Technical Analysis of Thanos Ransomware

Ikuromor Ogiriki, Christoper Beck and Vahid Heydari
Rowan University, Glassboro, NJ, USA
ogirik95@students.rowan.edu
beckch36@students.rowan.edu
heydari@rowan.edu

Abstract: Ransomware is a developing menace that encrypts users' files and holds the decryption key hostage until the victim pays a ransom. This particular class of malware has been in charge of extortion hundreds of millions of dollars every year. Adding to the problem, generating new variations is cheap. Therefore, new malware can detect antivirus and intrusion detection systems and evade them or manifest in ways to make themselves undetectable. We must first understand the characteristics and behavior of various varieties of ransomware to create and construct effective security mechanisms to combat them. This research presents a novel dynamic and behavioral analysis of a newly discovered ransomware called Thanos. It was founded in 2020 and is building up to be the leading malware used by low-to-medium-level attackers. It is part of a new ransomware class known as RaaS (Ransomware as a Service), where attackers can customize it for their desired target audience. So far, it is more prevalent in the middle east and North Africa and has over 130 unique samples already. As part of this investigation, the Thanos ransomware is carefully being analyzed. A testbed is created in the virtual artificial environment that mimics a regular operating system and identifies malware interactions with user data. Using this testbed, we can study how ransomware generally affects our system, how it spreads, and how it continually persists to access the user's information. We can design a new security mechanism to detect and mitigate Thanos and similar ransomware based on behavior examination results.

Keywords: Ransomware, malware detection, Virtual Machine, Threat, Thanos

1. Introduction

Since the advent of cryptos like Bitcoin, cybercrime has been on the rise that led to the development of a new type of malware called ransomware. Ransomware is a malicious software that enables attackers to access users' data and restrict them until a ransom is paid. Ransomware has grown to become the most financially rewarding form of malware for cyber attackers over the years. There are two major types of ransomware: 1) encrypting these data until a ransom is paid to decrypt (crypt) using a strong cryptography algorithm, 2) locking users out without tampering with their data until the ransom is paid (locker). Since the introduction of ransomware, the threat landscape has witnessed an increase of an annual rate of 267 percent. This research presents a novel dynamic and behavioral analysis of a newly discovered ransomware called Thanos.

It's essential to look at previous approaches that have been aimed at analysing and detecting ransomware. Previous research has attempted to identify and eliminate malware infections using the signature-based strategy. However, so far though, this has shown to be ineffective in properly identifying all types of malware. While signature-based solutions are rapid and efficient, they are also easily circumvented by very new or much older kinds of malware as discussed by Mahdavifar and Ghorbani (2019) and Zhang et al (2020). The behavior-based malware detection approach, on the other hand, has stronger resistance to this older virus but falls short since it is exceedingly time-consuming. We may conclude from these two distinct types of malware detection systems that, while both are effective solutions, they are not comprehensive enough to recognize malware.

As a solution, we propose a more hybrid system, utilizing both the signature and dynamic behavior-based detection techniques, with the added layer of the machine learning algorithm. This hybrid system would give a more robust answer to the malware detection challenge.

The primary purpose of this research is to present multiple potential approaches for training malware classifiers using machine learning. Another objective is to prevent ransomware from spreading laterally inside a network. To prevent the lateral movement of ransomware, we will use various techniques such as SDN, Moving Target Defense, and Zero Trust.

2. Related Work

This section includes a brief review of previous ransomware detection methods classified based on machine learning, file system/process monitoring, and network traffic.

2.1 Classifications Based on Machine Learning

2.1.1 DNAAct-Ran

According to Khan et al (2020), the traditional signature-based malware detection is no longer efficient in identifying ransomware. As a result, their answer presented a new and better way using the ground-breaking Digital DNA sequencing engine. This engine employs a machine-learning algorithm to classify ransomware based on its digital genome and phenotype to identify its numerous destructive functionalities appropriately. The suggested DNAAct Ran technique uses machine learning to determine if a software is a ransomware. This technique achieves this aim by first selecting the significant traits, generating digital sequences for those selected futures, and detecting the ransomware. The algorithms utilized are the Multi-Objective Grey Wolf Optimization (MOGWO, an extension of GWO by Mirjalili, Mirjalili and Lewis (2014)) and the Binary Cuckoo Search (BCS algorithm) by Yang and Suash (2009). The classifier's performance in comparison to other ML approaches is used to assess the accuracy of the DNAAct-capacity Ran's to identify ransomware. Some other active machine learning approaches, in addition to this, suggested one, include naive Bayes, decision stump, and the Adaboost classification algorithm.

2.1.2 Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence

Homayoun et al (2020) advocated employing sequential pattern mining algorithms to discover the best attributes of ransomware programs from benign apps and to identify ransomware software. The efficiency of their detection characteristics was examined by using them with the J48, random forest, bagging, and MLP classification algorithms. The criteria for this investigation were the usual types of True Positive (for total samples now recognized), False Positives (mistakenly identified samples), True Negative (number of correctly rejected samples), False Negative (number of incorrectly rejected samples) (Incorrectly rejected samples).

They begin by identifying and defining detectable patterns and occurrences to identify the appropriate attributes for classification. The sequence pattern mining approach will next be applied to each dataset in order to identify the best sequence pattern. Each sequence in each dataset is then cross-matched based on the maximum sequence pattern to highlight the characteristics of the training classifiers. The following are examples and descriptions of maximum sequence patterns: 1) R (for all events must be registry), 2) D (all events must be DLL), 3) F (all events must be file), 4) RF (multiple transitions, but the first transition is from the registry to the file event), 5) RD (multiple transitions, but the first transition is from the registry to the DLL event), 6) FR (multiple transitions, but the first transition is from file to registry event), 7) FD (more than one transition, although the initial transition is (more than one transition, but the first transition is from DLL to file event).

2.2 Classifications Based on File System/Process Monitoring

Prior attempts to detect malware have primarily focused on monitoring its low-level file system operations. UNVEIL by Kharraz et al (2016) is one such technique. UNVEIL detects ransomware by attempting to monitor file activity. They were divided into three types based on file system operations (whether a file was read, written/encrypted, deleted, or overwritten) to monitor these actions. They may be able to detect ransomware assaults as a result of this. The Redemption by Kharraz et al (2017) method was also used to examine the request pattern of a file I/O to see whether there was any potential ransomware for each process. If this is restored, the processes that have been labelled as dangerous will be terminated. These are good solutions in general, but their drawback is that many harmless apps, such as encryption and compression of applications, also have such file access characteristic features. If this is the case, these solutions risk producing a large number of false-positive findings since they consider those features to be the same when identifying the activity of various ransomware file systems. Below are a few more characterizations based on file system/process monitoring.

2.2.1 RWGUARD: A Real-Time Detection System Against Cryptographic Ransomware

Mehnaz et al (2018) presented RWGUARD a decoy-based ransomware technique rigorously tested to analyze 14 of the most common ransomware and detect their operations in real-time. It used both the file change and the process change to identify files encrypted by ransomware. Three monitoring strategies were used in this approach: file change monitoring, decoy monitoring, and process monitoring. By employing the correct CryptoAPI function and learning characteristics identical to the user's encrypted file, it was possible to distinguish between a benign and an encrypted ransomware file. Using this comprehensive decoy system, it was nearly hard for ransomware to distinguish their fake files.

2.2.2 RANSOMSPECTOR

This method is based on the virtual machine introspection approach which was presented by Garfinkel and Rosenblum (2003). This solution integrates file operations like opening, renaming, closing, reading, and writing with network operations like connecting, binding, receiving, sending, and disconnecting. They then matched them to discover which corresponded to specific system calls in the kernel of the operating system. The virtual machine can also collect this, which includes context information like the system call's return value, the parameters, and, lastly, the caller's process. Tang et al (2020) also discovered that a significant number of crypto-ransomware samples linked to a network create a huge number of network patterns with similar patterns that differ from their file activities. As a result, by analyzing how these ransomware programs interface with the network and file system, they will gain a bit more precision and inform the user if there is evidence of a ransomware assault.

2.2.3 Crypto Ransomware Analysis and Detection Using Process Monitor

Kardile et al (2017) suggested a method for identifying ransomware attacks based on a process monitor implemented on top of Cuckoo Sandbox. They intentionally picked sandbox because it removes the danger of data loss because Cuckoo sandbox returns to its original state after the malicious sample has been executed. This method first builds a genuine and bogus environment to run these ransomware strains. They may then capture the file system calls trail and record the I/O access using a process monitor. Their research discovered that when suspected malware attacked the system being targeted, the behaviors and activities of files in the system altered dramatically. They found that the time stamp for the entries in the Master File table was quite close to a ransomware assault occurring on that system by observing the Master File Table.

2.2.4 Cryptodrop

Scaife et al (2016) presented a solution designed for the Windows operating system, which has been known to be frequently targeted by ransomware. This method of identifying ransomware relies on indicators to track the many ways in which a file is changes. If all of these indicators are discovered to be true in a file, it may be determined that the file has been corrupted and contains harmful elements. These signs include categorizing ransomware activity based on their actions into three categories. For class one, the ransomware would try to rewrite what was in the original file by opening it, reading it, encrypting it, and closing it. For class two, alter the location of the user's file, read and encrypt the file, and then return it to its original position. The file name may change from the original name by shifting the file back and forth. The ransomware would examine the file, produce an encrypted copy, and then destroy or replace the original for the final class.

2.2.5 SSD-assisted Ransomware Detection and Data Recovery Techniques

On the storage side, SSD insider++ method presented by Baek et al (2020) incorporates sophisticated features like online ransomware detection, flawless data recovery, and sluggish detection. The technique described for online detection is one of the key distinctions between this suggested approach and signature-based alternatives. The algorithm watches and analyzes the host machine's I/O pattern and makes a judgment during run time by analyzing invariant traits that characterize the I/O behavior of ransomware-affected host computers. This is especially significant since it now allows for identifying ransomware attacks in their early phases. The SSD-insider++ overcomes the drawbacks of earlier software and hardware in detecting ransomware by combining a ransomware detection and a data recovery algorithm onto a single SSD.

The SSD-architectural insider++'s architecture comprises ransomware detection and backup/recovery. To identify any abnormal behavior, the SSD Insider++ employs two distinct file operations known as "update-after-read" and "trim-after-read." When ransomware attacks files, its goal is to remain undetected for the longest time feasible by the user. As a result, if many I/O patterns are discovered, we can interpret this as a symptom of a ransomware assault. Baek et al. studied the behavior of six prominent real-world malware to capture ransomware behaviors. Zerber, Locky, Cryptoshield, WannaCry, Mole, and Jaff are examples of malware among them. To identify the traits capable of differentiating this malware by comparing their I/O footprints to those of common apps. After training and testing with various combinations of this ransomware and programs, the SSD insider++ was able to identify new or undiscovered malware by recognizing their distinctive I/O patterns.

2.3 Classifications Based on the Network Traffic

2.3.1 The Case of BadRabbit

Alotaibi et al (2021) developed a technique for detecting efforts to distribute ransomware at the network level rather than preventing the device from being encrypted, which was already addressed in prior studies using badRabbit as a case study. To accomplish this analysis, they employ two VMs, one Windows 10 and one REMnux, for static analysis. They operated four virtual machines for the dynamic analysis: one with a REMnux acting as a gateway, two with Windows 10 (one infected with BadRabbit), and one with Windows 7. The investigation showed that Bad Rabbit did not need to contact the command-and-control server to exchange an encryption key; instead, it accessed these files using a public key. Because Bad Rabbit is self-propagating ransomware, our solution employs five modules to identify and fight self-propagating malware. Deep packet inspection (dpi) and packet header inspection are examples of these modules (phi), honey pot-based, ARP scanning-based detection, and SMB Packet size checkers.

2.3.2 Ransomware Early Detection by the Analysis of File Sharing Traffic

Morato et al (2018) presented Ransomware Early Detection from FIle SHaring traffic which is usually referred to as REDFISH. This solution may be regarded as a framework for detecting and blocking different ransomware behaviors when the infection encrypts data on a network volume from a Network Attached Storage.

This solution examines the difference in traffic behavior between infected and non-infected hosts. The characteristics they investigated for these host behaviors include how files on a shared file are opened, read, written, and deleted. Their approach is derived from studying SMB/SMB2 traffic over a single TCP connection.

3. Proposed Approach

For this work, our major goal is to implement a novel approach to detect the Thano Malware. The analysis would be carried out in three major layers. The first two layers would be the behavioral and signature-based approaches respectively. A figurative illustration is presented in Figure 1. By using both layers, we would leverage the benefits that both malware strategies implement. Layer three acts as a bonus layer to catch any pitfalls from the previous two layers. In this layer, we introduce machine learning techniques to enable us to train the malware classifiers. Figure 2, Diagrammatically depict the procedure of all the layers. We can completely optimize all three approaches for their resistance in identifying malware samples and run-time efficiency by adding the machine learning algorithm layer.

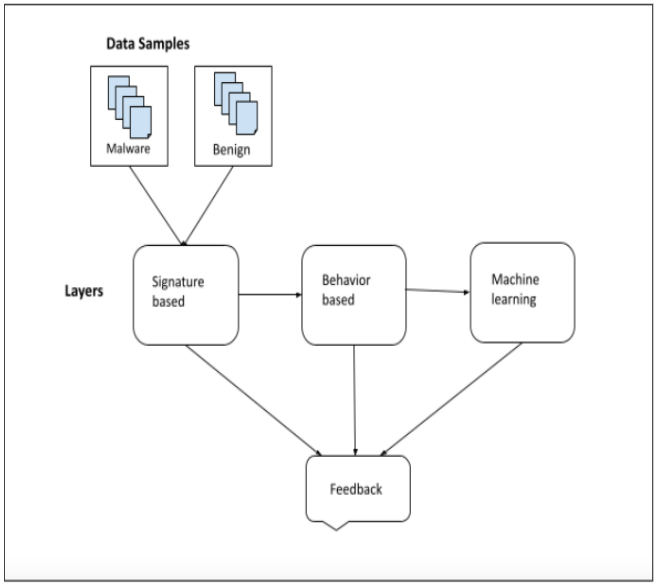

Figure 1: Diagram of the Architectural layer of the hybrid layer system

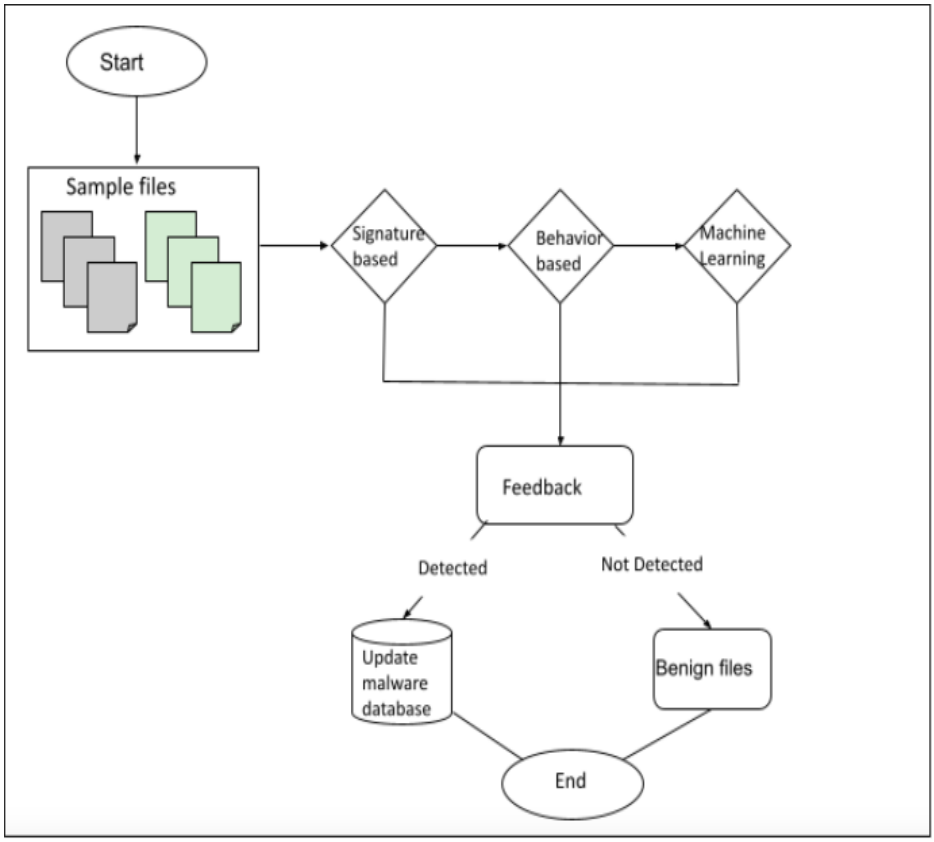

Figure 2: Diagram of the processes of the proposed hybrid technique

4. Research Plan

The primary purpose of this research is to present multiple potential approaches for training malware classifiers using machine learning. Another objective is to prevent ransomware from spreading laterally inside a network. here are two types of malware analysis techniques: signature-based malware analysis and dynamic malware analysis. Each of these strategies has advantages and limitations; the main problem of signature-based detection is that new and undiscovered malware cannot be identified, whereas Dynamic analysis takes time and is rigid. We may utilize the strengths of both analytical approaches while reducing their limitations by combining them to build a hybrid system. To prevent the lateral movement of ransomware, we will use various techniques such as SDN, Moving Target Defense, and Zero Trust.

The objectives are:

- To be able to obtain malware sample data set to perform experiments.
- To be able to properly detect malware, either those that are known or even the new ones.
- To be able to use machine learning algorithms to differentiate between benign and malicious programs.
- To be able to prevent the lateral movement of ransomware.

4.1 Data collection

In previous methods, data were obtained from various websites such as VirusTotal, Virus shares, Zelster, MWanalysis.org, Vxheaven, PCHome Malware Repositories, etc. For research purposes, three major types of datasets are available/ being used and they are:

- Publicly available datasets: These are currently being offered and provided publicly. They are also being updated and maintained by research enthusiasts for the purpose of research all over the world for free in the area of cyber security.
- Artificially generated datasets: These are classified as datasets generated manually using special tools or collected from the network traffic by cyber security researchers.
- Commercial datasets: As the name implies, these are datasets that are not freely offered to the public for use. They are provided as commercial projects and supported by companies for commercial purposes.

For the sake of this research, we shall be obtaining our dataset from publicly available datasets. This is because public datasets are freely available, generate new insights into data collected by fellow researchers, and possess a larger sample size. In order to obtain the biggest sample data, we will obtain malicious ransomware samples from public sites such as virusTotal, Vx heavens, NetLux, Anubis, nexginre for malicious data and Benign samples from portableapps.com. By collecting ransomware sample datasets from a wide variety of publicly available datasets, we can achieve one of our goals of building not only the biggest sample but also the more diverse dataset.

4.2 Data Processing

A testbed similar to what is presented by Kardile et al (2017) will be used. In this testbed, two environments are used. The first is a realistic user environment and the second is an artificial environment. We can then proceed to run the malicious sample in these two different environments. These two environments are built on the Cuckoo sandbox because it is impossible to lose users' data because it reverts to normal after the malicious sample has been completely run. After this, the file system call trace and the I/O access are being recorded using the process monitor. Finally, the proposed system would then detect and analyze the ransomware through manipulating the process monitor. It is also important to note that a virtual machine is often being set up for many dynamic analyses such as this. The need for a virtual machine is to emulate the regular Host OS. Since malicious samples can harm the host OS, a virtual machine OS is being used in its place because even though it looks identical, it is completed isolated from the host OS. Two of the most commonly used virtual machines are VM Workstation and Oracle Virtual Box. Other tools involved in the monitoring of ransomware behaviors include;

- ProcessMonitor: This is a tool that is available on Windows, and it is used for monitoring file systems, processes, threads, networks, and Registry.
- Ollydgb: this is an x86 debugger that is useful for examining the execution of another program.
- WireShark: This is an open-source sniffing and analysis tool that captures network traffic.
- Netcat: Networking tool that can monitor data transfer over a network.

We are combining both the static and dynamic methods of analysis for our hybrid method to provide a more hybrid solution. This hybrid solution is then tested again using four different machine learning algorithms to find the best and most accurate malware detection. The tools we can use for the static analysis in order to be able to extract the static features such as the string, the imports, the exports, and much more, include IDA pro disassembler, capstone, Peid, PsStudios. For the dynamic extraction, we can use tools such as ProcMon, Ollydbg, Wireshark, RegShot, and Cuckoo Sandbox to gather runtime features. As stated above for the dynamic analysis, we would use the OS of a virtual machine so as not to endanger our host OS. Our testbed is being set up this way because while the static analysis has the upper hand by consuming fewer resources and having a higher analysis speed, it cannot detect unknown and obfuscated malware. With the dynamic analysis alone, we can detect unknown malware, but we would consume more resources and find it difficult to detect malware that hides their behaviors during run time. We can catch any fallout from the previous two methods with the additional layer of machine learning algorithms.

4.3 Data Analysis

Classification algorithms are classified into symbolic learning algorithms (CART. C4.5, NewID, AC2, ITrule, Cal5, CN2), statistical algorithms (Naive Bayes, k-nearest neighbor, kernel density, linear discriminant, quadratic discriminant, logistic regression, projection pursuit, Bayesian networks), neural networks (backpropagation, radial basis functions), and Random Forest. However, for the purpose of this research, we will be comparing Logistic Regression, Naive Bayes Classifier, Random Forest, and a Decision Tree.

The aim is to record the performance of Logistic Regression, Naive Bayes, Radom Forest, and Decision Tree Classifier on a data set to classify. The result will be tabulated and graphed to show the recommended algorithm for classifying data sets.

The project will consist of the following stages:

- The data will be collected (and cleaned if necessary) from the various open-source ransomware database.
- The data will be separated into training and test data.
- Use the data set to build a decision tree, perform the necessary tests, and record findings.

- Use the data set to train a model using logistic regression, perform the necessary tests, and record findings.
- Use the data set to train the model using Naive Bayes Classifier, perform the necessary tests, and record findings.
- Compare the performance of the algorithms and provide proof (if any) of the recommended algorithm for the input data.

The goal is to train the four selected classification algorithms to predict whether a malware sample is malicious or benign. The data set used for the training and testing will be the same. The performance of all algorithms will be measured, calculated, and compared based on accuracy, speed, and error recorded. The reason for choosing a random forest classifier is that compared to other machine learning classifiers, they are faster and a lot more scalable to deploy. From previous research carried out by Mehnaz et al (2018) for RWGward, it was also noticed that another tree-based classifier known as decision tree had a high performance like the Random Forest for Non-Linear datasets. For these reasons, we can consider using both tree classifiers for non-linear datasets or use the naive Bayes and Logistic regression for others.

4.4 Evaluation

Recent work by Scaife et al (2016) shows that instead of testing the program that was making the changes to the file, their method tested the user's data that was being changed. This means that they are able to create an early alert for ransomware based on how the users' data was changing.

We can test our research by obtaining and modifying some ransomware to develop new ransomware. So, these new ransomware tools will have new signatures and different behavior. These ransomware samples would also be collected from various ransomware families to have a wide range of classifications.

The four algorithms will be evaluated against the data provided through the various publicly available datasets. The data has been collected and correctly labelled, thus serving as good training and test data. The evaluation of the algorithms would be based on their speed (performance), accuracy (in %), and their marginal error (in %).

In order to be precise with the performance of all algorithms, they will be compared with different input sizes and measured accordingly. We would also employ the use of Machine learning tools. There are several machine learning tools available, and they are divided into two: python-based tools and Java-based tools. For the python-based tools, we have tools such as Scikit learn, Keras, and Theano. For the Java-Based tool, we have Weka.

4.5 Lateral Movement Prevention

We will set up another testbed with multiple VMs to capture how ransomware moves through the network. We will implement various techniques to prevent the lateral movement of ransomware and compare the results. These methods include SDN, network segmentation, zero trust architecture, and IP hopping by Moving Target Defense methods. We will compare their success in preventing lateral movement of all ransomware in our dataset. The cost of implementation, the effect on the network performance will also be measured and compared.

4.6 Expected Outcome

We can present a rather comprehensive and detailed solution by combining the various methods and features present in the hybrid-based detection system. Also, we can overcome the deception and confusion that comes with just one type of detection technique to a large extent. This way, it would allow us to understand the program better more systematically, comprehensively, and accurately. Based on the results of our second testbed, we can also classify solutions for preventing the lateral movement of ransomware based on their accuracy and network performance.

References

Alotaibi, Fahad & Vassilakis, Vassilios. (2021). SDN-Based Detection of Self-Propagating Ransomware: The Case of BadRabbit. IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3058897.

Baek, Sungha & Jung, Youngdon & Mohaisen, Aziz & Lee, Sungjin & Nyang, Daehun. (2020). SSD-assisted Ransomware Detection and Data Recovery Techniques. IEEE Transactions on Computers. PP. 1-1. 10.1109/TC.2020.3011214.

Garfinkel, T., Rosenblum, M., (2003) "A virtual machine introspection-based architecture for intrusion detection." In: Proceedings of the 10th Network and Distributed System Security Symposium, pp. 191–206.

Homayoun, S., Dehghantanha A., M. Ahmadzadeh, S. Hashemi and R. Khayami (2020), "Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence," in *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, 1 April-June pp. 341-351, doi: 10.1109/TETC.2017.2756908.

Kardile, A.B. (2017), Crypto Ransomware Analysis and Detection Using Process Monitor. Ph.D. Thesis, The University of Texas at Arlington, Arlington, TX, USA.

Khan, Firoz & Ncube, Dr & Ramasamy, Lakshmana & Kadry, Seifedine & Nam, Yunyoung. (2020), "A Digital DNA Sequencing Engine for Ransomware Detection Using Machine Learning", IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.3003785.

Kharaz, A., Arshad, S., Mulliner, C., Robertson, W., Kirda, E. (2016): Unveil: a large-scale, automated approach to detecting ransomware. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 757–772. USENIX Association, Austin

Kharraz, A., Kirda, E., (2017). Redemption: Real-time protection against ransomware at end-hosts. In: Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses, pp. 98–119.

Mahdavifar, S. and Ghorbani, A.A. (2019), Application of deep learning to cybersecurity: A survey, Neurocomputing 347 149–176, http://dx.doi.org/10.1016/j.neucom.2019.02.056, http://www.sciencedirect.com/science/article/pii/S0925231219302954.

Mehnaz, Shagufta & Mudgerikar, Anand & Bertino, Elisa. (2018). RWGuard: A Real-Time Detection System Against Cryptographic Ransomware: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, Proceedings. 10.1007/978-3-030-00470-5_6

Mirjalili, S., Mirjalili S. M, and Lewis A. (2014), ''Grey wolf optimizer,'' Adv. Eng. Softw., Mar, vol. 69, pp. 46–61.

Morato, Daniel & Berrueta, Eduardo & Magaña, Eduardo & Izal, Mikel. (2018). Ransomware early detection by the analysis of file-sharing traffic. Journal of Network and Computer Applications. 124. 10.1016/j.jnca.2018.09.013.

Scaife, N., Carter, H., Traynor, P., Butler, K.R.B. (2016): Cryptolock (and drop it): stop-ping ransomware attacks on user data. In: 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), June, pp. 303–312. https://doi.org/10.1109/ICDCS.2016.4627.

Tang, Fei & Ma, Boyang & Li, Jinku & Zhang, Fengwei & Su, Jipeng & Ma, Jianfeng. (2020). RansomSpector: An Introspection-Based Approach to Detect Crypto Ransomware. Computers & Security. 97. 101997. 10.1016/j.cose.2020.101997.

Yang, X.S., and Suash Deb (2009), ''Cuckoo search via Lévy flights,'' in Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC), pp. 210–214.

Zhang, W., Wang, H., He, H. and Liu, P. (2020) DAMBA: Detecting android malware by ORGB analysis, IEEE Trans. Reliab. 69 (1) 55–69, http://dx.doi.org/10.1109/TR.2019.2924677.