CRYPTO RANSOMWARE ANALYSIS AND DETECTION USING PROCESS MONITOR

by

ASHWINI BALKRUSHNA KARDILE

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2017

Acknowledgements

I would like to thank Dr. Ming for his timely guidance and motivation. His insights for this research were valuable. I would also like to thank my committee members Dr. David Levine and Dr. David Kung for taking out time from their schedule and attending my dissertation. I am grateful to John Podolanko; it would not have been possible without his help and support. Thank you, John, for helping me and foster my confidence. I would like to thank my colleagues for supporting me directly or indirectly.

Last but not the least; I would like to thank my parents, my family and my friends for encouraging me and supporting me throughout my research.

November 16, 2017

Abstract

CRYPTO RANSOMWARE ANALYSIS

AND DETECTION USING

PROCESS MONITOR

Ashwini Balkrushna Kardile, MS

The University of Texas at Arlington, 2017

Supervising Professor: Jiang Ming

Ransomware is a faster growing threat that encrypts user's files and locks the computer and holds the key required to decrypt the files for ransom. Over the past few years, the impact of ransomware has increased exponentially. There have been several reported high profile ransomware attacks, such as CryptoLocker, CryptoWall, WannaCry, Petya and Bad Rabbit which have collectively cost individuals and companies well over a billion dollars according to FBI. As the threat of ransomware has become more prevalent, security companies and researchers have begun proposing new approaches for detection and prevention of ransomware. However, these approaches generally lack dynamicity and are either prone to a high false positive rate, or they detect ransomware after some amount of data loss has occurred.

This research represents a dynamic approach to ransomware analysis and is specifically developed to detect ransomware on the user's data. It starts by generating an artificial user environment using Cuckoo Sandbox and monitoring system behavior using Process Monitor to analyze ransomware in its early stages before it interacts with the user's files. By utilizing a Cuckoo sandbox with Process Monitor, I can generate a detailed report of system activities from which ransomware behavior is analyzed. This

model also keeps a record of file access rates and other file-related details in order to track potentially malicious behavior. In this paper, I demonstrate the ability of the model to identify Ransomware by providing a training set that consist of known ransomware families and samples listed on VirusTotal.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

Ransomware is a special type of malicious software, also known as malware, which infects a system and limits a user's access to the system and its files until a ransom is paid. It does this by encrypting the user's files and locking the user's desktop. Ransomware made its first appearances within the last decade and has quickly become one of the more critical threats to security. New ransomware programs are showing up every day, and they are generating enormous profits for the program authors. While most users refuse to pay the ransom, there are enough ransoms being paid to make ransomware a billion-dollar industry, and as a result, cybercriminals are exploring and acclimating new ways to extort money. In 2017, the ransomware landscape has shifted dramatically with the emergence of two new self-propagating threats WannaCry and Petya [17]. WannaCry attacked known Windows network vulnerabilities using exploits like EternalBlue, which allowed an intruder to execute arbitrary code on a targeted system by transmitting customized data packets [17]. WannaCry made global headlines after infecting more than 230,000 systems in over 150 countries and causing an estimated $5 billion in damages [17]. Another example of rapidly growing ransomware that bubbled up in Ukraine and in Russia is Bad Rabbit. It started appearing shortly after the WannaCry and Petya ransomware families made headlines. Bad Rabbit targeted Ukraines Ministry of Infrastructure and Kievs public transport system.

Compared to typical malware, ransomware shows a significant difference in its behavior. For example, traditional malware generally maintains a goal of achieving furtiveness so it can create a backdoor, or so it can gather information such as credentials for financial websites, keystrokes or a user's webcam stream and send it to a remote command and control server without arousing suspicion. On the other hand,

ransomware does not establish backdoors nor collect private information. Rather, ransomware simply creates a denial of service of a system to its own user by encrypting files and/or locking the desktop.

Given the rapid growth of ransomware attacks [17], it is important to come up with a technique to protect a users system against this malicious software. However, we require a low-level understanding of how ransomware interacts with a users system before we can build proper defenses in order to protect the user's files. Many existing techniques for ransomware defense focus mainly on their levels of sophistication and their incremental improvements over ransomware attacks, but they lack a more practical and dynamic approach in ransomware defense techniques by only focusing on keeping their signature definitions up-to-date and only performing the most basic of analysis. In this paper, I take a much deeper dive into the analysis of the key functionalities of ransomware and its effects on a system using dynamic approach.

Today, behavior-based malware detection can be achieved because of the breakthroughs in dynamic analysis. Code-based static analysis analyzes the suspicious files by examining its static properties that consist of header details, hashes, packer signature, date of creation, etc. A static approach for malware detection is usually performed on binary files without executing it and disassembling it using an interactive disassembler. In contrast, dynamic analysis is carried out by keenly observing the behavior of the malware while executing it on the system. It provides new insights by tracking the changes in the system as well as any unusual behavior. In dynamic analysis, certain changes in the system should raise a flag alerting the administrator of the files that have been modified and/or added and deleted, new processes that are running, registry modifications indicating what changes took place in the system, any new services or applications that has been installed, and any the modification of certain system

settings. While both static and dynamic analysis strives to accomplish the same goal, the dynamic approach is often performed in a virtual sandbox environment to prevent the malware from actually infecting a real user's files. By using a sandbox environment, a virtual machine can easily be easily rolled back to a previous state once the malware has completed and been analyzed. Unfortunately, behavior-based malware detection system generally fails to detect ransomware because it is either looking for the placement of backdoors or the gathering of private information. It also fails because ransomware engages in activity that appears similar to benign applications that use encryption or compression. Moreover, many behavior-based systems look for very specific behaviors are constantly plagued by misclassification of ransomware as seen in several antiviruses [1, 5].

In this paper, I present an analysis system which follows a dynamic approach to detect ransomware attacks and model its behavior. In this approach, the system generates a realistic, artificial user environment in which ransomware samples are executed and their interactions with the system environment monitored. Close observation of the interaction of the ransomware with the file system permits the system to identify cryptographic ransomware behavior. In order for a ransomware attack to succeed, ransomware will need to access the user's system, interfere with the files and lock the system leaving it inaccessible. In my approach, many ransomware samples are analyzed allowing for detection of ransomware by observing the file system. In addition, this approach provides insights on how to differentiate between distinct ransomware families such as Cerber, CryptoWall, Crysis, etc. by examining their file system calls. This approach utilizes the open-source Cuckoo Sandbox which creates a safe environment for executing untrusted and potentially malicious executables that prevent them from spreading and doesn't sacrifice a user's files or private information. Cuckoo accepts any

executable binary file and generates a detailed report drafting the behavior of the file when executed in a realistic environment. Once the malicious executable is submitted to the Cuckoo environment, a software suite called Process Monitor tracks the changes in the system in terms of file system activities, registry modifications, and network communication. Process Monitor is an advanced system tool for the Windows operating system that monitors and records all the system calls and other file-related activities. Process Monitor can be tailored to filter out unwanted information to focus on the ransomware behavior.

In my research, I analyzed 495 of the latest malware samples, and this approach was able to correctly identify and detect 479 ransomware samples from multiple ransomware families with no false positives. The 96.7% detection rate of this approach suggests that it performs better than the current behavior-based analysis systems in regard to identifying and detecting ransomware samples expediently [5]. This model can easily be deployed and used on any malware analysis system.

Chapter 2

Background

Like other types of malware, ransomware continues to try new techniques to evade detection, new programming language, new naming conventions and more and more vehement demand tricks to pressure victims into paying ransom. One of the new techniques involves packaging ransomware in RarSFX executable files. Ransomware uses a number of different strategies to increase its potentially harmful behavior and attack the user. For example, it can corrupt the user's file in multiple ways; inject a process or multiple processes, capture users information to a third party for extortion, encrypts users files and make it unreadable and establish a communication with command-and-control servers. Command and control servers, known as C&C servers, are used by attackers to establish and manage secure transmission with the targeted systems within the network. This detection approach expects that the collected samples of the ransomware can use all of the techniques and strategies that other malware may use, such as lack of user activity can be an indication of a Virtual environment, observing system configurations including screen resolution, name of the user, etc. In addition to this, the proposed system assumes that a successful ransomware attack performs one or more of the following activities [5].

Perpetual desktop warning: After the malicious sample submission to Cuckoo sandbox is followed by a successful execution of ransomware infection, a malicious program typically displays a warning message to the victim. This message, also known as a ransom note, informs users that their system has been infected by the ransomware and locks the desktop of the system. The contents of this ransom note provide and instruct users on how to pay the demanded ransom to regain the access to the system and unlock the computer. This ransom note or message can be generated in many different

ways. Malware authors can write an HTML code or can create other forms of continual Window to display this kind of message. But, a very prominent technique to generate the ransom note is to call dedicated API functions such as createDesktop() that creates a new desktop. Call this API function to create a new desktop displaying the ransom note and make it the persistent and configure a system to lock the victim out of the compromised system. Displaying such a ransom note is a usual action in many of the ransomware attacks.

Discriminatory encryption and deletion of users private files based on attributes such as size and extension: Rather than deleting users files, some ransomware families encrypt the user's private files selectively. To encrypt the user's private files in more sophisticated manner, the ransomware sample can list the files based on its size and can even open an application and look for the recently accessed files. Some of the ransomware families can also inject malicious code into any Windows application that can be used to read process memory.

Extensive encryption and deletion of user's files: Ransomware families such as CryptoWall, CryptoLocker, Crysis attacks the system and lists the victims files. It contentiously encrypts the user's private files it discovers and makes it unreadable and inaccessible by withholding the decryption key. The user can regain the access once the desired ransom is paid off. Encryption is carried out by using an encryption key which is either calculated locally by the malware on the targeted system or can be transmitted remotely on Command-and-control servers and then delivered to the targeted machine. An intruder may use Windows API functions such as DeleteFile, DeleteFileTransacted or customized threatening functions to delete the original files of the victim. Ransomware often calls DeleteFile function to remove files the user's file from the system. The attacker can also overwrite the user's files with the encrypted version of a file using Windows API.

In this approach, once the malicious sample file is submitted to the Cuckoo Sandbox [3], it restores the virtual machine environment and starts executing the malicious sample along with the Process Monitor [4] running at the startup. This process monitor monitors and tracks all the system calls, registry alteration, thread activities and network activities. Procmon displays customizable columns that consist of the information about individual events, the name of the process causing that event, event sequence number and timestamp. The result can easily be characterized by its column value. It can utilize one of the following features to carry out ransomware analysis and detection.

Filtering: It is not easy to look for a particular result in Procmon when you have thousands of events. That's where the ability to filter the events plays an important role. This feature is specifically useful in malware analysis as it provides a way to filter on individual system calls such as CreateFile, WriteFile, RegSetValue or any other suspicious calls.

Processes with Process Explorer: This is an extremely powerful task manager tool included in the Process Monitor that must be running when performing dynamic analysis of malware. Process Explorer outlines all the active processes, process properties, and overall system properties. It shows five columns: Process, Process ID, CPU usage, Description and Company Name and updates its view every second.

In this work, I address all these features and scenarios where an attacker has already compromised the targeted system with the malicious code and is able to initiate frivolous ransomware related operations on the user's system.

Chapter 3

Related Works

In [9], Malone et al. propose using hardware performance counters to detect malicious program modifications at load time and runtime by acting as dynamic integrity checkers. The main benefit of this is that it incurs almost no hardware cost since they are built into most processors. The authors claim that hardware performance counters are very efficient at detecting program modifications.

The biggest limitation to the research in [9] is that they tested their approach on programs running solo instead of in a dynamic environment where multiple programs are running simultaneously. Also, this only serves to protect benign programs and does nothing to detect standalone malware. Given that ransomware is generally standalone [12], this approach would require significant modification before it could expand its capabilities.

In [11], Tang et al. continue previous works on hardware performance counters and use them to detect anomaly-based malware by looking at micro-architectural execution patterns. The author's approach goes beyond that of recent works in behavior-based malware detection to detect a much wider range of malware to include zero-day by using machine learning to establish a baseline of benign program executions and use them to detect deviations, and the detector can be used in complement to existing signature-based detections.

Because the approach of using hardware performance counters for malware detection is a relatively new idea, it is still far from production ready. It is prone to a high false-positive rate, and it also requires baselines for every individual program on a computer to detect anomalies in the benign programs themselves in the event they are exploited or are under attack. This doesn't do much for standalone malware detection.

The feasibility of building a malware detector in hardware using data from existing hardware performance counters is examined by Demme et al. in [12]. In this paper, the authors find that they can detect multiple variations of malware within families with ease given a small control set. They also propose modifications in hardware which allow the malware detector to run smoothly beneath the system software which is a great improvement over and fewer buggies than existing software-based antivirus solutions. When used in combination with software-based antivirus, the authors claim that their approach advances the state of the art in online malware detection.

This paper provides a lot of solid insight and can serve as a foundation on which many future works can be built. As with anyone who develops malware-fighting solutions, the author's voice concern over the potential for malware authors to once again adapt their programs to combat even hardware-based approaches such as the one discussed in their paper, but on the other hand it is good to see the hardware community finally join the fight against malware because it forces malware authors to step into a new arena in which they lose the advantage they have had over the anti-malware developers in the software arena since Brain was first discovered.

In [6], Kharraz et al. view the evolution of ransomware in the wild from 2006-2014 and determine that the sophisticated destructive capabilities of most ransomware families lack growth despite the improvements of encryption, deletion and communications techniques in general. They insist that stopping advanced ransomware attacks is not as complex as commonly believed and that defenses involving file system monitoring can be practical and effective because ransomware generates file system requests much differently than benign programs. While the file system monitoring approach is indeed practical as evidenced by my own research, it alone is not enough due to the overwhelming percentage of ransomware samples analyzed by the authors that produced some amount

of data loss before being detected. Other than that, their analysis of the destructiveness of ransomware is thorough and was very useful in developing my own approach to combating ransomware.

Kharraz et al. take this work even further in [5] by focusing on the difference between ransomware and all other existing families of malware. In the malware arena, ransomware stands alone in comparison with all the rest which is why nearly all generic malware detection systems are losing the fight against ransomware. The authors create a dynamic analysis system called UNVEIL which is designed to specifically detect ransomware and is to be used in combination with other malware detection systems. UNVEIL essentially generates a honeypot environment and detects ransomware as soon as it interacts with the user's data. UNVEIL also monitors the desktop to detect any ransomware-like behavior such as a lock screen preventing the user from accessing their files. The authors boast that UNVEIL greatly improves upon the state of the art by demonstrating its capability to identify known evasive ransomware currently immune to detection by existing antivirus systems.

This paper really does improve upon the state of the art given a zero false-positive rate from over 13,000 samples and detects superficial and sophisticated as well as zero-day ransomware attacks. One limitation of the paper is the potential for ransomware to detect the artificially generated environment to avoid it. While it certainly raises the bar of difficulty for the ransomware author to circumvent, it is not infeasible. The other limitation is that most ransomware samples still incur some amount of data loss before detection.

In [8], Kolodenker et al. propose a new system, PayBreak, to effectively combat ransomware and prevent any data loss. It does this by essentially creating a key escrow inaccessible to ransomware that holds every key used in encryption in a secure manner

thus allowing the decryption of any files encrypted by ransomware. PayBreak demonstrates the ability to restore all files lost to twelve different ransomware families, and it does so with negligible performance overhead. While complete data recovery or complete prevention of data loss is the ideal result of combating ransomware, PayBreak only manages to effectively work with only 60% of all ransomware families leaving eight common families of ransomware that can decimate a users system to go uninhibited. PayBreak also lacks a basic robustness allowing it to be evaded simply by ransomware authors rolling back to older versions of crypto libraries or through basic obfuscation and evasion techniques as stated by the authors themselves. Their approach was essentially just a proof-of-concept, and it is uncertain whether the authors will pursue any future work on PayBreak or not.

Chapter 4

System Design

In this chapter, I describe the techniques for detecting ransomware attacks. The malicious samples belong to distinct ransomware families. The ultimate goal of the proposed system is to detect the ransomware attack. As the system is built on the top of Cuckoo Sandbox, it is highly impossible to experience any data loss (which I have created for generating realistic user environment) as the Sandbox used here reverts back to its clean state once the malicious sample execution is completed. Detailed information about the system design and its approach and is described in this chapter.

I first describe the need for creating an artificial, realistic user environment in each malicious sample run. Then, I mention how to capture the file system call traces and record the I/O access using the Process Monitor and describe how the proposed system manipulate the output of the Process monitor to detect and analyze the ransomware.

Generating artificial user environment: It is important to prevent the malware analysis environment against the fingerprinting techniques. Sophisticated malware writers capitalize on and manipulate the static features inside the malware analysis systems such as the name of the computer, IP address of the computer, etc. and launch exploration attack to copy and use both public as well as private malware analysis systems. One static feature that can have a compelling impact on the effectiveness of the malware analysis systems is the user data that can be used efficiently to fingerprint the analysis environment. That is, in some tricky environments, where simple gimmicks such as virtualization checks are impossible, an impractical user environment can be a common sign that the code is running in a malware analysis system.

A potential approach to address such exploration attacks is to create the user environment in such a way that the user data is real, persuasive and non-deterministic in

malicious sample execution. This automatically generated user environment serve as an appealing target machine to boost the ransomware to tamper user files and attack the user's data while at the same time, prohibiting the feasibility of being recognized by the attacker. Apparently, generating a user environment is an important factor to be considered, especially if this is to be done automatically. This is treated as a non-trivial problem because the content generator should not permit the malware writer to capture the automatically generated user data resided in the malware analysis environment as well as demonstrate that it does not belong to a real user. In this approach, Cuckoo Sandbox provides four classic ways to prepare the guest machines such as VMware Workstation, KVM, XenServer and Virtual machine. I describe how to automatically generate a realistic yet artificial user environment for ransomware execution in each malware sample submission in Chapter 6.

Monitoring file system activities: The file system monitor in the proposed system has direct access to the data buffers involved in I/O patterns, permitting the analysis system full control of the file system alterations. Every I/O operation contains process name, process ID, operation type, file system path, offset and the result of the I/O events carried out by each malware run. There are multiple ways to read, write, or list files in kernel mode and all of these functions are conclusively converted to a sequence of I/O requests.

Procmon monitors all the system calls it can gather as soon as it is run. As multiple system calls exist on the Windows machine, it's normally unreasonable to trace them all at once. Therefore, Process Monitor provides various features to perform file system activity trace. In this malware analysis system, procmon is run at the startup of each malicious sample execution as soon as the execution starts. As mentioned earlier, Procmon exhibits the configurable columns that contain information about the events

caused by the process. Figure 1 shows a collection of procmon events caused by a process that captured on a Virtual machine running a suspicious file.

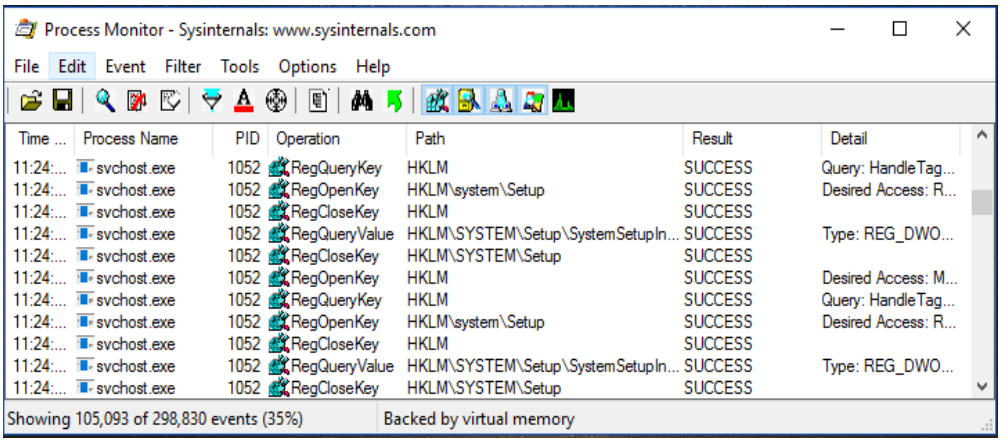


Figure 4-1 Process Monitor Sample Run

Reading the operation column, one can easily read the information about the currently running process, operations that are performed on the system including registry access and file system access. There are multiple entries to look for which makes it difficult to identify the specific entry. Procmon can be set to filter the events caused by the running process, individual system calls, registry activities or other suspicious calls. It filters through all the recorded events when filtering is turned on. Procmon also provides four automatic filters on its toolbar:

- File system: Analyzing the file system interactions can exhibit all the files that are created and used by malware.
- Registry: By reviewing registry operations, self-propagating malware and self-installing malwares can be identified.
- Process activity: Malware spawned processes can be determined by observing the process activity.

14

- Network: Network connection analysis can exhibit the ports that malware is listening on.

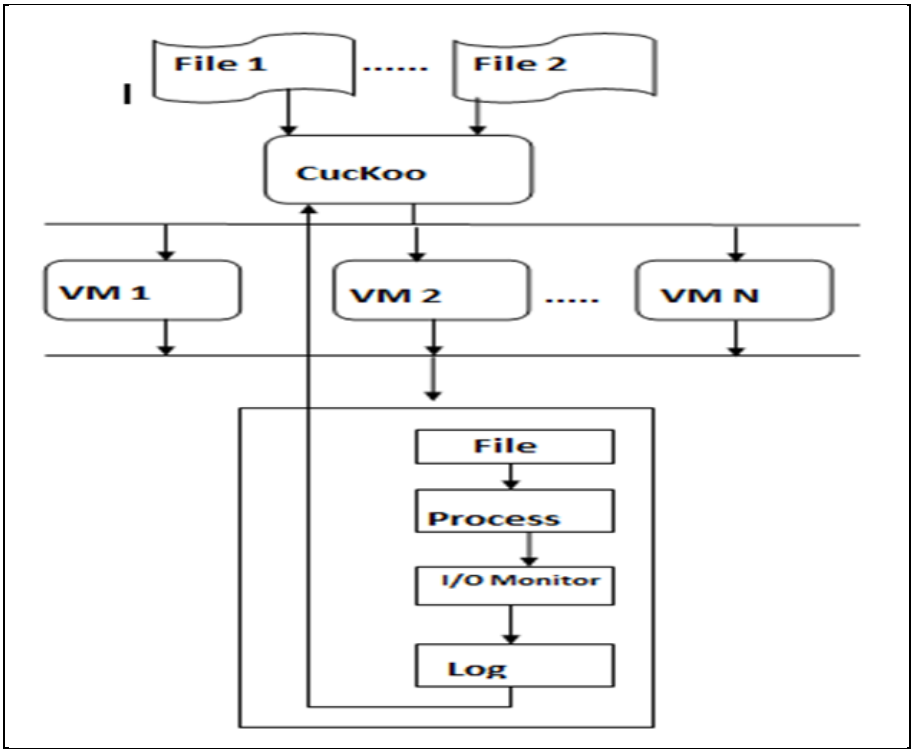
File system activities in Procmon set callbacks on I/O request to the file system. I considered the access pattern in terms of I/O traces such that T is an access pattern of tn and tn = (F;O; P;E) where:

- F is files available in the system
- O is the sequence of I/O access pattern
- P is the processes running at each execution
- E is the entropy of data buffers

For each of the ransomware family, I observed that the malicious samples display distinctive yet repetitive I/O traces. Thus, this strategy to read, write, delete or deny/grant access to the file is reflected in the form of I/O access patterns. Therefore, these file access patterns can be treated as a different I/O sequence for a particular family. It is important to note that, the proposed system mainly considers read, write and delete file access.

Calculating Shannon Entropy: Entropy is a classic indicator that provides information about the ambiguity of data. Encrypted or compressed data has high-value entropy whereas; normal user data has comparatively low-value entropy. For every read and write access request to a file is recorded in the I/O pattern. Generally, a successful ransomware attack outlines a consistently high-value entropy output as the malware reads the user's files and writes the encrypted contents to the files. The system uses Shannon entropy for this computation as correlating the entropy value of read and write I/O requests serve a magnificent indicator of crypto-ransomware behavior. Assuming a uniform random distribution of bytes in the user's file, Shannon entropy can be calculated [4, 7].

15



16

Figure 4-2 System Architecture

Access Patterns: For every execution, once the Procmon collects I/O access traces for the running sample, the system extract the I/O access pattern for each file submitted to the analysis system and check which processes accessed the file. This allows observing the repetition of I/O access patterns generated on behalf of the malicious sample in the run. The specific detection benchmark used by the system to detect the ransomware samples is mainly to identify write and delete operations in I/O pattern in each sample execution. Typical ransomware attack aims to encrypt, overwrite or delete the user's file in order to tamper the user's data. Such I/O request patterns are detected as suspicious file system activity. I studied and examined ransomware samples of different ransomware families, which shows that these attacks can be very different in their tampering strategies such as connecting to Command-and-Control servers, key generation, etc.

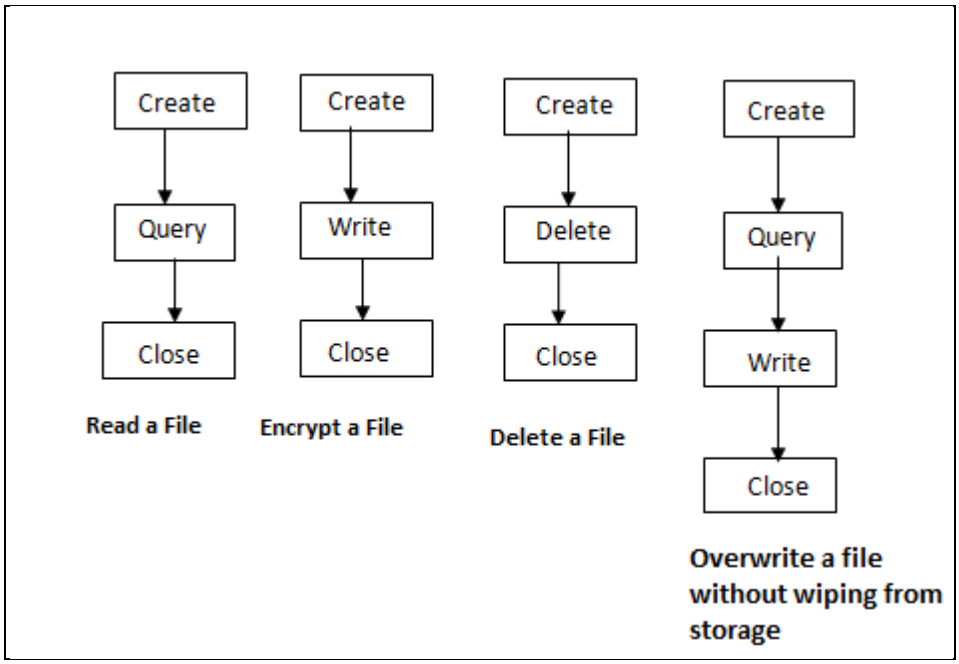


Figure 4-3 High-level access patterns for various ransomware samples

Figure 4-3 exhibits the high-level access patterns for various ransomware samples. The access pattern shown in the figure is displaying I/O sequence of CryptoWall. Its access pattern remains persistent with respect to the ransomware family. I observed the similar I/O file access activity for samples in the Cerber family as well. While both of these families are classified as two different ransomware families, they have similar I/O access patterns while running malicious samples, since they use the same encryption functions to encrypt the user's files.

Considering another example, in Petya ransomware family, the ransomware creates a new file, reads the data from the user's file, generates an encrypted version and simply write the encrypted data to the newly generated file without removing the original file from the disk storage. Such ransomware attacks have a high possibility of recovering the user's files. In another type, the ransomware sample creates a new file

and overwrites the file with the encrypted version based on the original file and then deletes the user's file securely using customized overwriting implementations or using standard Windows APIs such as DeleteFile.

18

Chapter 5

Experimental Setup and Evaluation

Since ransomware attacks user's files, the prototype is built on the top of Cuckoo Sandbox in order to carry out the ransomware analysis and detection without fearing of any data loss and file encryption. Cuckoo is a tool that allows performing sandboxed malware analysis. Cuckoo Sandbox typically provides basic services such as malicious sample submission, performing classic human interaction tasks during analysis, managing multiple Virtual Machines and simulating command line user input. It can record and achieve following results [3]:

- Screenshots taken during the execution of the malicious sample.
- Logs all the system calls carried out by all the processes produced by malware.
- Files that are being created or deleted by the malware being executed.

As mentioned earlier, Cuckoo Sandbox is an improved and commutable design- it can be used standalone or can be integrated with any other large framework. It can analyze distinct file types such as PDF documents, generic Windows executables, compressed files, PHP/Python script, etc along with untrusted URLs and untrusted websites. However, fundamentally, the proposed system could be implemented using any other dynamic analysis system, for example, NorMan Sandbox, Anubis, BitBlaze, etc.

I evaluated my approach using two virtual machines running Windows 7 (64-bit) on Ubuntu that uses Linux kernel. While, particularly Windows 7 (64 bit) is not required to perform the ransomware analysis and detection, it was chosen because Cuckoo Sandbox works well along with Windows 7. Each of the two virtual machines has more than one NTFS drives that provide security to local and network along with some additional features such as fault tolerance, compression, maximum file and partition size, etc. I considered the popular tricks used by the malware writers to change the IP address

range and MAC addresses of the virtual machines to prevent the configured virtual machines from being identified by the malware and took antievasion measures such as configuring DHCP to obtain fixed IP address using default gateway. Furthermore, I permitted bounded and controlled access to the network by specifying a host-only adapter rather than using NAT for the internet access. Network bandwidth was restricted to lessen likely to cause DoS attacks. However, the filtered host-only adapter network setting granted limited DNS, IRC and HTTP traffic so that the malicious samples could communicate with command-and-control servers making it look like a genuine network bridge. In order to manage destructive network traffic such as Spam during the execution of the malicious samples, SMTP traffic was redirected to local honeypot well-supported by and configured in Cuckoo Sandbox [3].

The operating system environment mounted on the proposed ransomware analysis system consists of generic user data such as valid browsing history, saved valuable credentials and other related files and customizations. I also ran some startup programs in each malware run and emulated basic user activity during the execution. Such a user interaction was arbitrarily generated, but it was constant across each run of the malicious sample. Each sample was then executed in the analysis environment for a stipulated time required for the malicious sample to execute; I believe that the required time given for the malicious sample for its execution is sufficient for almost all of the ransomware samples to model their ransomware-like malicious behavior. As described in the earlier chapters, artificial user environment was generated for each ransomware sample run; file system I/O accesses were recorded and execution screenshots were captured. All these results were sent back and stored in the Cuckoo Sandbox segregating its types like screenshots, reports, network activity, etc. After each sample execution, the entire system that is, virtual machine environment is rolled back to its

clean state to hinder any interference across any other execution. All experiments of executing distinct ransomware families and analyzing the ransomware were performed according to well-established experimental guidelines for judicious malware analysis [10]. These guidelines include:

- Correct database: Check whether training dataset and evaluation or test dataset should have different ransomware families.
- Transparency: Acknowledge the ransomware family names and explain the selection of ransomware sample families. Mention the name of the system and its configuration used during the execution.
- Realism: Allow considerate internet connectivity for the malware to communicate with the command and control server and perform real world executions using relevant ransomware families.
- Safety: Expand and mention well-designed containment policies compliment realistic experiments while reducing the potential harm.

Labeled Database: In this experiment, collecting a ransomware dataset was a crucial part of my research. I collected numerous samples of distinct ransomware families from VirusTotal. VirusTotal is a free service that analyzes files and URLs for viruses, untrusted websites and other malicious contents. To obtain defined and exact labeled ransomware samples, I cross checked the list of MD5 hashes with VirusTotal [1]. Being cautious about the ransomware malware sample selection, I considered a malware to be ransomware if at least four antivirus agencies identified it as belonging to the mentioned classification. To get the family names, I considered the commonly used naming conventions and schemes followed by most of the antivirus vendors to assign malware labels and selected the most common label. I captured 782 samples in total. These 782 samples were tested on cloud-based analyzer known as VMRay analyzer [2]. This analyzer also follows

dynamic analysis approach to perform malware analysis and produce the result in a tree-like structure indicating process and thread activities carried out by a malicious sample submitted to the VMRay. I ran all of 782 ransomware samples in VMRay to make sure that these samples were certainly active ransomware. Out of which, I confirmed 715 samples to be active ransomware instances.

After executing each sample, I observed and analyzed the file system activities, registry activities and other file related traces of each sample for any signals of attacks on user's files. I looked for the screenshots captured to check whether running the sample displayed a message or ransom note on the user's desktop. Furthermore, in order to mitigate biased outcomes due to polymorphic techniques as well as to conduct a balanced experiment over ransomware families, I performed this analysis not only based on families and different alternatives per family but also based on individual samples.

Table 5-1 Number of Ransomware Samples per family

Family	**Samples**
Cerber	94
Jaff	30
WannaCry	64
Petya	33
CryptoWall	53
Sage	64
TeslaCrypt	64
Spora	39
Locky	44
Total	**495**

Table 5-1 shows the total number of samples as well as different variants in each of the selected ransomware family.

Non-ransomware samples were also collected in addition to the known and labeled ransomware dataset. I again cross checked the non-ransomware samples in VMRay as well as in VirusTotal to make sure that the selected samples are benign executables including applications that have ransomware-like behavior such as compression, encryption, secure deletion, etc. Table 5-2 shows a list of benign applications.

Table 5-2 List of benign applications and relevant I/O access patterns

Sample	Capability	Operation	Description
7-zip	Compression	Read Write	Read data from file Write data to file
Eraser	Secure Delete	Write Delete	Write data to file Delete the file safely

I also tested few non-ransomware malware samples from distinct malware families to calculate the false positive rate of the proposed system. Table 5-3 shows an example of system calls traces Petya ransomware and NotPetya non-ransomware malware. It indicates that victim's file is first read and then overwritten with an encrypted version by making the API function calls ReadFile and QueryInformation. The file access system calls for NotPetya are similar to that of the Petya ransomware. The main difference is that the Petya ransomware takes an encrypted copy of master boot record and overwrites it with its own malicious code that displays a ransom note, leaving the system unable to boot. However, NotPetya does not keep a copy of overwritten master boot record rather it removes all the files from the targeted system [17].

Table 5-3 Example of file system traces and I/O access

Sample	Operation	Description
Petya	Read Write	Read data from file Write data to file
NonPetya	Read Delete	Read data from file Delete the file

System calls trace genuine applications with possible ransomware-like behavior: The very first goal of this approach is to examine and describe how malicious process interacts with the file system and other related files when a targeted system under the compromised network is experiencing ransomware attack. To achieve this goal, I investigate the trivial characteristics of the ransomware attacks from a file system viewpoint regardless of the technical differences that these attacks expose to, such as the system infection and the key generation techniques. While achieving this goal, one question that pops up is that whether genuine applications such as compression, encryption or secure deletion can generate similar system calls and other file related I/O accesses, resulting in false positives. It is important to note that in case of the benign applications, the original file content is evaluated carefully as the ultimate goal of such applications is to generate an encrypted version of the original file rather than limiting the access to the file. Therefore, the default operation in these applications is that the original files remain untouched even after encryption or compression process. If automatic deletion is willfully activated by the user after the encryption, it possibly results in a false positive.

However, in this approach, I assume that the original data is preserved and the normal default behavior is displayed. This is a legitimate assumption, believing that this

approach is a malware analysis system that focuses on analyzing potentially suspicious samples collected and submitted for analysis. Moreover, I/O access sequence is inspected and displayed in Table 5-4. Considering the contents of Table 5-4, it indicates that the benign programs exhibit distinct I/O access patterns.

Table 5-4 File system calls of benign applications

Sample	Operation	Description
7-zip	Read	Read data from file
	Write	Write data to file
	Create	Create a new file
Eraser	Write	Write/Overwrite data to file
	Delete	Delete the file safely

It is not required that the genuine applications must perform encryption or deletion on the user's files, but it can modify the contents of the files. For example, altering the contents of a Microsoft document and modifying a Microsoft PowerPoint file (by inserting or deleting an image, etc.) generates I/O patterns similar to the ransomware. However, the main difference is that such applications normally generate I/O request pattern for a particular file at a time and iteration of a same I/O requests does not occur over multiple files owned by the user. Also, it is important to note that, benign applications generally do not encrypt, delete, modify or compress the random files but it needs refined user input such as a filename or other options. Therefore, most of the genuine applications would expect some user input or simply perform exit operation when executed in the proposed system.

Evaluation of false positives: As I used the same dataset for detecting false positives, it is less possible to get an accurate precision-recall analysis. It makes the verification of

the detection result real challenging. Re-submitting the samples while checking for false positives is not worth while in all cases as although the dataset contains latest samples, there is a possibility that samples may have become inactive at the time of re-analysis. Therefore, detection results are verified manually. That is, the samples that provided ransom note on the desktop of the targeted machine, manual verification of such detection result is done using post execution screenshots. Using this, I confirmed that the proposed system correctly reported 479 samples that delivered a ransom note at the end of the sample analysis and execution.

It is important to note that, the proposed system is only based on the I/O access sequence and the file system calls. It does not take into account the changes in entropy in the detection phase rather it is only considered for the evaluation. For every sample in the collected database, I keep track of I/O access patterns and the other file related activities. If there are multiple write or delete I/O request patterns for the same user files and there is an exponential increase in the entropy or creation of the new entropy which is way higher than that of the original files, it was confirmed that the detection as a true positives. The generation of new entropy file based on user file is a solid insight into the ransomware detection. For example, a malicious sample that carries out secure deletion techniques may first overwrite the files and such files may have low entropy value. On the contrary, malicious encryption application first needs to encrypt the original file and then overwrite it with high entropy value. In either case, generating high entropy raises a flag in this evaluation.

By applying these two approaches and analyzing the results of these approaches, I did not find any false positives during the evaluation. There were a few sample runs that had noticeable modifications in the execution environment such as desktop exhibited a nonfunctional dialogue box with some unreadable characters or the

malicious program generated a large window having nothing to display on it. However, after careful investigation and observation, it was clear that the extracted text within the dialogue box was not related to any ransomware activity, the proposed system termed it as a non-ransomware sample.

Evaluation of false negatives: Determining the false negative rate is a challenging since manually checking each and every ransomware sample is not feasible. The proposed system of ransomware detection using the labeled dataset, false negatives mainly occurred in samples that make constant alterations on the desktop of the targeted system. From malicious ransomware samples, I eliminated the samples that were not detected as malware by any of the antivirus scanners in VirusTotal and in VMRay after multiple resubmissions. Although the number was very less, by applying this strategy, I could reduce the number of samples up to some extent to check for the false negative.

After each run, I checked for system calls traces and I/O access patterns and verified whether any process has demanded an access to write to the user's files. Along with I/O patterns, I also checked the entropy values. If there is an indication of a write access to the user's files with high increase in the entropy data than the entropy data of the read access, the detection of false negative occurs. By using these two factors, I confirmed that this ransomware analysis and detection system does not have any false negatives.

Decision Model: Manual analysis process of all the captured file system activities is time consuming and may lead human errors if a large dataset is considered. For a large dataset that contains thousands of ransomware samples, it is required to automate the process of analyzing such I/O accesses and file system activities. It not only speeds up the analysis but also helps to mitigate human errors and interactions, turning out to be an effective outcome in terms of providing early detection of a ransomware. To automate the

file system activities analysis process, I observed and analyzed the results produced by procmon and Cuckoo Sandbox such as reports and analysis logs. Considering the file system activities carried out in the first minute from the execution start time, I came up with the numerical values that back up the decision model and generate decision metrics. Based on the analysis result, the threshold value is considered. This threshold plays an important role in the ransomware detection as a metric value exceeding the threshold results into ransomware detection in the system.

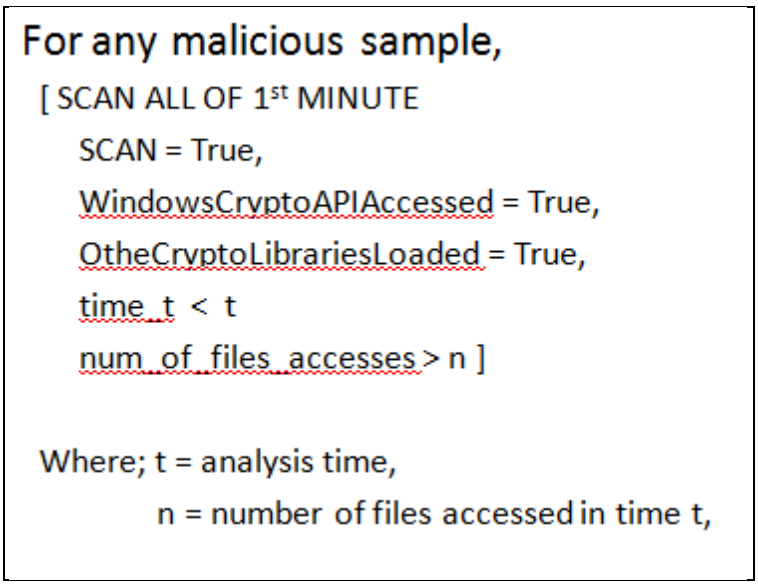


Figure 5-1 Decision Metrics

By applying above decision metrics, I analyzed the ransomware sample execution report and analysis logs and could get with following numerical values.

Depending on the number of files accessed in time t, threshold value 183 is calculated for WannaCry ransomware family whereas, the threshold value 53 is calculated for Cerber ransomware family. Any metrics exceeding these thresholds would result in the ransomware detection. Table 5-5 shows the decision metrics result for WannaCry and Cerber ransomware family samples.

Table 5-5 Decision metrics result

Ransomware sample	Decision Model	
	Time(t)	Number of files accessed(n)
WannaCry	20	183
Cerber	20	53

Chapter 6

Implementation

In this chapter, I describe the implementation details of the proposed system for ransomware analysis and detection for the Windows platform. The malware analysis system mainly focuses on detecting the ransomware through monitoring the file system access and I/O traces. The combination of these indicators provides a strong measure of suspiciousness of the process currently running in the system. The proposed system is implemented on the Windows platform as it is currently the main target of ransomware attacks. I explain how to generate realistically artificial user environments for malicious sample run, how the file system access and I/O traces monitoring was implemented using Process Monitor and how to setup Cuckoo Sandbox along with Virtual machine configuration.

Generating realistic user environment: The user environment is set up in the Windows 7 platform installed in the virtual machine. At every malicious sample execution, this user environment is made of distinct file contents that include digital images, audio files, and other pdf and Microsoft documents. These contents can be accessible during each execution. To generate an artificial user environment, a set of files with different extensions are created. However, the number of files in each extension group is sampled uniformly to obtain accurate Shannon entropy value. Each set of files belong to a particular extension type such .pdf, .docx. .jpg, etc forms a document space for the user environment to execute the ransomware sample. These document spaces that are generated for each set of files are identical to the original user data. The ransomware sample can potentially use following properties to identify the user environment.

File paths: Malware sample looks for the user's files at particular locations in the compromised system. Such locations are usually the Windows registry, directories, etc. Malware authors potentially look for a specific file path or a registry path. If the user's files are stored abnormally at a random location in any execution environment, currently running malware founds it fishy or a virtual user environment and terminates the execution of malicious sample without injecting the malicious code into the system. Thus, the system associates the files of certain types with standard locations in the Windows directory structure. For example, the system does not create document files in a directory with image or media files, rather creates it under My Documents. Furthermore, the length of the path of the user files is also randomly selected and even exhibit different path for the same set of documents. Moreover, the paths to the user files can have variable depths relative to the root folder as a folder may have a set of sub-folders.

Valid contents: According to the observation of successful ransomware attacks, common categories of the files that the malicious sample tries to look for and encrypts are documents, software licenses and authenticity provider keys, file archives and media. Document extensions include doc, docx, txt, ppt, pptx, pdf and py. Keys and license contains extensions such as .pem, .crt, .cer, etc. Compression class has various extensions such as .zip, .rar files. And media contents are a wide variety of audio and video data files with extensions like mp3, mp4, avi along with image files extensions such as .png, .jpg, .jpeg, etc. In order to have the valid and genuine file contents, I collected approximately 8300 sentences by querying 25 meaningful English words in Google. For every submitted word, I collected a meaning text from the search result. These formed sentences were used to generate the content for the user files and named the user files with valid dictionary words. The file names are of variable lengths that do not appear random as it serves some meaning. Moreover, the problem with random file name such

that asdfgh.pdf is that the malware author or the attacker can programmatically calculate the entropy of the file names and its contents to tamper the user data. Therefore, using the files that have meaningful contents, I make it difficult for the attacker to fingerprint the targeted system.

Monitoring file system activities using Process Monitor: Various techniques are developed and used to monitor malicious sample file system activity and I/O access in malware analysis environments. The file system activities exhibit significant changes when the ransomware successfully attacks targeted system. A keen observation of MFT (Master File Table) can be useful to detect the creation, encryption or deletion activities carried out on a file. For instance, significant number of status modifications occurs in a very little timestamp in MFT entries if the system is under the ransomware attack. For encrypted files, MFT entries have a large number of encrypted content. Furthermore, the classifier can be trained on genuine and tampered MFT entries to detect abnormal file system activities when the system is under the ransomware attack [6].

Another possible approach to monitor file system activity is hooking a list of related system calls or file system API functions using System Service Descriptor Table (SSDT) [6]. These API functions hooking can be bypassed simply copying a DLL that has the required code and dynamically loading it into the process address space with a different name. (E.g. Stolen Code, Sliding Calls) Moreover, the ransomware can make use of personalized cryptosystems neglecting the standard APIs to sidestep API hooking to encrypt user files. Hooking the system calls via the SSDT can also be prevented on 64-bit operating system using Kernel Patch Protection. Furthermore, many SSDT functions are subject to change depending on the versions of Windows as well as it lacks the proper documentation. Therefore, these approaches are not suitable for the proposed ransomware analysis and detection system [7].

Instead of API functions or system calls hooking, this system monitors file system activities and I/O access patterns using the Sysinternals advanced process monitoring tool called as Process Monitor. It is an advanced file system monitoring tool that achieves system wide file system monitoring in various Windows versions. Process monitor has various features which are essential for monitoring the files system access along with capturing registry access and network communications occur between the targeted system and the command and control server under compromised network. Each and every event is logged by the process monitor that resides in the startup of the user environment mounted on the virtual machines. Once the malicious sample is submitted to the Cuckoo Sandbox, Virtual machines are restored and the execution is started. It runs the procmon in the background accepting the request to capture the file access and monitor file activities. Procmon starts capturing the I/O traces for the submitted sample. It identifies the currently running process instantiated by the malicious sample from the ransomware family and records all the I/O requests performed by the current process based on its type. I/O requests can be any of the types read, write, delete or create. Process Monitor exhibits the output depending on the settings provided to it. Destructive filters can filter the results based on Registry activities, File system activities and Network activities.

Chapter 7

Performance and Limitations

Chapter 5, Experimental Setup, demonstrates that the proposed malware analysis system accomplishes meaningful, practical and advantageous detection results for the experiment that is carried out on a real-world dataset of latest ransomware families. Unfortunately, malware writers deliberately look for defensive techniques and alter the attack strategies. In this chapter, I elaborate the limitations of the proposed malware analysis system and possible strategies to escape. There is always a potential chance for the attackers to find out the ways to fingerprint the artificially generated user environment and terminate the execution process. However, it comes at a high cost and increases the difficulty for the attacker to programmatically identify the artificially generated execution environment. However, implementing these approaches to detect the ransomware can possibly make the detection much easier because these strategies require hooking particular functions in the Operating system. In addition, these approaches delay commencing the attack by injecting the malicious code or by encrypting the files that increase the risk of being detected by antivirus scanners on the targeted system before a successful attack occurs.

Although I have not come across any sample with another potential behavior that can be observed is, the malware might only encrypt a specific portion of the user's file instead of encrypting the entire file. It can also disorder the file contents using a definite pattern to make it unreadable. Developing such ransomware is quite possible for the malware author. The main idea is that in order to carry out such activities, the malicious ransomware sample should access the users file with write permission and handle the file contents. In this approach, there is no way possible that in the generated user environment, the genuine software will need to open a random file with write permission.

Thus, if a malicious program accesses the files, Process monitor will log this activity and such behavior can be tested and ransomware can be detected in such cases as well. Precisely, there is always the possibility that an attacker will be able to tamper the dynamic approach and the malware analysis environment. For example, stalling code has gained a lot of popularity to delay the launching, masquerade the behavior and prevent dynamic analysis of a sample [13]. Table 6-1 shows the overall performance of the proposed system.

Table 6-1 Overall performance of the proposed system

Evaluation	**Results**
Analyzed Samples	715
Detection Rate	96.7%
False Positives	0.0%

Another example that counters the dynamic approach is that the malware authors use sleep calls. Using sleep calls, malware refrains from suspicious behavior during the monitoring process. Such code takes a longer time to execute giving an attacker the privilege to actively look for indicators of dynamic analysis. Therefore, the proposed malware analysis system uses Sandbox that is more immune to such kind of evasion techniques. The ultimate contribution of this system is the introduction of new techniques for the automated detection of the ransomware using dynamic approach and not just the dynamic analysis of malware. This system aims to detect the user-level ransomware by monitoring the file access and I/O traces. As a result, there is a risk that ransomware may run at the kernel level and counter some hooks which may be used by Process monitor to capture registry access activities and to monitor the file system

activities. However, this would require the ransomware to run with root privileges to exploit kernel vulnerability.

Chapter 8

Conclusion

In this paper, I presented a classic approach for analysis and detection of the latest ransomware. This system specifically identifies the general behavior of ransomware such as malicious encryption of the user files. This behavior is difficult for the malicious program to obscure or alter. The evaluation of the system exhibits that the approach followed in the malware analysis was able to correctly detect 479 ransomware samples from multiple distinct families in a real-world data feed with zero false positives. In fact, this system outruns all existing antivirus scanners and a modern industrial sandboxing technology in detecting both trivial and technically and programmatically sophisticated ransomware attacks. With the help of Cuckoo sandbox with Process Monitor, I could generate a detailed report of system activities from which ransomware behavior is analyzed. In this approach, the classification of the ransomware attack was based on 495 ransomware samples of distinct 9 ransomware families that have emerged recently. Outcomes of this system show that a significant number of ransomware families share similar features and file system access along with I/O records in the core portion of the ransomware attack, but it still has the shortcoming of reliable destructive functions that successfully target the user's file.

This paper also describes how a malicious sample interacts with the file system when a targeted system is under a ransomware attack. I noticed that malicious file system activity of distinct ransomware families can be monitored using Process monitor. While looking at the I/O traces of the ransomware execution recorded by procmon, I observed distinguishable files system activities and I/O patterns of malicious code and benign applications. I propose a typical methodology that allows us to detect a compelling

number of ransomware attacks without making any assumptions on how the malicious program infects user's data.

References

[1]. VirusTotal: Sample provider and analyzer.

[2]. VMRay analyzer: Cloud based analyzer.

[3]. Cuckoo Sandbox: An automated sandbox uses dynamic analysis approach.

[4]. Process Monitor : Sysinternals tool for file system activity monitor

[5]. AMIN KHARAZ, SAJJAD ARSHAD, COLLIN MULLINER, WILLIAM ROBERTSON AND ENGIN KIRDA, *NORTHEASTERN UNIVERSITY. UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware.* Usenix - 2016.

[6]. AMIN KHARRAZ, WILLIAM ROBERTSON, DAVIDE BALZAROTTI, LEYLA BILGE, AND ENGIN KIRDA, NORTHEASTERN UNIVERSITY, BOSTON, USA LASTLINE LABS. *Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attack.* Dimva-2015.

[7]. NOLEN SCAIFE, HENRY CARTER, PATRICK TRAYNOR, KEVIN R.B. BUTLER. *CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data.* IEEE-2016.

[8]. EUGENE KOLODENKER*Z*, WILLIAM KOCH, GIANLUCA STRINGHINI*Y*, AND MANUEL EGELE. *Paybreak: Defense against Cryptographic Ransomware*. Asia CCS-2017.

[9]. COREY MALONE, MOHAMED ZAHRAN, RAMESH KARRI. *Are Hardware Performance Counters a Cost Effective Way for Integrity Checking of Programs?* STC' 2011.

[10]. CHRISTIAN ROSSOW, CHRISTIAN J. DIETRICH, CHRIS GRIER, CHRISTIAN KREIBICH, VERN PAXSON, NORBERT POHLMANN, HERBERT BOS, MAARTEN VAN STEEN. *Prudent Practices for Designing Malware Experiments: Status Quo and Outlook.* IEEE-2012.

[11]. ADRIAN TANG SIMHA SETHUMADHAVAN SALVATORE STOLFO. *Unsupervised Anomaly-based Malware Detection using Hardware Feature. 2014.*

[12]. JOHN DEMME MATTHEW MAYCOCK JARED SCHMITZ ADRIAN TANG ADAM WAKSMAN SIMHA SETHUMADHAVAN SALVATORE STOLFO. *On the Feasibility of Online Malware Detection with Performance Counter.* ISCA 2013.

[13]. CLEMENS KOLBITSCH, ENGIN KIRDA, CHRISTOPHER KRUEGEL. *The Power of Procrastination: Detection and Mitigation of Execution-Stalling Malicious Code.*CCS-2011.

[14]. DANIELE SGANDURRA, LUIS MUÑOZ-GONZÁLEZ, RABIH MOHSEN, EMIL C. LUPU. *Automated Dynamic Analysis of Ransomware-Benefits, Limitations and use for Detection. In ArXiv e-prints, arXiv 1609.03020.*

[15]. Internet Security Threat Report, Symantec [July 2017].

[16]. Rethink Your Strategy To Defeat Evasive Attacks, Palo Alto Networks. White Paper.

[17]. DICK O'BRIEN. *ISTR Ransomware 2017, an ISTR Special Report.*July-2017.

Biographical Information

This report belongs to Ashwini Balkrushna Kardile. Ashwini has obtained a Bachelor's degree in Computer Engineering and Master of Science degree in Computer Science and she has successfully defended her master's thesis in Information Security under the supervision of Dr. Jiang Ming.

Ashwini is interested in pursuing her career in the field of Information Security, Cyber Security and Software Security and Privacy. She has worked on several projects in Software security area involving Static analysis tools, Cloud services, malware analysis and various open source platforms such as Cuckoo Sandbox, Kali Linux, IDA Pro, IDS/IPS, Wireshark, etc. Ashwini was appointed as a Graduated Teaching Assistant/ Grader for the Spring 2017 and Fall 2017 semester and had successfully conducted the labs along with the CTF competition as a part of Information Security (CSE 5380).

Ashwini explored her expertise in the field of Information Security by working as a Research Assistant in Summer 2017 which improved her thought process and gave an insight into her research in the Ransomware Analysis and Detection.

Ashwini is willing to pursue her career in the area of her major and looking for a full-time opportunity in the field of Cyber Security to utilize her skills and R&D knowledge of Malware Analysis and Detection in the corporate world.