Recovery Method for Ransomware Encryption Attacks with File Extension Changing on File Server

Rintaro Nagano*, Ryuku Hisasue*, Hiroshi Inamura†, Shigemi Ishida†
* Graduate School of Systems Information Science, Future University Hakodate, Japan
Email: {g2123044, g2122054}@fun.ac.jp
† School of Systems Information Science, Future University Hakodate, Japan
Email: {inamura, ish}@fun.ac.jp

***Abstract*—Cryptographic ransomware, which encrypts and disables accessing files and demands money in exchange for decryption, is causing damage to businesses. Conventional automatic restoration running on a PC as a countermeasure against an encryption attack requires the installation, which increases PC maintenance costs. Unlike the conventional method where data was stored on each terminal, if data is handled only on the file server, only the file server needs to be protected from encryption attacks, thus reducing management costs for the entire organization. In this study, focusing on the behavior of known samples that change file extensions to specific ones, we propose a method to identify and roll back file operations caused by cryptographic attacks on file servers. We evaluated the proposed system by performing an encryption attack on it using mock ransomware implemented according to the typical behavior of ransomware that changes the file extension to a specific one. The proposed system can identify and roll back file operations performed by the ransomware and automatically recover files that have been encrypted in a short time.**

***Index Terms*—Ransomware, File Server, File Extension, Automatic Recovery**

I. Introduction

Cryptographic ransomware, which disables accessing files and demands money, is a threat [1]. Encryption attacks by ransomware have caused enormous damage. For example, the data of Electronic Medical Records in Osaka General Medical Center in Japan was encrypted, and it took 60 days to restore the service, with the total damage amounting to more than one billion yen [2]. Thus, the total amount of damage expands by the longer the recovery period. When organizations are attacked by ransomware, the damage can be reduced by restoring files quickly and restarting the business.

The existing method for automatically restoring files that are encrypted by ransomware [3] provides fast-automatic restoration. However, building a countermeasure system on each PC would be costly to manage, and deploying or updating the system on every PC is burdensome. It is difficult for non-IT expert members in organizations to manage complex countermeasure mechanisms by themselves, and it costs if the organization hire system management staffs.

On the other hand, if all data is stored on the file server, unlike conventional system where each PC stored data, only one countermeasure against encryption attacks on the file server is needed, reducing the management cost.

Typical cryptographic ransomware changes the file extension of encrypted files to a specific extension to make it easier to recognize that the files are encrypted and upset the victims [5]. Based on this feature found in the known samples, it allows automatic identification limited to actions that change the file extension to a specific one. Using this, we propose a mechanism to enable rapid recovery from cryptographic ransomware attacks by automatically identifying and rolling back encryption attacks against files on file servers. Since the goal is to reduce the number of devices to be managed, our system does not need to be installed on each client device, just on the file server. We implemented a mock ransomware that changes file extension to specific one during encryption and evaluated the time for automatic restoration from encryption attacks.

Our main contributions are twofold:

- To reduce management costs, we proposed a method to automatically restore files encrypted by ransomware on file servers, and implemented a prototype that initiates identification and restoration triggered by user actions.
- We evaluated the time for restration from an encryption attack by mock ransomware by having the ransomware encryption attack and other normal file operations executed in parallel, and then identifying the file operations caused by the attack and rolling back. The results showed that the time for resarection was short.

II. Related Work

This section describes two types of studies: one that detects and stops encryption attacks on client PCs and automatically restores encrypted files and the other that detects and stops encryption attacks using network traffic between file servers and client PCs. We then describe the information used as a clue to detect ransomware and clarify the information effective in realizing countermeasure mechanisms on file servers.

A. Stopping of Encryption Attacks on Client PCs, and Automatic Restoration

It has been shown that existing methods on client PCs can detect and stop encryption attacks and automatically restore them with high accuracy using machine learning and other methods based on the characteristics of operations such as encryption attacks and the creation of threat text.

Hodota et al. use a hypervisor to collect storage access patterns for ransomware detection [6]. Scaife et al. use metrics

978-4-907626-52-5/23/$31.00 ©2023 IPSJ

about the behavior of encryption attacks to create a kernel driver that detects encryption attacks, thereby stopping ransomware before a large amount of user data are lost [7].

However, the high management costs associated with deploying, maintaining, and updating hypervisors and kernel drivers make it difficult to implement these techniques in organizations. For example, managing complex mechanisms such as hypervisors in an organization requires advanced knowledge, and it is difficult for members of the organization who need to be proficient in information technology to implement these detection mechanisms on their own. In other words, it is necessary to ask other organizations to set up and update PCs to implement existing methods, which is time-consuming and expensive. Furthermore, these methods cannot automatically restore files, and require additional time for file restoration, which may increase the damage due to the time needed to restore the business.

Continella et al. developed an add-on driver for Windows native file systems that detects ransomware encryption attacks and performs automatic data recovery [3]. They implemented an IRPLogger (I/O Request Packet Logger) module to collect low-level I/O file system requests and logs of user and ransomware file accesses. This method copies files when the process first accesses the file and uses the copied data to recover the file if attacked by ransomware. Using the classification model generated by supervised learning with the collected data, the system classifies benign and ransomware processes with an accuracy of 97.7%. However, when reevaluated in Ref. [8], it was reported that the classification accuracy dropped to 65.7% because this method uses machine learning.

As described above, the three problems with the method for client PCs are *high management cost, some methods only stop ransomware and do not automatically restore it,* and *accuracy dropped unless they are relearned*, because they use machine learning for detection.

B. Stopping Encryption Attacks using Network Traffic

To detect and stop encryption attacks using network traffic, Daniel et al. have developed a system to collect traffic data using the system that converts packet-based communication into an analyzable metadata format, and intercepts communication with the client PC when detects behaviors specific to encryption attacks [9]. They developed a system that detects encryption attack-specific behavior and intercepts communication with client PCs. They have confirmed that this method can stop a file server from being attacked by encryption attacks. However, this system using network traffic only detects and stops encryption, doesn't restore files automatically.

C. Issues in Automatic Restoration from Encryption Attack on A File Server

Attempting to recover files that have suffered encryption attacks on file servers can solve the management cost problem, but there are two research tasks; realizing 1) automated and 2) short-time recovery from file encrption attacks.

When operating countermeasure systems in organizations such as a company, the ability to quickly restore business operations after an encryption attack helps limit economic damage. In order to prevent the total amount of damage from increasing due to a prolonged recovery period, fast file recovery is necessary by not only stopping encryption attacks but also automatically restoring files.

D. Characteristics of Encryption Attacks

The change of file extensions by ransomware is valuable information for solving the two problems described in the previous section. Several kinds of ransomware changes the file extensions of files to be encrypted [4, 5]. It is able to get changed file extensions information on file servers.

In the literatures [7, 10], The following four typical behaviors of encryption attacks are described:

1) Read data from the file and overwrite the same file with the encrypted data
2) Read data from the file, save the encrypted data in new file, and then delete the original file
3) Read the data in the file, save the encrypted data in a separate file, and then overwrite the original file with another set of data to destroy it.
4) Move the file from the user's directory to a temporary directory, then read the file, write the encrypted contents, and return the file to its original location.

When ransomware changes file extensions to specific extensions, there is a possibility to identify file operations by ransomware based on the file extension change operations. Among the file operations, those that include a specific file extension in the filename of the target file can be identified as operations related to the encryption attack.

III. Recovery method from encryption attack on file server

We propose a system to quickly and automatically identify and roll back file operations of encryption attacks on files on file servers for the ransomware that behaves of known samples that change file extensions to specific ones.

We use the file-extension modification associated with encryption attacks to detect ransomware on file servers. Whenever a write operation occurs on the file server, the system backs up the contents before the write operation, and the system identifies the backup data before encryption from the "file operations" that list the file operations executed on the client PCs in chronological order.

Our method does not use machine learning or other time-consuming mechanisms to recover files, but uses file extensions that can identify file operations of ransomware quickly. We attempt to identify file deletion and file creation operations of ransomware from file extensions and roll back file operations by encryption attacks in a short time. Reducing false positives is an important point because an automatic file recovery system might unintentionally reverse legitimate user changes by misdetection. Using the feature that ransomware changes the extensions of encrypted files to specific ones to enhance their threatening impact, we can reduce false positives while maintaining reliable detection based on these specific ransomware extensions.

A. Overview of Proposed System

The flow from file destruction by ransomware to file recovery by the proposed system is shown in Fig. 1.

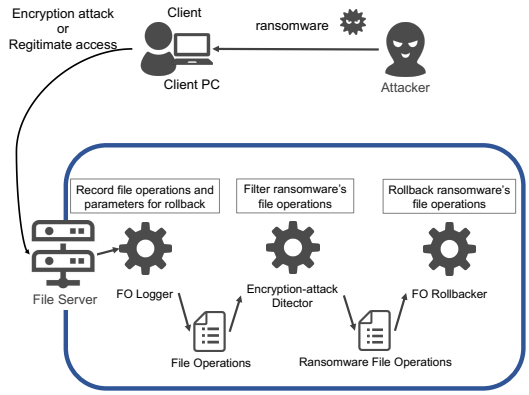

Fig. 1. Workflow of encryption to restoration

Our proposed system is composed of three main components: the *FO (File Operations) Logger*, which produces sequences of file operations, the *Encryption-Attack Detector*, which identifies file creation and deletion operations with specific extensions during ransomware encryption attacks, and the *FO Rollbacker*, which reverts the file creation and deletion attacks identified by the Encryption-Attack Detector. The sequences generated by the FO Logger feed into the Encryption-Attack Detector. The operations identified as part of an encryption attack are then passed to the FO Rollbacker for automatic file restoration.

B. File Server used for Prototype Implementation

In this study, we implemented our proposed system as an extension of an open-source file server, UNFS3 [11]. UNFS3 is a file server implementation that supports the Network File System version 3.0 (NFS ver.3.0) [12] protocol and operates in user space. Our system records and rolls back operations based on the procedures defined in the NFS ver.3.0 protocol. The release version of UNFS3 we used is 0.9.22.

C. File Operation Function

In building our proposed system, we focus on a specific function that activates when a file on a file server undergoes manipulation. We call this *file operation function* in this paper. When a client with an NFS file server mount conducts a file operation, the file server uses ONC RPC [13, 14] to initiate the file operation function. ONC RPC, a protocol developed by Sun Microsystems, enables function execution on another PC via a network. On this protocol, a client program's function call prompts a request to the file server from the client's PC. Upon receiving this request, the file server executes the corresponding procedure and responds the result to the client's PC. In this way, the file server initiates the appropriate file operation function each time a client manipulates a file.

Executed file operation functions can be logged on the file server side, independent of the client PC. Thus, it is able to use these file operation functions in our proposed system, which is designed to file restoration solely on the file server side.

D. FO Logger

The FO logger logs sequences of file operations. Each time a file operation function runs, FO logger logs a single record of the file operations. Each record includes the execution timestamp, the invoked file operation function, the function's arguments, and the data before the operation's alterations.

E. Encryption Attack Detector

The encryption attack detector identifies file deletion and file creation operations associated with encryption attacks using ransomware-specific file extensions. This function can be executed at any time by this system's administrator. When a file deletion or file creation is performed, the path name of the target file is passed to the corresponding file operation function as an argument. These path names can be output to the file operations. If those path names contain ransomware-specific extensions, the file operation is categorized as a ransomware file operation. Finally, the specified file operations are output for input to the rollback function.

The rollback function undoes the specified file operations. It rolls back the file operations included in the encrypted attack file operations obtained by Encryption Attack Detector which is described in the previous section. Using the file operations log saved by the FO logger, undo the executed file operations.

The flow of the rollback function is described below. First, the proposed system temporarily stops accepting client requests to prevent new file operations from being executed while identifying encryption attacks. Next, the file operations are read and stored in a list structure. Then, the file operations are sorted in reverse order of timestamps. Finally, a procedure is executed that performs the reverse operation of the executed file operation function for each record in the file operations.

IV. EVALUATION

A. Experiment Setup

The experimental environment is constructed by file server and mock ransomware. A file server is started on the server, a public directory is self-mounted, and the files in the mounted directory are conducted encryption attacks. The nginx directory and the target directory are placed in a shared directory, and execute the nginx build during the encryption attack.

Referring to the literature [15], we set the file size of files to be encrypted to 80 kB, the median file size of a file used by Windows users in a real environment. In this literature, data was collected over one month during the first half of 2000 when a diverse group of people used the files for various tasks. In this experiment, the median file size used by Windows NT users was about 80 kB, and about 20% of all files used were also around 80 kB.

In this study, we use our implemented mock ransomware for evaluation to remove unexpected behavior during evaluation. The mock ransomware is implemented based on typical ransomware behavior described in the literature [7].

The mock ransomware used in our experiments performs an encryption attack as follows. First, it reads the file to be encrypted. Second, it creates a file to write encrypted data. Third, it writes encrypted data and change the file extension. Finally, it deletes the original file.

In actual usage of this system, other file operations are performed in parallel with the encryption attack.

The more complex the parallel operation is, the more difficult it is to identify the file operation. In the real environment, Word, Excel, and other tools will likely perform

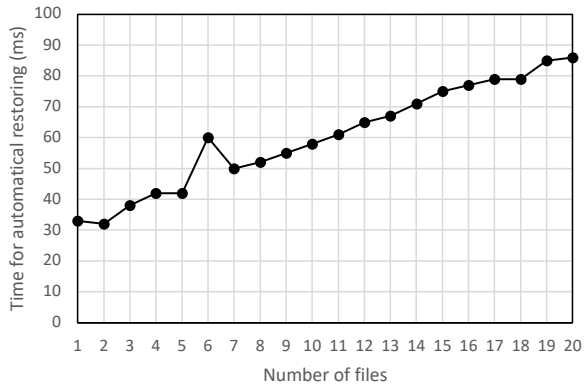

Fig. 2. Relation between recovery time and number of files

the operations. Still, the file operations by these application software are likely to be sporadic and have a small number of file operations per client. An example of many file operations would be program builds. In this experiment, we evaluate the ability to automatically roll back only the files subjected to encryption attacks among multiple file operations; building software from source code that intensively reads and writes files parallel to encryption attacks by mock ransomware. We used nginx version 1.23.3 as the target software.

B. Selective Rollback Evaluation to Encryption Attacks

We measure the execution time of the restoration process by the encryption attack detector function in our method identifying file operations of files that have suffered encryption attacks triggered by user operations. The execution time of this process is the downtime of the file restoration process. Since it is expected that the execution time of the attack invalidation process will vary depending on the number and size of target files, we evaluate the incremented execution time by changing the number of the target files. In this paper, we change the number of target files from 1 to 20, conduct experiments with 20 trials for each file, and measure the time of restoring attacked files.

C. The Results of Restoration-time Evaluation

Fig. 2 shows the results of executing the encryption attack by the mock ransomware while performing normal file operations in parallel, identifying the file deletion/creation operations by the mock ransomware, and restoring the files. The vertical axis represents the execution time required to restore files, and the horizontal axis represents the number of restored files. The file operations executed in parallel are nginx builds, as described in Sec. IV-A.

As a result, even when the largest number 20, all files are automatically restored in 86 ms. The graph is almost a straight line, which indicates the recovery time is proportional to the number of files for each about 3 ms per a file.

The recovery time is relatively short and is expected to have a negligible impact on business continuity. This short-recovery time is critical because our method assumes that restoring files from encryption attacks on the file server used in organizations such as a company.

We showed that the proposed method could automatically restore files in a short time from ransomware encryption attacks that delete original and create encrypted files, even when in parallel to regular file operation.

V. CONCLUSION

To reduce the management costs required to disable cryptographic ransomware attacks, we proposed a mechanism to automatically and quickly identify and roll back encryption attack that change file extensions to specific ones on the file server. This restore operation is triggered by a user operation. Using the proposed mechanism, we measured the time required to automatically restore files from a mock ransomware encryption attack. In our implementation on a PC with typical performance, we restored up to 20 files of 80 kb, and the process was completed within 86 ms. Furthermore, we confirmed that the proposed method works without problems even when other file operations occur in parallel with the encryption attack.

REFERENCES

[1] EUROPOL, 2016. Internet Organised Crime Thread Assessment (IOCTA) 2016. Tech. rep. Europol - European Police Office, https://doi.org/10.2813/275589.

[2] Information Security Incident Investigation Committee on Osaka General Medical Center, Investigation report, https://www.gh.opho.jp/pdf/report_v01.pdf (in Japanese, 2023-5-23 accessed).

[3] Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barenghi, A., Zanero, S. and Maggi, F.: ShieldFS: A Self-healing, Ransomware-aware Filesystem, *Proceedings of the 32nd Annual Computer Security Applications Conference*, ACM.

[4] Tomas Meskauskas, Babuk Locker ransomware virus - removal and decryption options, https://www.pcrisk.com/removal-guides/19783-babuk-locker-ransomware (2023-6-25 accessed)

[5] GROUP, Y. T. Conti Ransomware source code: a well-designed COTS ransomware, https://yoroi.company/research/conti-ransomware-source-code-a-well-designed-cots-ransomware/ (2023-5-22 accessed)

[6] M. Hirano and R. Kobayashi, "Machine Learning Based Ransomware Detection Using Storage Access Patterns Obtained From Live-forensic Hypervisor," *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, Granada, Spain, 2019, pp. 1-6, doi 10.1109/IOTSMS48152.2019.8939214.

[7] Scaife, N., Carter, H., Traynor, P. and Butler, K. R. B.: CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data, *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pp. 303–312 (2016).

[8] Gupta, A., Prakash, A. and Scaife, N.: Prognosis Negative: Evaluating Real-Time Behavioral Ransomware Detectors, *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 353–368 (online),

[9] Morato, D., Berrueta, E., Magac3b1a, E. and Izal, M.: Ransomware early detection by the analysis of file sharing traffic, *Journal of Network and Computer Applications*, Vol. 124, pp. 14–32 (online),

[10] T. Hagiwara,R. Kobayashi,M. Kato A Study on Detection and Process Identification of Encrypting Ransomware Using Decoy Files, *DPS* (in Japanese), Vol. 2021, No. 50, pp. 1–8 (2021).

[11] Schmidt, P.: UNFS3 Homepage, https://unfs3.sourceforge.net/ (2022-12-22).

[12] Staubach, P., Pawlowski, B. and Callaghan, B.: NFS Version 3 Protocol Specification, RFC 1813 (1995).

[13] Sun Microsystems, I.: RPC: Remote Procedure Call Protocol specification: Version 2, RFC 1057 (1988).

[14] Srinivasan, R.: XDR: External Data Representation Standard, RFC 1832 (1995).

[15] Roselli, D. S., Lorch, J. R., Anderson, T. E. et al.: A Comparison of File System Workloads., *USENIX annual technical conference, general track*, pp. 41–54 (2000).