Ransomware Network Traffic Analysis for Pre-Encryption Alert

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

Ransomware Network Traffic Analysis for Pre-Encryption Alert

Routa Moussaileb[1,2], Nora Cuppens[1], Jean-Louis Lanet[2], and Hélène Le Bouder[1]

[1] IMT Atlantique, SRCD, France
[2] Inria, LHS-PEC, France
[1] name.surnameb@imt-atlantique.fr
[2] ruta.moussaileb@irisa.fr
[2] jean-louis.lanet@inria.fr

Abstract. Cyber Security researchers are in an ongoing battle against ransomware attacks. Some exploits begin with social engineering methods to install payloads on victims' computers, followed by a communication with command and control servers for data exchange. To scale down these attacks, scientists should shed light on the danger of those rising intrusions to prevent permanent data loss. To join this arm race against malware, we propose in this paper an analysis of various ransomware families based on the collected system and network logs from a computer. We delve into malicious network traffic generated by these samples to perform a packet level detection. Our goal is to reconstruct ransomware's full activity to check if its network communication is distinguishable from benign traffic. Then, we examine if the first packet sent occurs before data's encryption to alert the administrators or afterwards. We aim to define the first occurrence of the alert raised by malicious network traffic and where it takes place in a ransomware workflow. Logs collected are available at http://serveur2.seres.rennes.telecom-bretagne.eu/data/RansomwareData/.

Keywords: Ransomware · Network Traffic · Machine Learning.

1 Introduction

Ransomware attacks represent a widespread phenomenon of this decade. None of the operating systems or electronic devices are spared. This pandemic affected more than 200 000 computers at the beginning of 2017 [34].

There is no doubt that ransomware is spreading at a high rate infecting not only governmental organizations but also end users and hospitals. Its timeline represents a lucrative business that combines intelligence of the attacker and the fear of the victims for the loss of his/her files.

Motivation Our motivation to join the malware battle specifically ransomware is its frequency. In fact, one attack occurs every 40 seconds (October 2016) infecting various sectors like education, entertainment, financial services, healthcare, etc. Unfortunately, these sectors are not immune to such attacks or extortion techniques (Deutsche Bahn, Honda, Renault, etc.) [2].

Even though previous operating system targets were essentially Windows computers, nowadays a wider range of infected equipment and OS is noticed: MacOS, IoT devices and Cellphones (Android OS) [35,41]. Previous work has been carried out knowing the signature of the ransomware on the file system that is the encryption of all victims' files. It is an obvious footprint representing solely that malicious software. In fact, data distribution diverges

between an encrypted and a non-encrypted file. Thus, any statistical tool (Shannon Entropy, Chi Squared, ...) applied on these elements displays distinct results enabling an accurate detection. As follows, an emerging ransomware can neither go unnoticed in these conditions nor easily be detected. Our work does not represent a real-time based solution, but rather a study on ransomware families to potentially extract an additional signature besides the obvious encryption phase.

Contribution The objective of this paper is to find a way to spot the same traceability, however, based on network analysis. The main contributions of this paper are summarized as follows:

1. Providing a mechanism for data filtering based on open source tools.
2. Creating ransomware models via machine learning on network flows.
3. Evaluating ransom notes and encrypted files to check whether the detection occurred at a time t inferior at the start of the encryption.

Outline The paper is structured as follows. Ransomware's description is presented in Section 2. Ransomware state of the art is defined in Section 3. Data collection, filtering and detection mechanisms of ransomware are developed in Section 4. The results of the experiments are outlined in Section 5. Finally, the conclusion is drawn in Section 6.

2 Context

2.1 Ransomware

Ransomware is a specific type of malware that encrypts users' files or locks the screen where access to data is only granted if a ransom is paid. The payment is accomplished via Bitcoin or any other cryptocurrency. Another ransomware type such as Doxware operates in a distinct way: you can accept paying the ransom or you can provide information about two of your friends for cyber criminals. Ransomware's stages are outlined below:

1. **Infection** process by spam emails, self-propagation.
2. **Encryption** process (AES, RSA...) of specific file types (.doc, .xls, .jpg, .pdf).
3. **Deletion** of original files and Microsoft Shadow Volumes.
4. **Ransom Request** key exchanged for money.

Since 1989 (first known malware extortion attack), researchers have been aware of these attacks. Concurrently, cyber criminals have been improving their techniques for better gain. A glimpse of the recent attacks is presented below. Cyber attackers are shifting their focus to large companies for more fruitful hunts. For instance, National Health Service (NHS) England, and Telefonica have been infected with WanaCrypt0r [3].

Various categories of malware exist. Each one of them presents specific characteristics [22]. For example in 2012, the attacker of Reveton ransomware displayed a fraudulent message to the user. It declared that he/she has been illegally using some software or acquiring illegal documents/movies. In addition to that, to pressure the victim to pay, the attacker displays the IP address of the victim and an actual footage from his/her webcam representing scare tactics [4].

Another wave of ransomware named "CryptoLocker" emerged in 2013 using 2048-bit RSA for encrypting a whitelist of files. As a result, data is unrecoverable since decryption would take decades. This pandemic did not end here. It goes on with diverse attacks like CryptoWall, WannaCry, Petya and Bad Rabbit that infect currently just computers. However, Fusob that appeared in 2015-2016 targeted mobile devices. The most recent cyberattack struck Baltimore city on the 7th of May 2019. Multiple services were shutdown due to this ransomware attack [1].

Targeted files may vary from a ransomware to another. A fine sample that encompasses the majority of these extensions is: .odt, .docx, .xlsx, .pptx, .mdb, .jpeg, .gif, .bmp, .exif, .txt, etc [32].

2.2 Ransomware Timeline

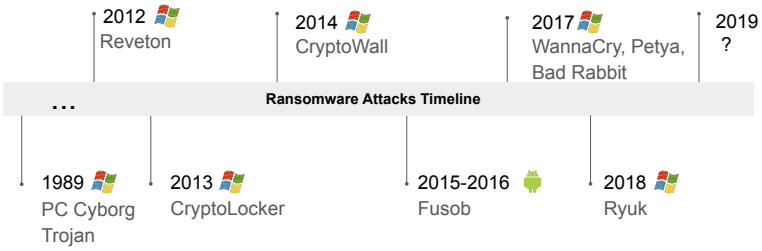

Fig. 1. Ransomware Families Debut Timeline [17].

The timeline shown in Figure 1 presents a glimpse of ransomware families initial release dates and their targeted systems. It will be used in Section 5 as a comparison means between the release date and the network activity observed throughout the collected data of various samples. Thus, the question remains, based on previous observations can we predict what will happen in ransomware's universe, or prevent a plausible attack.

3 Ransomware Detection State of the Art

In the worst case scenario, if no noticeable characteristics alerted the user, encryption process will begin.To eradicate malicious threats, a myriad of solutions are available in the current literature [40,37,8,19].

Some researchers delve into investigating ransomware's signals on the filesystem for instance API calls or relevant system calls [14,24]; or they rely on metrics collected from encrypted files such as entropy[42,36,23]. However, some researchers delve into the study of network activity for ransomware detection [12,13,9]. Therefore, the analysis presented in the literature could be divided in two parts: host and network based ransomware detection.

3.1 Host Based Ransomware Detection

Previous work conducted in the literature emphasizes that each malware, even though is sometimes obfuscated, can present different patterns enabling researchers its detection at an

early stage. Indeed, ransomware metrics have distinct values compared to benign processes such as I/O Request Packets Sequence, API Calls Flow Graph, and in our case, acquired system and network logs from ransomware execution.

Gomez Hernandez *et al.* illustrate in their work (RLocker) a ransomware early detection mechanism that could prevent its operations [20]. One of RLocker advantages is the detection of zero day's ransomware attacks. It is achieved by deploying a honeyfile structure to lure ransomware, which will block any process that passes through it, or try to modify it. However, previous requirements are crucial for the success of such operation. Nevertheless, RLocker has a limitation: it could be bypassed if a process passes randomly through the files without intercepting the lure folders. A similar limitation related to files traversals could be noticed in Moussaileb *et al.* work for early mitigation mechanisms [28]. Even though their solution, solely based on file system exploration, is effective (up to 100% detection rate), their detection would be delayed if a ransomware uses multithreading for simultaneously traversing and encrypting the filesystem (some encrypted files are inevitable in this case).

Most of the current detection mechanisms of ransomware rely on an obvious signature of the latter: the encryption process that occurs [42,24,36,23,25,16]. In fact, many alerts based on monitoring the system activity are reported. For example, studying bits distribution, whether performing machine learning algorithms on various dlls or on functions called referring to a cryptographic context [14,25]. This means that the encryption process is already taking place and losses are inevitable. All the solutions mentioned previously are viable for ransomware detection. Yet, some file losses occur and they are solely based on system activity when the malware is executing its payload.

A set of indicators can represent viably ransomware behavior as presented in CryptoLock by Scaife *et al.* [36]. Shannon entropy was one of their main metrics since encrypted or compressed files have high entropy. In their solution, few files are lost. Another statistical technique was adopted by Mbol *et al.* for ransomware detection [27]. Their focus was on JPEG files since they initially have high entropy. In order to distinguish between an encrypted image and a clear one, Kullback Leibler divergence was used and 128 KB up to 512 KB of JPEG files were analyzed. A high detection rate was achieved by their solution: 99.95%. A similar approach is used by Palisse *et al.*, however, using the goodness-of-fit test to distinguish between encrypted and non encrypted files achieving 99.3% true positive rate [30].

Wolf described in his latest work [39] multiple ransomware detection approaches. It includes a comparison of the metadata of a file before and after encryption is used. A high similarity indicates that little changes were made to the file thus indicating the absence of encryption.

3.2 Network Based Ransomware Detection

A victim's computer is like a zombie machine (compromised computer under remote direction) under ransomware instructions. It needs to communicate with the C&C system for information exchange such as the key needed for encrypting victims' files. Therefore, an additional step is taken into consideration to check if the communication between those two entities based on network activity is sufficient to detect malware attacks before the payload's fully execution and file losses. Some recent search topics switched from host to network analysis.

For example, Ganame *et al.* stated that to have a better coping mechanism with existent cyber threats researchers should move from signature to behavioral based detection [18]. Thus,

they developed a data breach system that is able to identify zero-day malware: WannaCry. Its main advantage lays in its capacity to block the source of compromise preventing further ransomware propagation.

Cabaj *et al.* presented in their work clear characteristics of CryptoWall and Locky families [13]. They both communicate with the server via HTTP POST requests. Despite providing valuable insight on these two distinct families, very little information can be retrieved for instance from Shade ransomware since all its traffic is encrypted. In addition to that, other type of ransomware for example Bitman and Teslacrypt have some samples that communicate with the C&C (Command and Control) via GET requests only as shown in our experiments (for example 2e7f7cc9a815efd0a6590042ce27f6da Teslacrypt ransowmare). Thus, these ransomware samples can go unnoticed.

Alhawi *et al.* are able to detect Windows ransomware while only tracing the network traffic achieving a 97.1% detection rate [9]. Yet, no indication is provided as to whether the detection occurred before or after the encryption process. Moreover, conversation flow is used for the classification. In other words, N suspicious records are required to detect an anomaly or ransomware attack.

Our work is an extension of Alhawi *et al.* [9]. The same approach based on machine learning to flag malicious ransomware communication is used. However, our analysis is performed on packet inspection rather than the conversation flow. We take one step further in the analysis to localize this traffic in the ransomware workflow: When does it occur? Is it sufficient to separate benign vs ransomware activity?

We extend the current work sphere by analyzing packet flows rather than conversation flows. We are aware of the challenges encountered by using machine learning for network intrusion detection (lack of training data, data variability, malware polymorphism, adversarial training, ...) [10,31,38], we take them into consideration and we adopt the Machine Learning method as a means for classification.

4 Proposed methodology

We develop in the current paper a proof of concept that can be implemented in a driver as a future work to avoid being detected by a ransomware process.

We endeavor to thwart ransomware nefarious behavior by analyzing the network traces. The malware analysis routine consists in executing the ransomware and gathering the needed information: system logs for reconstructing the malware session and displaying the timestamp of the first encryption process and network logs for traffic classification. Then, the machine learning process enables a classification between malicious and benign records. Finally, an evaluation is made to check whether the detection of malicious traffic occurs before the encryption process or afterwards. To accomplish this task, our proposed mechanism is divided into 3 main parts: Data Collection, Session Reconstruction & Data Filtering and Analysis & Model Development. It will be thoroughly explained in the following sections. All the steps are summarized in **Ransomware Network Alert** Algorithm.

4.1 Data collection

The ransomware is downloaded and executed on the pc of the victim (our bare metal platform) for 2 up to 3 minutes which is the time required until the encryption note or encrypted files

Algorithm Ransomware Network Alert RNA

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

pop up on the file system. Moreover, the time constraint is also due to the encryption process involved in the ransomware infection that could also encrypt the collected information.

The Wireshark and Process Monitor executables are launched on Windows OS 7 32 bits as in [21]. Each one of them has an independent task for collecting the following information: Wireshark collects the information about network activity whereas Process Monitor gathers the whole system activity (including network information).

Log formats are present below:

- PCAP (Packet Capture) File: Data created by Wireshark that contains the network packet data generated during a live network capture (source and destination IP address, ports, data exchanged, length of the data, ...). It can be analyzed later on for network intrusion detection purposes.
- PML (Process Monitor Log) File: File created by Process Monitor and contains the log of system activities (process name, process id (PID), path, ...).

The information from Process Monitor and Wireshark is acquired to perform a network analysis. Needed data is collected from an automated bare metal malware analysis platform built from scratch using Clonezilla Server [15].

A crawler downloads a ransomware from two databases Virus Share *(https://virusshare.com)* and Malwaredb.Malekal *(http://malwaredb.malekal.com)* (currently down), then it is executed on Windows 7 32 bits machines for a period of 2 to 3 minutes. A dump corresponding to this malware behavior is saved for further post mortem analysis. For scalability reasons, parallel machines are used to perform the tests as well as an improved disk image distribution.

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

1054 ransomware were executed on Windows7 OS (table 1). **Howbeit, 100 ransomware executables are kept for the machine learning phase since they were active** (encrypted the files of the victim). Even though only 100 samples are used for experiments but the machine learning is performed on packets. For example for 12 Bitman samples, we extracted 714 network records. Whereas if we consider the network flow as shown in [9] we get only 62 malicious records to evaluate.

Stratosphere IPS dataset is used for normal captures only [5]. In fact, it contains recent normal traffic captured since 2013 until 2017. An additional information is the description of the behavior captured making the labeling process feasible. Moreover, in the project's malware section, it contains ransomware packet capture: it is a mean of comparison between the ransomware traces provided by their dataset and our own generated in a bare metal platform explained in the following sections.

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

Table 1. An overview of the active ransomware families (total of 100 active samples from 1054 tested), ranked in descending order according to their samples number.

4.2 Session reconstruction & Data filtering

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

In the following section, we describe the preprocessing (filtering) of the collected data to gather solely ransomware activity.

Let $\mathbb{P}$ be the ensemble of all the Processes running in Windows.
Let *p_name*, *p_pid*, *p_ppid* be respectively the name, the PID (process identifier) and the PPID (parent process identifier) of a specific process p.

Ransomware executable names are associated with their MD5 or SHA-256 hash. They represent a unique identifier, that is known prior to the execution of the ransomware for testing purposes (*line 4 of Algorithm Ransomware Network Alert*). To extract the information from the PML file, an initial lookup is made on all the processes that have a name constituted

of the concatenation of the (Ransomware_MD5Hash or Ransomware_SHA256) and (.exe) a filename extension representing an executable file on Windows. The operator + denotes the concatenation operation.

$Ransomware\_name = Ransomware\_MD5Hash + .exe$

$Ransomware\_name = Ransomware\_SHA256 + .exe$

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

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

However, any process running on Windows creates different children as threads or new processes to accomplish its tasks or parallelize the workload. In ransomware's case, one thread is created for listing the files, another one for encryption. For this reason, the tree/graph of the current processes is essential since it displays the relation among all of them (*line 8 of Algorithm Ransomware Network Alert*).

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

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

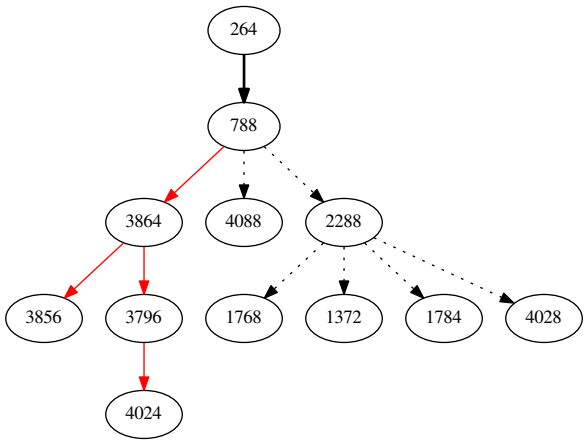

Fig. 2. TeslaCrypt Process IDs Tree: Red Arrow.

Figure 2 displays a sub-tree of some subprocesses running on the machines. The red arrow marks the beginning of TeslaCrypt's execution. It is clear that TeslaCrypt malware creates many processes to accomplish its tasks, even though having a benign parent and siblings. Therefore, it is essential to build this "relation" graph (*line 9 of Algorithm Ransomware Network Alert*).

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

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

Ransomware Network Session Reconstruction

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

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

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

Table 2 displays the different features found in a PML file (table 3) with the basic network elements (for instance IP addresses and ports) while detailed and additional characteristics (TCP checksum, flags, windows size) can be extracted from a PCAP file (table 4).

Table 2. TeslaCrypt Network Information extracted from:

Table 3. PML File.

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

Table 4. PCAP File.

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

4.3 Supervised Machine Learning

The goal of this machine learning step is to develop a model for ransomware detection via network traffic analysis. Point anomaly represents a suspicious record at a given time t: when a specific data instance deviates from the normal pattern. Whereas, collective anomaly represents a collection of similar events that are abnormal [7]. For example, point anomaly can be flagging Record#1 from table 3 since it is not similar to benign records. Therefore,

it is used for the machine learning process to flag any malicious network communication established by the ransomware. Its main advantage compared to collective anomaly is an early detection of ransomware presence rather than having to analyze n packets to expose the malicious behavior.

The supervised approach is effective since labeling the data is possible in our system. Thus, it enables us to detect other variants of ransomware based on an extrapolation of the data acquired throughout our experiments (line 17). Most of the research done in the literature on network intrusion detection via machine learning uses the following algorithms: decision trees, k-nearest neighbors and random forests [11,33,29]. Therefore, they will be adopted to detect ransomware behavior as a deviation from normal traffic. To perform this classification, Scikit-learn, a free software machine learning library, is used.

Our analysis addresses point anomaly subdivided in two, whether TCP or UDP protocol is used. In fact, each packet is different than the other and presents few common features such as IP addresses and ports.

Whereas, for the collective anomaly, the conversation flow is used. Each row in the list displays the statistical values for exactly one conversation (ports, addresses, duration and packets exchanged). This work has been already covered in the literature in [9].

Benign traffic is downloaded from Stratosphere IPS, an open source dataset used to reproduce the experiments [5]. It contains sufficient captures for our analysis.

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

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

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

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

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

Data Preprocessing can have a significant impact on the performance of various ML algorithms [26]. It handles, among other things, missing values and categorical variables. In fact, an intervention is needed since classification models can not handle these elements on their own. In our samples, empty values are replaced by zero, as for the IP addresses and flags they are transformed into integers. Overall, the whole dataset consists of solely numerical values.

5 Experimental results

UDP Results

For the Cerber and Zerber samples, we achieve a 100% detection rate using any of the Decision Trees, Random Forest or K Nearest Neighbors. In fact, the difference is explicit. More than 16000 UDP packets are sent through incremental IP addresses having the same length in a matter of seconds. Additionally, the same information is being conveyed to all those different servers or zombies. The protocol used is solely UDP, very rare in a user normal environment, and is blocked in some companies. Moreover, it is comparable to a Denial of Service (DOS) attack due to the important number of contacted servers via UDP that is not common in normal behavior in just few seconds.

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

TCP Results

The results of the other samples are presented in tables 5 to 9.

Supervised Learning Algorithm	True Positive Rate	True Negative Rate	False Positive Rate	False Negative Rate	Training Time (seconds)
K nearest neighbor (n=2)	99.56	98.13	1.86	043	0.004
Decision Tree	100	100	0	0	0.01
Random Forest	100	99.79	0.2	0	0.03

Table 5. Bitman Classifiers Performance Metrics.

Supervised Learning Algorithm	True Positive Rate	True Negative Rate	False Positive Rate	False Negative Rate	Training Time (seconds)
K nearest neighbor (n=2)	100	99.97	0.02	0	0.13
Decision Tree	100	100	0	0	0.16
Random Forest	100	100	0	0	0.24

Table 6. Cerber Classifiers Performance Metrics.

Supervised Learning Algorithm	True Positive Rate	True Negative Rate	False Positive Rate	False Negative Rate	Training Time (seconds)
K nearest neighbor (n=2)	100	99.99	1.4*10e-2	0	3.76
Decision Tree	100	100	0	0	1.02
Random Forest	100	100	0	0	1.57

Table 7. Shade Classifiers Performance Metrics.

Supervised Learning Algorithm	True Positive Rate	True Negative Rate	False Positive Rate	False Negative Rate	Training Time (seconds)
K nearest neighbor (n=2)	99.31	97.88	2.11	0.68	0.004
Decision Tree	99.31	99.34	0.65	0.68	0.009
Random Forest	100	100	0	0	0.035

Table 8. TeslaCrypt Classifiers Performance Metrics.

Supervised Learning Algorithm	True Positive Rate	True Negative Rate	False Positive Rate	False Negative Rate	Training Time (seconds)
K nearest neighbor (n=2)	100	100	0	0	1.59
Decision Tree	100	100	0	0	0.15
Random Forest	100	100	0	0	0.32

Table 9. Zerber Classifiers Performance Metrics.

Supervised Learning Algorithm	True Positive Rate	True Negative Rate	False Positive Rate	False Negative Rate	Training Time (seconds)
K nearest neighbor (n=2)	38.35	98.11	1.88	61.64	8.02
Decision Tree	98.46	100	0	1.53	0.54
Random Forest	95.7	100	0	4.29	0.7

Table 10. Zero Day Classifiers Performance Metrics.

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

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

A benchmark comparison is possible with the proposed work in [9]. In fact, the authors perform machine learning algorithms on protocols regardless if they were TCP or UDP based. However, we separate them since UDP records outweigh TCP ones, thus, this separation will enable us to have a balanced dataset. In addition, raw features are used such as described in section 4.3. It means that any record or communication can be flagged without delaying the alert mechanism that relies on having n malicious conversation flows. Decision trees lead to more accurate results (98.46% vs 97.10% in [9]).

5.1 Zero Day Ransomware Detection

The experiments conducted are divided into two parts. Signature based ransomware detection explained in the aforementioned sections where the training and the testing is performed on samples from a specific ransomware RA, RB, ..., RN (see Figure 3, Part a).

Yet, to detect zero day attacks, an administrator should test on new variants of ransomware. To implement this task, training is carried out on malware samples that appeared earlier or in the beginning of 2016. As for the tests, they will be executed on different ransomware families excluded from the training set (see Figure 3, Part b).

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

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

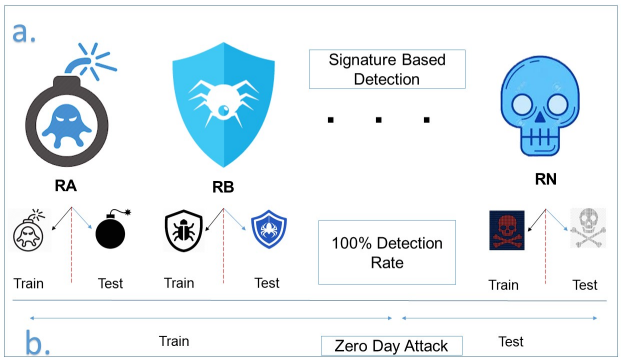

Fig. 3. ML on Ransomware Families.

Since test samples did not figure in the training set, we have 98.46 % as true positive rate and 100% as a true negative rate (table 10). They still represent a high value since the strategy of requests sent between the victim and the C&C is shared among the majority of ransomware families.

5.2 ALERT Time

Checking if encrypted files with ransom notes exist or not is crucial, in other words, if the detection occurs before the beginning of the encryption process or at the end. Consequently, it is essential to recapture the time of the last packet sent and the start of ransom notes. For example, after Cerber's network communication, it creates 9 different threads and immediately after that, the encryption process takes place. It leaves some nanoseconds for the prevention mechanism to take a decision (blocking or killing the process, freezing the pc) before any file loss. Table 11 shows that the first alert, a network UDP send request, appeared before the ransom note #DECRYPT MY FILES#.html.

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

Table 11. Snippet of Cerber PML File, hash: 534da47881eba8a6d3cf676e6c89d1be.

Table 12 presents the percentage of samples that made a communication with the server before an obvious ransom note or encrypted file (RansomAlert). For Bitman and TeslaCrypt families, only 1 sample communicated with the C&C before it displays a ransom note. It means that the detection mechanism based only on network traffic is not appropriate for those families. Howbeit, network traffic detection for Shade ransomware is efficient and will spare file losses for victims.

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

An example of Bitman ransom Notes is given below:

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

Table 12. Encryption Alert.

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

Some Cerber samples killed Process Monitor process several times during their execution, so the PML file retrieved is corrupted. Therefore, a difference is found between the number of active samples in table 1 and in Figure 12. To scale down any possible error, we did not consider those 10 truncated samples in the Alert Analysis because the acquired data was incomplete.

5.3 Results Overview

Based on the timeline mentioned in the context (Figure 1) and on the network traffic, ransomware have evolved throughout the years and are polymorphic. They used to communicate via non encrypted HTTP traffic (TCP requests), then other families moved to GET requests. Shade ransomware for example uses only TLS protocol for its communication. In addition to that, it was one the pioneers for IPv6 communication. In 2016, UDP communication emerged. Based on the data gathered, new variants of ransomware can be detected if the divergence between new samples and existent ones are low. However, many cases are covered in our work. Attackers will have to work on covert channels for exfiltrating information or keep encrypted communication similar to benign application.

Tests are also performed on 18 samples from Cerber, Zerber, TeslaCrypt and Bitman without any Internet connection. The encryption still took place. Nonetheless, we know that the keys were generated locally, enabling us to retrieve them via a simple hook to Windows Crypto API or is hard-coded in ransomware's executable, highly unlikely. Two identical ransomware samples are found in Bitman/TeslaCrypt and two others in Cerber/Zerber. It denotes a resemblance between those families. For example, 2d2c589941dd477acc60f6b8433c3562 MD5 hash is flagged as Bitman by 7 anti-virus companies and as TeslaCrypt by 8 other anti-virus companies [6]. They are kept for signature based detection (no duplicate records in the same family since it appears just once), but removed from zero day analysis.

6 Conclusion

In this work, we are able to detect ransomware through network traffic monitoring. We conclude that the majority of ransomware behave similarly. We found some common patterns

among various families. To get a precise ransomware detection, we use machine learning techniques.

To sum up, network alerts represent a first suspicion means informing the user of the presence of a potential ransomware. However, some drawbacks exist. This first alarm can take place after the creation of encrypted files or ransom notes as we noticed in some families. In addition, few elements are needed for a classification, we have underfitting problems (Zerber samples), prone to adversarial attacks. Besides, only Decision Trees among the tested algorithms provided high detection rates for zero day attacks. For all the reasons mentioned above, Network Alerts should be backed up with system data to provide a general detection mechanism, working on all types of ransomware. As for our future work, we will gather additional information from the file system to present a multi-layer alert strategy to detect ransomware's payload as early as possible. In this work, we chose multiple malware samples and executed them. We should examine the puzzle of infection mechanism such as spam email to detect ransomware download process before its installation on the victim's computer. Furthermore, merely 10% of the samples have encrypted files, that means we have only 10% active ransomware. We will check if it is due to evasion mechanisms (Sysinternal Tools, Wireshark,...) or if the ransomware database is outdated (C&C servers are down).

References

1. Baltimore ransomware attack. https://www.bbc.com/news/technology-48423954
2. Kaspersky Press Release. https://www.kaspersky.com/about/press-releases/2016_attacks-on-business-now-equal-one-every-40-seconds
3. Malwarebytes Blog. https://blog.malwarebytes.com/
4. Reveton Attack. https://krebsonsecurity.com/2012/08/inside-a-reveton-ransomware-operation/
5. Stratosphere IPS. https://www.stratosphereips.org/l
6. Virus Total. https://www.virustotal.com
7. Ahmed, M., Mahmood, A.N., Hu, J.: A survey of network anomaly detection techniques. Journal of Network and Computer Applications **60**, 19–31 (2016)
8. Al-rimy, B.A.S., Maarof, M.A., Shaid, S.Z.M.: Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. Computers & Security **74**, 144–166 (2018)
9. Alhawi, O.M., Baldwin, J., Dehghantanha, A.: Leveraging machine learning techniques for windows ransomware network traffic detection. Cyber Threat Intelligence pp. 93–106 (2018)
10. Amit, I., Matherly, J., Hewlett, W., Xu, Z., Meshi, Y., Weinberger, Y.: Machine learning in cyber-security-problems, challenges and data sets. arXiv preprint arXiv:1812.07858 (2018)
11. Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials **18**(2), 1153–1176 (2016)
12. Cabaj, K., Gawkowski, P., Grochowski, K., Osojca, D.: Network activity analysis of cryptowall ransomware. Przeglad Elektrotechniczny **91**(11), 201–204 (2015)
13. Cabaj, K., Gregorczyk, M., Mazurczyk, W.: Software-defined networking-based crypto ransomware detection using http traffic characteristics. Computers & Electrical Engineering **66**, 353–368 (2018)
14. Chen, Z.G., Kang, H.S., Yin, S.N., Kim, S.R.: 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. pp. 196–201. ACM (2017)
15. Clonezilla: The Free and Open Source Software for Disk Imaging and Cloning,. http://clonezilla.org/
16. Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barenghi, A., Zanero, S., Maggi, F.: Shieldfs: a self-healing, ransomware-aware filesystem. In: Proceedings of the 32nd Annual Conference on Computer Security Applications. pp. 336–347. ACM (2016)
17. F-Secure: Evaluating the customer journey of crypto-ransomware and the paradox behind it. Tech. rep. (Jul 2016)
18. Ganame, K., Allaire, M.A., Zagdene, G., Boudar, O.: 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. pp. 169–181. Springer (2017)

19. Genç, Z.A., Lenzini, G., Ryan, P.Y.: Next generation cryptographic ransomware. In: Nordic Conference on Secure IT Systems. pp. 385–401. Springer (2018)
20. Gómez-Hernández, J., Álvarez-González, L., García-Teodoro, P.: R-locker: Thwarting ransomware action through a honeyfile-based approach. Computers & Security **73**, 389–398 (2018)
21. Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., Khayami, R.: Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence. IEEE transactions on emerging topics in computing (2017)
22. Idika, N., Mathur, A.P.: A survey of malware detection techniques. Purdue University **48** (2007)
23. Kharaz, A., Arshad, S., Mulliner, C., Robertson, W., Kirda, E.: UNVEIL: A large-scale, automated approach to detecting ransomware. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 757–772. USENIX Association, Austin, TX (2016), https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kharaz
24. Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., Kirda, E.: Cutting the gordian knot: a look under the hood of ransomware attacks. In: Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 3–24. Springer (2015)
25. Kolodenker, E., Koch, W., Stringhini, G., Egele, M.: Paybreak: Defense against cryptographic ransomware. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. pp. 599–611. ACM (2017)
26. Kotsiantis, S., Kanellopoulos, D., Pintelas, P.: Data preprocessing for supervised leaning. International Journal of Computer Science **1**(2), 111–117 (2006)
27. Mbol, F., Robert, J.M., Sadighian, A.: An efficient approach to detect torrentlocker ransomware in computer systems. In: International Conference on Cryptology and Network Security. pp. 532–541. Springer (2016)
28. Moussaileb, R., Bouget, B., Palisse, A., Le Bouder, H., Cuppens, N., Lanet, J.L.: Ransomware's early mitigation mechanisms. In: Proceedings of the 13th International Conference on Availability, Reliability and Security. p. 2. ACM (2018)
29. Muniyandi, A.P., Rajeswari, R., Rajaram, R.: Network anomaly detection by cascading k-means clustering and c4. 5 decision tree algorithm. Procedia Engineering **30**, 174–182 (2012)
30. Palisse, A., Durand, A., Le Bouder, H., Le Guernic, C., Lanet, J.L.: Data aware defense (dad): towards a generic and practical ransomware countermeasure. In: Nordic Conference on Secure IT Systems. pp. 192–208. Springer (2017)
31. Papernot, N., McDaniel, P., Sinha, A., Wellman, M.: Towards the science of security and privacy in machine learning. arXiv preprint arXiv:1611.03814 (2016)
32. Rajput, T.S.: Evolving threat agents: Ransomware and their variants. International Journal of Computer Applications **164**(7), 28–34 (2017)
33. Revathi, S., Malathi, A.: A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection. International Journal of Engineering Research & Technology (IJERT) **2**(12), 1848–1853 (2013)
34. Sahi, S.K.: A study of wannacry ransomware attack. International Journal of Engineering Research in Computer Science and Engineering **4**(9), 5–7 (2017)
35. Salvi, M.H.U., Kerkar, M.R.V.: Ransomware: A cyber extortion. ASIAN JOURNAL FOR CONVERGENCE IN TECHNOLOGY (AJCT) **2** (2016)
36. Scaife, N., Carter, H., Traynor, P., Butler, K.R.: Cryptolock (and drop it): stopping ransomware attacks on user data. In: Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on. pp. 303–312. IEEE (2016)
37. Sgandurra, D., Muñoz-González, L., Mohsen, R., Lupu, E.C.: Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. arXiv preprint arXiv:1609.03020 (2016)
38. Sommer, R., Paxson, V.: Outside the closed world: On using machine learning for network intrusion detection. In: 2010 IEEE symposium on security and privacy. pp. 305–316. IEEE (2010)
39. Wolf, J.: Ransomware detection
40. Yang, T., Yang, Y., Qian, K., Lo, D.C.T., Qian, Y., Tao, L.: Automated detection and analysis for android ransomware. In: 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems. pp. 1338–1343. IEEE (2015)
41. Yaqoob, I., Ahmed, E., ur Rehman, M.H., Ahmed, A.I.A., Al-garadi, M.A., Imran, M., Guizani, M.: The rise of ransomware and emerging security challenges in the internet of things. Computer Networks **129**, 444–458 (2017)
42. Young, A.L., Yung, M.M.: An implementation of cryptoviral extortion using microsoft's crypto api (2005)