EPiC Series in Computing

Volume 58, 2019, Pages 127–136

Proceedings of 34th International Conference on Computers and Their Applications

Behaviour Based Ransomware Detection

Christopher J. W. Chew[1] and Vimal Kumar[1]

University of Waikato, Hamilton, New Zealand cc246@students.waikato.ac.nz, vimal.kumar@waikato.ac.nz

Abstract

Ransomware is an ever-increasing threat in the world of cyber security targeting vulnerable users and companies, but what is lacking is an easier way to group, and devise practical and easy solutions which every day users can utilise.

In this paper we look at the different characteristics of ransomware, and present preventative techniques to tackle these ransomware attacks. More specifically our techniques are based on ransomware behaviour as opposed to the signature based detection used by most anti-malware software. We further discuss the implementation of these techniques and their effectiveness. We have tested the techniques on four prominent ransomware strains, WannaCry, TeslaCrypt, Cerber and Petya. In this paper we discuss how our techniques dealt with these ransomware strains and the performance impact of these techniques.

1 Introduction

Malware is malicious software that aims to harm or compromise a user's system. According to Kaspersky Labs, in the second quarter of 2018 they detected and blocked 962,947,023 attacks launched from web resources located in 187 countries around the world, with the United States having one of the highest detection rates of malware at 45.87%. [8]. Malware has been and still is a significant threat to computing resources the world over. The threat of malware however, has evolved over time and in the past year or so Ransomware has emerged as the predominant form of malware. The WannaCry ransomware attack that started on May 12, 2017 had infected more than 300,000 computers in 150 countries within a day. [5]

Ransomware is a type of malware and is used by an attacker to infect a user's system with the end-goal of receiving ransom money from the user. There are many different types and variants of ransomware that exist today. Many of the more complex and devastating ones such as WannaCry, Locky, Petya, NotPetya etc. have gained worldwide notoriety through the media but there are many others that have flown under the radar but are still a threat to unsuspecting users and corporations.

The attackers employ a number of infection methods to infect unsuspecting user machines. Once infected, the ransomware will either lock or restrict the user's files using methods such as putting the user's files in password protected zip folder, or providing a permanent screen overlay which stops the user from interacting with anything else in the system. Ransomware that restrict user access to their files are known as the locker-type ransomware. Another type of ransomware are crypto-type ransomware. This type of ransomware will encrypt the user's files

through the use of an encryption scheme, rendering their valuable files no longer accessible. The encryption schemes can range from the usage of public key encryption, homebrew encryption, or the more common, hybrid encryption. Once the user's files have been encrypted or locked, the attacker will seek out a form of payment from the user. These days the ransom is predominantly sought out in crypto-currencies like Bitcoin due to the anonymous nature of transactions. The user will need to pay the demanded ransom if they wish to retrieve the files again which may not necessarily be the case in most situations.

From statistics acquired from Symantec's 2018 Internet Security Threat Report, there is an estimated rise of 46% in ransomware variants, targeting many users or businesses [4]. While businesses may have cyber-security experts working for them, this is not the case for home users who are often left defenceless.

While signature based detection techniques developed generally for malware detection have been used for ransomware detection by many anti-malware solutions, in this paper we explore behaviour based techniques for detecting ransomware. Our focus is on creating basic and easily-implementable solutions for day-to-day user to halt or completely mitigate the attack of ransomware through the use of behaviour based detection. These solutions are designed to aid the user in dealing with the increased threat, and giving the responsibility to the user for their own security.

2 Background and Related Work

In signature based malware detection, malware analysts detect patterns or sequence of bytes unique to the malware in its code. Such sequences are stored in a database and the during scanning, the anti-malware software tries to detect such patterns in executable files. Signature based malware detection techniques have traditionally been very popular because they have a low false positive ratio. However, they are unable to cope with obfuscated code in malware and cannot detect new strains before they have been studied by an analyst.

Behaviour based detection on the other hand is the notion of observing the characteristics of how a malware operates. In this type of detection the anti-malware looks for certain activities that may indicate the presence of a malicious software.

Most work in malware and specifically ransomware detection is focused on signature based methods however lately other techniques have also been used. Authors in [1] adopt an authentication based access control mechanism under the name of AntiBotics that has three components. The first component is the Policy Enforcement Driver which acts as an initial gate that records and halts any file modification attempts such as, renames or deletions. For the file modifications to be made, a challenge is created such as CAPTCHA or biometric authentication to authenticate the user's actions. The next component is the Policy Specification Interface, which is a GUI program that allows admins to configure the policies for the system. The last component is the Challenge Response Generator, this component looks at controlling the challenges which are generate, i.e. the time-out rate, and mechanisms to prevent large generations of challenges.

The Maltester system in [3] uses a HoneyPot system to monitor network traffic. The paper focused on the CryptoWall ransomware and the Maltester HoneyPot system was able to identify the sources of where the C&C servers were being hosted through connection logs.

In [6] two different implementations were created. The first one is Filesystem Activity Monitor which observes I/O activity using Windows Filesystem Minifilter Driver framework. The second implementation is targeted more towards locker-type ransomware which takes screenshots of the desktop. The screenshots are analysed against samples of ransom note that have been clustered into the respective ransomware family. Each cluster is then filtered of

words that may result in false positives to finally receive a list of words that is identified with each family of ransomware.

There are many behaviour based malware detection platforms such as Crowdroid [2] with a client server architecture. Such platforms collect data from from multiple devices and send it to a centralised server that analyses the data for malware behaviour and performs the detection. Our approach on the other hand is to perform the detection independently of any centralised server and create techniques that can be easily implemented and run on any machine.

3 Ransomware Behaviour

For our behaviour based ransomware detection, we targeted three Indicators of Compromise (IoCs). These IoCs are a result of the behaviour of a ransomware malware on a machine.

Our first IoC was **file changes**. From the ransomware samples we acquired we observed that, once a file has been encrypted, the ransomware will append an unknown file extension to the file which will mark it as encrypted. Thus an unknown file extension is an indicator of the presence of ransomware.

Our second IoC was **file entropy**. Entropy is a measure of the amount of randomness in a file. Many ransomware variants encrypt files residing on the system. Encryption changes the plaintext in a file to ciphertext that has high entropy.

Our third IoC was manipulation of **canary files**. Canary files or sparse files are essentially files that contains filler data that are of no relevance to the user. Because of this, these files can be placed around the user's system and act as a trap for ransomware. If one of the canary or sparse files were to be tampered by a malicious software, the user will be notified. This idea is simple and effective in most cases as it is not as resource intensive as monitoring an entire file system.

The above IoCs used individually may result in a high false positive ratio. As pointed out in [7], to simply rely on entropy alone to determine if a file has been encrypted would not be enough as there are other files such as Portable Executable (PE) format files that contain high levels of entropy. We therefore pair the measurement of entropy together with reading the magic bytes of a file. Magic bytes are essentially the starting bytes of a file that determine the file extension. With encrypted files, an unknown extension will be appended to the file to mark the file as encrypted. Therefore, if files are high in entropy and contain an unknown file extension, we can determine the encrypted files and lower the occurrence of false positives.

4 Disabling Methods

Before we discuss the implementation of the ransomware detection methods, we will go over the technique that uses those methods to disable ransomware. We use a simple counter that counts the number of file changes, or high entropy level files during a given interval. In our current implementation the interval has been set to 5 seconds. If the counter hits a predefined threshold within that interval then the following methods are used to take action.

- Use Cacls or ACL to restrict the user's file or folder access privileges
- Stop user processes to prevent any further destruction of the user's systems.

Our disabling methods are modular so they can be used with any of the detection methods.

Table 1: The general design structure of the implementations

Implementations	**Behaviour**	**Disabling methods**
File Monitoring	Canary files & fake drives	Restrict access using Cacl
File Entropy	Entropy and file extension	Terminate user processes
ACL Authentication	File changes	Revoke ACL writing privileges

5 Implementations

Based on the IoCs discussed in section 3, we came up with three detection implementations. These are described in detail in this section.

5.1 File Monitoring

The file monitoring system combined the **file changes** and the **canary files** IoCs. The implementation consisted of a setup phase and a detection phase.

A simple implementation of inserting canary files in the actual physical system drive will theoretically detect ransomware as user processes generally do not edit canary files while ransomware will encrypt them. The success of this implementation however will depend on where the canary files are placed in the drive and the order in which ransomware starts to access files. With a random file access, there is a high probability that half of the files will already be encrypted by the time a canary file is accessed and something suspicious is detected.

In our implementation, in the setup phase, two artificial network drives were created. The artificial drives were inserted before and after the letter of the real drives to ensure a lower chance of the ransomware encrypting the user's real drive. Once the drives were created, canary files within the artificial network drives were inserted. It was found that the more sophisticated ransomware such as Cerber or WannaCry have mechanisms that are able to detect canary files. We suspect that this is determined by the size of the file, or if the files contain the same hash signature but this needs more investigation. We created the canary files from user's actual files and made them to be larger than 1KB to bypass any ransomware mechanism that detects them. The artificial drives were then populated by only these files. This approach has two benefits, first, when a ransomware is detected it will only have encrypted canary files and not actual user files and second, the implementation would not be as resource intensive as opposed to monitoring individual canary files that are placed within the user's system.

In the detection phase we created and registered two events. The first event was a timer event, which was used as a periodic timer for counting the number of file renames in a five second interval. The idea is that a ransomware will continuously rename files as it encrypts them one by one. The 5 second interval was chosen to lower the risk of false detection, as some legitimate programs such as games also often alter file names. The second event we created was through the use of FileSystemWatcher filter in Windows, which is used for monitoring file operations. The FileSystemWatcher filter monitors the two artificial network drives. If a file rename is detected within those drives, a counter is incremented once to indicate to the user that a file has been changed. When three file changes occur within the time frame the disabling phase is triggered.

As the disabling methods are modular and can be implemented with the other two scripts, we mainly used the method of Cacls, which is a command-line utility developed by Microsoft to alter Access Control Levels of files and folders.

5.2 File Entropy

File Entropy as mentioned previously is the measure of randomness in a file. For this implementation we used the tool described in [7]. It uses the Shannon Entropy algorithm which is one of the more efficient and accurate ways of measuring entropy levels. The Shannon Entropy algorithm returns a number on the scale of 1-8 for a file, where 1 is low entropy and 8 is high. Text and alphanumeric data generally has low entropy while encrypted data which has a high degree of randomness has high entropy. We tested the Shannon Entropy algorithm on a collection of files. Some files in our collection were ciphertexts from various encryption schemes, some were outputs of cryptographic hash and some were ordinary files found on a system. The result of our testing is shown in tables 2 and 3.

Table 2: Entropy level of ciphertexts and hashes

Cryptographic operation	**Entropy level**
RSA 4096	6.01
AES 256	5.99
DES	5.92
SHA-1	3.76
MD5	3.59
XOR	3.15

Table 3: Entropy level of other files

File type	**Entropy level**
dll	6.075
zip	7.913
exe	4.787
ps1	4.705
txt	1.837

From table 2 we deduced that a threshold of 5.5 was a safe number when it came to detecting encrypted files based on entropy. However table 3 shows that certain files such as the zip and dll files owing to their structure also have high entropy and with our threshold may be classified as encrypted files. To counter this, along with entropy we also checked the magic bytes in the files for determining the file type. An encrypted file will have high entropy with the magic bytes encrypted as well, while a legitimate file such as a *dll* or *zip* will have high entropy with readable magic bytes. To accomplish this the SigCheck utility from Microsoft was used.

Similar to file monitoring, for disabling, a timer and counter were used to count the number of suspicious files. Following a file operation if a file contained high entropy and contained an unreadable extension, that specific file would be marked as a suspect file. Once two suspect files were found in quick succession, the disabling process began.

5.3 ACL Authentication

We also implemented an authentication based method similar to the one described in [1]. This implementation is not exactly a behaviour based detection method. In this implementation the users were asked to determine which directory they wanted to monitor for file change operations. The file operations included file additions, file renames, and file removals.

A file change in the monitored directory triggers User Access Control and uses ACL to restrict any further file change within the user's system. When the user identifies and confirms the file change, the script releases all locked directories and stops until rescheduled to start again. This also gives the user the freedom to specify how frequent the this method should run.

6 Performance Evaluation

The evaluation of the techniques was done in two phases. First we evaluated the detection performance of our techniques against some well known ransomware samples. Second we measured the resource intensiveness of the techniques on a system.

For the implementation we created a sandbox environment by using virtual machines running on Virtual Box. The operating system used for all the testing was Windows 8.1. The sandbox environment was isolated by disconnecting it from any network and internet. The techniques were implemented through scripts running on Powershell so that Windows system internals could be accessed.

6.1 Ransomware Detection Performance

We tested our techniques on four ransomware samples, WannaCry, Cerber, Petya and TeslaCrypt.

Our measure of success will whether our implementations were able to detect the the presence of ransomware based on their behaviours and whether we could prevent or halt the attacks.

6.1.1 WannaCry

The first test was against the infamous WannaCry ransomware. With the file monitoring implementation, it appeared that all the canary files that formed the artificial drives were encrypted but the implementation was eventually able to restrict file and folder access. During the testing, WannaCry posed as a problematic ransomware as some of the actual files in the course of our runs were encrypted. This was due to the disabling part taking time to kick in.

With the file entropy implementation we were able to acquire satisfactory results as the setup files that were created by WannaCry had high levels of entropy which caused our script to trigger instantaneously, and we were able to kill the process before any files were encrypted.

With the ACL Authentication implementation, the ACL permissions were revoked instantly from the user thus preventing any encryption to occur, as WannaCry required a few setup files before the encryption process began. Since WannaCry could not access any folders the administrator user did not have access to, it was not able to encrypt any files. Through five different tests done with the ACL Authentication script against WannaCry, we were able to always prevent the attack of the ransomware.

6.1.2 TeslaCrypt

File monitoring returned a much more satisfactory result on Teslacrypt and only the trap files were encrypted from the artificial network drives. Although, similar to the previous iterations, the user's desktop background was altered and the ransom note appeared. However, no casualties of encrypted files were encountered throughout the testing period.

With the file entropy implementation, we managed to successfully prevent the attack. No files were encrypted and due to the disabling method used, where user processes were

terminated, the ransomware was prevented from changing the background of the user's system and displaying the ransomware note.

The ACL Authentication implementation yielded some interesting results. Although we were able to prevent any of the files from being encrypted by blocking ACL permissions, the ransomware was still able to change the user's background and displayed the ransom note. Once ACL permissions were granted back, the files were still unencrypted and readable. However, this ransomware was persistent on start-up, therefore if the user was to restart their computer while the ransomware was active, their files would be encrypted. This is due to a modification to the start-up registry key.

6.1.3 Cerber

While testing the file monitoring program against Cerber we found that it ignored files that were less than a certain size (2KB). Hence, the file monitoring program was unable to detect this ransomware initially. As a remedy, we inserted canary files of varying sizes into the artificial drives which gave us successful results in the detection phase. However, we were unable to attain any satisfactory results with this implementation at the disabling phase, as the user files were still being encrypted.

The file entropy implementation was successful against Cerber. The setup files had enough entropy to be detected and the process was terminated before any files were encrypted.

The ACL implementation too was able to detect the creation of the setup files when Cerber was executed and was able to quickly lock up the valuable files and folders in time before the ransomware was able to encrypt anything.

6.1.4 Petya

Petya is a ransomware that is difficult to detect and prevent via the use of behaviour based detection. The Petya ransomware's behaviour is a combination of a virus's and a ransomware's behaviour. Petya causes the user's system to trigger a Bluescreen of Death (BoD) stopping any process at that point and forcibly shutting the user's system down. Once the system has been shut down and restarted the ransomware encrypts the Master Boot Record. At this stage, the process of detecting the ransomware is too late, thus our implementations were unable to achieve any satisfactory results.

6.2 Runtime Performance

In this section we will discuss runtime performance of our implementations. We will first discuss the Central Processing Unit (CPU) and Memory usage of the implementations after which, each implementation will be evaluated individually discussed. Each method was tested by running it for WannaCry 5 times. In each of the below line charts the numbers show the worst and the best case. The x axis labelled as 'Runs' represents each iteration, an iteration or run is determined when the prominent method(s) of interest is completed. For example, the File Monitoring implementation, a run is completed when the network drive creation has finished.

6.2.1 CPU & Memory Utilization

Figure 1 shows the CPU usage of the three implementations and figure 2 shows the memory usage across the 5 runs.

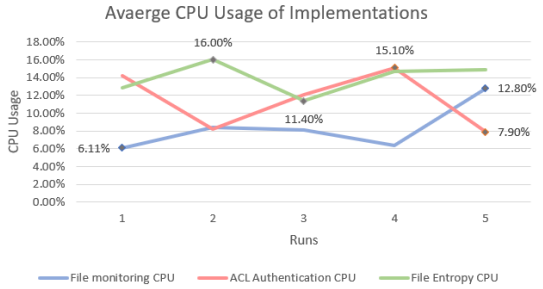

Figure 1: CPU Averages of implementations

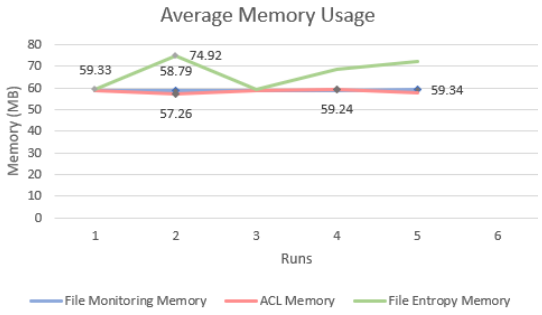

Figure 2: Memory Averages of implementations

From the two figures above we can see that the file monitoring implementation is on average more efficient in terms of both CPU and memory usage. File entropy turns out to be the most resource intensive of the three on both the parameters. The ACL authentication method had a low memory usage performance and an average CPU usage performance.

6.2.2 File Monitoring Performance

We further analyzed the three different implementation on various Components of Interest (CoI)s that provided us insights into them. The CoI for the file monitoring implementation was the creation of artificial network drives.

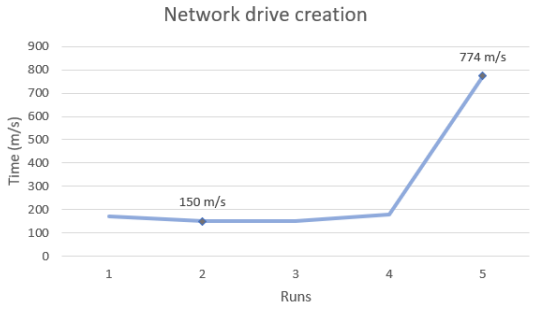

Figure 3: CoI for File Monitoring implementation

One of the major components of the file monitoring implementation was the artificial network drive creation. We observed that this CoI was quite fast on average but there is a spike in the last run as seen in Figure 3. The spike does not affect the memory usage but does affect the CPU usage as shown in Figure 1. Our analysis suggests this is a random spike as nothing was changed in the environment and thus can be ignored as an outlier.

6.2.3 File Entropy Performance

For testing the file entropy implementation, our CoIs were the execution time of determining the file entropy of a detected file, how long it took to check the file header of a file, and the execution time of terminating the user processes.

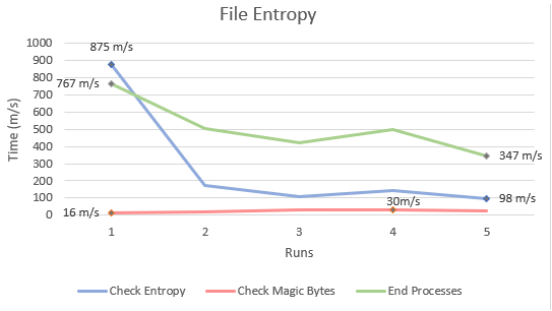

Figure 4: The results of prominent methods for File Entropy implementation

From Figures 1 and 2, we can see that the file entropy implementation was the most resource intensive out of the three. This is because there was significant amount of computation involved in checking for entropy and checking for the file's magic bytes as well as terminating user processes. From Figure 4 we can see that checking the entropy and terminating a process make up a large part of the total execution time.

6.2.4 ACL Authentication Performance

For the ACL authentication our CoIs were the execution time of the registration of the monitoring events, the time in which it took to revoke the ACL permissions after detecting a file change, and the time it took to grant the permissions back to user.

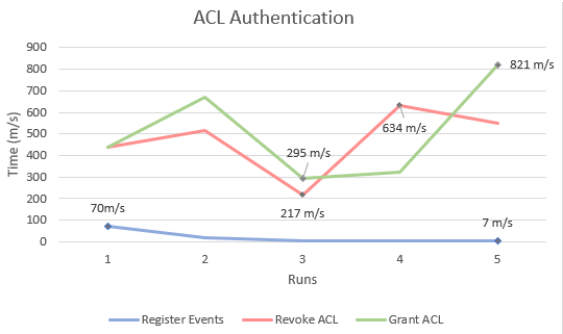

Figure 5: Prominent methods for ACL implementation

Observing the results from Figure 5, we see that while registering the events takes an almost constant amount of time the revoking and granting of ACL are fluctuating and this means are affected by the environment.

The two CoIs also required significant amount of time and of CPU resources which resulted in the ACL authentication implementation overall requiring considerable resources as shown in Figures 1 and 2. However, we do not deem that as a problem as the most significant figure was the average execution time of revoking ACL permissions since that was the deterrent that halted the ransomware attack, which was still less than a second to execute and complete.

7 Conclusion and Future Work

The threat of ransomware is ever increasing and evolving as attackers continuously create new and unique strains. Signature based detection is unable to keep up with this pace of ransomware development. In this paper we showed that behaviour based approaches can be the way forward. We implemented some simple detection methods and tested them against some well known ransomware samples. Our preliminary investigation has shown promise and we can see that their implementation is not very resource intensive as signature based detection methods generally are.

However there are many questions that are unanswered that will need to be addressed in future, such as how will sophisticated strains such as Petya be handled? Can the monitoring be done on physical drives instead of artificial drives without putting a lot of strain on the system? What are some other ransomware behaviour that can be used for detection etc. Moreover, while our preliminary testing has shown promise, the implementations still need to be tested rigorously and on a variety of samples.

References

[1] Or Ami, Yuval Elovici, and Danny Hendler. Ransomware prevention using application authentication-based file access control. 2018.

[2] Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, pages 15–26. ACM, 2011.

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

[4] Orla Cox Hon Lau Benjamin Nahorney Dick O'Brien Brigid O'Gorman John-Paul Power Scott Wallace Paul Wood Candid Wueest Gillian Cleary, Mayee Corpin. Internet security threat report. Technical Report 23, Symantec Corporation, 2018.

[5] Chris Graham. Nhs cyber attack: Everything you need to know about 'biggest ransomware' offensive in history, 2017, May 20.

[6] Amin Kharraz, Sajjad Arshad, Collin Mulliner, William K Robertson, and Engin Kirda. Unveil: A large-scale, automated approach to detecting ransomware. In *USENIX Security Symposium*, pages 757–772, 2016.

[7] R VandenBrink. Using file entropy to identify "ransomware" files, 2016, August 16.

[8] Denis Parinov Alexander Liskin Oleg Kupreev Victor Chebyshev, Fedor Sinitsyn. It threat evolution q2 2018. statistics, 2018, August 6.