# CryptoDetect: Ransomware Defense

Bharadwaj Thirumal
*University of Colorado Boulder*
Bharadwaj.Thirumal@colorado.edu

Kannan Subramanian
*University of Colorado Boulder*
Kannan.Subramanian@colorado.edu

## Abstract

Ransomware is a type of malware that encrypts files in a system and demands ransom from the owner of the system. In this paper, we present a tool that backs up important files and recovers them if the files are encrypted by the ransomware. We create a service that detects ransomware by monitoring changes to the files and shuts down the computer. We also present a consolidated view of the evolution of ransomware, existing defenses, other methods we tried to implement to counter ransomware, CryptoDetect tool, limitations of our method, related work and future work.

## 1. Overview

The term malware encompasses a wide variety of unnecessary programs that plague the computer system. This include viruses, Trojan horses, worms, adware, spyware, backdoors, and ransomware. Microsoft Windows is a common target for most malware authors since it is a popular and widely used operating system. Malware were used to steal information, cause interruption in the normal operation of the system (DoS), taking control of the systems for illegal/criminal purposes and now to demand money. The working mechanism of ransomware is simple. When a person opens a suspicious attachment from an email, ransomware is downloaded into the system. It encrypts files with certain extensions throughout the entire system using the public key it gets by contacting its Command and Control Server. It then displays the ransom note. The private key is known only to the attacker and is not stored locally in the infected system. We will go through a brief overview of the history of ransomware to understand the evolution of ransomware.

## 2. History of Ransomware

The first known ransomware was AIDS Trojan which infected systems in 1989 by encrypting files and demanding money for decryption. Since it used symmetric key encryption, the decryption key could be easily found by reverse engineering the code. In 1996, Young and Yung came up with the idea of using asymmetric key cryptography for *cryptoviral extortion.* This concept began to take shape during 2005 and 2006 using RSA encryption schemes.

Ransomware made a comeback with Reveton in 2012 which displayed a fake warning message from a law enforcement agency. This was followed by CryptoLocker in 2013 which was the first ransomware to demand ransom in the form of Bitcoin. This collected $27 million in 3 months by encrypting certain files (choosing from a white list of file extensions) using 2048-bit RSA key. This technique was used by CryptoLocker 2.0 (not the successor of CryptoLocker). Another ransomware TorrentLocker made the mistake of using the same keystream for all the systems and so was easily overcome. But it extracted email IDs from the infected systems to send copies of itself to those addresses. Cryptowall appeared in 2014, used attachments from email to execute a JavaScript and download the ransomware. It then latched itself to admin processes like service host and deleted the volume shadow copies, disabled system restore. Moreover, it deleted the backup copies without alerting the user. Cryptowall 4.0, the latest version encrypts both the file contents and the file name.

Other ransomware like CTB-Locker, uses the combination of elliptic curve cryptography, Tor and Bitcoin. It uses networked peers to outsource the infection process. TeslaCrypt is available for purchase from the dark web. Cerber, found recently, is ransomware as a service.

## 3. Existing Solutions

The solutions that currently exist are very primitive and focus on not letting the ransomware infect the system. The solutions are explained in this section and some of them are quite obvious.

**Paying the ransom.** This is not exactly a defensive solution. Ransom could be paid as a last resort to get the data back. But, there are some cases where the ransom was paid but the decryption key was not given back. So, this method is not recommended. Initially, the ransom was demanded in Dollars but the latest ransomware demand the payment in Bitcoin using the Tor network.

**Don't open attachments from suspicious emails.** Most ransomware infections occur as a result of users opening malicious email attachments. Malicious JavaScript is disguised as pdf, word documents, which when

clicked, install ransomware into the system. The attachments can be opened in a sandbox and tested but doing this every time a file is downloaded is not practical. However, there are certain variants that detect that they are being run in a sandbox and exit.

**Don't use Microsoft Windows.** As mentioned before, Microsoft Windows has been the primary target for malware authors especially ransomware authors. One of the solutions is to not use Windows but this is not viable as it is widely used and most of the programs are developed for Windows. Fragmentation is also an issue. However, during our research we found that Windows is open to attack by ransomware but really hard to defend as the kernel is closed for modification and loadable kernel module is restricted. Ransomware injects itself into service host and explorer.exe processes that have administrative privileges by using DLL. Windows doesn't sign these processes, hence there is no way to authenticate a process. Because of these issues and the popularity of windows, most ransomware have been designed for windows.

Linux is open source and has lesser number of bugs. The first ever OS X ransomware infection discovered recently was the result of a compromised Mac development certificate. All applications in Mac are signed by Apple and it is improbable for an unsigned process to install and run with administrative privileges.

**Create a network backup.** Since ransomware infect the files on the hard disk, backups can be created and stored in a network location so that the ransomware cannot reach them.

**CryptoPrevent [2] software.** CryptoPrevent is a tool developed to prevent ransomware infection. It uses group policies to restrict certain kind of software installing and running from specific locations by enforcing these policies using windows registry. This is not a foolproof approach.

**Anti-virus.** Most of the anti-virus software detect and block malware using their signature. This would detect malware whose signature is known. The malware authors could change the signature of their product and the anti-virus cannot detect them.

All of these solutions are designed to be proactive. They try to prevent the entry of malware into the system. But these won't be able to help if the system is infected with the ransomware.

## 4. Initial Ideas

We tried to build a tool that prevents the ransomware from infecting the system by analyzing attachments in a sandbox. Since there are a lot of infection vectors, it is difficult to prevent all kinds of infection vectors.

Detecting a ransomware by examining its opcode [3] was the next idea in our list. We soon realized that the code is obfuscated and it is difficult to automate such a process. Signature based detection is infeasible. Behavioral detection is interesting but some files could be encrypted by the time our tool detects the ransomware and shuts down the system. Therefore, some sort of file backup mechanism is important to recover the lost files.

## 5. Volume Shadow Copy [4]

Volume Shadow Copy is Microsoft's way of creating snapshots of volumes. But most ransomware delete the volume shadow copy by taking over the vssadmin process which is the admin process used to manage the volume shadow copies. The ransomware deletes the backup without alerting the user. It uses the switch **\quite** so that the user is not notified. There is no option to disable this switch.

VSS (Volume Shadow Service) provides APIs [5] that can be used to create and restore backups. But there are no APIs to read the volume shadow copy. It backups an entire volume instead of specific files. This has a huge storage overhead. This is the reason that we had to move away from the idea of using VSS. We would like to point out that this is one of reasons we state to not using Windows – the malware authors can easily delete the VS copies but a developer cannot defend against it.

## 6. ELAM: Protected Process

When a service is launched as a protected service, other non-protected services will not be able to inject threads into this process. They won't be able to write to the virtual memory of the protected service. This doesn't prevent the injection of legitimate DLLs but they must be signed with an appropriate certificate. [6]

This is where ELAM comes into picture. ELAM (Early Launch Anti-Malware) driver is a driver that Microsoft came up with. From Windows 8, Microsoft has added a new feature called secure boo that will load the ELAM driver before any others. This Anti-Malware that gets launched early, evaluates other drivers to determine if they should be allowed to launch. Microsoft's Developer Network states that the ELAM drivers have to be specially signed by Microsoft so that they can be started early in the boot cycle. But Microsoft will sign the driver only if it passes the tests included in its Windows Hardware Certification Kit (Windows HCK).

In addition to the getting their driver signed by Microsoft, the anti-malware vendor must include within

their driver, a resource section that contains the information of the certificates used to sign the user mode service binaries. During the boot process, this resource section is extracted from the ELAM driver and used to validate the certificate and register the anti-malware certificate.

Our tool will be effective only when it is signed by Microsoft. This will not require much change in our code – we just need to install our signed ELAM driver and couple it with the service that we wrote.

## 7. File History [7]

File History is a feature introduced in windows 8 that backups files periodically. File History API can be used to create, configure and restore backup. [8] It also provides API to open a communication channel to the File History Service and read the backup data. There are three problems with this approach. First, it will backup files stored only in libraries, desktop, favorites and contacts folders. Moreover, the API provides a method to only exclude the files and folders included in the backup and does not provide any way to add inclusion rules. Hence, important files in other folders cannot be backed up. An unwieldy workaround is to add all important files to a user library. Second, File History will store the backup either in an external drive or a network drive. In standalone computers, these may not be available. A hacky way is to share a local folder and trick the FH to accept it as a network drive. Third, over time, FH builds a complete history of changes made to each file. This will have a higher storage overhead compared to just storing the latest version.

## 8. Our approach

### Backup and Restore Tool

We created a WPF Form using Visual Studio that provides an UI to create and restore backups. When this process runs for the first time, the user has to specify the files to be protected. This service will perform the following actions:

1) Copy these files into a folder.

2) Zip the folder.

3) Split the zip file into random sized parts.

4) Hide the parts in random folders.

5) Change file creation time of each part to random date-time.

If the zip file is split into equal sized parts, the ransomware can perform the following attack:

**A.** Estimate the lower bound of the zip file size (containing files that were accessed recently).

**B.** Estimate the upper bound of the zip file size (containing all files*).

**C.** Estimate the number of parts**.

**D.** Find the size of each part. (zip file size divided by number of parts)

**E.** Search files in all the drives whose size is roughly equal to estimated value from 4.

Steps A to E might have to be run multiple number of times to accurately find the parts. Step 3 will make this attack harder. Changing the file creation date-time of the parts is important, because otherwise the ransomware can search for files created within a short date-time range. These parts either have: A) no extension or B) exe or any other file extension that the ransomware excludes from encrypting. The rationale behind hiding the parts in random folders is that the ransomware has to spend more time in searching for the parts. Shamir's secret sharing algorithm [9] can be used to generate k-of-N parts of the file.[10] Any K parts are sufficient to reconstruct the original file. This tool can be scheduled to run daily.

It also provides a restore feature to restore the files from the backup parts.

### Ransomware Detection Service (Protected Process)

This protected process maintains a lock on all the parts so that they can't be deleted. This windows service always runs in the background. It is configured to run in safe mode, auto run after system startup. It registers a handler function to monitor any changes (modification or deletion) to the important files. If any one of these files is modified, then an event is raised and the handler function will be called. The handler function will shut down the system to prevent encryption of other files. The time taken for the shutdown is the delay between the event generation (modification of a file) and the handler function invocation. Based on our experiments, this is instantaneous. The ransomware was able to encrypt, on average, two files before the shutdown. More testing has to be done in cases where a ransomware spawns multiple processes to encrypt the files.

In order to avoid false positives, for example, preventing the system shut down when a user modifies a file, a threshold can be defined. If the threshold is 5%, this service will initiate the shut down if 5% of the total number of files are modified in short time.

* the file extensions that ransomware targets

** the number of parts cannot be truly random because we don't want to divide a small file into huge number of very small sized parts. Dividing a huge file into small number of huge sized parts is undesirable as well. This algorithm can be reverse engineered by the ransomware authors and the number of parts could then be estimated.

## 9. Making CryptoDetect Robust

### Code Obfuscation

To deter reverse engineering and obscure the code, code obfuscation tools can be used to hinder ransomware authors. These tools can strip debug symbols, obfuscate control flow and provide method call redirection. [11]

### Self-Replication

To obscure the CryptoDetect process from detection by the ransomware, CryptoDetect can be configured to create a copy of itself in a random directory, run the copy and terminate itself. Self-Replication is possible by using the Reflection technique in C#. [12]

### Self-Mutation

Ransomware can try to detect CryptoDetect using signatures. This could be made difficult by designing CryptoDetect to be polymorphic. This could be challenging to implement.

### Defeat Persistence of Ransomware

Ransomware adds itself to the startup folder, configure the system to execute itself in safe mode. It achieves this by adding entries in the windows registry. The protected service can be configured to monitor the windows registry to detect this behavior and delete the malicious registry entry.

## 10. Future Work

Rather than detecting the ransomware process behavior and shutting down the system, a better approach would be identifying and terminating the ransomware process. But, the keep alive process trick [13] makes this difficult to implement.

Ransomware, typically, generates a symmetric key and uses it to encrypt the files. This symmetric key is then encrypted by the public key of a C2 server and then sent to the C2 server. If a ransomware process is encrypting the files, the symmetric key must be stored somewhere in the memory. Instead of shut down, a full memory dump can be triggered. Some post mortem tool could be used to analyze the memory dump and retrieve the key.

Further research is required to analyze how machine learning techniques can be used to detect and stop ransomware infections.

## 11. Conclusion

Behavioral based detection of ransomware is possible by monitoring modifications to important files. If the number of files modified is above a threshold, the system is shut down. Backup of the important files is created, zipped and split into random sized parts and hidden inside random folders. This backup can be used to restore files that are encrypted by the ransomware before it is detected by CryptoDetect.

Different techniques like code obfuscation, self-replication and self-mutation can be used to make the defense more robust. This defense is not foolproof because of bugs in the implementation of protected services and other bugs in windows.

A foolproof solution involves multiple steps. First, Microsoft should sign system processes like explorer.exe and svchost.exe with administrative privileges to prevent tampering and injection of malicious code. Second, critical operations like deleting a volume shadow copy should alert the user and require him to act before they are run. Third, Microsoft should allow loadable kernel modules. These can be used as hooks into critical events in windows that are otherwise not exposed to user space applications. They can follow Apple and issue Windows Application developer certificates, only allow the signed applications to use loadable kernel modules.

## 12. References


[1] Wikipedia: Ransomware

https://en.wikipedia.org/wiki/Ransomware

[2] Foolish IT: CryptoPrevent

https://www.foolishit.com/cryptoprevent-malware-prevention/

[3] McAfee Security Blog

https://blogs.mcafee.com/mcafee-labs/locky-ransomware-rampage-javascript-downloader/

https://kc.mcafee.com/resources/sites/MCAFEE/content/live/PRODUCT_DOCUMENTATION/25000/PD25203/en_US/Ransomware_Update_RevI.pdf

https://kc.mcafee.com/resources/sites/MCAFEE/content/live/PRODUCT_DOCUMENTATION/26000/PD26383/en_US/McAfee_Labs_Threat_Advisory-Ransomware-Locky.pdf

http://www.mcafee.com/us/resources/solution-briefs/sb-quarterly-threat-q1-2015-2.pdf

http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2015.pdf

[4] MSDN: Volume Shadow Copy Service

https://msdn.microsoft.com/en-us/library/windows/desktop/bb968832(v=vs.85).aspx

[5] MSDN: Volume Shadow Copy API Reference

https://msdn.microsoft.com/en-us/library/windows/desktop/aa384648(v=vs.85).aspx

[6] MSDN: Protecting Anti-Malware Services

https://msdn.microsoft.com/en-us/library/windows/desktop/dn313124(v=vs.85).aspx

https://msdn.microsoft.com/en-us/library/windows/hardware/dn265159(v=vs.85).aspx

[7] TechNet: Windows 8 – File History Explained

https://technet.microsoft.com/en-us/magazine/dn448546.aspx

[8] Windows Blog: A New Way to Backup: File History

https://blogs.windows.com/windowsexperience/2012/12/20/a-new-way-to-backup-file-history-in-windows-8/

[9] Wikipedia: Shamir's Secret Sharing

https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing

[10] Ubuntu Manpages: gfshare

http://manpages.ubuntu.com/manpages/xenial/en/man7/gfshare.7.html

[11] Wikipedia: List of obfuscators for .NET

https://en.wikipedia.org/wiki/List_of_obfuscators_for_.NET

[12] Wikipedia: Reflection (computer programming)

https://en.wikipedia.org/wiki/Reflection_(computer_programming)

[13] Keep alive processes or preventing app termination with immediate restart

http://www.codeproject.com/Articles/672843/Keep-alive-processes-or-preventing-app-termination