5-2022

Ransomware and Malware Sandboxing

Byron Denham

Ransomware and Malware Sandboxing

An Undergraduate Honors College Thesis

in the

Department of Computer Science and Computer Engineering
College of Engineering
University of Arkansas
Fayetteville, AR
May 2022

by

Byron Denham

Ransomware and Malware Sandboxing

Byron Denham
Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR 72701, USA
bedenham@uark.edu

Dale R. Thompson
Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR 72701, USA
drt@uark.edu

***Abstract*— The threat of ransomware that encrypts data on a device and asks for payment to decrypt the data affects individual users, businesses, and vital systems including healthcare. This threat has become increasingly more prevalent in the past few years. To understand ransomware through malware analysis, care must be taken to sandbox the ransomware in an environment that allows for a detailed and comprehensive analysis while also preventing it from being able to further spread. Modern malware often takes measures to detect whether it has been placed into an analysis environment to prevent examination. In this work, several notable pieces of ransomware were placed into sandbox environments to discover how they might obfuscate themselves for evading analysis and to determine ways they propagate. The goal of the work is to identify and understand these how these obfuscation and propagation techniques function in a sandbox, so that mitigation methods can be developed.**

Keywords— ransomware, sandboxing, malware, analysis environment, Wannacry, Cryptolocker

I. Introduction

Although the idea of malware that locks users from their files until a ransom is paid is not a novel one, the existence of asymmetric encryption and the growing ubiquity of cryptocurrency have made the development of ransomware more effective and thus it has become a prevalent cybersecurity threat over the last ten years. While there does exist ransomware that holds the user's data by merely restricting access to particular files, most ransomware gets its leverage to demand a ransom by encrypting the victim's files, rendering them useless. The encryption process of ransomware begins by the generation of an RSA key pair on the side of the attacker, the public key of which is sent to the victim's machine. This public key can be used for one of two things depending on the design of the ransomware. If the designer of the ransomware wants to avoid generating keys on the victim's machine, the ransomware will use the public key for the encryption of the files. In this case a new RSA key will need to be generated for each victim meaning that the public key is likely retrieved from a command-and-control server. If the designer of the ransomware wants to avoid needing to generate a new RSA key for each victim, they will generate symmetric keys on the victim's machine instead, encrypt them with their public key, and send the encrypted symmetric keys back to a command-and-control server. In this case the file encryption would be done with the symmetric keys. Finally, if the victim decides to pay the ransom, the attacker will either send them the private key or decrypted symmetric key needed for the user to decrypt their files.

The increasing popularity of cryptocurrency has made it easier for attackers to demand payments for decrypting files that cannot be tracked. During the first half of 2021, the FBI's Internet Crime Complaint Center received 2084 incidents of ransomware which was a 62% increase from the same timeframe of the previous year causing $16.8M in losses [18]. Because of its use of modern encryption techniques, recovery of files encrypted by the ransomware is often impossible without paying the ransom. This makes the recovery of machines infected with ransomware significantly more difficult than the recovery of machines infected with other types of malware. Recovery from ransomware can rarely be done by actually decrypting the files targeted by the ransomware and must almost always be done by restoring a machine to a previous state from a backup, losing any newer data that was not backed up. However, it is difficult to keep backups up to date manually and off site, and ransomware's targeting of things like shadow copies compounds this problem. Because of the difficulty of recovery from ransomware, effort has been and will need to be put into methods of mitigating and recognizing it as early as possible, potentially recording keys it generates if this occurs, and ideally preventing it from encrypting anything at all. To achieve this level of ransomware mitigation, detailed analysis of existing and new samples will be necessary. A key component of malware analysis is sandboxing: the creation of an environment that allows for the execution of malicious code such that its effects are contained and it does not infect anything outside of the sandbox. The purpose of this work is to detail the techniques and setup used to sandbox general malware for analysis, then to run ransomware samples in these environments and observe how the analysis of ransomware might differ from that of general malware.

II. Related Work

In this section, ransomware, sandboxing for malware analysis, and ransomware mitigation and detection are described.

An overview of how ransomware fits into the landscape of malware as a whole and details of how it operates are described in [10][2][8]. The work in [2] gives a synopsis of malware in general then discusses the specifics of ransomware. It cites phishing emails and drive-by downloads as the two primary

vectors of infection used by ransomware. Ransomware generally follows the steps of infection, execution, attacking backups, encryption, and user notification. In [8], the way ransomware scans to spread from the infected machine to others on the network is described. Self-propagation is a potential source of infection along with software security exploits, redirecting Internet traffic to sites that can infect a victim, injecting malicious code into non-malicious websites, and malvertising campaigns [10]. Six common ransomware families as well as the use of command-and-control servers to communicate encryption keys are described in [10]. Command-and-control servers are owned by the attacker and function to send instructions to malware, or retrieve information gathered from it. Many types of malware use obfuscation techniques and [10] discusses observed obfuscation methods used in ransomware such as the use of TOR, domain shadowing, polymorphic behavior, and dormancy. TOR's onion routing is used to obscure malware's interaction with resources like command-and-control servers. Sometimes communication to these servers will be blacklisted by an IDS on a victim's machine. Domain shadowing allows for such communication to occur through the registration of domain names that are not yet blacklisted, but that point to malicious addresses. Polymorphic behavior allows for the ransomware to create mutations of itself that are functionally the same but by merely being different are more difficult to identify. Dormancy allows for the ransomware to remain inactive until the ideal time to execute occurs. All three of these sources, as well as [1][5] mention the existence of two main types of ransomware: locker and crypto. Locker ransomware restricts access to a computer or parts of a computer until the ransom is paid while crypto ransomware encrypts a user's files and will only decrypt them once the ransom is paid. This distinction is notable because crypto ransomware is more distinct from the majority of malware than locker because of its use of cryptographic encryption. Cracking the modern encryption algorithms used by crypto ransomware is considered a nearly unsolvable problem unless the ransomware has a specific inherent flaw that allows for recovery of its encryption keys. Because of the uniqueness of this type of ransomware, the malware that is sandboxed in this project is crypto ransomware.

Because recovery from crypto ransomware is considered nearly impossible, much of the research has gone into methodology for ransomware's early detection and mitigation [1][5][6][7][9]. The authors of [1] cite a growing increase in the existence of ransomware that avoids detection and contribute to the ability to recognize ransomware by focusing on how ransomware propagates through a network, rather than how it behaves on an already infected machine. Work in [5] looks at the actions of ransomware one step later in its process by examining file system traversal prior to encryption. This paper points out the issues with detection methods that focus on the entropy generated from the encryption process as it allows for the ransomware to do some damage before being halted. One detail of the ransomware process that [5] discusses that is worth considering for sandboxing is how ransomware chooses which files to encrypt. Specifically, the authors state that ransomware may operate by blacklisting or whitelisting file extensions to encrypt. The focus of contribution in [6] to the problem of ransomware is the automation of the analysis of logs generated by ransomware to aid in detection. The work in [7] analyzes logs from ransomware attacks with high detail logs being used for detection and low detail logs to aid in recovery attempts. Highlights of common detection methods used in literature are described in [9]. Machine learning and artificial intelligence have been used to analyze the behavior of ransomware to discern patterns that can be used in its detection [1][5][8][9]. Most literature that focuses on detection uses data sets that must be gathered from sandbox environments to develop analysis tools [1][5][6][7][9]. All of this indicates the significance of examining how the nuances of ransomware should be considered in an analysis environment.

The authors of [3] and [4] examine the tools, techniques, and practices used in modern malware analysis. The work in [4] primarily discusses tools used, and briefly discusses sandboxing. It divides analysis into two types: static and dynamic analysis. Static analysis is defined as information gathering done without executing the malicious code. It mentions tools like PeStudio, Resource Hacker, and IDA Pro for things like string extraction and reverse engineering. Dynamic analysis is defined as analysis done during the execution of the malicious code. It mentions using tools like Wireshark and Process Monitor to track how the malware executes. Lastly, it describes the importance of sandboxing in dynamic analysis to prevent malware from escaping. The work in [3] emphasizes the widespread use of virtualized environments rather than physical systems for sandboxing, listing sandboxing products such as Cuckoo that can be used. The work in [4] consolidates and analyzes information gathered from career malware researchers in an effort to create models for malware research. The authors cite five primary goals that malware researchers have with analysis: 1) finding IP addresses, domain names, and hashes to blacklist, 2) determining the specifics of the executable's malicious behavior, 3) labeling the malware into families of similar type, 4) finding indicators of compromise in the malware, and 5) generating a report for their clients. Information gathered from the researchers also suggests an increasing focus on the importance of dynamic, behavioral analysis further indicating the significance of understanding sandboxing approaches. Sources for obtaining samples of malicious code are also discussed, citing sources like VirusTotal, The Zoo, and Malware Bazaar [17][14][19]. All models of malware researchers' workflow in [4] use static and dynamic analysis, but one of the models contains an emulating malware stage. The idea behind this is to emulate things like responses from command-and-control servers to make the malware operate to its full extent. Because of ransomware's use of command-and-control servers, the need for emulation was kept in mind for the sandboxing of ransomware. The authors in [4] discuss how sandboxing is done and why analysts use the sandboxes they do. Sandboxes can be physical machines or virtual machines and can use proprietary or open-source tools. The broad consensus among analysts in [4] is to use virtual machines rather than physical because of the expense of potentially damaging costly physical machines as well as issues with testing large quantities of samples not scaling well with physical machines. Analysts also preferred open-source software rather than proprietary because of the greater potential for custom configuration with open-source. Lastly, custom or premade sandboxes (such as Cuckoo) can be used. Many of the analysts preferred the customizability of non-premade sandboxes.

III. Sandboxing Setup

Based on the broad consensus of researchers in [4], the sandbox environments used in this work were custom made virtual environments. Because most ransomware is designed for the Windows operating systems [11], the virtual machines used Windows 7 and Windows 10 as operating systems to study previously analyzed malware. All virtual machines exist on a host operating system. Even though the sandboxes used were in the virtual machines, the specifics of their host were still considered. In the setup used in [1], one of the physical machines was responsible for monitoring the network traffic between the other machines and thus was not intended to be infected. The monitoring machine had a different operating system than the machines to be infected because the ransomware was not designed to infect it. For this project, because the sandboxes were Windows 7 and Windows 10, the host machine was the most current version of Ubuntu Linux (Ubuntu 20.04.3). While it is likely possible to create a virtual sandbox with the same operating system as the host, having the two sandboxed operating systems be different from the host operating system is one of the best preventative measures to avert the host machine being infected. Lastly, it is best to use a host machine dedicated to hosting sandbox environments that uses no important passwords and does not contain important files.

Oracle's VirtualBox [12] was used as the virtual machine manager. There are many options for virtualization software, but VirtualBox was chosen for several reasons. VirtualBox is open-source software as per the preference of the analysts in [4]. In addition, VirtualBox is highly customizable, with the ability to change certain specs that are very useful for malware sandboxing such as memory and network settings. VirtualBox also supports snapshots which is helpful for the easy customizing of sandboxes that the analyst's in [4] considered valuable. Snapshots allow the state of the virtual machine to be saved, and those saved states can be returned to at any time. This feature has numerous benefits. The most apparent benefit being that the state of the machine can be saved before infection by the malware and can easily be returned to no matter how much damage is done to the virtual machine. In addition, many tools are used for analysis in sandbox environments. This means the setup of the variety of analysis tools need only be done once per operating system as long as a snapshot is taken of that state. It is possible for malware to need a specific environment to run to its full extent. If there exist two samples that need the same operating system but different configurations, snapshots allow for both configurations to be created without needing to setup multiple virtual machines. In the setup of this project, four main phases of snapshots were used: first a snapshot of the operating system after initial setup, second a snapshot of the machine once the desired analysis tools were installed, third a snapshot of the machine with the malware samples put onto the machine but prior to infection, and lastly a snapshot of the infected machine.

There are many instances of malware that are designed to detect that they are being run in a virtual environment. Neither of the ransomware samples analyzed in this project fall into this category but as ransomware continues to evolve there will likely be a growing number of ransomware instances that will employ these tactics. One of the ways malware will attempt to recognize being run in a virtualized environment is through observation of system resources. This can be mitigated by giving the virtual machine as realistic of an amount of resources as possible, or at the very least more than the default settings of the virtual machine manager. Also, installing more software than just analysis tools and doing normal operations to generate logs can make it more difficult for the malware to detect being in a sandbox.

Once the virtual machine with the desired operating system has been created and a snapshot taken, analysis tools must be installed. The process of installing analysis applications into a virtual environment can be a long and difficult procedure as different software may have unique dependencies and system requirement that must be met. If only a small number of analysis applications are wanted in the sandbox it would not be difficult to download a few; however, there is a large quantity of tools that are useful for malware analysis. Flare-VM [13] is a free and open-source tool that automates the process of installing a wide variety of applications useful for malware analysis as well as adding repositories to aid in the installation of other desired packages. Flare-VM includes many tools such as hex editors like HxD, debuggers like x64dbg, disassemblers like IDA, portable executable analysis tools like PeStudio, and networking tools like Wireshark. It also installs utilities like wget, Yara, and Python. The Mandiant distribution of Flare-VM assumes the installation is occurring on Windows 7 Service Pack 1 or later and 50-60 GB of total storage. It also requires the .NET framework to be at least 4.5 and WMF to be at least 5.1. The Windows 7 machine used for this project did not meet these requirements initially, but the installation of these two dependencies highlights some nuances of sandboxing with older operating systems. The Internet Explorer web browser on the Windows 7 virtual machine was too outdated to travel to the download pages for .NET 4.5 and WMF 5.1. This problem can be solved using VirtualBox's shared folder functionality. Shared folders exist to aid in the transfer of files between a host machine and a virtual machine and can be used to transfer the installers for .NET 4.5 and WMF 5.1 or for a modern browser's installer that can be used to obtain other installers. Once these requirements were met, the zip file for the Flare-VM installation package was downloaded and extracted, an instance of Powershell was run as administrator and its execution policy was set to unrestricted per the Flare-VM install instructions. Lastly the installation script was run. One potentially useful tool that was not included in Flare-VM that was downloaded was Microsoft Network Monitor. Microsoft Network Monitor is like Wireshark but can associate captured packets with the process that they originated from.

As mentioned in [4], emulating responses expected by malware can be important for getting it to exhibit all the behavior a researcher might want to observe. Ransomware almost always contacts a command-and-control server for key generation or storage and may not run if communication is not established. Ideally, a malware sandbox environment will be isolated from network connections with any other machines and thus will not be connected to the internet meaning the ransomware will not be able to reach its command-and-control servers. One of the tools provided by Flare-VM that has the potential to solve this issue is FakeNet-NG [16]. FakeNet-NG is designed to intercept and redirect network traffic while

TABLE I. VIRTUALBOX VIRTUAL MACHINE COMMUNICATION ABILITY UNDER DIFFERENT NETWORK SETTINGS [15]

Mode	VM→Host	VM←Host	VM1↔VM2	VM→Net/LAN	VM←Net/LAN
Host-only	+	+	+	–	–
Internal	–	–	+	–	–
Bridged	+	+	+	+	+
NAT	+	Port forward	–	+	Port forward
NATservice	+	Port forward	+	+	Port forward

simulating network responses. For example, in a virtual machine put on a host-only network, a ping command to google.com would be unable to find an IP address that it could send those ICMP packets to. However, with FakeNet-NG using its default settings and running on that same machine, a ping request to google.com would result in all four packets being "received" the response being that the destination host is unreachable. Once the Flare-VM install script completed and Microsoft Network Monitor and FakeNet-NG were installed, another snapshot was taken.

After tools have been installed, it is time to acquire samples and finalize isolating the sandbox for the safe execution of ransomware. All malware samples used in this project were obtained from the Github repository, theZoo [14]. In order to complete isolating the sandbox, any shared folders were disconnected, the VirtualBox guest additions were removed, and the virtual machine was put onto a host-only network. Getting rid of folders shared between the host and the virtual machine prevents the ransomware from potentially placing malicious data onto the host or from encrypting files on the host. The removal of the VirtualBox guest additions (which may be present on the virtual machine if shared folders were used) is a way to further prevent the malware's detection of being in a virtualized environment. Future ransomware that attempts to resist analysis might look for the existence of things like VirtualBox's guest additions that would reveal that it is being run in a virtualized environment. Lastly, the virtual machine was put onto a host-only network as suggested in the Flare-VM documentation. As said in the VirtualBox network documentation and shown in Table I, a host only network allows for communication with the host and with other virtual machines on the network while preventing communication with the Internet and other machines on the host computer's local area network.

VirtualBox networks can be created and configured in the network tools of VirtualBox's VM manager. The adapter can be configured automatically or manually and the DHCP server settings for the network can also be configured. In order for the virtual machines on the same network to be configured the DHCP server must be enabled. If the host machine has the same operating system as the sandbox, it would likely be better to use an internal network as it would not be able to communicate with the host machine.

IV. ANALYSIS OF WANNACRY

Malware analysis typically begins with a static analysis as it does not require the execution of the malware which will, in the case of ransomware, cause parts of the system to become unusable. In this section, the analysis of Wannacry is described [21]. The Wannacry ransomware attacks occurred in 2017 using

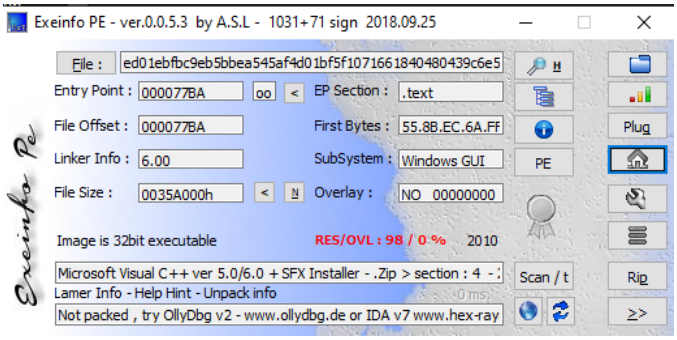

Fig. 1. Wannacry Exeinfo PE results

the EternalBlue exploit, which targeted a vulnerability in the SMB protocol on Windows. The attack impacted individuals and organization across the world, notably affecting the National Health Service in the United Kingdom, impacting their ability to provide medical care [20]. The attack was eventually halted by malware researcher Marcus Hutchins after the discovery of a DNS kill switch found in the code of the ransomware.

During static analysis, the sample was first put into PeStudio which revealed that the first two bytes were 4D 5A (M Z), that it contained the DOS stub message "!This program cannot be run in DOS mode.", and that the original name file was "diskpart.exe" all of which are indicators used to identify disguised executables. PeStudio is a static analysis tool that examines an executable to search for indications that it might be malicious. All three of the hashes generated by PeStudio (md5, sha1, sha256) on the sample were flagged as malicious by 62 out of the 73 scanners on VirusTotal [17]. PeStudio was also able to recognize the use of a PKZIP resource. As shown in Fig. 1, the Exeinfo PE tool was able to recognize that the sample was not packed. Lastly the sample was passed to the strings command line tool which was able to recognize more notable strings than were shown by PeStudio. Some interesting strings that appear in Fig. 2 showed the existence of multiple ".wnry" files many of which seemed to contain messages in different languages, such as "msg/m_english.wnry". It also references some potential executable that may be called by the main sample such as "taskdl.exe" and "taskse.exe". Lastly, there were multiple strings that referenced cryptography and encryption such as "CryptGenKey" and "CryptEncrypt" indicating that the sample is crypto ransomware.

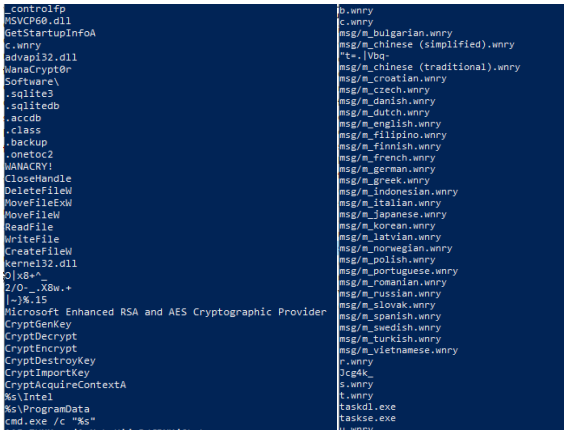

Fig. 2. Exa Wannacry Exeinfo PE results

Fig. 3. Wannacry warning message

For dynamic analysis the sample was run alongside Process Hacker, Microsoft network monitor, and Wireshark. The first notable attribute of the Wannacry sample was the fact that it would execute to its full extent in both the Windows 7 and Windows 10 sandboxes without needing any network emulation. The sample was able to encrypt files and display the Wannacry warning message as seen in Fig. 3.

Beyond being convenient for the dynamic analysis in the sandbox, this reveals several things about the sample. The fact that Wannacry will go through with encryption regardless of whether it is able to communicate with a command-and-control server could mean that Wannacry is storing the keys used to encrypt files on the host machine until it is able to send them to the command-and-control server. This would be unlikely unless Wannacry is storing encrypted forms of the keys on the infected machine that could be decrypted to reveal the actual keys to the command-and-control server. Secondly, the fact that encryption occurred discloses that key generation is performed on the infected machine, as it was not possible for Wannacry to retrieve them from a server.

Only one core was used by the sandbox initially, as ideally a sandbox will not need to use a large amount of system resources unless necessary to bypass malware's virtual machine detection which was not necessary in this case. Keeping this in mind, resource hacker showed the CPU's available resources hovering between 90%-100% prior to execution of the sample. At the beginning of the sample's execution the available resources dipped to 40%-50% range and eventually stabilized to the 70%-80% range. It was also notable that the initial executable stayed in the range of using 5%-20% of CPU resources and many subroutines were called by the initially run executable including: "cmd.exe", "cscript", "conhost", and "taskdl" all of which may be further analyzable by extracting the pkzip resource detected in PeStudio. Eventually the processes being run by the malware stabilized to what is pictured in Fig. 4 with the above being from the Windows 7 sandbox and the below being from the Windows 10 sandbox.

As mentioned prior, a famous attribute of the original Wannacry virus was the presence of the DNS kill switch

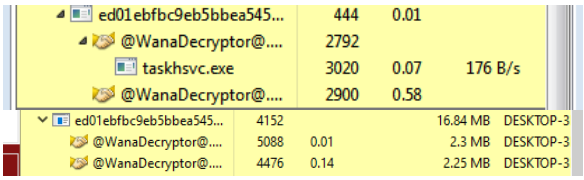

Fig. 4. Wannacry processes in process hacker

discovered and registered by malware researcher Marcus Hutchins, preventing Wannacry from spreading further; however, more permutations of Wannacry that were not affected by the kill switch were eventually released. The DNS kill switch is a section of Wannacry's code that attempts to confirm if a certain domain name exists and only allow it to execute further if the domain does not exist. One reason such a kill switch might have been implemented into Wannacry originally was the fact that it spread through worm-like behavior, meaning it could spread without action being taken by the victim. The presence of the kill switch may have been a way for the authors of Wannacry to eliminate its effects if they ever decided it had spread further than they had intended. Something noticeable from the Wireshark capture was a lack of DNS queries originating from the IP of the infected sandbox machine, suggesting the sample was not susceptible to the DNS kill switch. All packets captured with the source IP address being that of the sandbox machine used NBNS, LLMNR, and IGMPv3 protocols; however, it was difficult to discern which if any of these packets were associated with Wannacry. Also, no packets captured were able to be associated with the Wannacry executable by Microsoft Network Monitor. The NBNS protocol is a name resolution protocol like DNS, but all the queries originating from the sandbox's IP address were to the name "B-PC<1c>", a domain controller. All these requests were to the broadcast address of the subnet. LLMNR is also a name resolution protocol, however, all its queries were to "b-PC" through the multicast address 224.0.0.252. The IGMPv3 packets were also to a multicast address, 224.0.0.252, possibly to search for other devices in the subnet.

One interesting behavioral component of Wannacry was that Wannacry offers to decrypt some files for free, even if it has no network connection. This could mean that Wannacry stores at least some of the file encrypting keys unencrypted on the infected system, or the files that it claimed to decrypt were never actually encrypted. Lastly, both sandboxes were put onto the same host only network that was setup to allow them to communicate with each other (in VirtualBox, a DHCP server must be enabled to do this). The sample was run on only one of the sandboxes to see if infection would eventually spread to the other sandbox. Despite running for about an hour, there was no indication of the ransomware spreading to the other machine.

V. Analysis of Cryptolocker

In the static analysis of Cryptolocker [22], PeStudio also found the first bytes to be 4D 5A (M Z) and the DOS-stub "!This program cannot be run in DOS mode." indicating the sample as an executable. All three hashes generated by PeStudio were recognized by 63 out of the 73 of the scanners on VirusTotal. Although PeStudio was unable to recognize the original name of the file, one of its indicators was that "The file references

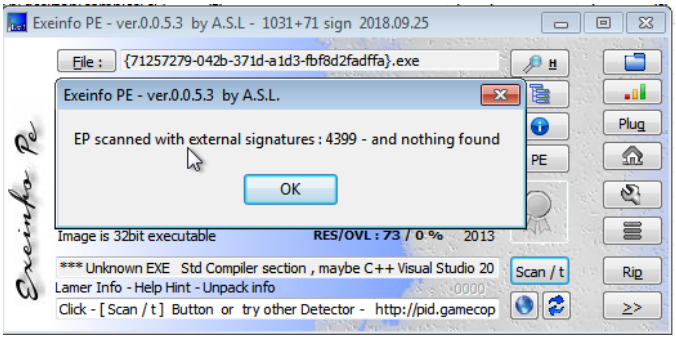

Fig. 5. Cryptolocker Exeinfo PE results

extensions like a Ransomware | Wiper". It was also able to recognize the use of an RTF resource. The Exeinfo PE scan shown in Fig. 5 indicated that the sample was not packed.

As shown in Fig. 6, the strings command line tool was also able to recognize more in this case as well. Some of the recognized strings seem to be text for a warning message explaining how to make payments with cryptocurrency such as "Bitcoins can be transferred through a computer of smartphone without an intermediate financial institution." There were also strings indicating the use of cryptographic encryption such as "Microsoft Enhanced RSA and AES Cryptographic Provider" and strings indicating HTTP GET and POST requests.

Unlike Wannacry, Cryptolocker was not able to run to its full extent in the initial sandbox environment. After execution, there were no encrypted files and no warning messages appeared. Unlike Wannacry, which upon execution will generate multiple files in the same directory that the initial sample was executed, Cryptolocker deletes the initial executable upon running it. However, in both the Windows 7 and Windows 10 sandboxes the process remained running while barely using any resources.

The process remained running on reboot of the sandbox, possibly trying to do more after reboot than it did beforehand which was suggested by the fact that, the Windows 7 sandbox requested permission for the process to continue to run after reboot but not upon initial execution. Microsoft network monitor was unable to associate any traffic with this specific process but some packets in Wireshark showed NBNS queries to suspicious addresses like "pfwnssjgmdxb.ru".

Fig. 6. Cryptolocker strings output

Fig. 7. Cryptolocker processes in process hacker

Six queries to these domains would occur before being switched to another domain. There were also LLMNR and IGMPv3 packets like in Wannacry. The IGMPv3 packets didn't seem any different from the Wannacry analysis, but there were LLMNR and other NBNS packets that did not occur in the Wannacry analysis. All three types of packets had the same destination addresses as in the Wannacry analysis, but besides there being NBNS queries to suspicious addresses like those mentioned prior, there were LLMNR and NBNS queries to the name "ISATAP", which is used for IPv6 addresses. The final difference in packets captured was the existence of NBNS queries to the name "WORKGROUP". The fact that the sample was unable to run and the existence of persistent queries to suspicious domains indicates that Cryptolocker does not generate keys on the victim's machine and instead retrieves them from their command-and-control servers. If this is the case, then Cryptolocker is likely trying to hide itself and quietly continue to achieve connection with the command-and-control servers until it can get keys to encrypt. The sample would also not run to its full extent when Fakenet-NG was running using default configurations. Based on responses from Fakenet-NG, it was able to automatically intercept the suspicious requests but clearly was unable to reply with the responses expected by the Cryptolocker process.

VI. Evaluation

The two instances of ransomware tested in this project proved to be good examples of both ransomware that was easy to analyze in a sandbox and ransomware that was more difficult to analyze. The samples also helped to demonstrate some of the nuances of sandboxing ransomware. The most noticeable takeaway was the fact that isolation and safe execution was considerably easier than emulation. The use of VirtualBox and Flare-VM made the creation and setup of a virtual machine designed for the execution and analysis of malware simple and straight forward. VirtualBox's snapshot capability allowed for easy reversion to uninfected states in the sandboxes for cases when something might have been missed in a dynamic analysis. The use of VirtualBox's host only network in combination with a host operating system different than that of the sandboxes prevented the spread of the ransomware to the host machine or any machine other than the sandboxes, while allowing the sandboxes to communicate with each other. While it would be possible to use an internal network such that the host was not in the network, there is a distinct advantage in having the host on the same network, especially with ransomware. Because ransomware encrypts files on the machine it infects, this could cause difficulty if the ransomware targets pcap files created by programs like Wireshark for packet capturing. This will become increasingly likely as ransomware is developed to resist analysis. Having the host be attached to the network allows for the monitoring of the network without the files created for monitoring being at risk of encryption by the ransomware. After all the analysis, which included running both the Wannacry and Cryptolocker samples multiple times in both the Windows 7 and Windows 10 sandboxes, there was no indication of infection on the host machine. All files on the host stayed unencrypted and

the host's system monitor showed no suspicious processes or any process named similarly to those created by either of the ransomwares.

Emulating things for the malware to allow it to run to its full extent proved to be the more difficult problem. Neither Wannacry nor Cryptolocker is known to have mechanisms designed specifically for analysis and sandbox resistance. However, some of the known functionality of both samples was unable to be executed in the environments created. Despite being temporarily run on the same host only network where both sandbox machines were able to communicate with each other, Wannacry never infected the Windows 7 machine, though it contained the EternalBlue exploit used by Wannacry to spread. Cryptolocker's functionality did not reach the encryption or warning phase, as it was unable to retrieve the necessary encryption keys from its command-and-control server. These demonstrate future work that could be done for the sandboxing of ransomware, namely finding ways to emulate responses expected by ransomware from command-and-control servers. This problem might be more straightforward than it may seem, as most ransomware contacts these servers for the same or similar types of responses. Future work could be done towards the configuration of FakeNet-NG such that it could emulate these types of responses, or into the creation of software that could whitelist Internet communication in a sandbox such that ransomware in the sandboxes could only communicate to its command-and-control servers and not propagate through the network.

VII. Conclusion

Malware analysis and sandboxing is a complicated and risky topic, and with the rise of ransomware, becoming familiar with the specificities of sandboxing ransomware will become more and more important. There are two main considerations when creating these sandboxes. The first is creating an environment that prevents the ransomware from spreading outside of the protected environment and infecting undesired machines. The second is creating an environment that allows for the ransomware's full execution so that all of its behavior can be observed. As demonstrated in this work, the first of these turns out to be a simpler task than the second. This is primarily because of ransomware nearly always contacting command-and-control servers. As with the Cryptolocker ransomware analyzed in this work, other ransomware may be unable to execute unless contact to these servers is established. This issue arises because of sandbox environments' isolation from the Internet. Such isolation prevents the spread of the sample but also prevents direct connection to servers that might be necessary for execution. Finding ways to emulate these types of network connections or ways to safely communicate with these servers, without potentially infecting other machines, is likely the most important work for ransomware sandboxing in the future.

References

[1] A. O. Almashhadani, M. Kaiiali, S. Sezer and P. O'Kane, "A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware," in IEEE Access, vol. 7, pp. 47053-47067, 2019, doi: 10.1109/ACCESS.2019.2907485.

[2] Wira Zanoramy A. Zakaria, Mohd Faizal Abdollah, Othman Mohd, and Aswami Fadillah Mohd Ariffin. 2017. The Rise of Ransomware. In Proceedings of the 2017 International Conference on Software and e-Business (ICSEB 2017). Association for Computing Machinery, New York, NY, USA, 66–70. DOI:https://doi.org/10.1145/3178212.3178224

[3] Rakesh Singh Kunwar and Priyanka Sharma. 2016. Malware Analysis: Tools and Techniques. In Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies (ICTCS '16). Association for Computing Machinery, New York, NY, USA, Article 144, 1–4. DOI:https://doi.org/10.1145/2905055.2905361

[4] Miuyin Yong Wong, Matthew Landen, Manos Antonakakis, Douglas M. Blough, Elissa M. Redmiles, and Mustaque Ahamad. 2021. An Inside Look into the Practice of Malware Analysis. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21). Association for Computing Machinery, New York, NY, USA, 3053–3069. DOI:https://doi.org/10.1145/3460120.3484759

[5] Routa Moussaileb, Benjamin Bouget, Aurélien Palisse, Hélène Le Bouder, Nora Cuppens, and Jean-Louis Lanet. 2018. Ransomware's Early Mitigation Mechanisms. In Proceedings of the 13th International Conference on Availability, Reliability and Security (ARES 2018). Association for Computing Machinery, New York, NY, USA, Article 2, 1–10. DOI:https://doi.org/10.1145/3230833.3234691

[6] Q. Chen and R. A. Bridges, "Automated Behavioral Analysis of Malware: A Case Study of WannaCry Ransomware," 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017, pp. 454-460, doi: 10.1109/ICMLA.2017.0-119.

[7] Cornel Constantinescu and Sangeetha Seshadri. 2021. Sentinel: ransomware detection in file storage. In Proceedings of the 14th ACM International Conference on Systems and Storage(SYSTOR '21). Association for Computing Machinery, New York, NY, USA, Article 28, 1. DOI:https://doi.org/10.1145/3456727.3463834

[8] R. R. Sharma and S. Sharma, "A novel Approach to counter Ransomwares," 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2020, pp. 775-780, doi: 10.1109/Confluence47617.2020.9058190.

[9] R. Agrawal, J. W. Stokes, K. Selvaraj and M. Marinescu, "Attention in Recurrent Neural Networks for Ransomware Detection," ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 3222-3226, doi: 10.1109/ICASSP.2019.8682899.

[10] D. Gonzalez and T. Hayajneh, "Detection and prevention of crypto-ransomware," 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), 2017, pp. 472-478, doi: 10.1109/UEMCON.2017.8249052.

[11] T. Anderson "Google's VirusTotal reports that 95% of ransomware spotted targets Windows." Internet: https://www.theregister.com/2021/10/14/googles_virustotal_malware/, Oct. 14, 2021 [Mar. 7, 2022].

[12] VirtualBox. Internet: https://www.virtualbox.org [Oct. 21, 2021]

[13] Flare-VM. Internet: https://github.com/mandiant/flare-vm, Oct. 22, 2021 [Nov. 8, 2021]

[14] theZoo. Internet: https://github.com/ytisf/theZoo, Jan. 10, 2022 [Jan. 13, 2022]

[15] Virtual Networking. Internet: https://www.virtualbox.org/manual/ch06.html [Mar. 11, 2022]

[16] FakeNet-NG. Internet: https://github.com/mandiant/flare-fakenet-ng/blob/master/README.md, Nov. 11, 2020 [Dec. 3, 2021]

[17] Virustotal. Internet: https://www.virustotal.com/gui/home/upload [Jan. 17, 2022]

[18] "Ransomware Awareness for Holidays and Weekends." Internet: https://www.cisa.gov/uscert/ncas/alerts/aa21-243a, Aug. 31, 2021 [Mar. 10, 2022]

[19] MalwareBazaar. Internet: https://bazaar.abuse.ch, [Jan. 13, 2022]

[20] A. Morse "Investigation: WannaCry cyber attack and the NHS." Internet: https://www.nao.org.uk/report/investigation-wannacry-cyber-attack-and-the-nhs/, Oct. 27, 2017 [Jan. 20, 2022]

[21] "WHAT IS WANNACRY/WANACRYPT0R?" Internet: https://www.cisa.gov/uscert/sites/default/files/FactSheets/NCCIC%20ICS_FactSheet_WannaCry_Ransomware_S508C.pdf, [Apr. 3, 2022]

[22] "CryptoLocker Ransomware Infections" Internet: https://www.cisa.gov/uscert/ncas/alerts/TA13-309A, Nov. 5, 2016 [Apr. 3, 2022]