# Ransomware Encrypted Your Files but You Restored Them from Network Traffic

Eduardo Berrueta[1], Daniel Morato[1,2], Eduardo Magaña[1,2], Mikel Izal[1,2]
[1] Public University of Navarre, Department of Electrical, Electronic Engineering and Communications, Pamplona, Spain
[2] Institute of Smart Cities, Pamplona, Spain
email: {eduardo.berrueta, daniel.morato, eduardo.magana, mikel.izal}@unavarra.es

***Abstract*— In a scenario where user files are stored in a network shared volume, a single computer infected by ransomware could encrypt the whole set of shared files, with a large impact on user productivity. On the other hand, medium and large companies maintain hardware or software probes that monitor the traffic in critical network links, in order to evaluate service performance, detect security breaches, account for network or service usage, etc. In this paper we suggest using the monitoring capabilities in one of these tools in order to keep a trace of the traffic between the users and the file server. Once the ransomware is detected, the lost files can be recovered from the traffic trace. This includes any user modifications posterior to the last snapshot of periodic backups. The paper explains the problems faced by the monitoring tool, which is neither the client nor the server of the file sharing operations. It also describes the data structures in order to process the actions of users that could be simultaneously working on the same file. A proof of concept software implementation was capable of successfully recovering the files encrypted by 18 different ransomware families.**

## I. INTRODUCTION

In recent years, a new type of malware named 'ransomware' has raised in popularity due to its massive impact on business [1] and home users alike [2]. This kind of malware locks the access to user's computer (lockscreen ransomware) or encrypts user's files (crypto ransomware or cryptoware) and it asks for a payment in order to recover them. During 2016, Europol declared that encrypting ransomware had become "the most prominent malware threat (...) for citizens and enterprises alike" [3].

In medium and large enterprises, files are stored on network shared volumes. Those volumes could be shared among several users and hosts. This architecture allows simple file sharing between end users and the implementation of centralized backup policies for those volumes. However, when crypto ransomware infects one user host in this LAN (Local Area Network) it will be able to access and encrypt the files in the shared volume. The result is a much larger impact than for isolated hosts. In this paper we focus on this scenario of LAN shared volumes and the recovery of lost files due to an infection.

In situations of infection by ransomware there are cases when the affected users pay the ransom [4]. This is the reason for the lucrative illicit business around ransomware [5], including the appearance of ransomware-as-a-service offerings [6]. For the users that select avoiding the payment, the prevalent solution is the recovery of lost files from the most recent backup. However, all the new files and modifications made after the last backup are lost. As most enterprises execute nightly-backups [7], it could represent a considerable lost of information. The backup solution must keep its snapshots inaccessible from the user hosts and any ransomware [8], or else it risks getting also the backup files encrypted.

There have been several proposals for the detection of ransomware [9] [10] [11]. Most of them require the malware to take destructive actions before an alarm can be raised [12], [13], which implies the loss of some files. The numbers are in the range of 5-20 deleted files, depending on the detection algorithm and the strain of ransomware. Proposals like [14] [15] [16] avoid data loss by intercepting I/O (Input/Output) system calls and keeping a backup of lost data. However, these solutions present a burden on the user's CPU and disk. They must also be installed in all hosts and they could be deactivated by ransomware that escalates privileges.

In this paper we propose the recovery of lost files thanks to the monitored network traffic between the user hosts and the shared volume. It is common the deployment of network traffic monitoring tools for security auditing, performance evaluation or accounting. They receive a copy of traffic thanks to mirroring techniques implemented on the network switches (Figure 1). Such a deployment does not require any software installation on the user hosts or the file sharing appliance. Compared to an in-path firewall or proxy server it does not impair network traffic and it is not a point of failure in the network. Also, the probe cannot be targeted by malware as it is an off-path deployment that only receives a copy of the traffic. Traffic mirroring is implemented at line-rate thanks to modern switching electronics, without any impact on data plane forwarding. Some examples of analysis tools (hardware and software) are [17] [18] [19] [20]. They are capable of storing traffic coming from 10Gb/s links.

If the computing power in the network monitoring tool is enough, it could implement a ransomware detection algorithm based on the file sharing activity. We assume a detection algorithm is already implemented (such as [12] or [13]) and we describe how the lost files can be recovered from the traffic stored in the network probe.

Although there exist tools for recovering files based on network traffic, they are generic and they do not take into account the specifics of a ransomware alert scenario, where the data to recover must not be the one encrypted by the

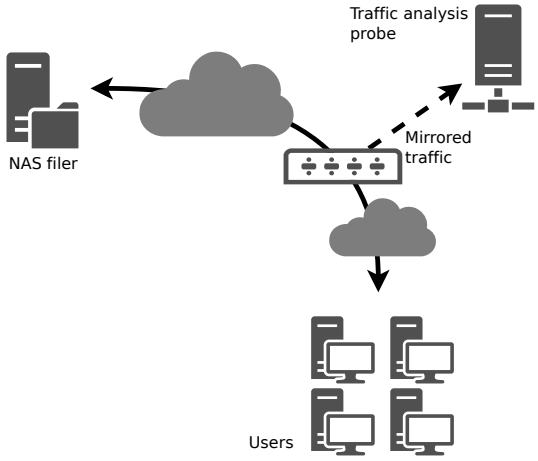


Fig. 1. Network monitoring scenario. The traffic between the hosts and the NAS (Network Attached Storage) filer is replicated from a switch into the traffic analysis probe

ransomware but the previous version. Tools like [21] or [22] extract files from HTTP or FTP traffic, which are not common file sharing protocols in the corporate environment. Wireshark is the reference traffic analysis tool and it supports a module [23] for the recovery of files from file sharing traffic that uses Server Message Block (SMB) protocol [24] [25]. However, it recovers as many instances of the file as they were opened in the trace file, without any distinction between original or encrypted data. Finally, [26] uses file format signatures in order to look for the beginning of a file, and from there it recovers the file. However, it does not consider the concurrent actions of different user on the same file and according to [23], it losses data and it cannot correctly recognize common file formats.

This paper describes the design, testing and evaluation of a tool for the recovery of files transferred using SMB2 or unencrypted SMB3 sessions, as this is the most common file sharing protocol in the corporate environment. It serves as a proof of concept implementation which can be extended to other network file sharing protocols, as it does not require any capability unique to SMB. It is based only on the protocol messages used to open, read, write, close and remove a network shared file. These messages are present in SMB, NFS (Network File System) [27], AFP (Apple Filing Protocol) [28] or any other protocol for file level access. The unencrypted data flow case is the usual deployment scenario in the corporate environment where the data network between clients and server is controlled, so the performance impact from traffic encryption can be avoided. Of course the file content could be encrypted by the user (not the ransomware) and it does not affect the tool behavior.

The tool is capable of recovering files that have been edited in multiple connections and from different users in the time period when the network traffic was captured. Based on the estimated time for the beginning of ransomware action, it can take advantage of the typical ransomware behavior in order to recover the file content, using its own actions to defeat it.

The following sections are organized as follows: section II describes the algorithms and data structures in the proposed software for file recovery. Section III is devoted to the evaluation of correctness and completeness in the tool. Finally, section IV presents the conclusions.

## II. Software structure

We assume that the network probe stores all the SMB traffic since the previous backup snapshot, for example in *pcap* format [29]. A ransomware detection algorithm has triggered an alarm, it has identified the destroyed files and the start of the file sharing operations that destroyed the data. However, the detection software could not block the ransomware early enough because several destructive actions were required in order to trigger the alarm [30] [13]. In this section we describe the software structure for a tool that can recover the files that were lost before the ransomware was blocked.

SMB is transported over TCP and each host maintains one TCP connection with the SMB server. This connection transports all file access commands to the shared volumes. Each of these TCP connections can be analyzed individually. However, the target file can be modified by the actions from different users, even simultaneously. Therefore, any file operation, from any connection, must be taken into account in a time ordered manner for the successful reconstruction of the file content. We have not found any previous tool with this capability.

For each TCP connection, SMB traffic cannot be analyzed on a per-packet basis because SMB commands could extend along several TCP segments. Therefore, each of the TCP streams (one for each direction) must be reconstructed. Tools like [31] [32] offer this functionality. For a proof of concept we have implemented a TCP reconstruction function that for each new packet added to any stream in a TCP connection offers the possibility to request the continuous available data in the reconstructed stream in any direction (Figure 2).

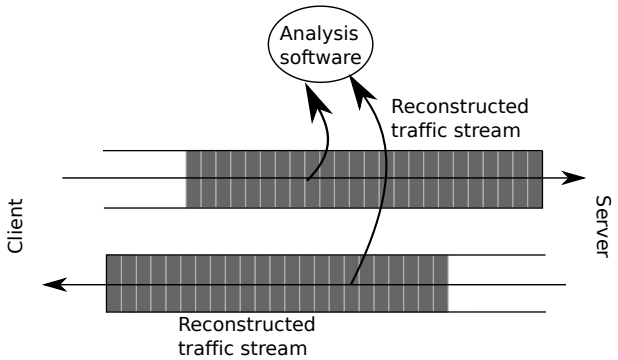


Fig. 2. TCP stream reconstruction API behavior. Data from both streams in the connection are available to the analysis software

The remaining of this section describes the internal software architecture and the specific problems it must solve in order to reconstruct the sequence of SMB commands and finally reconstruct the lost file from the operations on its original data and not on the ransomware encrypted version.

### A. SMB traffic analysis

SMB, used as a file transfer protocol, is a binary protocol based on a Request/Response design, where the client sends the request messages to the file server. Every 'Request' message contains a unique client generated identifier called a 'MessageID'. The 'MessageID' is replicated in the corresponding 'Response' message in order to pair both messages (Figure 3). This protocol characteristic allows the client to queue several request messages in the server (usually 'Read' or 'Write' requests). It also allows the server to avoid Head-of-Line blocking by reordering the responses, sending first whichever data it gets ready. Even if the responses do not arrive at the client in the same order in which it sent the requests, they can be paired using the 'MessageID' field. An example could be the client sending several read operations, for different files, and the server responding to them in a different order, perhaps due to the data being obtained from different hard disks with different response times (see Figure 3 for an example).

Using the above-mentioned functionality, an SMB client implementation must keep a queue with the pending requests sent to the server. On the other hand, a server implementation could continue receiving requests from the SMB session while it waits for the previous requested data to be ready. However, an analysis probe is neither in a client nor in a server position. An analysis probe is in a midpoint in the communication, seeing both requests and responses in the order they cross a network link. The probe is analyzing in parallel the traffic from all the SMB sessions with the same file server, therefore the ordering in processing the messages is relevant to the task of obtaining the final state of the file. Causality is preserved at the monitoring point (no response is seen before its corresponding request), however, depending on the order the TCP stream data is processed in the analysis probe, a 'Response' message could be processed before its request was seen.

In a naive algorithm the monitoring probe could try to follow the Request/Response causality by reading first from the client to server stream in order to see a 'Request' and then read from the server to client stream in order to obtain the 'Response' to the previous message. However, due to the possible reordering in the SMB messages, it could obtain the response to a different request. Figure 3 shows an example of this situation. At time $t_1$ the monitoring probe has seen the first 'Read request', sent by the client at $t_0$. If the probe waits for data in the opposite direction it will not obtain the corresponding response message but the response to the request message sent at $t_2$. Note that this situation is unrelated to any network traffic disorder, as ordering is recovered in the TCP stream reconstruction function. It is due to the application level requests being served in a different order than they were issued.

This situation requires the probe to simulate the queue of client pending requests. In order to reproduce this queue, any data available in the client to server stream must be immediately processed. Whenever data is available in the server to client stream it will correspond to one of the queued pending requests. This queue must be implemented for any concurrent SMB session. In order to describe the proposed data structures we describe first the different types of identifiers used in SMB messages.

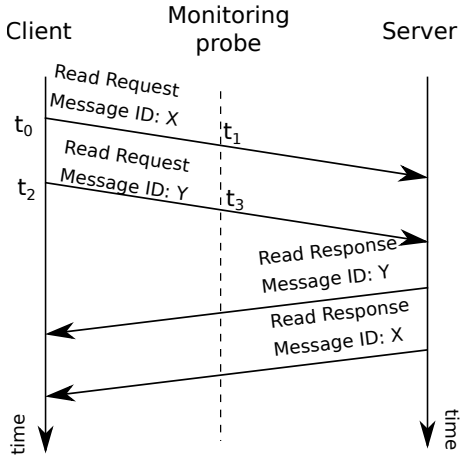


Fig. 3. Example of SMB messages with reordering

*1) SMB identifiers:* In an SMB session, after the initial negotiation messages, the client requests access to a volume using a 'Tree Connect Request' message (Figure 4). The server assigns a 'TreeID' to the session, identifying the accessed volume. Several 'Tree Connect' message pairs could be sent using the same SMB session in order to grant concurrent access to different shared volumes. These messages must be analyzed as they contain the volume path string and the rest of the commands will only specify paths relative to the tree.

The messages to request access to a file (or to 'open' a file) use the 'TreeID' in order to refer to the volume containing the file. The request message is named a 'Create Request' and it is the only message that contains the file name and the relative path. The response message assigns a 'fileID' (or FID in SMB documentation [25]) which identifies the opened file. Subsequent read or write commands are implemented using 'Read Request/Response' and 'Write Request/Response' messages. These request messages contain the 'fileID' but their responses contain only the 'messageID' and the 'TreeID'. The 'TreeID' is required in the responses because the 'messageID' values are local per tree connection. Note that none of the read/write messages contains the file path or name.

*2) Data structures:* Figure 5 shows the data structures used to follow the sequence of SMB messages when multiple TCP connections are present and several files have to be recovered. A list, hash list or hash tree contains the currently established TCP connections. For each connection, a list, named 'TreeList', is maintained. Each element in a TreeList refers to the files accessed after a 'Tree Connect' to the specific shared volume. The only 'Tree Connect' structures stored are those for volumes that contain files that have to be recovered.

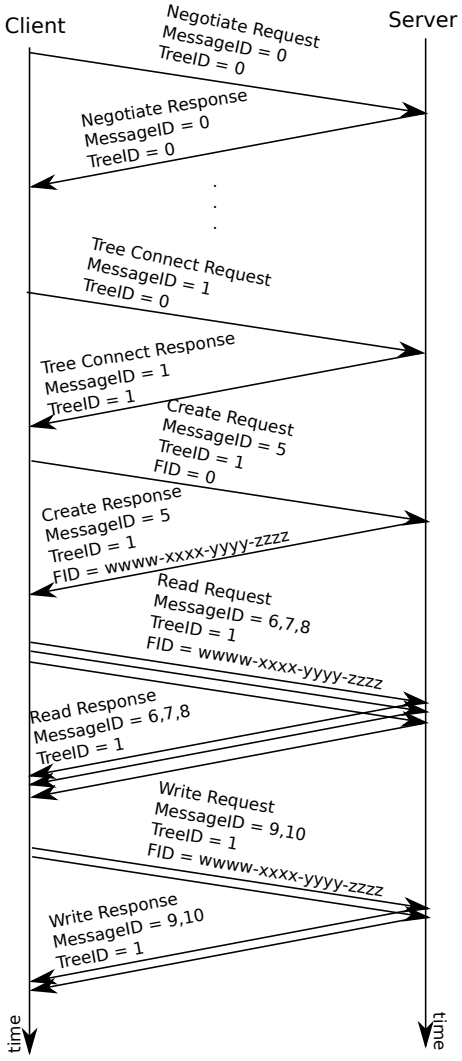


Fig. 4. Example of SMB messages including the different identifiers

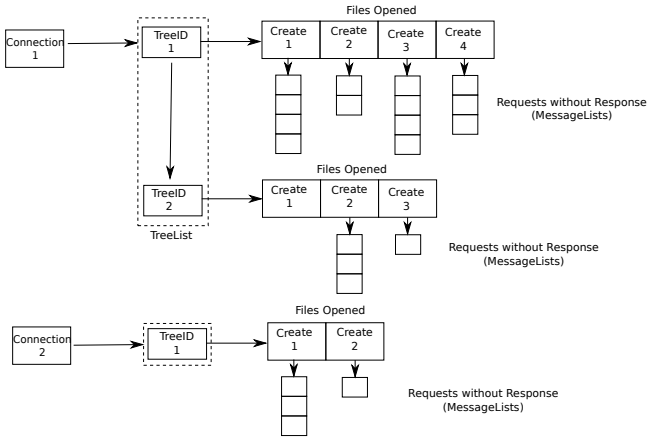


Fig. 5. Data structures for SMB message processing

Every element in a TreeList stores a TreeID and a list of opened files in that tree for that SMB session. Initially, the element for an opened file contains only the information seen in the 'Create Request' message and after processing the response message a MessageList structure is created in order to store the read and write request messages for this file. This MessageList implements the client pending requests queue and its elements disappear when the corresponding response messages are processed.

### *B. File reconstruction*

Several files can be recovered simultaneously, hence a list of target file names (FileList) is implemented. Each element in the list will contain a file path and each one could be located in a different shared volume.

The file content can be obtained from the read and write operations. If any user has opened and read the file then the whole original content is available in the traffic trace. Any posterior file modification can be reconstructed from the write commands (implemented using 'Write Request/Response' messages). Any read or write operation will be considered valid only when the corresponding confirmation is observed in the form of a response. In the case of read operations the data are in the response but the file offset is only in the request. For the write operations, both the offset and the data are in the request but the operation could fail, therefore its data cannot be used until the response has been processed. These operations could come from different users working on the same file and their actions must be correctly interleaved following the temporal order stored in the traffic trace. For example, a later write operation could overwrite the data that was recovered from a previous read operation.

Some ransomware strains read the whole file and create a new one with the encrypted content, then they delete the original file. Other strains overwrite the original file with the encrypted version. In any case, the whole file content is read, therefore it is stored in the traffic trace as payload of file sharing packets and it can be recovered from them. The only requirement is that the ransomware detection algorithm identifies the 'Create Request/Response' where the ransomware started its action on the target file. After this event, any write operation on the file could be due to the ransomware and the data could be the encrypted version, therefore it must no be included in the reconstructed file. However, any other user could still be reading the file. If this new user reads data that have not been overwritten by the ransomware, they are still valid data from the file and they can be used to recover it.

Figure 6 shows an example of the reconstruction procedure. Read operations are marked with an arrow above the time axis, while write operations use an arrow below the axis. The timestamp $t_r$ marks when the ransomware opened the file for its encryption. Any read or write operation before $t_r$ contains data that can safely be included in the recovered file, maintaining its time order. Any write operation after $t_r$ is a suspected ransomware operation, however, any read operation after $t_r$ *on data that has not been overwritten after* $t_r$ is valid data that can be included in the restored file.

As a design option, if no read operations were included in the reconstructed file after $t_r$, we would not recover data that is read by the ransomware or by other software during

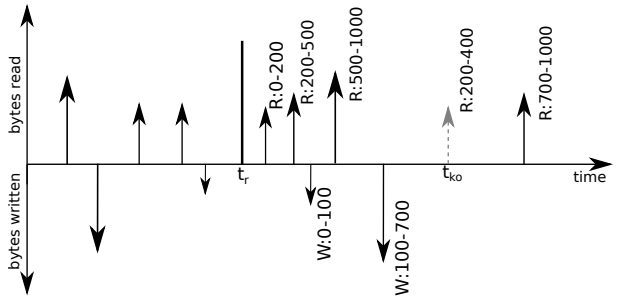


Fig. 6. Example of included and excluded operations in file recovery. Each read (R) and write (W) operation after $t_r$ shows the file offset where they read or write. Data read at $t_{ko}$ is not included in the reconstructed file because it was previously overwritten

ransomware operation. However, if every read operation after $t_r$ was used, then if the ransomware reads what it writes or any other process reads the encrypted version of the file we would recover these data instead of the original one. Hence, for each file to recover, a list of blocks (Figure 7) where the ransomware has possibly written is maintained after $t_r$. This allows the inclusion or exclusion of read operations after $t_r$.

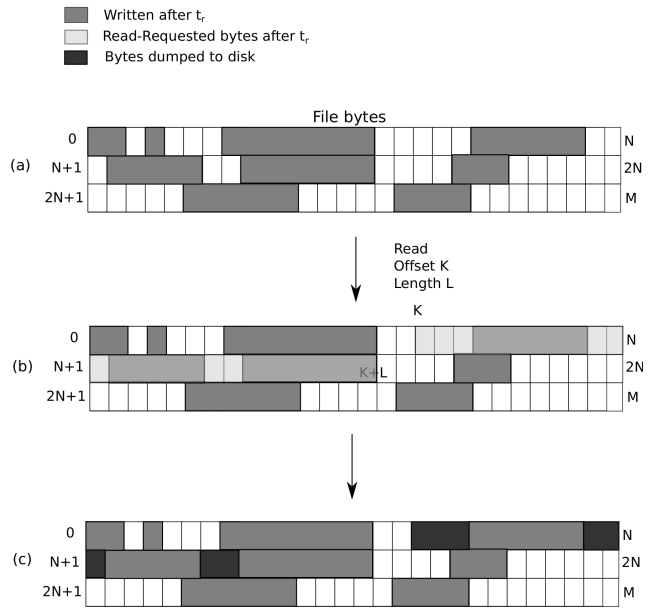


Fig. 7. List of blocks for a file being reconstructed. (a) Some blocks in the file (dark filled) have been overwritten after the ransomware started; (b) a process reads data form the file but some blocks are from those overwritten (light gray filled); (c) only bytes not overwritten (black filled boxes) are included in the reconstructed file

A last event that must be taken into account is the possibility of a rename or moving operation on the target file. We assume that the file path to recover is the original one when the traffic trace starts (from the previous backup). If afterwards any SMB operation renames or relocates the file then the program must take into account this new location. This situations usually take place when the ransomware renames the file to encrypt.

## III. Evaluation and Experimental Results

In order to evaluate the proposed recovery mechanisms we have carried out two different sets of tests: (a) tests of correctness, evaluating different user actions in the traffic and (b) tests of completeness, evaluating whether any file encrypted by a ransomware could be recovered.

For all the tests, the network scenario has been created using several virtual machines (VMs) running in a single host. One of the VMs acts as the SMB file server and the rest act as clients that read or write files shared by the server (Figure 8). The trace from the traffic crossing the virtual switch is obtained using functionalities from the hypervisor [33].

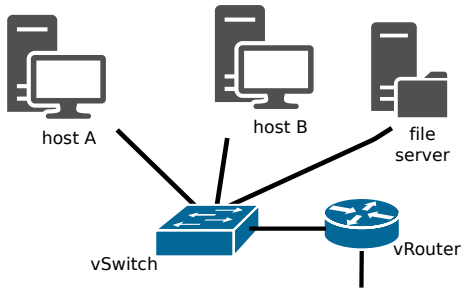


Fig. 8. Testing scenario

For the completeness tests, 54 samples of ransomware from 18 different families were obtained (Table I). Each sample encrypted sets of files in the range of 2 Gbytes to 5 Gbytes, containing thousands of files.

| Family | Versions | Number of samples | Samples with all files recovered |
|---|---|---|---|
| VirLock | VirLock | 1 | 1 |
| CTBLocker | CTBLocker v4.0 | 3 | 3 |
| Teslacrypt | Teslacrypt v3.0 | 1 | 1 |
| TorrentLocker | CryptoFortress | 1 | 0 |
| DMALocker | DMALocker | 1 | 1 |
| Locky | Locky v1.0, Aesir, Odin, Osiris, Diablo6 | 10 | 10 |
| Cerber | Cerber v2.0, Cerber v4.0, Cerber v4.1.6, Cerber v5.0, Cerber v4.1, Red Cerber | 15 | 15 |
| CryptXXX | CryptMIC v5.001 | 1 | 1 |
| Bart | Bart v2.0 | 1 | 1 |
| CryptoMix | CryptFile2, Mole, CryptoShield, Revenge | 10 | 9 |
| Crysis | Crysis, Dharma | 2 | 1 |
| Sage | Sage v2.0 | 1 | 1 |
| MRCR | MRCR1 | 1 | 0 |
| Spora | Spora | 1 | 0 |
| WannaCry | WannaCry v2.0 | 2 | 2 |
| Jaff | Jaff | 1 | 1 |
| Globe | GlobeImposter v2.0 | 1 | 0 |
| Zeus | Zeus | 1 | 1 |

TABLE I
Ransomware samples and success in the recovery

### A. Test of correctness

In order to check the file reconstruction algorithm described in section II-B in a complex scenario, three programs were created that edited the same file. These programs emulate the behavior of two concurrent normal users and

a ransomware in action. Program 'benign1' was run in host A (Figures 8 and 9). It read the first third of the file content. Afterwards, program 'benign2' was run in host B and it read the second third of the file. Next, program 'benign1' wrote inside the first third of the file content and 'benign2' wrote in the second third of the file. Afterwards, programa 'malign' was run in host A. This program emulated ransomware behaviour by reading the whole file and overwriting its content. Finally, both programs 'benign1' and 'benign2' read the whole file content. The start time $t_t$ of ransomware action is identified as the time when program 'malign' opens the file. The program must use any read and write operation previous to that instant and any posterior read operation that is not accessing bytes overwritten after $t_r$. The result is that no data read in the last operations of 'benign1' and 'benign2' should be included in the recovered file, as they are reading the encrypted version of the file.

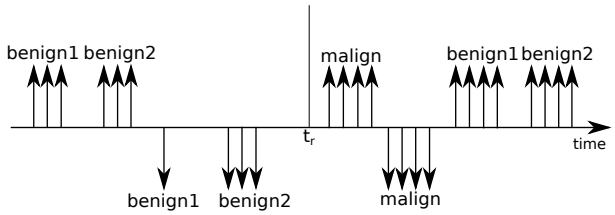


Fig. 9. Test emulating user and ransomware behaviour

The tool successfully recovered the original content of the file, independent of its size.

After this detailed test, a bulk evaluation was conducted. More than 5000 files, filling more than 5 Gbytes of data were copied to and from the shared volume. This generated large traffic traces containing the set of files. All the files were recovered and it was checked that they were binary identical to their original version. This test evaluated the TCP stream reconstruction functionality and the SMB analysis module.

### *B. Test of completeness*

The ransomware samples shown in Table I encrypted whole sets of thousands of files, as no detection algorithm was run. Also, there were no user actions prior or simultaneous to ransomware activity. Therefore, the starting time of ransomware action is the beginning of each traffic trace. The tool was requested to recover all the files in the encrypted volume (5034 files). For 48 of the 54 ransomware samples, *all files were recovered correctly*.

In the 6 remaining samples the tool could not recover the whole content of all the files for the following reasons:

- 'CryptoFortress', 'Spora' and 'GlobeImposter' do not encrypt the whole files. They only encrypt the beginning and the end of each file. The consequence is that the ransomware does not read the whole file, therefore its content is not in the traffic trace. However, as the rest of the bytes are not encrypted they are still in the file and it could be reconstructed using its 'half encrypted' version and the recovered content.
- 'Mole', 'MRCR' and 'Dharma' do not completely encrypt a file when it is large (approximately above 100 Mbytes). This results in the same consequences as in the previous case.

It must be noted that the traces described in Table I are captured in a worst case scenario, as there are no real users. In a deployment scenario the users would have opened and read many of the shared files before the ransomware started, therefore their content would be in the trace and it could be recovered.

In every scenario, if any content is not recovered, it is because it was not read by the ransomware, therefore it was not encrypted and so the data remain unencrypted in the shared volume. Then, the file could be reconstructed by combining the remaining data with the recovered ones.

## IV. Conclusions

This paper has described the design and deployment architecture of a software tool that recovers files lost due to the actions of ransomware in a scenario with network shared volumes.

The software takes all the user actions on the files from the recorded traffic and it reconstructs the file content by incorporating any modifications to the file from user actions.

Based on the information from a ransomware detection tool, it can recover the file from operations previous and posterior to the ransomware action without taking the encrypted content as valid data.

The tool design was evaluated in a proof of concept implementation and it successfully recovered files from test traffic traces but also from the traffic created by the actions of 18 different families of ransomware. It recovered all the files in traces with thousands of encrypted files. It failed to recover the whole content only in those cases where the ransomware did not encrypt the whole file, therefore the remaining content was not lost and it was really not necessary to recover it.

## ACKNOWLEDGMENT

This work was supported by Spanish MINECO through project PIT (TEC2015-69417-C2-2-R).

## References

[1] Ransomware and businesses 2016. Technical report, Symantec Corporation, 2016. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/ISTR2016_Ransomware_and_Businesses.pdf.

[2] Adrien Gendre. Ransomware statistics, 2017. https://www.vadesecure.com/en/ransomware-statistics-2017/, Last access May 9, 2018.

[3] EUROPOL. Internet organised crime thread assessment (IOCTA) 2016. Technical report, Europol - European Police Office, 2016. https://doi.org/10.2813/275589.

[4] Samir Thakkar. Ransomware - exploring the electronic form of extortion. *International Journal for Scientific Research & Development*, 2(10):123–126, jan 2015.

[5] Masarah Paquet-Clouston, Bernhard Haslhofer, and Benot Dupont. Ransomware payments in the bitcoin ecosystem. In *Proceedings of the 17th Annual Workshop on the Economics of Information Security (WEIS)*, June 2018.

[6] A. K. Mauraya, N. Kumar, A. Agrawal, and R. A. Khan. Ransomware: Evolution, target and safety measures. *International Journal of Computer Sciences and Engineering*, 6(1):80–85, jan 2017.

[7] Understanding the depth of the global ransomware problem. Technical report, Osterman Research, Inc., August 2016. https://www.malwarebytes.com/pdf/white-papers/UnderstandingTheDepthOfRansomwareIntheUS.pdf.

[8] onQ Ransomware Edition, 2017. https://quorum.com/resources/onq-ransomware-edition, Last access May 9, 2018.

[9] Tianliang Lu, Lu Zhang, Shunye Wang, and Qi Gong. Ransomware detection based on v-detector negative selection algorithm. In *Proceedings of the 2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, Dec 2017.

[10] Md Mahbub Hasan and Md. Mahbubur Rahman. Ranshunt: A support vector machines based ransomware analysis framework with integrated feature set. In *Proceedings of the 2017 20th International Conference of Computer and Information Technology (ICCIT)*, Dec 2017.

[11] R. Vinayakumar, K. P. Soman, K. K. Senthil Velan, and Shaunak Ganorkar. Evaluating shallow and deep networks for ransomware detection and classification. In *Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sept 2017.

[12] N. Scaife, H. Carter, P. Traynor, and K. R. B. Butler. Cryptolock (and drop it): Stopping ransomware attacks on user data. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 303–312, June 2016.

[13] Manish Shukla, Sutapa Mondal, and Sachin Lodha. Poster: Locally virtualized environment for mitigating ransomware threat. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*. ACM Press, 2016.

[14] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. ShieldFS: a self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC 16*. ACM Press, 2016.

[15] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 3–24. Springer International Publishing, 2015.

[16] Amin Kharraz and Engin Kirda. Redemption: Real-time protection against ransomware at end-hosts. In *Research in Attacks, Intrusions, and Defenses*, pages 98–119. Springer International Publishing, 2017.

[17] Victor Moreno, Pedro M. Santiago Del Rio, Javier Ramos, Jose Luis Garcia Dorado, Ivan Gonzalez, Francisco J. Gomez Arribas, and Javier Aracil. Packet storage at multi-gigabit rates using off-the-shelf systems. In *2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS)*. IEEE, aug 2014.

[18] Paul Emmerich, Maximilian Pudelko, Sebastian Gallenmuller, and Georg Carle. FlowScope: Efficient packet capture and storage in 100 Gbit/s networks. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, jun 2017.

[19] Intel. Data plane development kit. http://dpdk.org, Last access May 9, 2018.

[20] Endace. Introducing EndaceProbe. Technical report, Endace. https://www.endace.com/introducing-endaceprobe.pdf.

[21] http-sniffer, 2012. http://xiaming.me/http-sniffer/, Last access May 9, 2018.

[22] Chaosreader. http://chaosreader.sourceforge.net/, Last access May 9, 2018.

[23] Stephen Deck. Extracting files from network packet captures. Technical report, SANS Institute, December 2015. https://www.sans.org/reading-room/whitepapers/tools/extracting-files-network-packet-captures-36562.

[24] Microsoft Corporation, Common Internet File System (CIFS) protocol. https://msdn.microsoft.com/en-us/library/ee442092.aspx, Last access January 2018.

[25] Microsoft Corporation, Server Message Block (SMB) Protocol versions 2 and 3. https://msdn.microsoft.com/en-us/library/cc246482.aspx, Last access May 9, 2018.

[26] Nicolas Harbour. tcpxtract. http://tcpxtract.sourceforge.net/.

[27] T. Haynes and D. Noveck. Network File System (NFS) Version 4 Protocol. RFC 7530, RFC Editor, March 2015. https://www.rfc-editor.org/rfc/rfc7530.txt.

[28] Apple filing protocol concepts. Technical report. https://developer.apple.com/library/archive/documentation/Networking/Conceptual/AFP/Concepts/Concepts.html.

[29] ntopng high-speec web-based traffic analysis. https://www.ntop.org/, Last access May 9, 2018.

[30] Manaar Alam, Sarani Bhattacharya, Debdeep Mukhopadhyay, and Anupam Chattopadhyay. RAPPER: Ransomware prevention via performance counters, 2018. https://arxiv.org/abs/1802.03909, Last access May 9, 2018.

[31] tcpflow - a tcp ip session reassembler. https://github.com/simsong/tcpflow/wiki/tcpflow, Last access May 9, 2018.

[32] Rafak Wojtczuk et al. Libnids. http://libnids.sourceforge.net/, Last access May 9, 2018.

[33] Network tracing. https://www.virtualbox.org/wiki/Network_tips, Last access May 9, 2018.