Signature-less ransomware detection and mitigation

Article *in* Journal of Computer Virology and Hacking Techniques · December 2021

DOI: 10.1007/s11416-021-00384-0

CITATIONS
5

READS
340

5 authors, including:

Harsh Mahajan

1 PUBLICATION **5** CITATIONS

SEE PROFILE

Aarti Amod Agarkar

Sinhgad college of engineering, pune, india

11 PUBLICATIONS **59** CITATIONS

SEE PROFILE

Journal of Computer Virology and Hacking Techniques
https://doi.org/10.1007/s11416-021-00384-0

ORIGINAL PAPER

Signature-less ransomware detection and mitigation

Yash Shashikant Joshi[1] · Harsh Mahajan[1] · Sumedh Nitin Joshi[2] · Kshitij Pradeep Gupta[1] · Aarti Amod Agarkar[3]

Received: 19 August 2020 / Accepted: 12 April 2021

Abstract
Ransomware is a challenging threat that encrypts a user's files until some ransom is paid by the victim. This type of malware is a profitable business for attackers, generating millions of dollars annually. Several approaches based on signature matching have been proposed to detect ransomware intrusions but they fail to detect ransomware whose signature is unknown. We try to detect ransomware's behaviour with the help of a mini-filter driver using a signature-less detection method. The proposed technique combines the working of Shannon's entropy and fuzzy hash to provide better results in detecting ransomware. Not only this technique has been practically tested but has been successful in detecting over 95% of the tested ransomware attacks on windows operating systems.

Keywords Ransomware · Cyber-security · Protection mechanism · Data security · Malware detection

1 Introduction

Ransomware is a kind of malware that threatens to block access to user's data unless and until a ransom is paid. For as long as computers will be in existence, whether connected to the internet or not, there is always a risk of any ransomware attack. Bitcoin is demanded as the usual form of ransom payment so that attackers are not exposed [1]. The most common way of a ransomware attack is through phishing emails [2]. Ransomware attacks have increasingly widespread and are causing tens of millions of dollars in consumer loss annually [3]. This malware has been a prominent threat to enterprises and individuals which has created a need for developing a solution against its attacks.

In [4], authors explained the working of ransomware in the case of Windows as, "there are some main stages that every crypto family follows. Each variant enters the victim's machine via some malicious website, email attachment, or any malicious link and progress from there. The malware deletes all their store points, backup folders, and shadow volume copies." The ransomware either encrypts the user files or blocks user access to the system making the files inaccessible to the user.

Signature matching is a technique that detects ransomware based on some known features of malware. Early signature detection systems use several different features to detect ransomware attacks [5, 6]. However, the biggest drawback of signature matching is that it is difficult to detect ransomware whose signature is not known. Furthermore, the new ransomware is designed with different encryption methods to counter static signature matching techniques [7]. A good anti-ransomware program should be able to protect a computer even from ransomware whose signature is not known. The proposed technique detects ransomware by a signature-less method.

In this paper, the proposed technique combines the working of Shannon's entropy and fuzzy hash to detect ransomware before it could cause harm to the system and try to mitigate its effect by a signature-less method. The prevention mechanism starts running in the background as soon as the

✉ Sumedh Nitin Joshi
joshisumedh13@gmail.com

Yash Shashikant Joshi
yashjoshi009@gmail.com

Harsh Mahajan
harsh321mahajan@gmail.com

Kshitij Pradeep Gupta
kshitijgupta139@gmail.com

Aarti Amod Agarkar
pratibha26@gmail.com

[1] Accenture Solutions Pvt Ltd., Pune 411057, India

[2] ByteSeq Security, Pune 411046, India

[3] Marathwada Mitra Mandal's College of Engineering, Pune 411052, India

Published online: 04 May 2021

system boots. The proposed technique had been practically tested against ransomware samples on different operating systems and has achieved a success rate of over 95% in detecting the tested ransomware samples.

The paper is structured as follows: Sect. 2 discusses the work related to the detection and prevention of ransomware attacks using various techniques. Section 3 specifies and explains the proposed technique which uses Shannon's entropy and Fuzzy hash to detect ransomware by a signatureless method. Section 4 includes the trials performed on the proposed technique to detect ransomware samples and analyses its efficiency in preventing ransomware attacks. Section 5 concludes the proposed technique.

2 Related work

Dekel et al. [8] and C. Moore [9] proposed the methods of using honeypots for ransomware detection. A possible solution to detect ransomware attacks can be the use of honeypots [10]. Honeypot is nothing but a decoy computer system that helps to trap hackers and also newly emerging hacking methods. Honeypots are designed to purposely identify malicious activities performed over the system. But there are also some limitations with honeypots. Mohammed and Rehman [11] stated the limitation of honeypots. Honeypots will be unaware of the ransomware attack unless it is directly attacked. If the attacker manages to identify honeypot files, the attacker can now avoid that honeypot and penetrate the system.

Gonzalez and Hayajneh [12] discussed a method to detect an application's behaviour using API call monitoring, without actually implementing the method. The proposed method is predicted to help in detecting suspicious sequences of Windows API calls. The author also proposed close monitoring of MFT (Master File Table) to detect a compelling amount of deletion, creation, and encryption of the files.

In [13], authors developed CryptoDrop that detects ransomware based on monitoring the real-time change of user data. CryptoDrop alerts the user whenever ransomware suspicious behaviour is detected. However, it fails to understand the intent of the changes which have been detected, resulting in the blocking of some of the user performed activities. Hosfelt [14] proposed a machine learning-based method to detect crypto-ransomware attacks. Different machine learning models were tested to detect ransomware samples. Though the models provided a detection rate of above 95% for small number of samples, it performed poorly with a larger dataset of ransomware samples due to complexities in machine learning models.

Use of a file system filter driver for detection of ransomware is shown in [15]. According to authors," We recognize two different patterns: Default Programs request access to relative files against nondefault Programs. Taking such a simplified approach, it is possible to build a bullet-proof system defence." However, the main limitations of the system included the likelihood of high false positives.

The proposed technique is based on I/O request interception. The behaviour of the I/O requests will be analysed with the help of a mini-filter driver. Whenever a process asks for write permission on the file the mini-filter driver will calculate the fuzzy hash and Shannon's entropy of the file after the process has completed its operation and based on that a process will be marked suspicious or genuine.

3 Proposed technique

3.1 Indicators used for detecting ransomware

3.1.1 Shannon's entropy

Entropy refers to the randomness of the data in the file [16]. The bytes in a regular text file are not evenly distributed because of which their entropy is low. Files that are encrypted or compressed have a high entropy value since each byte in ciphertext should have a uniform probability of occurring [13]. Since ransomware encrypts files, this method can be used to detect whether the file is encrypted or not. Shannon's Entropy of file is computed using the following equation:

$$H(X) = -\sum_{i=0}^{N-1} P_i \log_2 P_i \tag{1}$$

where, (1) P_i is the probability of given symbol and N is the size in bytes.

The probability of occurrence is computed per byte. Hence, a single byte has 2^8 possible values, meaning that one value as a probability of occurrence (in the worst case with the maximum uncertainty), of $\frac{1}{2^8}$. Thus, computing the entropy presented in Eq. 1 results to $\log_2(2^8) = 8\log_2(2) = 8$.

Hence, Eq. 1 returns a value between 0 and 8 [13]. Higher the entropy the higher the probability of an encrypted file.

3.1.2 Fuzzy hash

Since ransomware encryption produces an output which is dissimilar to its original content, such changes can be captured using similarity-preserving hash functions [17, 18]. Cryptographic hashing (MD5, SHA, etc.) is the traditional method to check the integrity of the file. This requires two copies of the same file. Hash is generated of the copied file and it is compared to the hash of the original file. If the hash is the same then we can say that both the files are the same. But the drawback with cryptographic hashing is that

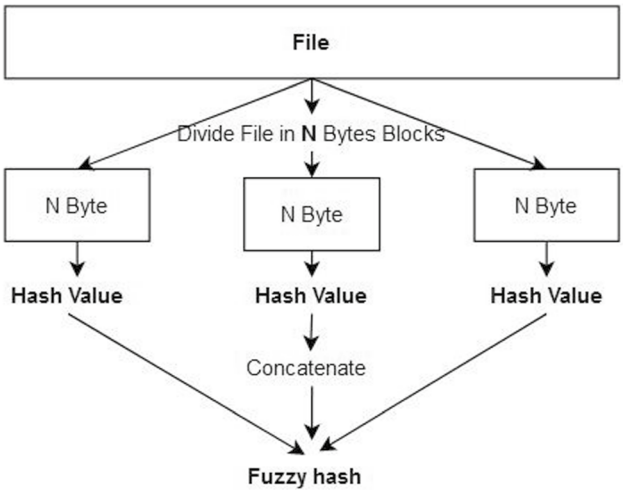

Fig. 1 Working of fuzzy hash

changing a single bit in a file will change its hash value, which renders the file is different from the original file. Using cryptographic hashing it is not possible to find any similarity/association between two files [19].

Fuzzy hashing can efficiently and effectively help to identify highly similar files. Fuzzy hash is calculated by dividing the file into multiple blocks and the hash value is calculated for each block, at last, hash values of all the blocks are combined as shown in Fig. 1. The block size is determined based on the content of the file using the rolling hash algorithm. The rolling hash algorithm produces pseudo-random value using a small context of a few bytes [17, 20]. It determines block boundaries in the file depending on the content of sections of seven bytes at a time, based upon Adler32 function. Afterwards, the hash value of each block is calculated separately and the fuzzy hash value is produced by concatenating all the hashes into one hash. Damerau-Levenshtein distance function is used to calculate the similarity between two files. It produces a similarity percentage which is between 0 and 100, where 0% indicates the files have low similarity and 100% indicates high similarity [21, 22].

3.2 Minifilter driver

A file system filter driver is a minifilter driver and a kernel-mode component that interacts with the behaviour of a file system. A file system filter driver can filter I/O operations for one or more file systems and because of this, it can log, modify, and prevent I/O operations.

As shown in Fig. 2, whenever a process or a user requests a file, the request is forwarded to the I/O Manager. Before forwarding the request to the File System Driver, the request is intercepted by the Filter Manager.

Filter Manager provides a rich set of functionalities required for developing a file system filter driver. For developing a minifilter driver we can adopt these functions instead of implementing a legacy filter driver from scratch. Only when a Minifilter driver is loaded into the operating system is the Filter Manager involved. The Minifilter driver is indirectly connected to the target volume file system stack, reporting its I/O operations to the Filter Manager. For each I/O operation, minifilter driver can register functions to be called at different events: before/after the requested I/O operation [15].

Filter Manager consists of many registered minifilter drivers; these drivers are called in order of their altitude. All minifilter drivers must have a unique identifier called altitude. The altitude of a minifilter driver defines its position concerning other minifilter drivers in the I/O stack when the minifilter driver is loaded. A minifilter driver having low numerical altitude is loaded into the I/O stack below a minifilter driver that has a higher numerical value. Filter manager passes I/O request packets to the minifilter driver in a predefined order according to their altitude [15].

In Fig. 2, assuming that all the three minifilter drivers are registered for the same I/O operation, the filter manager would call registered preoperation callback routine of a minifilter driver in order of altitude from highest to lowest (i.e. A, B, C) and then the I/O request is forwarded to the next lower driver for further processing. When the filter manager receives an I/O request for completion, it calls the registered postoperation callback routine of the minifilter driver from lowest to highest altitude (i.e. C, B, A).

The minifilter driver that has been proposed filters IRP_MJ_CREATE, IRP_MJ_WRITE, and IRP_MJ_CLOSE requests. So, whenever a filter manager receives these requests from a process, our minifilter driver will monitor the changes made by that process. The proposed minifilter driver prevents the creation of specific files or file content and hence the filter driver falls into "FSFilter Content Screener" load order group. The altitude range of this load order group is 260000–269,998 [23].

3.3 Requests filtered by minifilter driver

Following are the requests filtered by minifilter driver:

- IRP_MJ_CREATE: IRP_MJ_CREATE request is generated when a new file or directory is being created, or when an existing file, device, directory, or volume is being opened. If the request is completed successfully the process receives a handle to the file object [24].
- IRP_MJ_WRITE: IRP_MJ_WRITE request is generated when a process wants to perform write operation on a file [25].

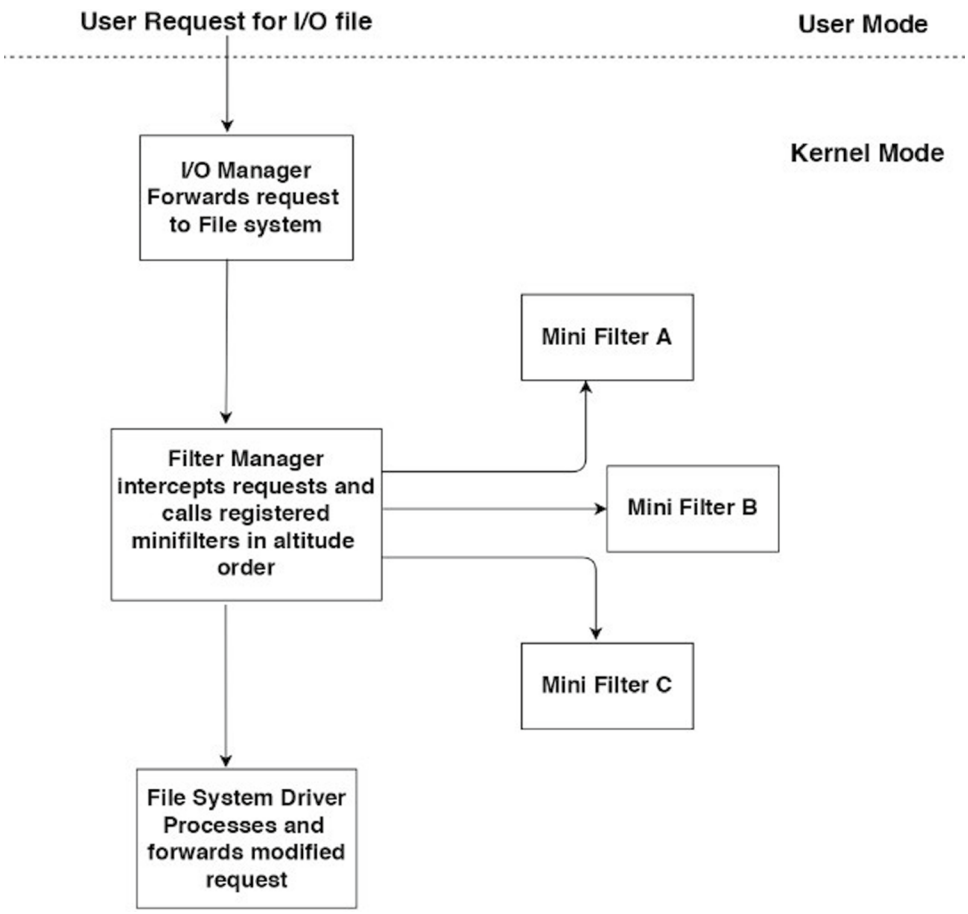

Fig. 2 Architecture of the file system filter driver

- IRP_MJ_CLOSE: IRP_MJ_CLOSE request is generated when the last handle of the file object has been closed or released [26].

3.4 Implementation

Figure 3 describes the proposed architecture of the anti-ransomware application. When a process requests read/write access for a file, the I/O manager sends IRP_MJ_CREATE to open the file and get handle to file object. IRP_MJ_CREATE request is further passed to filter manager. As our minifilter driver filters this request preoperation callback routine corresponding to the IRP_MJ_CREATE request is called by the filter manager.

• *Preoperation callback routine for IRP_MJ_CREATE:*

In the preoperation callback, the anti-ransomware application is called. In the application, a list of file extensions that may contain important data of users is maintained. The files accessed will be matched with the extensions mentioned in our list. If the extensions matched then the file will be monitored by the anti-ransomware. User can add or remove extension in the extension list using the GUI provided by the anti-ransomware application. Once the preoperation callback routine is completed the control is transferred back to the filter manager.

If the process wants to perform a write operation on the file then an IRP_MJ_WRITE request is generated. As our minifilter driver filters IRP_MJ_WRITE request preoperation callback routines registered for IRP_MJ_WRITE request will be called by the filter manager.

• Preoperation callback routine for IRP_MJ_WRITE:

In this preoperation callback routine the anti-ransomware application calculates the fuzzy hash of that file for further comparison once write operation is completed.

Once the process has completed its operation, I/O manager sends an IRP_MJ_CLOSE request to the filter manager. When the filter manager receives IRP_MJ_CLOSE request, it calls the postoperation callback routine which is registered with IRP_MJ_CLOSE request.

• Postoperation callback routine for IRP_MJ_CLOSE:

At this stage, the process has completed its I/O operation on the file. In the anti-ransomware application, we calculate Shannon's entropy of this modified file. If the entropy calculated is higher than the threshold then the file might be encrypted by the process. If the entropy is higher than the threshold, we calculate the fuzzy hash of the modified file. The fuzzy hash that had been calculated before

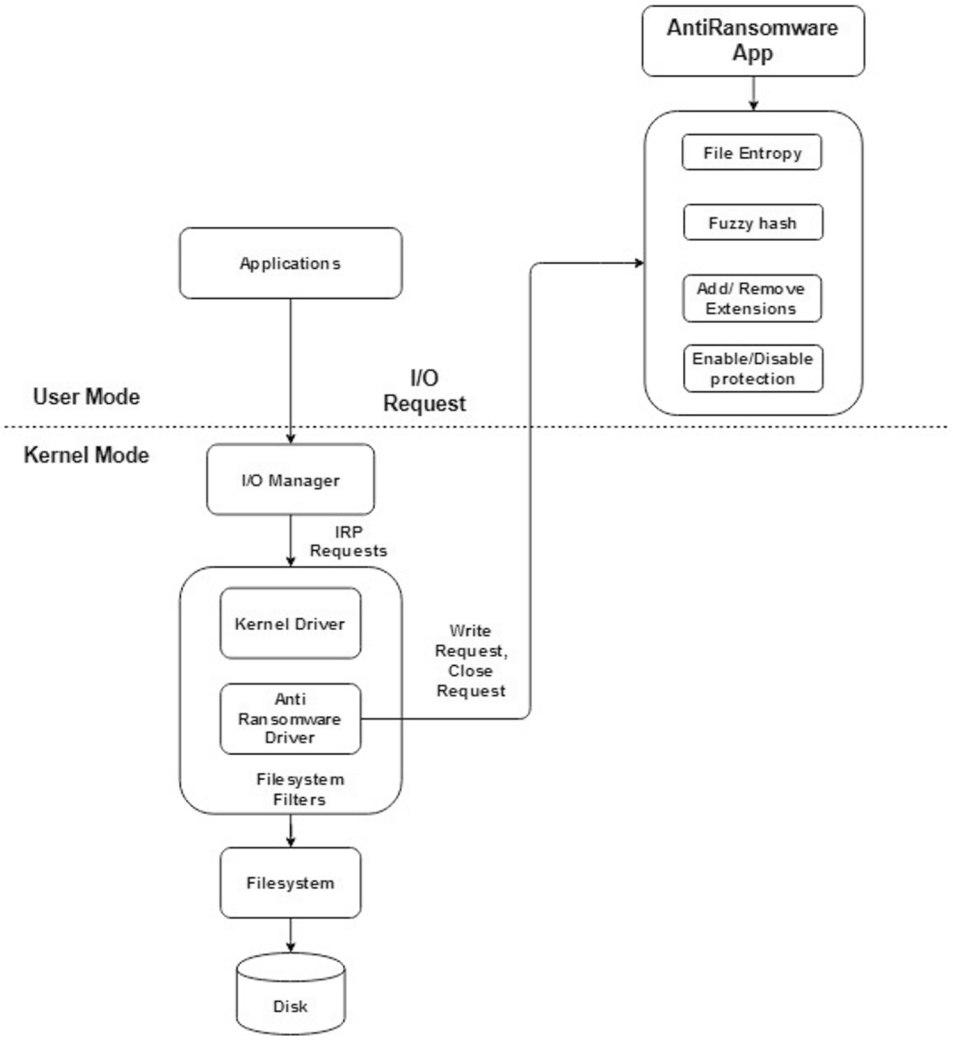

Fig. 3 Architecture of the anti-ransomware application

file modification in the preoperation callback routine is then compared with the fuzzy hash of the modified file and it produces a value that signifies the degree of similarity between the original file and modified file. If the original file is encrypted then the degree of similarity between the modified file and the original file will be low. If the degree of similarity is below a threshold value, this implies that the file is encrypted by the process and the user is notified.

Now, to allow the user access to the files and to alter them the anti-ransomware application allows certain processes to perform operations, such as the explorer.exe which is the process which carries the changes in files whenever the user alters it.

3.5 Threshold value of Shannon's entropy and fuzzy hash

By studying the work done in [27] and [28] to calculate the entropy of various types of files such as plain text, packed executable, native executables, and encrypted executables, the entropy threshold is set to 5.5.

Sarantinos et al. [19] and Roussev [20] has conducted various experiments to determine the similarity of files. The research work presented in [19] uses the fuzzy hash to determine similar malware files and the work presented in [20] determined similarity between different files such as JPG, PDF, TXT, DOC, XLS, PPT. By studying these papers, the fuzzy hash threshold is set to 21.

4 Result

4.1 Setting test environment

For evaluating the proposed technique, we used a virtual box for testing on different versions of Windows: Windows 7 32 bit, Windows 7 64 bit, Windows 8 32 bit, Windows 8.1 32 bit, Windows 8.1 64 bit, Windows 10 32 bit, Windows 10 64 bit. Majority of ransomware attacks are targeted towards

Windows operating systems [29]. After installing Windows, the following steps were taken for proper testing of the solution in a virtual machine environment:

- Disabling Windows Defender- To avoid possible blocking of our ransomware samples by Windows real-time protection.
- Disabling Windows Firewall- To allow the proper execution of malicious code because some malicious code acts in a restrained manner when the Windows firewall is active.
- Disabling LAN- So that there is no impact on other computers connected through LAN.

Libraries for visual C and.NET required by ransomware for execution were also installed. We populated the system with a large amount of artificial data (PDF, JPEG, HTML, PNG, JPG file, and directories) to keep the testing environment as real as possible.

After setting up the environment the most important task is monitoring activities. Following tools were used for monitoring the system-

- Process Monitor- It is a monitoring tool for Windows from Windows SYS-internals that displays in real-time all file systems, process/thread, and registry activity [30].
- Process Explorer- It shows the list of currently active processes and also displays information regarding handles and DLLs opened or loaded by the process [30].

For checking whether ransomware was successful in encrypting artificial data we monitored files, drivers output log, and desktop background as generally a ransom message is displayed after encrypting data.

4.2 Testing phase

We obtained ransomware samples from crawling VirusTotal API and through the VirusShare repository. Although these samples were marked as ransomware by various antivirus engines does not imply that these are actual ransomware samples. Some of the samples do not perform any operation to harm user's data and therefore these types of samples were removed from the dataset. The final dataset for testing consisted of 508 ransomware samples.

Figure 4 shows the distribution of different ransomware family samples obtained for the purpose of testing our proposed ransomware detection technique.

Table 1 depicts the number of samples of different ransomware families tested against our proposed technique and the detection result of those samples. The table also mentions the first attack year of the respective ransomware families.

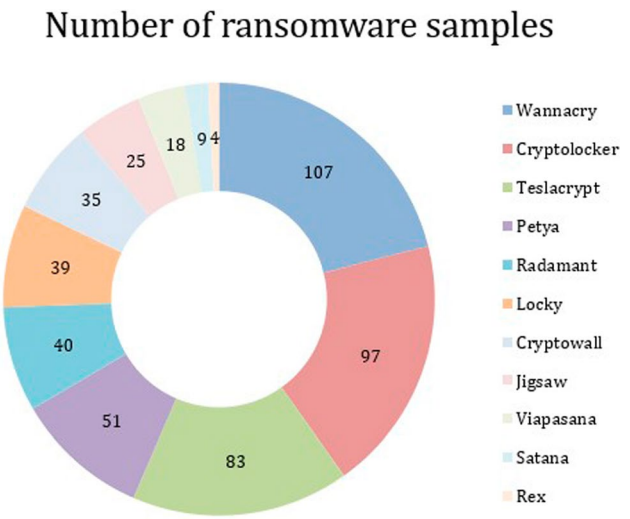

Fig. 4 Distribution of the number of ransomware samples

As shown in Table 1, the proposed technique was successful in detecting and blocking 483 out of 508 ransomware samples tested, which does provide a detection rate of 95.07% against ransomware attacks. The jigsaw ransomware was not blocked by the proposed minifilter driver because this particular ransomware used methods different than any other ransomware to access files on the machine, therefore its behaviour was not detected by the anti-ransomware. The working of Jigsaw looked as if it was only trying to rename the files thus it got through the blocking mechanism. Further research would be required to understand the exact behaviour of Jigsaw.

4.3 False positive

The proposed technique should block the ransomware applications but it should not block any legitimate applications. To check the robustness of our system, we tested our technique against some of the common windows applications in the same virtual environment. Windows application which we tested were: 7-zip, Mozilla Firefox, Microsoft Word, Spotify, VLC media player, Microsoft Excel, Adobe Reader, uTorrent, Skype, Microsoft Outlook, Dropbox, LibreOffice, Paint.Net, Netflix, Microsoft Sticky Notes, Adobe Photoshop Express. None of the applications was falsely detected as ransomware when tested against our proposed technique resulting in zero false positives. The use of filter driver along with the combined working of Shannon's entropy and Fuzzy Hash techniques allow legitimate user processes to carry out the required tasks.

Table 1 Detection of different ransomware samples by our technique

Ransomware families	No. of samples	% of samples	First attack year	Result
Cryptolocker	97	19.09	2013	Detected
Cryptowall	35	6.88	2014	Detected
Jigsaw	25	4.93	2016	Not Detected
Locky	39	7.68	2016	Detected
Petya	51	10.03	2016	Detected
Radamant	40	7.87	2015	Detected
Rex	04	0.78	2016	Detected
Satana	09	1.78	2017	Detected
Teslacrypt	83	16.34	2015	Detected
Viapasana	18	3.55	2015	Detected
Wannacry	107	21.07	2017	Detected

4.4 Limitations

The proposed technique does have a few limitations such as a DLL injection in explorer.exe would bypass the security of the system or malware can access a lower altitude mini-filter driver and disable a mini-filter driver at a higher altitude. In further work, mechanisms to overcome the limitations can be integrated with our proposed technique to provide total protection from all types of malware attacks. Currently, we suggest a backup mechanism along with our proposed system as a precautionary measure. These limitations however do not affect the results of the tested ransomware samples and are far beyond the scope of this paper. In further work, mechanisms to overcome the limitations can be integrated with the proposed technique to provide total protection from all types of malware attacks.

5 Conclusion

Ransomware is not a new phenomenon but it has increased on a large scale in recent years. It is highly important to build an anti-ransomware tool that can ensure confidentiality, integrity, and availability of private data.

In this paper, we had proposed a mini-filter driver that allows us to analyse the I/O request (read, write, and attribute change requests) to the file system driver. The mini-filter driver is deployed at a certain altitude such that it has access to most of the objects of the operating system. This detection technique is robust because, to bypass the security of the mini-filter driver, a new mini-filter driver needs to be developed and deployed whose altitude needs to be equal to or less than the proposed mini-filter driver. The implementation of the signature-less technique by combining the working of Shannon's entropy and fuzzy hash allows us to detect even new and unknown ransomware attacks. Our proposed technique was successful in detecting more than 95% of the tested ransomware families in all Windows operating systems.

The proposed anti-ransomware system can be developed in the future by combining signature-less detection with signature detection. This will allow the system to detect known ransomware signatures in a faster way along with providing better results by detecting the signature of ransomware which remained undetected in the signature-less detection.

References

1. Poriye, A.M., Kumar, V.: Ransomware detection and prevention. Int. J. Comput. Sci. Eng. **6**(5), 900–905 (2018). https://doi.org/10.26438/ijcse/v6i5.900905
2. Luo, X., Liao, Q.: Awareness education as the key to ransomware prevention. Inf. Syst. Secur. (2007). https://doi.org/10.1080/10658980701576412
3. Gorman, G.O., McDonald, G.: Ransomware : a growing menace. Symantec (2012).
4. Monika, Zavarsky, P., Lindskog, D.: Experimental analysis of ransomware on windows and android platforms: evolution and characterization. Procedia Comput. Sci. **94**, 465–472 (2016). https://doi.org/10.1016/j.procs.2016.08.072
5. Kumar, S., Spafford, E.H.: A generic virus scanner in C++. In: Proceedings - Annual Computer Security Applications Conference, ACSAC (1992). https://doi.org/10.1109/CSAC.1992.228218.
6. Traynor, P., Chien, M., Weaver, S., Hicks, B., McDaniel, P.: Non-invasive methods for host certification. ACM Trans. Inf. Syst. Secur. (2008). https://doi.org/10.1145/1341731.1341737
7. Moser, A., Kruegel, C., Kirda, E.: Limits of static analysis for malware detection. In: Proceedings - Annual Computer Security Applications Conference, ACSAC (2007). https://doi.org/10.1109/ACSAC.2007.21.
8. Dekel, K., Mizrachi, L., Zaikin, R., Vanunu, O.: Method and system for mitigating the effects of ransomware (2018).
9. Moore, C.: Detecting ransomware with honeypot techniques. In: Proceedings of 2016 Cybersecurity Cyberforensics Conference. CCC 2016, pp. 77–81 (2016). https://doi.org/10.1109/CCC.2016.14.
10. Lance Spitzner, Honeypots: Tracking Hackers. 2002.
11. Mohammed, M., Rehman, H.U.: Honeypots and routers: collecting internet attacks (2016).

12. Gonzalez, D., Hayajneh, T.: Detection and prevention of crypto-ransomware. In: 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2017, pp. 472–478 (2017). https://doi.org/10.1109/UEMCON.2017.8249052
13. Scaife, N., Carter, H., Traynor, P., Butler, K.R.B.: CryptoLock (and Drop It): stopping ransomware attacks on user data. In: Proceedings - International Conference on Distributed Computing Systems, pp. 303–312 (2016). https://doi.org/10.1109/ICDCS.2016.46.
14. Hosfelt, D.D.: Automated detection and classification of cryptographic algorithms in binary programs through machine learning. 2015, [Online]. Available: http://arxiv.org/%0Aabs/1503.01186.
15. Bottazzi, G., Italiano, G.F., Spera, D.: Preventing ransomware attacks through file system filter drivers. In: Proceedings of the Second Italian Conference on Cyber Security (ITASEC18), At Milan (2018).
16. Rajaram, R., Castellani, B., Wilson, A.N.: Advancing Shannon entropy for measuring diversity in systems. Complexity (2014). https://doi.org/10.1155/2017/8715605
17. Kornblum, J.: Identifying almost identical files using context triggered piecewise hashing. Digit. Investig. **3**(Supplement), 91–97 (2006). https://doi.org/10.1016/j.diin.2006.06.015
18. Roussev, V.: Data fingerprinting with similarity digests. IFIP Adv. Inform. Commun. Technol. (2010). https://doi.org/10.1007/978-3-642-15506-2_15
19. Sarantinos, N., Benzaïd, C., Arabiat, O., Al-Nemrat, A.: Forensic malware analysis: the value of fuzzy hashing algorithms in identifying similarities. In: Proceedings - 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 10th IEEE International Conference on Big Data Science and Engineering and 14th IEEE International Symposium on Parallel and Distributed Proce, 2016. https://doi.org/10.1109/TrustCom.2016.0274.
20. Roussev, V.: An evaluation of forensic similarity hashes. Digit. Investig. (2011). https://doi.org/10.1016/j.diin.2011.05.005
21. Naik, N., Jenkins, P., Savage, N., Yang, L., Naik, K., Song, J.: Augmented YARA rules fused with fuzzy hashing in ransomware triaging. IEEE Symp. Ser. Comput. Intell. (SSCI) **2019**, 625–632 (2019). https://doi.org/10.1109/SSCI44817.2019.9002773
22. Naik, N., Jenkins, P., Savage, N.: A ransomware detection method using fuzzy hashing for mitigating the risk of occlusion of information systems. In: ISSE 2019 - 5th IEEE International Symposium on Systems Engineering, Proceedings (2019). https://doi.org/10.1109/ISSE46696.2019.8984540.
23. Kim, H., Hollasch, L.W., Coulter, D., Matt: Load order groups and altitudes for filter drivers (2017). https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/load-order-groups-and-altitudes-for-minifilter-drivers.
24. Hollasch, L.W., Coulter, D., Marshall, D.: IRP_MJ_CREATE (IFS) - Windows drivers | Microsoft Docs (2017). https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/irp-mj-create#:~:text=TheI%2FO Manager sends, or volume is being opened.
25. Hollasch, L.W., Coulter, D., Kim, A., MacMichael, D.: IRP_MJ_WRITE (IFS) - Windows drivers|Microsoft Docs (2017). https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/irp-mj-write.
26. Hudek, T., Graff, E., Sherer, T.: IRP_MJ_CLOSE - Windows drivers | Microsoft Docs (2017). https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/irp-mj-close.
27. Lyda, R., Hamrock, J.: Using entropy analysis to find encrypted and packed malware. IEEE Secur. Priv. (2007). https://doi.org/10.1109/MSP.2007.48
28. Paul, C.B.: Entropy-based file type identification and partitioning (2017).
29. Mansfield-Devine, S.: Ransomware: taking businesses hostage. Netw. Secur. (2016). https://doi.org/10.1016/S1353-4858(16)30096-4
30. Russinovich, M.: Sysinternals suite. Microsoft (2018).