their approach relies on decoy folders where they flag any abnormality if a particular process passes through more than three different decoy folders. It is unlikely that a normal process passes through the Recycle Bin and Prog Data. This approach is reinforced by adding other features such as the total number of explored paths and the traversal's duration, which is fed to a supervised learning algorithm to perform an accurate classification. Nevertheless, their approach would be limited if ransomware uses multithreading techniques simultaneously for traversing the file system on the one hand and encrypting the files on the other hand.

Akin to previous research area, Lee and Hong introduce a novel mechanism to make efficient decoy files [81]. 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 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 [62, 92]. 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 would 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 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.

3.3.4 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 [84]. 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 ransomware scans just the first couple of bytes of any file, it will get its format and encrypt it if it belongs to the whitelist of files [67].

3.3.5 Files Monitoring (Encryption, I/O requests). Kharazz *et al.* present a dynamic based approach to detect ransomware by identifying any tampering of users files in a created artificial environment [76]. 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 will not be able to fingerprint the user-generated content and will be more likely to attack the system. 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 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 will evade detection.

Palisse *et al.* create two contributions in their work Data-Aware Defense (DaD) [98]. 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 relied 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 developed 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 [89]. 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 [102]. 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 [82]. 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 will not be synchronized to the backup system. However, their solution does not tackle the core of the subject: ransomware attack will not be stopped; they relatively propose a clean system recovery. In addition, the use of machine learning for statistical tests is skeptical since the values will fluctuate around specific numbers that are theoretically known. Thus, a dynamic change will not be 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 [3]. They incorporate attention to learn ransomware sequence known as ARI (Attended Recent Inputs). 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.

3.3.6 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 [11]. 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 will 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.

3.3.7 Multiple Stage/ IOC (indicators of compromise). Chew *et al.* propose a behavioral-based approach to detect ransomware to thwart its malicious intent [58]. 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 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 will increase. Windows 8.1 running on Virtual Box is used for the experiments. Results are satisfactory for all 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 [77]. 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 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 [104]. 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 [25]. 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 [89]. Besides, if ransomware equally distributes the tasks on different processes using multithreading techniques, some files will be lost before detecting the malware.

In a like manner, Mehnaz *et al.* combine multiple modules to detect crypto ransomware at early stages [41]. Those modules consist of monitoring changes occurring in the decoy files and monitoring the IRPs and fast I/O requests. File changes are checked as well including the similarity, the entropy, the file type and size change to identify a malicious process. The authors rely on CryptoAPI function hooking to retrieve the keys, restoring thus the encrypted files. A minimal overhead is noticed by the system (up to 5% or few milliseconds), so it does not impact the overall performance. Ransomware authors can bypass this solution if they add a spyware on the system to monitor user's daily activity thus omitting decoy elements from the encryption process.

The authors in [108] 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 ransomware without losing more than 20 files.

3.3.8 Keys Backup. Lee *et al.* present a prevention mechanism based on the encryption keys used to restore the data after the encryption process [83]. 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 [78]. 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 ransomware. Finally, the file recovery is achieved by testing multiple decryption schemes at different offsets since the encrypted file contains metadata of 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.

3.4 P4: Dealing

The final stage in a 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 4.

Articles	Type	Approaches		Tested	Detection/Protection Mechanism
		Static	Dynamic	Solution	
[4]	Network Analysis	-	✓	✓	DNS requests generated by DGA detection via Markov chains
[12, 13, 59, 115]	Network Analysis	-	✓	✓	Flagging suspicious communication via machine learning
[55]	Network Analysis	-	✓	✓	SDN mechanism capable of flagging malicious POST requests
[54, 75]	Network Honeypot	-	✓	X	Proposition of using Network Honeypot
[93]	System Honeypot	-	✓	✓	ML applied to file system traversal features to detect ransomware
[81]	System Honeypot	✓	✓	X	Novel mechanism to make efficient decoy files
[62, 92]	System Honeypot	-	✓	X ✓	Detection via monitoring honeypot folders
[84]	MTD	-	✓	✓	MTD applied to file extension to prevent the encryption process
[76]	Files Monitoring	-	✓	✓	Monitoring I/O requests and files' Shannon entropy for ransomware detection
[98]	Files Monitoring	-	✓	✓	Chi-squared test to check encrypted files
[89]	Files Monitoring	-	✓	✓	Kullback-Liebler divergence to locate JPEG encrypted files
[82]	Files Monitoring	-	✓	✓	Ransomware detection (in the backup system) by applying ML on file format and entropy
[11]	HPC	-	✓	✓	ANN applied on cache events to flag ransomware
[58]	Multiple IOC	-	✓	✓	Monitoring file changes and entropy, manipulation of decoy files to detect ransomware
[25]	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
[77]	Multiple IOC	-	✓	✓	Monitoring I/O request and changes in the MFT to detect ransomware
[104, 108]	Multiple IOC	-	✓	✓	File attributes, modification, features used to flag ransomware
[78, 83]	Keys Backup	-	✓	✓	Hooking to Microsoft cryptographic function to restore the keys and decrypt the files

Table 3. Ransomware Detection Mechanisms for the Destruction Phase P3.

3.4.1 Bitcoin Tracking. Spagnuolo *et al.* developed BitIodine, a framework that helps to track the irreversible transactions publically available on the blockchain [51]. 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 will be 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 CryptoLocker 1467 addresses belonging to 12 clusters. This step is carried out by analyzing flows of 0.5, 2, or 10 BTC, the ransom demanded by the attackers. 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 clustered can be analyzed to check the possibility of classifying them as ransomware.

Kuzuno and Karma propose as well an analytical process environment for bitcoin [80]. 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 authors' private database. 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 [73]. The authors collect the seed addresses (ransom addresses) from real victims or by executing a ransomware sample. 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 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 pre-deïňĄned categories: exchange, gambling, ransomware, etc [71]. 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 [1]. 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 [7]. TDA helps to extract hidden patterns fundamental elements to distinguish 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.

4 POLYMORPHIC RANSOMWARE

Numerous techniques exist to create a polymorphic version of an existent code. The authors in [100] 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.

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

Table 4. Ransomware Detection Mechanisms for the Dealing Phase P4.

Genc *et al.* delved into analyzing current solutions' weaknesses and projected themselves in the future for an anticipation of potential ransomware attacks [67]. 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 [105]. 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.

Ransomware attacks are evolving, and their authors find multiple ways to bypass some of the current detection mechanisms. Nonetheless, the goal of this malware is limited to altering the availability and integrity of the data. Researchers must be aware of the next pandemic that is just around the corner: Doxware. Indeed, every pillar of the CIA is attacked: even though local data recovery is possible, it will be exposed on the World Wide Web. In other words, once sensitive data is disclosed and proliferated, there is no turning back at this point: the damage can not be undone.

5 RANSOMWARE ANALYSIS

5.1 Ransomware Static & Dynamic Analysis

Quantitative analysis represents a major pillar in the research community. Indeed, it highlights the divergence between the data collected from ransomware versus benign applications behavior. This distinction is found in the system as well as in the network level.

The delivery phase presents the best practices and the recommendations that need to be taken into consideration to protect the files from ransomware attacks. No ransomware detection mechanisms are developed besides ransomware signatures. Therefore, the quantitative analysis concerns specially the second and the third phase of the attack (deployment and destruction) and partially the signature based detection in the first one.

This section presents the features proposed by the research community to detect ransomware based on static and dynamic analysis that covers to some extent the quantitative analysis.

PART I: Static Analysis

(1) **Malware triage** examines automatically unknown samples to decide if they should be dispatched to humans for further analysis or directly classifying the executable as malware. Triaging unknown binaries is accomplished using fuzzy hashing, import hashing and YARA rules [95]. Cryptographic hashing solely is not convenient in malware forensics. In fact, if a single binary digit is changed, the output hash is not identical to the original file even though they share most of the code. Therefore, **Fuzzy hashing** technique solves the aforementioned problem. Fuzzy hashing identifies the similarity between 2 chosen files by dividing each one of them into multiple blocks, calculating the hash for each block separately. The final hash represents the concatenation of all the produced hashes. It offers a degree of similarity rather than a binary classification.
Import Address Table (IAT) of a Portable Executable (PE) file contains function pointers to get the addresses of functions when the DLLs are loaded. **Import hashing** detects the similarity between two files based on the library/API names and their specific order in the executable file. It determines a binary similarity.
Finally, **Yara rules** provides a rule-based approach based on the strings found within a file. It supports OSs like Windows, Linux, and Mac OS.
If the hash of the unknown sample is similar to previously known malware, it is dumped into the malware database. Else, further analysis is required to identify the sample as ransomware.
(2) **Reverse engineering** is performed on three levels: assembly instruction level, libraries used in PE file structure level, and function calls used in the libraries. To this end, *objdump* is used to extract the assembly code in the *Intel* syntax.
(3) If the file is packed, the packer should be identified and **the appropriate unpacker should be used** if possible.
(4) Ransomware is identified as follows:
- The frequency distribution of the instructions^{A} is used to generate the vector representation of each ransomware.
The authors in [110] develop a tool to extract code segments which contain all DLLs required to execute the PE file.
Cosine similarity applied to the normalized frequency of the x instruction in ^{A}. The x instructions are: add, mov, cmp, imul, data16, or, test, xor, movzx, and. Association rule mining is used to detect ransomware pattern.

For example, whenever X is true, Y is true as well.
Multiple DLLs are used by ransomware.
- ADVAPI32: CryptReleaseContext, CryptAcquireContextA, CryptGenRandom, CryptEncrypt, CryptGetKeyParam, CryptAcquireContextW, CryptDestroyKey, CryptImportKey
- CRYPT32: CryptQueryObject, CertFreeCertificateContext, CertFindCertificateInStore, CryptMsgGetParam, CryptDecodeObjectEx

Some of the created association rules to detect ransomware families are:
- $[COMCTL32.DLL, SHELL32.DLL, USER32.DLL] \implies [KERNEL32.DLL]$
- $[MPR.DLL, ADVAPI32.DLL, SHELL32.DLL, USER32.DLL] \implies [KERNEL32.DLL]$

If any unknown binary file matches 60 percent of these rule sets, it is categorized as ransomware with an accuracy of 70 percent.

- Another possible ransomware signature is based on the properties of the PE files [116].
  Machine learning (DT, RF and Naive Bayes) is applied on 60 static properties of the PE files common to all malware and 9 ransomware specif properties.
  The common malware properties include bytes on last page of file, pages in file, relocations, number of section headers and section bodies in the file, symbols in the header, size of optional header, specific characteristics about the file, location of the entry point for the application, etc.
  The ransomware ones are presence of packer, identity of type of packer, checking and creating mutex for isolation, entropy of file, text and data sections, presence of common strings, DLL used for network connections and communication, command used to modify the registry, store information from the client, get access to file from given path.
- To sum up, ransomware can be spotted as shown in [91] based on the static features that include:
  - Keywords like “your files are encrypted”, “payment”, “to decrypt your files”, etc.
  - API functions and extensions: to search directories, rename files such as FindFirstFile or FindNextFile.
  - Cryptography signatures: Rijandel AES algorithm signature, the use of CryptEncrypt and CryptDecrypt from the Microsoft cryptography API.

PART II: Dynamic/Real Time Analysis

There is no doubt that static analysis spares file losses if ransomware executable is identified. However, this is not the case in most ransomware attacks. In fact, victims are unaware of the presence of a malicious software on their PCs or unintentionally click on a malicious URL. Therefore, real time analysis comes in hand as it recognizes and stops ransomware execution limiting the damage if the user has not the executable file.

Ransomware behavioral signature can also be extracted from API calls considering the sequence and frequency of the corresponding calls.

(1) API calls of running applications are extracted using API Monitor or a modified version of Cuckoo [57, 88]. Then, the sequences of APIs ($API_A \rightarrow API_B$) is extracted and converted in to a feature vector containing the frequency of each sequence. Data is normalized having each numeric value in the range [0,1]. The obtained vector represents the input of the training set of the machine learning algorithms. Some authors describe the technique used, for example converting a sequence of API calls into a vector. However, the specific API calls

used in the collection process are not mentioned. Table 5 shows the main API calls related to ransomware. Up to 131 API calls can be used for distinguishing between benign applications and ransomware [117].

Ransomware specific API calls	API calls used in ransomware at higher call frequency
InternetOpen	CryptAcquireContext
CryptDeriveKey	CloseHandle
CryptDecodeObject	FindNextFile
CryptGenKey	SetFilePointer
CryptImportPublicKeyInfo	GetFileSize
GetUserName	SetFileAttributes
NdrClientCall2	
Socket	
NdrClientCall2	

Table 5. Ransomware related API calls [69].

(2) In addition to the captured API calls, registry and DLL events as well as file system to DLL transitions can indicate the presence of ransomware within the first 10 seconds of its execution.

Registry Events	File system Events	DLL Events
RegQueryKey	QueryNameInformationFile	LoadImage
RegOpenKey	ReadFile	
RegQueryValue	CreateFile	
RegCloseKey	QueryBasicInformationFile	
RegCreateKey	CloseFile	
RegDeleteKey	QuerySizeInformationVolume	
RegLoadKey	WriteFile	
RegQueryMultipleValueKey	SetRenameInformationFile	

Table 6. Activities captured by Process Monitor [72].

PART III: From Deployment to Destruction

(1) Ransomware can be detected via **monitoring the DNS** requests made by applications **generated by DGAs** [4]. Each gibberish DNS request is flagged via building Markov chains for the required languages (English, Persian).

(2) **Machine learning applied on** the **network traffic** generated from goodware and ransomware samples is efficient to flag malicious communication. The retained features from the network traffic are presented in Table 7.

(3) **System honeypots** flag any abnormality found in processes file traversal. Generally, ransomware accesses various directories to establish a proper environment for the encryption.
Therefore, decoy files can be placed in those folders, and if they are accessed in a short period, a potential threat is discovered. The list of decoy folders used to detect ransomware activity in [93] is the following: C:\$Recycle.Bin, C:\Python, C:\PerfLogs, C:\Prog_data, C:\Prog_files.

(4) **Renaming file extensions** presented in the following tuples (Original File Extension, Changed File Extension): (.docx, .juev), (.hwp, .plaz), (.pdf, .ohiz), (.pptx, .nhru), (.txt, .umbc), (.xlsx, .qooi), (.zip, .imko) serve as an MTD technique in [84] protects users files without performance degradation.

Articles	[12]	[13]	[59]
Features used	protocol used	**Flow-level:**	inflow number of bytes
	address A, B	number of HTTP-POSTs	outflow number of bytes
	port A, B	number of DNS-NE	inflow standard deviation of packet lengths
	packets A →B, B →A	number of MDN	outflow standard deviation of packet lengths
	bytes A →B, B →A	**Packet-level:**	inflow mean burst length
		MDN	outflow minimal interarrival time
		DNS-NE	outflow to inflow packet ratio
		MNBNS	
		dns-ipv4/ipv6	

Table 7. Features used in the ML process.

(5) **Monitoring file changes** (entropy).
Statistical tools including Shannon (SE), cross (CE), relative entropy (RE), the chi-squared test (χ^2), and finally, the Kolmogorov-Smirnov test (KS) are applied on the following file extensions: doc, jpg, pdf, mp3, xls.
1000 samples of each file extension are collected. Then, with a 128 bits key, an AES encryption is performed on 256 bytes of each file. Table 8 represents the average values calculated based on the file type, before the encryption (ex, SE) and after the encryption process (E_SE) where E denotes the encrypted file type.

Files	SE	E_SE	CE	E_CE	RE	E_RE	(χ^2)	E_(χ^2)	KS	E_KS
doc	1.616	5.615	8.858	8.463	0.858	0.463	129.184	7.085	0.676	0.121
jpg	4.546	7.095	8.580	8.189	0.580	0.189	29.739	1.175	0.627	0.055
pdf	5.051	7.158	8.528	8.176	0.528	0.176	17.458	1.030	0.469	0.053
mp3	4.425	7.052	8.580	8.195	0.580	0.195	39.806	1.504	0.521	0.047
xls	1.602	5.623	8.859	8.461	0.859	0.461	130.165	7.059	0.681	0.120

Table 8. Average values of statistical tools applied on the first 256 bytes of 1000 files and their encrypted counterpart.

An extended analysis of the statistical tools is presented in [47] carried out by Pont *et al.*. They discuss why current statistical approaches fail to detect ransomware since the frequency of false positives is very high.

(6) **Multiple Indicators of Compromise**. The combination of the previous detection mechanisms provides a stronger protection against ransomware attacks. Indeed, if one indicator is bypassed by the malware, the remaining ones still flag an anomalous behavior.
- Monitoring file extensions changes.
- Monitoring file entropy.
- Monitoring canary files.
- Monitoring the frequency of read, write, and folder-listing operations.

(7) Finally, securing the encryption keys generated by ransomware using Microsoft's Crypto APIs is used for data restoration. The keys escrow will not stop the attack, however, will recover the encrypted data.

5.2 Ransomware Future Perspectives Overview

Future challenges and research propositions are presented in this section to offer an exhaustive countermeasure in the ransomware battle. A thorough analysis of the existing work in the literature helped to highlight the actual gaps and to provide a better roadmap for various phases of the ransomware attack presented below.

- Identify the root causes of ransomware attacks.
- Elaborate more on dealing with obfuscated or packed malware instead of omitting them from the analysis like in [91, 116] or classifying them directly as malicious software [45]. Further information about code obfuscation and evasion techniques are discussed in Section 4.
- Provide a benchmark dataset to validate or not future developed solutions.
- Provide a refined multiclass classification of ransomware samples to enhance actual binary classification (trustworthy application versus ransomware).
- Extract specific ransomware patterns to 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.
- Propose an evaluation of the resources consumed during the real-time analysis (overhead) and the performance impacts on the system.
- Re-design decoys generation of the deception-based techniques to improve the protection of the data of users, as mentioned in [31].
- Check whether the creation of a realistic environment (saved credentials, browser history, and realistic decoy files including images and documents [25]) is sufficient to trigger ransomware attacks.
- Provide an automated tool to populate an online shared repository containing the list of the blacklisted IP addresses of the C&C servers to prevent ransomware spreading.
- Evaluate the impact of different encryption schemes on the randomness of data since Shannon's entropy is not efficient in differentiating between an image, a zipped file, or an encrypted one raising the false-positive rates and the Chi-squared test can be bypassed by bytes permutation, as shown in [67].

Summary

Ransomware evolved over the last decade. The encryption mechanism adopted by the attackers shifted from including the symmetric keys in the malicious software to fetching them from the C&C. The current anti-ransomware ecosystem follows the rules of cat-and-mouse. Therefore, one size fits all solution to stop ransomware does not exist. New malware and ransomware emerge continually making the developed countermeasures inadequate to detect those new properties. 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. Even though ransomware has been studied thoroughly, we can not assert that this research area is "complete". Researchers must combine different aspects of ransomware behavior to build a safe countermeasure. However, several key elements can be elaborated to provide a proper ransomware countermeasure. The delivery phase is responsible for subsequent file losses. Therefore, the attack vectors graphs will help to identify the vulnerability points found in the systems. By automating the process, the latest patches can be applied as an incident response handling technique. New statistical tools or learning algorithms can be developed to detect the randomness aspect of the data caused by encryption. Besides, recent ransomware authors encrypt the files maintaining the same entropy before and after the encryption process. Thus, the frequency of the read and write operations would come in handy since the data distribution is invariant in the case of an unaltered entropy. The combination of those multi-layer techniques will generate an important overhead on the system. Analyzing the correlation between the countermeasures and their impact on the system will help administrators to tailor ransomware countermeasure based on their needs. Lastly, collaborative effort like the RAMSES project [48] and others will

provide a positive initiative for ransomware tracking and detection, thus, reducing the massive spread of the attacks.

6 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 (Section6.1 and Section 6.2). The proposed solutions are developed on the Android OS. Finally, we present the shortcomings of mobile ransomware in Section 6.3.

6.1 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 [39]. 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 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 [18]. 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 Naive Bayes are used as classification algorithms. They successfully detect ransomware instances.

6.2 Multiple IOC (indicators of compromise)

DNA-Droid developed in [32] 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 crypto 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) [29]. 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, Naive Bayes, 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.

6.3 Discussion

Ransomware affects mobile devices too. Being a different type of operating system, we investigated 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, crypto 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 [28, 30] and others [50, 73, 122]. However, current literature is not abundant with mobile ransomware journals or papers, as stated in [32, 39]. Even in the latest survey released in 2020 about ransomware in Windows and Android platforms, only six papers were mentioned related to mobile ransomware [21]. More general malware-oriented papers are reviewed.

Articles	Type	Approaches		Tested Solution	Detection/Protection Mechanism
		Static	Dynamic		
[39]	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.
[18]	API Calls	X	✓	✓	Random forest, J48, and Naive Bayes are applied to fifty-two collected system calls to detect ransomware samples.
[32]	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.
[29]	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 9. Mobile Ransomware Detection Mechanisms.

Most surveys classify mobile ransomware detection based on static, dynamic, or hybrid analysis [18, 29, 43, 61]. Model checking is also used to detect Android ransomware [42]. 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 [35, 36]. Therefore, we decided to focus solely on Windows ransomware in the current survey. We will propose another classification scheme for Android ransomware once the literature will be abundant with the proposed solutions.

7 CONCLUSION

To date, ransomware attacks are still spreading across the globe with a shifted target from end-users to businesses. The current literature is abundant with a variety of reliable solutions to flag ransomware's malicious behavior since its early stages, from the infection vectors to the installation process to the payload delivery and, finally, the tracking of cybercriminals. The combination of regular backup and an up-to-date antivirus is essential to avoid any file loss. To breakdown the suggested solutions, an attacker's effort should be equally distributed across all the stages from the infection until the ransom payment, which is not a trivial task. Detecting zero-day ransomware is possible since they have a predefined taxonomy and payload (encryption process). However, any alteration of the deployed mechanisms will be an extremely challenging task since we are exposing an unseen behavior. There will always be the first entity down (the first victim), but the following attacks can be prevented since we are capable of managing the incident response by acquiring useful information. Monetary and data loss are catalysts for conducting further research to tackle some limitations of current detection mechanisms and propose future perspectives to compete in this arms race against ransomware. To conclude, ransomware remains a dangerous threat that should be addressed soundly with continuous research.

A APPENDIX

This appendix gives an extended information about previously carried out experiments in the Deployment and in the Destruction phase (Tables 11 and 12) and the signature based detection in the Delivery phase (Table 10). Each table presents the tools used in a specific paper, if the execution on a bare metal (BM) or virtual machine (VM) platform, the number of samples and families.

Articles	Type	Tools	Tested on BM	Tested on VM	Number of Samples	Families
[119]	VSS	Process monitor Regshot	-	✓	-	CryptoWall, TeslaCrypt, CTBLocker, Locky
[91]	SBD	Yara rules objdump	-	✓	792	45 families[1]
[110]	SBD	pe-parser OllyDbg	-	✓	450	11 families [2]
[116]	SBD	WinDbg Hashing	-	✓	-	CryptoLocker, Locky, WannaCry
[95]	SBD	Yara rules	-	✓	200	WannaCry/WannaCryptor, Locky, Cerber, CryptoWall

[1] Badrabbit, Bitman, Blocker, BTCWARE, Crusis, Crykal, Crypmodadv, Crysis, Cryptodef, Fantom, FileCoder, GandCrab, Gimemo, Globeimposter, Gpcode, Hades, HDDCrypt, HiddenTears, HKNATA, Ishtar, Jigsaw, Karo, Lechiffre, Locky, Magniber, Matrix, MISCHA, Mole, Petya, Philadelephia, PUBG, Rack, RansomKD, Rapid, Samsam, Satan, Scatter, Striked, Tesla, Thanatos, Wannacry, Xorist, Zcrypt, Zenis, Zyklon.
[2] Locky, CryptoWall, FileCryptor, TeslaCrypt, Crypt, CryptoLocker, Cerber, CTB-Locker, Petya, Satana, WannaCry.

Table 10. Ransomware Detection Mechanisms Extended Overview for the Delivery Phase P1.

Articles	Type	Tools	Tested on BM	Tested on VM	Number of Samples	Families
[57]	API calls	API monitor, Weka	-	✓	83	CryptoWall, Kollah, Trojan-Ransom, TeslaCrypt
[88]	API calls	Cuckoo	-	✓	157	not mentioned
[112]	API calls	Cuckoo	-	✓	276	WannaCry, Cerber, Petya, CryptoLocker
[69]	API calls	Process Monitor	-	✓	14	14 families[1]
[117]	API calls	Cuckoo, TensorFlow	-	✓	755	Cerber, CryptoLocker, CryptoWall, Maktub, Ransomware, Sage, Torrentlocker
[8]	API calls	Cuckoo	-	✓	38152	Cerber, TeslaCrypt, CryptoWall, Petya, WannaCry
[10]	API calls	Cuckoo	-	✓	8152	Cerber, TeslaCrypt, CryptoWall, Petya, WannaCry
[99]	API calls	CSP	✓	-	39	Gpcode, CryptoLocker, CryptoWall, CTB-Locker, TorrentLocker, TeslaCrypt, CrypVault, Locky, Petya
[72]	WE	Process Monitor	-	✓	1624	Cerber, TeslaCrypt, Locky

[1] CTB-Locker, Cerber, CrypMIC, CryptFile2, CryptoMix, CryptoShield, GlobeImposter, Gryphon, JAFF, Mole, Revenge, TeslaCrypt, WannaCry, NemucodAES

Table 11. Ransomware Detection Mechanisms Extended Overview for the Deployment Phase P2.

Articles	Type	Tools	Tested on BM	Tested on VM	Number of Samples	Families
[4]	Network Analysis	-	-	-	20+	20 families [1]
[12]	Network Analysis	Weka	-	-	210	Cerber, Cryptolocker, Cryptowall, CTB-Locker, Locky, Padcrypt, Paycrypt, TorrentLocker, Teslacrypt
[13]	Network Analysis	Weka	-	-	4	Locky
[59]	Network Analysis	RaftLib	-	-	-	Cerber
[115]	Network Analysis	TensorFlow	-	-	155	23 families including CryptoWall, TeslaCrypt, CryptXXX, Locky, CrypMIC, Cerber
[93]	System Honeypot	Scikit-Learn	✓	-	770	TeslaCrypt, Xorist, Cerber, Bitman, Deshacop, Zerber, Locky, Yakes, Gpcode
[62]	System Honeypot	-	?	?*	50	Reveton, Cryptolocker, CryptoWall, Kovter, Filecoder, Winlock
[84]	MTD	-	-	✓	143	Tesla, Cerber, Locky, WannaCry, ZeroLocker, SatanLocker
[76]	Files Monitoring	-	-	✓	2121	12 families [2]
[98]	Files Monitoring	-	✓	-	798	15 families [3]
[89]	Files Monitoring	-	-	-	-	Not tested using ransomware
[82]	Files Monitoring	-	-	-	-	Not tested using ransomware
[58]	Multiple IOC	Cacls, ACL	-	✓	4	WannaCry, TeslaCrypt, Cerber, Petya
[25]	Multiple IOC	-	✓	-	688	CryptoWall, TeslaCrypt, Critroni, CryptoDefense, Crowti, Locky, CryptoLocker, TorrentLocker, DirtyDecrypt, PayCrypt, Troldesh, ZeroLocker
[77]	Multiple IOC	-	-	✓	1359	15 families [4]
[104]	Multiple IOC	-	-	✓	492	14 families [5]
[108]	Multiple IOC	-	-	✓	3	Authors developed ransomware
[78]	Keys Backup	-	-	✓	107	20 families [6]
[83]	Keys Backup	-	-	✓	-	Authors developed ransomware

[1] Cryptolocker (v2,v3), Cryptowall (v2,v3), CoinVault, CryptoGraphic Locker, CryptoDefense (v2), CryptorBit, TorrentLocker, ACCDFISA, BuyUnlockCode, CryptoFortress, PClock2, Critroni, ComputerCrime&IntellectualPropertySection, Harasom.

[2] Cryptolocker, CryptoWall, CTB-Locker, CrypVault, CoinVault, Filecoder, TeslaCrypt, Tox, VirLock, Reveton, Tobfy, Urausy.

[3] TeslaCrypt, Xorist, Cerber, Bitman, Deshacop, Zerber, Locky, Yakes, Gpcode, Gamarue, Shifu, Fsysna, Shade, Dalexis, Usteal

[4] Reveton, Cryptolocker, CryptoWall, Tobfy, Seftad, Winlock, Loktrom, Calelk, Urausy, Krotten, BlueScreen, Kovter, Filecoder, GPcode, Weelsof.

[5] CryptoDefense, CryptoFortress, CryptoLocker, CryptoTorLocker2015, CryptoWall, CTB-Locker, Filecoder, GPcode, MBL, PoshCoder, Ransom-FUE2, TeslaCrypt, Virlock, Xorist.

[6] Almalocker, Cerber, Chimera, CryptoFortress, CryptoLocker, CryptoWall, CrypWall, GPcode, Locky, SamSam, Thor, Tox, DXXD, MarsJokes, PokemonGo, Troldesh, VirLock, Androm, Razy, TeslaCrypt.

* Not mentioned if BM or VM.

Table 12. Ransomware Detection Mechanisms Extended Overview for the Destruction Phase P3.

The discrepancy in ransomware samples throughout the literature is due in part to inactive ransomware families after an epsilon time of their release. 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 ([68]). 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 suspects being monitored/analyzed (VM or debugging tools).
- The footprint of the test environment is already registered in the attacker's server.

The most important aspect in ransomware proposed countermeasures is the data loss (encrypted files) that occur in the system. However, among the 28 studied papers in the Deployment and Destruction phase, only 8 papers presented the recovery percentage by deploying their solution [25, 58, 78, 84, 98, 99, 104, 108]. Therefore, the accuracy is not included in the presented comparison tables (that exceeds 95% in most of the cases).

REFERENCES

[1] 2014. Building trust in blockchains. https://www.chainalysis.com/. (2014).

[2] 2014. YARA Rules. https://yara.readthedocs.io/en/latest/. (2014).

[3] Rakshit Agrawal, Jack W Stokes, Karthik Selvaraj, and Mady Marinescu. 2019. Attention in Recurrent Neural Networks for Ransomware Detection. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 3222–3226.

[4] Mohammad Mehdi Ahmadian, Hamid Reza Shahriari, and Seyed Mohammad Ghaffarian. 2015. 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)*. IEEE, 79–84.

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

[6] Maxat Akbanov, Vassilios G Vassilakis, and Michael D Logothetis. 2019. WannaCry Ransomware: Analysis of Infection, Persistence, Recovery Prevention and Propagation Mechanisms. *Journal of Telecommunications & Information Technology* 1 (2019).

[7] Cuneyt Gurcan Akcora, Yitao Li, Yulia R Gel, and Murat Kantarcioglu. 2019. BitcoinHeist: Topological Data Analysis for Ransomware Detection on the Bitcoin Blockchain. *arXiv preprint arXiv:1906.07852* (2019).

[8] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, Yuli Adam Prasetyo, Syed Zainudeen Mohd Shaid, and Asmawi Fadillah Mohd Ariffin. 2018. Zero-Day Aware Decision Fusion-Based Model for Crypto-Ransomware Early Detection. *International Journal of Integrated Engineering* 10, 6 (2018).

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

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

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

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

[13] Ahmad O Almashhadani, Mustafa Kaiiali, Sakir Sezer, and Philip O'Kane. 2019. A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware. *IEEE Access* 7 (2019), 47053–47067.

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

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

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

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

[18] Abdullah, Zubaile and Muhadi, Farah Waheeda and Saudi, Madihah Mohd and Hamid, Isredza Rahmi A and Foozy, Cik Feresa Mohd. 2020. Android Ransomware Detection Based on Dynamic Obtained Features. In *International Conference on Soft Computing and Data Mining*. Springer, 121–129.

[19] Alghoul, Ahmed and Al Ajrami, Sara and Al Jarousha, Ghada and Harb, Ghayda and Abu-Naser, Samy S. 2018. Email Classification Using Artificial Neural Network. (2018).

[20] Al-rimy, Bander Ali Saleh and Maarof, Mohd Aizaini and Shaid, Syed Zainudeen Mohd. 2018. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers & Security* 74 (2018), 144–166.

[21] Alzahrani, Abdulrahman and Alshehri, Ali and Alshahrani, Hani and Fu, Huirong. 2020. Ransomware in Windows and Android Platforms. *arXiv preprint arXiv:2005.05571* (2020).

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

[23] Berrueta, Eduardo and Morato, Daniel and Magaña, Eduardo and Izal, Mikel. 2019. A Survey on Detection Techniques for Cryptographic Ransomware. *IEEE Access* 7 (2019), 144925–144944.

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

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

[26] Cybersecurity and Infrastructure Security Agency. 2016. Ransomware. https://www.us-cert.gov/Ransomware. (2016).

[27] Delgado-Mohatar, Oscar and Sierra-Cámara, José María and Anguiano, Eloy. 2020. Blockchain-based semi-autonomous ransomware. *Future Generation Computer Systems* (2020).

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

[29] Ferrante, Alberto and Malek, Miroslaw and Martinelli, Fabio and Mercaldo, Francesco and Milosevic, Jelena. 2017. Extinguishing ransomware-a hybrid approach to android ransomware detection. In *International Symposium on Foundations and Practice of Security*. Springer, 242–258.

[30] Fuchs, Adam P and Chaudhuri, Avik and Foster, Jeffrey S. 2009. *Scandroid: Automated security certification of android*. Technical Report.

[31] Genç, Ziya Alper and Lenzini, Gabriele and Sgandurra, Daniele. 2019. On Deception-Based Protection Against Cryptographic Ransomware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 219–239.

[32] Gharib, Amirhossein and Ghorbani, Ali. 2017. Dna-droid: A real-time android ransomware detection framework. In *International Conference on Network and System Security*. Springer, 184–198.

[33] Hernandez-Castro, Julio and Cartwright, Edward and Stepanova, Anna. 2017. Economic analysis of ransomware. *Available at SSRN 2937641* (2017).

[34] Karapapas, Christos and Pittaras, Iakovos and Fotiou, Nikos and Polyzos, George C. 2020. Ransomware as a Service using Smart Contracts and IPFS. *arXiv preprint arXiv:2003.04426* (2020).

[35] Kaspersky Lab. 2016. 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).

[36] Kaspersky Lab. 2017. KSN Report: Ransomware in 2016-2017. https://go.kaspersky.com/rs/802-IJN-240/images/KSN_Report_Ransomware_2016-2017_ENG.PDF. (2017).

[37] Kitchenham, Barbara and Charters, Stuart. 2007. Guidelines for performing systematic literature reviews in software engineering. (2007).

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

[39] Maiorca, Davide and Mercaldo, Francesco and Giacinto, Giorgio and Visaggio, Corrado Aaron and Martinelli, Fabio. 2017. R-PackDroid: API package-based characterization and detection of mobile ransomware. In *Proceedings of the symposium on applied computing*. 1718–1723.

[40] Malwarebytes. 2018. Ransomware. https://www.malwarebytes.com/ransomware/. (2018).

[41] Mehnaz, Shagufta and Mudgerikar, Anand and Bertino, Elisa. 2018. Rwguard: A real-time detection system against cryptographic ransomware. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 114–136.

[42] Mercaldo, Francesco and Nardone, Vittoria and Santone, Antonella and Visaggio, Corrado Aaron. 2016. Ransomware steals your phone. formal methods rescue it. In *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*. Springer, 212–221.

[43] Mohan, Jithin Chandra and Kumar, Renuka. 2017. On the efficacy of android ransomware detection techniques: A survey. *International Journal of Pure and Applied Mathematics* 115, 8 (2017), 115–120.

[44] Norton. 2018. 7 tips to prevent ransomware. https://us.norton.com/internetsecurity-malware-7-tips-to-prevent-ransomware.html. (2018).

[45] Osaghae, Edgar O. 2016. Classifying packed programs as malicious software detected. *International Journal of Information Technology and Electrical Engineering* 5 (2016), 22–25.

[46] Park, Leo Hyun and Yu, Jungbeen and Kang, Hong-Koo and Lee, Taejin and Kwon, Taekyoung. 2020. Birds of a Feature: Intrafamily Clustering for Version Identification of Packed Malware. *IEEE Systems Journal* (2020).

[47] Pont, Jamie and Arief, Budi and Hernandez-Castro, Julio. 2020. Why Current Statistical Approaches to Ransomware Detection Fail. (2020).

[48] RAMSES. 2017. RAMSES. https://ramses2020.eu/. (2017).

[49] Saidani, Nadjate and Adi, Kamel and Allili, Mohand Said. 2020. A Semantic-Based Classification Approach for an Enhanced Spam Detection. *Computers & Security* (2020), 101716.

[50] Sarwar, Golam and Mehani, Olivier and Boreli, Roksana and Kaafar, Mohamed Ali. 2013. On the Effectiveness of Dynamic Taint Analysis for Protecting against Private Information Leaks on Android-based Devices. In *SECRYPT*, Vol. 96435.

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

[52] Yassine Lemmou. 2019. Ransom Note Files. https://github.com/lemmou/RansomNoteFiles. (2019).

[53] Mike Bond and George Danezis. 2006. A Pact with the Devil. In *Proceedings of the 2006 workshop on New security paradigms*. ACM, 77–82.

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

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

[56] Jason Castiglione and Dusko Pavlovic. 2019. Dynamic Distributed Secure Storage Against Ransomware. *IEEE Transactions on Computational Social Systems* (2019).

[57] Zhi-Guo Chen, Ho-Seok Kang, Shang-Nan Yin, and Sung-Ryul Kim. 2017. 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*. ACM, 196–201.

[58] Christopher Chew and Vimal Kumar. 2019. Behaviour Based Ransomware Detection. In *Proceedings of 34th International Conference*, Vol. 58. 127–136.

[59] Greg Cusack, Oliver Michel, and Eric Keller. 2018. 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*. ACM, 1–6.

[60] KAO Da-Yu, Shou-Ching HSIAO, and TSO Raylin. 2019. Analyzing WannaCry Ransomware Considering the Weapons and Exploits. In *2019 21st International Conference on Advanced Communication Technology (ICACT)*. IEEE, 1098–1107.

[61] Usama Desai. 2019. A Survey on Android Ransomware and its Detection Methods. (2019).

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

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

[64] 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. 2017. Social engineering as an attack vector for ransomware. In *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*. 1–6. https://doi.org/10.1109/CHILECON.2017.8229528

[65] Karim Ganame, Marc André Allaire, Ghassen Zagdene, and Oussama Boudar. 2017. 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*. Springer, 169–181.

[66] Shaunak Sanjay Ganorkar and Kamalanathan Kandasamy. 2017. Understanding and defending crypto-ransomware. *ARPN Journal of Engineering and Applied Sciences* 12, 12 (2017), 3920–3925.

[67] Ziya Alper Genç, Gabriele Lenzini, and Peter YA Ryan. 2018. Next Generation Cryptographic Ransomware. In *Nordic Conference on Secure IT Systems*. Springer, 385–401.

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

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

[70] Jordan W Han, Ong J Hoe, Joseph S Wing, and Sarfraz N Brohi. 2017. 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*. ACM, 222–226.

[71] Mikkel Alexander Harlev, Haohua Sun Yin, Klaus Christian Langenheldt, Raghava Mukkamala, and Ravi Vatrapu. 2018. 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*.

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

[73] 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. 2018. Tracking ransomware end-to-end. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 618–631.

[74] D Paul Joseph and Jasmine Norman. 2020. A Review and Analysis of Ransomware Using Memory Forensics and Its Tools. In *Smart Intelligent Computing and Applications*. Springer, 505–514.

[75] Nilesh Kakade, Mayur Shinde, Akshay Gawali, and Ajinkya Bhoite. 2018. JAVA Based Honeypot: Intrusion Detection System. (2018).

[76] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. 2016. {UNVEIL}: A Large-Scale, Automated Approach to Detecting Ransomware. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 757–772.

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

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

[79] Anjali Kumari, Md Zakirul Alam Bhuiyan, Jigyasa Namdeo, Shipra Kanaujia, Ruhul Amin, and Satyanarayana Vollala. 2019. Ransomware Attack Protection: A Cryptographic Approach. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. Springer, 15–25.

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

[81] Jeonghwan Lee, Jinwoo Lee, and Jiman Hong. 2017. How to Make Efficient Decoy Files for Ransomware Detection?. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*. ACM, 208–212.

[82] Kyungroul Lee, Sun-Young Lee, and Kangbin Yim. 2019. Machine Learning based File Entropy Analysis for Ransomware Detection in Backup Systems. *IEEE Access* (2019).

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

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

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

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

[87] Malwarebytes. 2019. 2019 State of Malware. https://resources.malwarebytes.com/resource/2019-state-malware-malwarebytes-labs-report/. (2019).

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

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

[90] Timothy McIntosh, Julian Jang-Jaccard, Paul Watters, and Teo Susnjak. 2019. The Inadequacy of Entropy-Based Ransomware Detection. In *International Conference on Neural Information Processing*. Springer, 181–189.

[91] May Medhat, Samir Gaber, and Nashwa Abdelbaki. 2018. 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)*. IEEE, 710–715.

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

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

[94] Routa Moussaileb, Nora Cuppens, Jean-Louis Lanet, and Hélène Le Bouder. 2019. Ransomware Network Traffic Analysis for Pre-Encryption Alert.

[95] Nitin Naik, Paul Jenkins, Nick Savage, and Longzhi Yang. 2019. Cyberthreat Hunting-Part 1: Triaging Ransomware using Fuzzy Hashing, Import Hashing and YARA Rules. (2019).

[96] Nitin Naik, Paul Jenkins, Nick Savage, and Longzhi Yang. 2019. Cyberthreat Hunting-Part 2: Tracking Ransomware Threat Actors using Fuzzy Hashing and Fuzzy C-Means Clustering. (2019).

[97] Daniel Nieuwenhuizen. 2017. A behavioural-based approach to ransomware detection. *Whitepaper. MWR Labs Whitepaper* (2017).

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

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

[100] Navneet Kaur Popli and Anup Girdhar. 2019. 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*. Springer, 65–80.

[101] Segun I Popoola, Samuel O Ojewande, Faith O Sweetwilliams, SN John, AA Atayero, et al. 2017. Ransomware: Current Trend, Challenges, and Research Directions. (2017).

[102] Toshima Singh Rajput. 2017. Evolving Threat Agents: Ransomware and their Variants. *International Journal of Computer Applications* 164, 7 (2017), 28–34.

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

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

[105] Sergiu Sechel. 2019. A Comparative Assessment of Obfuscated Ransomware Detection Methods. *Informatica Economica* 23, 2 (2019), 45–62.

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

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

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

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

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

[111] Symantec. 2019. 2019 Internet Security Threat Report. https://www.symantec.com/en/uk/security-center/threat-report. (2019).

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

[113] Aditya Tandon and Anand Nayyar. 2019. A Comprehensive Survey on Ransomware Attack: A Growing Havoc Cyberthreat. In *Data Management, Analytics and Innovation*. Springer, 403–420.

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

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

[116] Deepti Vidyarthi, CRS Kumar, Subrata Rakshit, and Shailesh Chansarkar. 2019. Static Malware Analysis to Identify Ransomware Properties. *International Journal of Computer Science Issues (IJCSI)* 16, 3 (2019), 10–17.

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

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

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

[120] Julian Wolf. 2017. Ransomware Detection. (2017).

[121] Syed Rameem Zahra and Mohammad Ahsan Chishti. 2019. RansomWare and Internet of Things: A New Security Nightmare. In *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 551–555.

[122] Jie Zhang, Cong Tian, and Zhenhua Duan. 2019. FastDroid: efficient taint analysis for Android applications. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 236–237.

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

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

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

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