Journal of Network and Computer Applications 124 (2018) 14–32

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Ransomware early detection by the analysis of file sharing traffic

Daniel Morato[b,*], Eduardo Berrueta[a], Eduardo Magaña[a,c], Mikel Izal[a]

[a] Public University of Navarre, Department of Automatics and Computing, Campus Arrosadia, 31006, Pamplona, Spain
[b] Institute of Smart Cities, Calle Tajonar 22, 31006, Pamplona, Spain
[c] Naudit High Performance Computing and Networking S.L., Calle Faraday 7, 28049, Madrid, Spain

ARTICLE INFO

Keywords:
Ransomware
Malware detection
Traffic analysis
Network security

ABSTRACT

Crypto ransomware is a type of malware that locks access to user files by encrypting them and demands a ransom in order to obtain the decryption key. This type of malware has become a serious threat for most enterprises. In those cases where the infected computer has access to documents in network shared volumes, a single host can lock access to documents across several departments in the company. We propose an algorithm that can detect ransomware action and prevent further activity over shared documents. The algorithm is based on the analysis of passively monitored traffic by a network probe. 19 different ransomware families were used for testing the algorithm in action. The results show that it can detect ransomware activity in less than 20 s, before more than 10 files are lost. Recovery of even those files was also possible because their content was stored in the traffic monitored by the network probe. Several days of traffic from real corporate networks were used to validate a low rate of false alarms. This paper offers also analytical models for the probability of early detection and the probability of false alarms for an arbitrarily large population of users.

1. Introduction

Ransomware is a type of malware that extorts computer users by locking access to their computers (lockscreen ransomware) or locking access to their files by encrypting them (encryption ransomware, crypto ransomware or cryptoware). During 2016, Europol declared that encrypting ransomware had become “the most prominent malware threat […] for citizens and enterprises alike” (EUROPOL, 2016).

Cryptoware is now recognised as the most profitable malware type in history (Cisco Systems, 2016) and hundreds of millions of dollars are estimated to be extorted to users every year (Symantec Corporation, 2016). A survey conducted in 2016 with 290 organizations from different industries in the United States, Canada, Germany and the United Kingdom found that nearly 50% of them had been victims of a ransomware attack during the previous 12 months (Osterman Research and Inc., 2016). Around 40% of the targets declared having paid the ransom. However, even if the organization pays the ransom, there is no guarantee that it will recover the files.

Not only local files to the infected computer are encrypted. Most organizations employ volume sharing protocols with networked disk arrays in order to make documents easily accessible to the users. All the documents could be in shared volumes, helping the backup procedures and allowing for user profile mobility and team work. However, this also makes them more vulnerable as the shared volumes are usually reachable from any infected computer in the organization. Recent ransomware incarnations include a worm behaviour that tries to spread the infection as much as possible through the local network, to computers reachable through Virtual Private Networks (VPNs) or to random targets in the public Internet (Selvaraj et al., 2017; Lee et al., 2017). Therefore, upon infection discovery the organization must usually stop business until its systems are cleaned and backup images are restored (Mathieu Rosemain and Le Guernigou).

In a volume sharing scenario a single infected host could encrypt a whole networked volume (Sjouwerman, 2017), with a global impact on the organization business. The files must be recovered from the most recent backup. Nightly backups are a common policy (Osterman Research and Inc., 2016) and the main recovery mechanism from a ransomware attack. They are easier to implement in scenarios with centralized volumes shared through a network. Upon suffering a ransomware infection, as much work time is lost as it took to detect the intrusion, because all the documents modified from the previous backup are only in the encrypted volumes. Including the IT (Information Technologies)

* Corresponding author.
E-mail addresses: daniel.morato@unavarra.es (D. Morato), eduardo.berrueta@unavarra.es (E. Berrueta), eduardo.magana@unavarra.es (E. Magaña), mikel.izal@unavarra.es (M. Izal).

https://doi.org/10.1016/j.jnca.2018.09.013
Received 21 February 2018; Received in revised form 21 June 2018; Accepted 21 September 2018
Available online 24 September 2018

personnel work-time for the recovery from the backups, at least 8 h of work per employee are usually lost (Osterman Research and Inc., 2016), and more time could be spent in order to put all the business back on track. If the event reaches the media, the organization must also account for the damage to its public image. Finally, financial and healthcare services are the most frequent and lucrative targets. When the latter are targeted (Cisco Systems, 2016; Lee et al., 2017), losses of lives due to delays in treatments or incorrect medications being administered could result.

Long delays in upgrading or applying critical software patches in these organizations offer great opportunities for attackers. This was the case in the "SamSam" ransomware campaign that affected JBoss servers in the Healthcare industry (Cisco Systems, 2016), as in the recent (May 2017) and extensive "WannaCry" infection (Selvaraj et al., 2017; Lee et al., 2017). These delays are due to the operational costs incurred by each update, as any change must be validated against all the software used in the organization, and deployment of the patch must be planned, usually disrupting work.

In this paper we present REDFISH (Ransomware Early Detection from FIle SHaring traffic), a framework for the detection and blockage of ransomware action when it tries to encrypt files contained in shared network volumes from a NAS (Network Attached Storage). This is a very frequent but unsolved scenario for present malware detection tools. We focus on network volumes shared using Server Message Block (SMB) protocol as it is the most common scenario in an office environment, however, the procedure could be exported to other volume sharing protocols. Our approach does not require the installation of software on any end-host, contrary to the common procedure in an anti-malware software deployment (Continella et al., 2016; Kharraz and Kirda, 2017; Scaife et al., 2016). We show that a network traffic inspection device can analyse traffic to the shared volumes and detect ransomware activity using behavioural patterns. This device and its updates are easier to deploy than updating the whole set of computers in an organization. It monitors every access to the shared volumes and it can program rules in an SDN environment for blocking traffic to the protected volumes from any computer that it detects to be infected. A network traffic inspection device can work outside the traffic path, analysing a copy of the packet traffic, received through a switch port mirror (see Fig. 1). Therefore, it does not introduce any extra delay to the user actions and as it is not installed on the user computer it is not vulnerable to being uninstalled by any malware.

We ran our experiments using more than 50 samples from 19 different ransomware families. All the samples were successfully detected. The results show detection times below 20 s. In more than a 99% percent of the detections, at most 10 files were encrypted before the alarm was raised and access from that computer to the volume could be blocked.

The main contributions of this paper are:

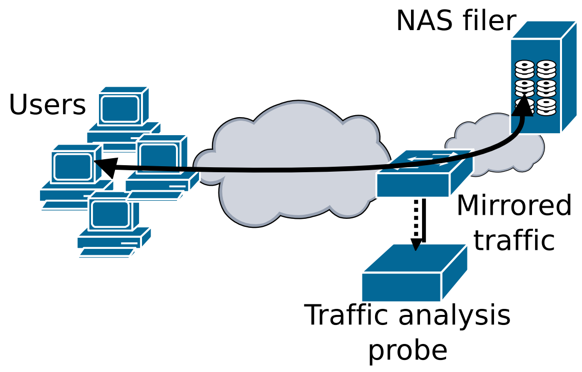


Fig. 1. Networking scenario. The traffic between the hosts and the NAS is replicated from a switch into the traffic analysis probe.

- Present a ransomware detection algorithm based on the analysis of network traffic to shared volumes. Shared volumes is the most common deployment in a corporate environment but no previous proposal has targeted this specific scenario.
- Validate the detection algorithm with 19 different families of ransomware and traffic from corporate networks with thousands of users. We avoid false positives (raising an alarm when no ransomware is present) by tuning the algorithm parameters to the typical behaviour of users and computer programs.
- Provide and validate analytical model approximations for the algorithm success and failure rate.
- Describe a network traffic analysis tool deployment capable of detecting ransomware infection without any software installation at the end-hosts and adding no delay to user actions. The architecture allows also easy deployment of file recovery tools that from nightly backups and network traffic can reconstruct file status before its encryption and destruction.

The remainder of this paper is organized as follows: section 2 describes the previous works in the literature on ransomware detection; section 3 describes the scenario of network shared volumes and the traffic traces used in the analysis. Section 4 describes the algorithm, selects the best configuration for fast and effective ransomware detection and presents analytical models for its analysis. Section 5 compares the results using traffic traces from different user scenarios; section 6 discusses the advantages and disadvantages of the proposed solution and section 7 concludes the paper.

2. Related work

Modern crypto-ransomware infections started in 2013 with CryptoLocker using public-private key cryptography (Ahmadian et al., 2015). It became a global problem in 2016, when more than 1,400,000 Kaspersky users were attacked, most of them by "Locky" and "CTB-Locker". About 22.6% of those users were in the corporate sector (Kaspersky Security Bulletin, 2017). In 2017, "WannaCry" ransomware infected in one day 400,000 machines in more than 150 countries, including the United States and China (Crowe). The adaptation of malware (Scaife et al., 2016) in order to avoid detection has made signature-based detection techniques obsolete (Vidal et al., 2017). They require frequent updates of the malware fingerprints database, and they are incapable of dealing with zero-day infections (Nieuwenhuizen, 2016). The research on ad-hoc ransomware detection procedures has offered in that period several alternatives with better or worse success rates.

Detection techniques capable of coping with 0-day ransomware attacks try to recognise the malware based on more general characteristics than basic software signatures. Some detection frameworks search for encryption primitives in malware code in order to block it or at least warn the user (Kolodenker et al., 2017). In Continella et al. (2016) the authors try to find the expanded AES encryption key in the process memory. This procedure is valid only for ransomware using this symmetric block cipher, although the authors comment that it could be extended to other ciphers. Solutions like the one presented in Kolodenker et al. (2017) try to store all the possible encryption keys being used by any program, therefore, if one of the programs was a ransomware the files could be restored. They require the dynamic inspection of cryptographic calls used by the process. All these alternatives could produce false positive detections for normal encryption software, they take CPU time from the user host and they could fail when the ransomware uses different encryption libraries and algorithms.

The use of decoy files or canary files is a completely different alternative (CryptoStopper, 2017; Feng and Liu, 2017). In those scenarios the anti-ransomware software monitors the modification of files created across the volume. Those files are not user-created documents, therefore no modifications are expected, however, the ransomware will probably try to encrypt those files. When the ransomware deletes one of those canary files, it is detected by the anti-malware tool. This procedure requires monitoring a large amount of files if it wants to detect the malware before many user files are lost, as it cannot guarantee that the ransomware will attack the canary files first. It is also vulnerable to ransomware samples that avoid those files; they could, for example, encrypt first the most recently used files or those from the recent files list in some common applications.

The proposal of using canary files is a precursor and simpler version of the detection frameworks based on monitoring any disk access by user programs. Ransomware activity can be detected based on statistics like the amount of read, written and/or deleted files, the access to many different file types, the search through a large quantity of directories and the creation of files with a much larger entropy than the original file, which is an indicator of possible data encryption. Recent proposals using this approach are (Kharraz and Kirda, 2017; Scaife et al., 2016; Kharraz et al., 2016; Shukla et al., 2016). All these papers are based on Microsoft Windows drivers that intercept I/O (Input/Output) system calls and analyse patterns, using simple linear combinations of measured indicators (Kharraz and Kirda, 2017) or using machine learning techniques (Defeat Ransomware). These algorithms produce good detection results, for example the one presented in Kharraz and Kirda (2017) has 100% ransomware detection rate with 0.8% of false positives (cases where it triggers an alarm when only benign applications were running). However, they take a toll on host performance, with CPU usage in the range of 2.8%–9% depending on the implementation. They produce false positive detection when the user takes actions similar to what a ransomware would do, creating files with high entropy (e.g. compressed files) and deleting files. They are also vulnerable to malware that escalates privileges and uninstalls detection tools (Continella et al., 2016).

Solutions based on network traffic analysis can be deployed on the end-host but also on firewalls or network traffic analysis probes. When installed out of the hosts they are solutions less vulnerable to being removed by the ransomware and they do not consume user computer resources. The traffic they use to detect the ransomware action is the communication that it requires with its command and control (C&C) servers in order to obtain the key for file encryption. This detection procedure fails in case of encryption with a local key without contact to any server; however, those strains of ransomware are infrequent nowadays as they contain the decryption key, which could be extracted by a skilled software analyst. Some ransomware strains try to hide their network traffic by accessing compromised web servers as a proxy to reach their C&C servers. They can be blocked using a list of addresses, which must be frequently updated (Cabaj and Mazurczyk, 2016; Umbrella, 2016). Some ransomware strains try to contact servers using DNS names generated by a DGA (Domain Generation Algorithm) which makes black-listing useless. In Ahmadian et al. (2015) the authors present heuristics for detecting these randomly generated names, however they must adapt their algorithm to many languages in order to detect randomly generated domain names without false positive alarms; moreover, they are vulnerable to generated domain names that are not random but dictionary based.

Commercial anti-virus solutions like (Defeat Ransomware; Umbrella, 2016) try to characterize network traffic using machine learning techniques in order to rise alarms in case of abnormal behaviour. They are prone to false positive detection and therefore they are not typically configured for automatic application blockage and require user validation (Defeat Ransomware), which increases the window of opportunity for ransomware action.

Finally, proposals like (Continella et al., 2016; Kharraz and Kirda, 2017; Shukla et al., 2016; Sophos Intercept X, 2017) intercept file access system calls and offer the added functionality of storing the original version of the files as well as any user modifications. They provide the capability of restoring the original file in case of encryption. However, they are vulnerable to uninstallation by ransomware getting administrator privileges, and they take CPU and hard disk resources.

The above-mentioned methods try to detect a ransomware when it is encrypting files in the user's computer. However, in most enterprise productivity deployments user documents are located in central network shared volumes (Eurostat Statistics Explained). They can be documents shared by groups of users or even the whole set of documents from a user for allowing mobility among hosts. The centralization offers better storage utilization with higher quality disks, group sharing capabilities, easier maintenance and simpler periodic backups. In fact, most enterprises hit by ransomware recover their documents thanks to nightly backups (Osterman Research and Inc., 2016). However, the same centralization and sharing opens the door to a single infected computer encrypting lots of documents with effects on many company departments. Locally installed malware detectors could prevent ransomware from encrypting network shared volumes, however, they require installation and updates on the whole set of company computers. As far as we know, no previous work has tried to detect ransomware action based on the traffic to a NAS system. In this paper we show how a single network probe can detect and stop any ransomware by the analysis of traffic to a network file server. Tens of gigabits per second of sustained traffic are supported, adding file recovery capabilities in order to reduce ransomware impact to a minimum.

3. Network scenario

In a LAN (Local Area Network), NAS volumes are usually enterprise-class disks shared using one or more network protocols over the Internet Protocol (IP). The applications are installed in the local hosts and only the documents are stored in network volumes. We center our case on volumes that store exclusively shared user documents, which can be modified frequently or rarely, but always by user actions. They could be spreadsheets, text documents, images, presentations, etc. We assume that guarded volumes do not contain application configuration files, user profiles, mailboxes, etc. Those are directories that suffer frequent file deletions due to normal application behaviour, but they are not critical.

The algorithm presented in this paper is based on analysing the IP traffic to the NAS appliance. We can get access to this traffic through several monitoring techniques. The algorithm could be implemented in an on-path firewall; however, it would incur in some processing delay (Lin and Lee, 2013). In order to avoid any delay to user traffic we propose an off-path deployment. In an enterprise network, Ethernet switches offer the capability of port mirroring, i.e. duplicating traffic from a network port to another port (SPAN or mirror port) where an analysis probe running REDFISH would be connected. This is the scenario depicted in Fig. 1. It is an off-path deployment where no extra delay is added to the traffic by its analysis (traffic mirroring is implemented in hardware switches at line rate).

Finally, being off-path, the probe cannot block traffic from an infected host as a firewall could, however, it can program discarding rules in an SDN-enabled switch, obtaining the same result.

3.1. Network storage traffic

In a NAS environment, volume access is provided at the file level, being the server called a *filer*. The protocols used are nowadays almost exclusively transported over TCP/IP and the most frequent are NFS (Network File System), SMB (Server Message Block) and AFP (Apple Filing Protocol). NFS is mostly deployed in the UNIX environment while AFP is restricted to macOS computers. SMB, in its several versions (SMB/CIFS, SMB2, SMB3), is the most common protocol for file sharing in the Microsoft Windows environment. The popularity of Microsoft Windows desktops makes ransomware more common for this operating system, even though there are strains for GNU/Linux or even Android (Liviu Arsene, 2016; Emm et al., 2016). In this paper we focus on network traffic to shared volumes using the SMB protocol. This is the default sharing protocol for all versions of Microsoft Windows, which is the *de-facto* desktop operating system in most companies. However, the proposed algorithm for ransomware detection does not require any feature specific to SMB protocol and it could be easily extended to other network file access protocols and versions.

In a NAS environment, SMB is transported over a TCP connection between the user's PC and the filer, using port 445 at the latter (IANA). It is a binary protocol with at least 75 different commands in its version 1 and 19 in its version 2. The most frequent protocol behaviour is a request-response one, although asynchronous notifications do exist (Microsoft Corporation).

We analysed host behaviour on opening, reading, writing and deleting shared files. The proposed algorithm is based on the differences between the traffic from an infected and a not infected host. Some locally installed ransomware detection proposals (Continella et al., 2016; Kharraz and Kirda, 2017; Sgandurra et al.,; Kharraz et al., 2015) intercept disk access API (Application Programming Interface) calls and can distinguish which program is responsible for each file access. The SMB protocol traffic does not offer an identification of the application that originated a file request, therefore we can only distinguish between accesses from different hosts.

Each host client maintains a TCP connection with the server. In case of disk array controllers that offer virtual volumes using different IP addresses, a single host could maintain several TCP connections, one for each server IP address. The algorithm proposed in this paper is based on the analysis of SMB/SMB2 traffic in a single TCP connection. When a single host maintains several connections or there is more than one client or server, the analysis is done in parallel for each TCP connection. In fact, we present results from the analysis of traffic traces with thousands of concurrent SMB sessions. The scalability results presented in section 6 show that a single CPU core on an analysis probe can process 10 Gb/s of this traffic. This is the scenario of an enterprise class dedicated file server for a large company.

{Th}e analysis of SMB traffic in the probe extracts the protocol commands. On the following we describe the SMB2 commands (Microsoft Corporation) that will offer most information to the ransomware detection algorithm. SMB2 is used from Microsoft Windows Vista to Windows 10.

- SMB2 CREATE [Request or Response]: The request command is sent by the client in order to create a file or get access to it. The response message contains the file size, which will be useful in order to detect write commands that overwrite existing data.
- SMB2 READ [Request or Response]: The request command is sent by the client in order to initiate a read operation on an opened file. It contains the amount of bytes to read and the file offset where the operation should begin. The response message contains the bytes read.
- SMB2 WRITE [Request or Response]: The request command is sent by the client in order to initiate a write operation on an opened file. It contains the bytes to write and the file offset where the operation should take place. The response message contains the amount of bytes written.
- SMB2 SET INFO [Request or Response]: Among other options, the request command can be used to mark a file for deletion. The file will be deleted when all handles to the file are closed (Microsoft Corporation).
- SMB2 CLOSE [Request or Response]: The request command is used by the client to close an instance of a file that was opened with a CREATE Request.

Our prototype implementation supports also SMB/CIFS (SMB version 1). SMB1 is commonly used in Microsoft Windows XP systems. The commands have a similar purpose but for brevity we omit their description.

3.2. User behaviour, ransomware behaviour and infected datasets

We have used two types of traffic traces:

- SMB traffic from enterprise offices where no ransomware was in action. We used these traces in order to tune the algorithm for a low false positive behaviour. This means not triggering the alarm when no ransomware is active. We also used these traces to estimate the probability of false alarms triggered by the algorithm.
- SMB traffic from cases of ransomware encrypting files in network shared volumes. We used these traces in order to measure how early the algorithm could detect different strains of ransomware and the amount of files that should be recovered from backups.

In the following subsections we describe the scenario where each traffic trace was obtained and their macroscopic characteristics.

3.2.1. Traffic traces for not infected scenarios

We have been capturing traffic from the Public University of Navarre Internet access link since 2006. We have extended this capture to links from a NAS filer, used internally by most of the non-academic staff. This is a scenario with thousands of office users, hundreds of which access simultaneously the volumes shared from a NAS. All non-SMB traffic has been removed from the traces. We have used an 8 h long traffic capture for the algorithm parameter tuning phase. We call this trace the *Campus0* trace in Table 1. Afterwards, we have validated the results using 6 traces, each one 24 h long, from the same scenario (from *Campus1* to *Campus6* traffic traces). We have extended the validation using a traffic trace captured in an office environment in a large company (*Private*). We present all the validation results in section 5.

Table 1
Traffic traces from not infected scenarios.

	Campus0	Campus1	Campus2	Campus3	Campus4	Campus5	Campus6	Private
Place	University	University	University	University	University	University	University	Large company
Date	2017-01-16	2017-02-22	2017-02-23	2017-02-24	2017-02-27	2017-02-28	2017-03-01	2015-04-24
Duration	8h40min	24 h	24 h	24 h	24 h	24 h	24 h	24 h
Size (Gbytes)	212	469	352	333	383	355	714	1100
Total SMB connections	46609	45414	43716	34864	42593	42162	43543	2717091
Stats for connections longer than 5 min that do READ and WRITE SMB operations								
SMB connections	401	424	391	375	362	369	386	21764
Client hosts	330	327	320	306	302	306	313	4882
Server hosts	1	1	1	1	1	1	1	1677
Avg. Connection duration (hours)	5.93	5.43	5.70	5.60	5.97	5.86	5.86	3.5
Files opened per connection	7640	9437	8589	6115	12224	7717	7870	1863
Mbytes read per connection	91.4	68.6	75.6	65.3	94.9	69.8	82.5	18.4
Mbytes written per connection	294.19	295.6	288.97	237.9	315.1	307.5	306.1	5.2

We show some basic statistics for these traces in Table 1. We have removed any connection that does not contain READ and WRITE SMB commands, as there is no file manipulation in them.

3.2.2. Traffic traces for infection scenarios

We have obtained ransomware samples from Hybrid Analysis and Malware Traffic Analysis. They were uploaded by different users and tagged as ransomware by antivirus tools. Table 2 lists the samples grouped by families. The naming of the samples comes from the same source as the binary files and it cannot be considered entirely reliable, as documented in previous papers (Canto et al., 2008). However, every sample used has been checked for ransomware activity encrypting files.

For running the samples we created a virtualized Windows 7 deployment using a host with an Intel Core 2 Duo CPU E6750 running at 2.66 GHz. We ran a VM (Virtual Machine) acting as the infected user (client) and a second VM acting as the SMB filer (server). The traffic between the VMs is captured in a *pcap* file while the ransomware is running. This emulates the traffic capture done by the analysis probe.

We have created a random file population in the filer. We recover from a snapshot the VM and the volume containing the documents for every new test. For the oldest ransomware samples we used a file population of 1916 files with an aggregated size of 1.88 GB and a maximum file size of 25.6 MB. For the most recent samples (since March 2017) we have used a larger file population whose sizes are obtained from a lognormal distribution with a Pareto tail. This model for file sizes and directory structure is described in Agrawal et al. (2009) and it was obtained from a dataset of over 60,000 Windows computer file system images in a large corporation. We used the default parameters recommended in that paper. The directory structure is also random, imitating a real file-system. The resulting new dataset contains 5138 files with an aggregated size of 5.3 GB, files of a maximum of 838 MB and a tree depth of 8 directories.

Table 2 shows a summary of the more than 50 ransomware samples obtained, clustered in 19 families. Each sample represents a different binary file, run for encryption of the files in our random population. Previous works like (Kharraz and Kirda, 2017; Scaife et al., 2016) present results with hundreds of samples, however, not every ransomware sample encrypts files in network shared volumes. We have removed any sample that does only encrypt local files. This is the reason why famous ransomware families like “TorrentLocker” and “Matrix” are not present in the experimental results. Other ransomware families are not active any longer due to the removal of their C&C servers, so no further experimentation with them is possible.

From the analysis of the traffic between the infected host and the filer we have recognised 3 types of general behaviours in the samples:

- Type I - The malware reads the original file, it creates a new file with a different name or extension and it writes in the new file the encrypted content from the original one. Finally, it deletes the original file and proceeds to the next one.
- Type II - It is similar to type I but the encrypted content is written over the original file, not in a new one.
- Type III - It is similar to type II but after overwriting the content it renames the file, adding an extension specific to the ransomware strain

This classification is similar to the one offered in Scaife et al. (2016), but we have a different view of the ransomware behaviour. We analyse the ransomware from the network traffic it creates instead of having access to I/O disk access system calls.

After analysing the traffic generated by each of these samples of ransomware in action we extract the following common characteristics:

(A) It must read files. This is necessary in order to create the encrypted versions.
(B) It must write files, with a similar amount of bytes to those read. These are the encrypted files. They can be newly created files or it can overwrite existing ones.
(C) It must destroy information, either deleting or overwriting files.
(D) Read and write actions will be close in time. They could even be in parallel.
(E) It tries to do the read, write and delete actions fast.

We have witnessed the above-mentioned behaviour in all the samples we have analysed. Crypto-ransomware can not avoid any of the these characteristics. Reading the files and writing a similar amount of bytes for the encrypted versions is obviously unavoidable (characteristics A and B). It must also delete the information in disk (C) or else there is no point in asking for a ransom. Read and write actions must be close in time (D) because in between the malware is encrypting the data. Both actions can be separated in time if the affected file is large but only to the extent of the available RAM. Finally, if the ransomware were not doing these actions fast (characteristic E) it would not delete many files before a periodic backup takes place, therefore losing much of its impact.

4. Ransomware detection algorithm

We have revised the literature on methodologies for ransomware detection, in order to possibly adapt previous proposals to the network

Table 2
List of ransomware samples.

Family	Versions	Date of appearance	behaviour type	Number of samples
VirLock	VirLock	December 2014	I	1
CTBLocker	CTBLocker v4.0	January 2015	I	3
Teslacrypt	TeslaCrypt v3.0	February 2015	III	1
TorrentLocker	CryptoFortress	March 2015	III	1
DMALocker	DMALocker	January 2016	III	1
Locky	Locky v1.0, Aesir, Odin, Osiris, Diablo6	February 2016	III	10
Cerber	Cerber v2.0, v4.0, v4.1.6, v5.0, v4.1, Red Cerber	March 2016	III	15
CryptXXX	CryptMIC v5.001	April 2016	II	1
Bart	Bart v2.0	June 2016	I	1
CryptoMix	CryptFile2	June 2016	I	10
Crysis	Crysis, Dharma	November 2016	III	1
Sage	Sage v2.0	December 2016	III	1
MRCR	MRCR1	December 2016	III	1
Spora	Spora	January 2017	II	1
WannaCry	WannaCry v2.0	February 2017	I	2
BTCWare	Aleta	March 2017	III	1
Jaff	Jaff	June 2017	III	1
Globe	GlobeImposter v2.0	June 2017	III	1
Zeus	Zeus	May 2018	I	1

shared volumes scenario. In this section we describe their parameters and applicability. Afterwards, we present the new algorithm we propose (REDFISH) and we adjust its parameters, based on a training scenario. Finally, we describe analytical models for both the successful detection and false positives from the algorithm.

4.1. Parameter selection

Table 3 shows the main proposals in the literature and the input parameters they use for ransomware detection. Most of them deploy the detection software locally on the possibly infected computer. They analyse the ransomware behaviour locally. Some of them take as input parameter the network traffic from the host, but we could not find any work on the literature about detecting ransomware action based on the traffic to network shared volumes.

This is the meaning of the columns in Table 3:

- File content: Methods based on file content require access to the byte values read and/or written. They usually compare the original content to the new values overwriting the file. They could also use entropy calculations in order to recognise possibly encrypted content.
  File sharing protocols are a network implementation of Input/Output (I/O) system calls, therefore their traffic offers the file content when the user reads or writes a file. However, we do not consider this parameter for three reasons: a) entropy computation increases the CPU load on the analysis probe, b) entropy increments could be very small if the original file was already encrypted or compressed and c) comparing old and new file content requires a large memory footprint and time consuming searches. This kind of file content analysis in real time at 10 Gb/s or larger speeds is expensive, so any proposal that requires file content analysis is discarded.
- I/O calls: The anti-malware software inspects all the disk-related system calls used by the programs. The analysis software knows when a file was opened or closed, its file path, when bytes were read or written, etc, but no access to the file content is required.
  A traffic monitoring probe that tracks network file sharing protocol messages has access to this kind of information.
- API (Application Programming Interface) calls: The anti-malware software inspects function calls used by suspected ransomware programs. The suspicious functions are usually related to cryptographic operations.
  A traffic monitoring probe has no access to function calls local to the host. Any proposal that requires inspecting software behaviour different from disk access must be discarded in the present scenario.
- File stats: The detection algorithm bases its decision on the file extensions in files created by the ransomware, the file extensions in files it accesses (amount of different extensions and values), etc. The detection uses meta-data about the files accessed by the ransomware.
  This information is available in the file sharing network protocol.
- Canary files: Special files are created in many directories. The anti-malware software monitors any access to these files. A user will not touch those files but ransomware software will try to encrypt them. A network monitoring probe could detect when a canary file is accessed. However, the procedure of creating the canary files in the file sharing volume is not straight forward to implement in the probe. All the proposals that use canary files ((Kharraz et al., 2015; Feng and Liu, 2017)) use also the analysis of API calls, therefore all of them are discarded.
- Binary analysis: The anti-malware software analyses the binary of the program to run, searching for strings or specific function calls.
  This information is not available in any network traffic.
- Network: Control traffic is analysed in order to detect the ransomware. Some methods monitor DNS resolution requests to specific black-listed domains ((Cabaj and Mazurczyk, 2016; Hasan and Rahman, 2017)), they try to detect dynamically generated domain names (Ahmadian et al., 2015; Quinkert et al.,) or they try to recognise the exchange of messages and encryption keys with the command and control servers ((Lu et al., 2017; Hasan and Rahman, 2017)).
  In order to monitor Internet traffic (DNS, traffic to command and control servers) the probe requires access to this traffic. This requires a complex probe deployment in some scenarios, compared to just installing the probe close to the file server. In some scenarios it is almost impossible to accomplish this deployment, for example when the users are spread through hundreds of remote offices and their Internet traffic does not use a single network link.

From this analysis we see that any method based on file content, API calls, canary files or binary analysis must be discarded. Those are the methods that check one or more of the first four columns in Table 3. Only (Cabaj and Mazurczyk, 2016; Quinkert et al.,) are left, however, network traffic information in these papers refers to Internet traffic, which is not available in the scenario under analysis.

I/O calls and file stats are the basic information accessible from the traffic with the file server. Every ransomware we have observed reads the contents of files and writes the encrypted version in a different file in the same directory or over the original content in the same file. We

Table 3
Parameters used in the literature.

Method	File content	API calls	Canary files	Binary analysis	I/O calls	File stats	Network
N. Scaife et al. (2016)	✓				✓		
A. Kharraz et al. (2015)		✓	✓		✓		
D. Sgandurra et al.		✓		✓	✓	✓	
F. Mbol et al. (2016)	✓						
A. Continella et al. (2016)	✓				✓	✓	
M. Shukla et al. (2016)	✓				✓	✓	
A. Kharraz and Kirda (2017)	✓				✓	✓	
R. Vinayakumar et al. (2017)		✓					
M. Alam et al.		✓					
M. M. Ahmadian and Shahriari (2016)		✓				✓	
Y. Feng and Liu (2017)		✓	✓				
K. Cabaj and Mazurczyk (2016)							✓
M. M. Ahmadian et al. (2015)	✓				✓		✓
F. Quinkert et al.							✓
A. Kharraz et al. (2016)	✓				✓		
H. Kim et al. (2017)		✓			✓		
E. Kolodenker et al. (2017)		✓			✓		
T. Lu et al. (2017)		✓			✓	✓	✓
M. M. Hasan and Rahman (2017)		✓		✓	✓		✓

Table 4
Parameters used in REDFISH algorithm.

Parameter	Units	Explanation
N		Minimum number of files deleted
T	seconds	Maximum time interval containing N deleted events
V_{thres}	bits per second	Minimum average read and write speed

will detect ransomware activity based on the destruction of information or the removal of several files in the same period as an intensive bidirectional (read and write) disk access activity is measured. The destruction could come from a delete command or from overwriting the data. We will refer to both cases as a deletion event. The parameters in the algorithm (Table 4), computed from the I/O calls in the file sharing protocols, are:

- N: The minimum number of files deleted in order to trigger the alarm. It must be a small value in order to detect ransomware activity before a large amount of files is deleted, which could incur in more work for their recovery (see section 6 about file recovery). However, if N is too small, the algorithm could confuse normal user action with ransomware activity.
- T (seconds): All the removal events must take place in a time interval of T seconds. Ransomware is expected to delete files quickly, so short values of T will be adequate.
- V_{thres} (bits per second): During the time interval with the N deletion events the throughput reading and writing files must exceed this average speed.

In the following subsections we present the details of the algorithm operation and provide values for parameters N, T and V_{thres} in order to achieve 100% ransomware detection.

4.2. Definitions for algorithm description

As a general framework, let $\{\rho_i\}, i \in \mathbb{N}$ be the discrete-time continuous random arrival process for the events of file read operations; $\{\omega_i\}$ is a similar process for the write operations and $\{\tau_i\}$ for the events of file deletion. A deletion event takes place at the time instant when a file handler is closed for a file where some data was overwritten or a deletion command was issued. We assume that two events cannot take place at exactly the same time. This is always true from the traffic monitor point of view as traffic is received in a serialized manner through the port mirror.

REDFISH algorithm processes the above mentioned events in a time sequential manner. For the file deletion events random process we also define the interarrival time process $\{t_k\}$ where $t_k = \tau_{k+1} - \tau_k,\ k \in \mathbb{N}$. We will use this last definition for the analytical model in the following sections.

For each read, write or delete event we can associate a file system path where the affected file was located. We extract this path from the SMB CREATE command when the file is opened, excluding the file name and extension. We name p_{ρ_i} the path for the file where a read operation took place at ρ_i. In an analogous way we define p_{ω_i} and p_{τ_i}. We call Q the set of all possible file system paths. We assume all operations are from the same shared volume.

We define b_{ρ_i} and b_{ω_i} as the amount of bytes in a corresponding read or write operation. For each file system path p we define $P_p(t)$ (equation (1)) as the cumulative amount of bytes read from files contained in that directory (not in subdirectories). In a similar manner we define $\Omega_p(t)$ (equation (2)) for the written bytes.

$$P_p(t) = \sum_i b_{\rho_i}, \forall i \in \mathbb{N} \mid \rho_i \le t \text{ and } p_{\rho_i} = p \quad P_p(0) = 0, \forall p \in Q \tag{1}$$

$$\Omega_p(t) = \sum_i b_{\omega_i}, \forall i \in \mathbb{N} \mid \omega_i \le t \text{ and } p_{\omega_i} = p \quad \Omega_p(0) = 0, \forall p \in Q \tag{2}$$

For the whole filesystem (the network shared volume) we define $P(t)$ and $\Omega(t)$ as the cumulative amount of bytes read or written in the volume (equations (3) and (4)).

$$P(t) = \sum_{\forall p \in Q} P_p(t) \tag{3}$$

$$\Omega(t) = \sum_{\forall p \in Q} \Omega_p(t) \tag{4}$$

The amount of bytes read by the ransomware in a file system path may differ from the amount written in the same path, but they should be similar, as the encrypted versions of the files will have a similar size to the original ones. In order to reduce the effect from the noise created by normal read and write actions by the user we define $m_p(t)$ as the minimum between the amounts of bytes read and written in directory p (equation (5)). We also define $m(t)$ as the minimum between the amounts of bytes read and written in the whole volume (equation (6)).

$$m_p(t) = \min\{P_p(t), \Omega_p(t)\} \tag{5}$$

$$m(t) = \min\{P(t), \Omega(t)\} \tag{6}$$

For the k-th deleted file ($k \ge N$), the time interval comprising the last N deletion events is $\Delta_N \tau_k$ (equation (7)).

$$\Delta_N \tau_k = \tau_k - \tau_{k-N+1} = \sum_{i=k-N+1}^{k-1} t_i \tag{7}$$

We compute $V[k]$ (equation (8)) as an average speed of read&write operation in the paths where those k files were deleted. $V^*[k]$ is computed as the average speed of read&write operations in the whole volume during those k delete operations (equation (9)). The set of paths where those events took place is $D = \{p_{\tau_i}\},\ \forall i \in \mathbb{N} \mid k - N + 1 \le i \le k$. The amount of bytes for the speed computations can be obtained from the increments in $m(t)$ and $m_p(t)$ for those paths. The time interval when this activity took place goes from the first time $m_p(t)$ got an increment in any of those paths until the last deletion event τ_k. We name τ^* the time when this first increment took place. τ^* is the minimum value among the time values η_i when the corresponding $m_{p_{\tau_i}}(t)$ got incremented, for any of the paths in set D (equation (10)).

$$V[k] = \frac{\sum_{i=k-N+1}^{k} (m_{p_{\tau_i}}(\tau_k) - m_{p_{\tau_i}}(\tau^*))}{\tau_k - \tau^*} \tag{8}$$

$$V^*[k] = \frac{\sum_{i=k-N+1}^{k} (m(\tau_k) - m(\tau^*))}{\tau_k - \tau^*} \tag{9}$$

$$\tau^* = \min_{k-N+1 \le i \le k} \{\eta_i\}, k - N + 1 \le i \le k \tag{10}$$

We can express η_i as shown in equation (11). For each deletion event τ_i the corresponding η_i is the first timestamp when $m_{p_{\tau_i}}(t)$ increases from the value at the previous deletion event τ_j in the same path.

$$\begin{aligned} &\eta_i \in \mathbb{R} \mid \eta_i < \tau_i \\ &\quad p_{\tau_j} = p_{\tau_i} \\ &m_{p_{\tau_j}}(\tau_j) < m_{p_{\tau_i}}(\eta_i) \\ &\ m_{p_{\tau_i}}(t) = m_{p_{\tau_i}}(\tau_j), \forall \tau_j \le t < \eta_i \end{aligned} \tag{11}$$

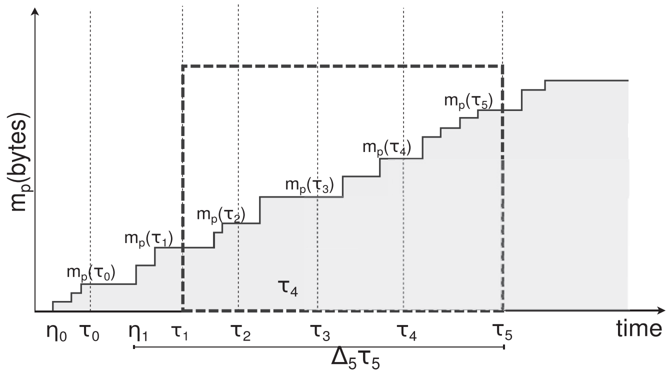


Fig. 2. Example of the increment in algorithm counters.

Based on these definitions, in the following subsections we describe two detection algorithms and analytical models for the estimation of detection and failure rates.

4.3. One-phase detection algorithm: REDFISH1

Benign applications that only read or only write files in the directory will not increment $m_p(t)$. Ransomware increments both $m_p(t)$ and $m(t)$ substantially while encrypting user files.

REDFISH1 algorithm will raise an alarm when files are deleted but only if two additional conditions are true.

The first condition requires the clustering of delete events in time. If $\Delta_N\tau_k > T$ then the events are too far apart to be consistent with ransomware action and an alarm will not be raised at τ_k. If $\Delta_N\tau_k \leq T$ then enough events are clustered together, which could indicate ransomware activity. This is the first necessary condition for raising the alarm.

We require a second condition, based on the speed reading and writing those files. If $V[k] > V_{thres}$ then the alarm is triggered at τ_k.

Fig. 2 shows an example of the events and values during REDFISH1 operation with N = 5. For clarity, all the deletion events shown take place in the same file system path p. The vertical lines mark these events. They take place when the file is truncated or when the file handle is closed for a file that has been marked for deletion using the SET INFO command or for a file whose content has been overwritten (partially or totally). The time series $m_p(t)$ for the selected path is also included in the figure. The time interval $\Delta_5\tau_5$ contains the last 5 deletion events. The time when $m_p(t)$ increased its value from τ_0 is η_1, also included in the figure. In the example, an alarm is raised if:

$$\tau_5 - \tau_1 < T \qquad \text{and} \qquad \frac{m_p(\tau_5) - m_p(\tau_1)}{\tau_5 - \eta_1} > V_{thres}$$

If more system paths contain deletion events then they are included in the computation of the amount of bytes read&written and also in computing the beginning of the time interval using equations (10) and (11).

We include a pseudo-code implementation of REDFISH1 (Algorithm 1).

4.4. Two-phase detection algorithm: REDFISH2

The value of $m_p(t)$ increases only if the encrypted versions of the files are created in the same path as each original file. This is the behaviour we have witnessed in every ransomware sample we have obtained, and it is unavoidable for any ransomware that overwrites the original file. However, a new ransomware sample, aware of REDFISH1 algorithm, could try to avoid detection by creating the encrypted file in a different path, not increasing the value of $m_p(t)$ and therefore showing low values of $V[k]$. This could result in undetected ransomware samples.

We must highlight that we have not found any ransomware sample showing this behaviour. All of them overwrite the original file or create a new one in the same path. However, we propose REDFISH2 in order to cope with potential future ransomware samples that could choose this strategy.

REDFISH2 is an extension to REDFISH1. It keeps the rules from REDFISH1, therefore any ransomware detected by REDFISH1 will be detected by REDFISH2 with the same efficiency. REDFISH2 adds a second set of rules, executed in case of no detection with REDFISH1. It detects ransomware samples that could be overlooked by REDFISH1 due to the same-path condition. To achieve this goal, it uses $V^*[k]$.

$V^*[k]$ provides the read&write speed from this user in the whole shared volume. It is affected by processes that read files in a filesystem path while other processes write files in a different path. Therefore, it can result in a high rate of false alarms. In order to not increase the number of false alarms using REDFISH2, the values for minimum number of files deleted and minimum $V^*[k]$ will be tuned differently.

We define a second triplet of values N^*, T^* and V^*_{thres}. These are equivalent to N, T and V_{thres} but they are only used in the second set of rules in REDFISH2.

The implementation in pseudo-code of REDFISH2 is very similar to the algorithm shown for REDFISH1 (Algorithm 1), so we omit it. First, the checks in REDFISH1 are carried. In case of no alarm, a new set of equivalent checks is carried, this time changing N to N^*, T to T^*, V_{thres} to V^*_{thres} and carrying the computation of *bytesR* and *bytesW* ignoring of the filesystem path.

4.5. Parameter tuning

We tune the values for the parameters (N, T, V_{thres}) in REDFISH1 and (N^*, T^*, V^*_{thres}) in REDFISH2 in order to reduce the amount of false alarms (false positives) and maximize true detections. We proceed first with the parameters for REDFISH1, as they are also used in REDFISH2.

4.5.1. Parameters in REDFISH1

Our goal is the fast detection of every ransomware strain. Our measurement of detection speed will be based on the number of files lost before the ransomware is detected. The parameter N is the minimum number of deleted files before the algorithm can trigger the alarm, therefore a low value of N is recommended. We will show that we detect the ransomware activity in all the samples (100% success rate) with a number of lost files close to the minimum value N, using small values of N (around N = 10).

Anomaly detection systems are prone to false positive alarms (Vidal et al., 2017; Sari, 2015). It is usually considered better to have some false positives than to fail on the detection of a zero-day attack (Liao et al., 2013). However, false positives are dangerous, as they reduce user confidence in the detection system, increasing the possibility that he could ignore a true alarm. Therefore, we aim not only for 100% ransomware detection rate but also for zero false positives. In order to reach this goal we use the eight one-day-long traffic traces we have collected in two different user and network scenarios. To summarize the procedure followed, we take the parameter value space (N, T, V_{thres}) and reduce it to the subset of configurations for which all the ransomware strains are detected and no false alarm would be triggered for the training trace. In order to achieve this objective, for every value of (N, T) we select the value of V_{thres} such that no false positive will be triggered. Within this subset we search for the values that offer 100% ransomware detection with the lowest N value, in order to detect the malware with the minimum number of files lost.

We search for the best combination of parameters using a single training trace. Once we find a suitable operational point, we check the number of false positives using the rest of the traces with normal user behaviour. Even though we select the parameters for no false positives, this will be guaranteed only for the training trace, so we must check,

Algorithm 1 REDFISH1.

```
deleteEvtList ← new empty list of Event(t,minBytes)
bytesR ← new empty dictionary of Bytes(path)
bytesW ← new empty dictionary of Bytes(path)
procedure REDFISH1CHECKFORRANSOMWARE
  path ← path from SMB message
  if type in SMB message = READ then
    bytesR(path) + = bytes read in SMB message
  if type in SMB message = WRITE then
    bytesW(path) + = bytes written in SMB message
  if (type in SMB message = WRITE) or (type in SMB message = REMOVE FILE) or (type
  in SMB message = TRUNCATE FILE) then
    deleteEvent ← new Event
    t in deleteEvent ← time in SMB message
    minBytes in deleteEvent ← min(bytesR(path), bytesW(path))
    add deleteEvent to deleteEvtList
    bytesR(path) ← 0
    bytesW(path) ← 0
    while t in deleteEvent − min(t from all Event in deleteEvtList) > T
do
      remove oldest Event from deleteEvtList
    if number of elements in deleteEvtList = N then
      RWbytes ← sum(minBytes from all Event in deleteEvtList)
      RWspeed ← RWbytes/(max(t in deleteEvtList) − min(t in deleteEvtList))
      if RWspeed > Vthres then
        rise Alarm
    else if number of elements in deleteEvtList > N then
      remove oldest Event from deleteEvtList
```

using other traces, that the parameters provide good results. We detail the results from each step in the following paragraphs.

We apply the detection algorithm to the traffic trace *Campus0* as the training trace, with a value of $V_{thres} = 0$ and a wide range of values for N and T. This is a trace where no ransomware was present. For every (false) alarm situation, we record the measured value of V. Fig. 3 shows the maximum value of V recorded in any alarm for each pair of values (N, T), which we call $V_{thres}(N, T)$. This is the minimum value of V_{thres} which assures that for this traffic trace there will be no alarms, as no one took place with a higher value of V.

High values of T take to more alarm cases in the training trace. The reason behind this is that a user is more likely to delete N files in a larger time interval. Having more alarm cases takes to a larger $V_{thres}(N, T)$, that corresponds to the worst-case alarm. This increase in $V_{thres}(N, T)$ reaches a maximum for values of $T > 30$ s, which depends on N. It is expected that a higher value of $V_{thres}(N, T)$ will take to more cases of real ransomware action that will not reach that throughput, therefore it takes to worse results in positive detection.

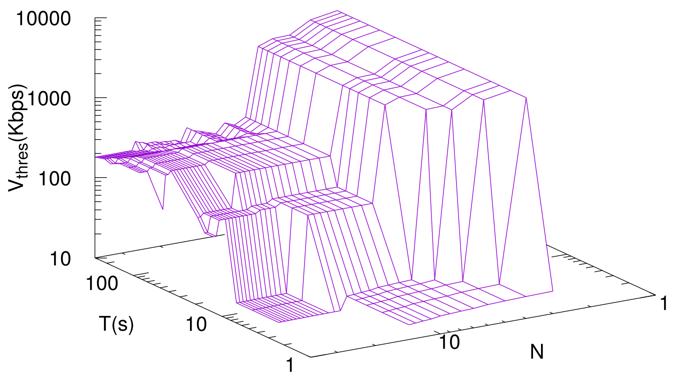


Fig. 3. Maximum measured V, which translates into the threshold value of V that assures no false alarms in the training trace *Campus0*. Values of $V_{thres} = 0$ are not plotted due to the logarithmic scale. A interpolation is added for clarity, even though N takes only integer values.

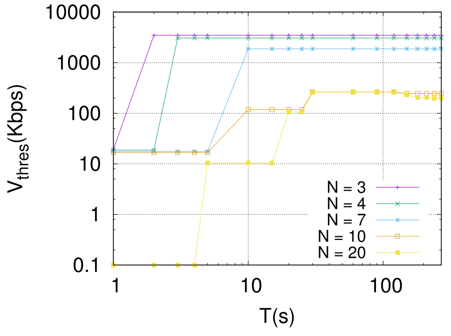
For small values of N it is easier to find cases where the user deletes N files in a short period of time, therefore increasing the number of alarm situations and the value of $V_{thres}(N, T)$. Fig. 4a shows the value of $V_{thres}(N, T)$ fixing the value of N. Values of N larger than 5 are advised; they provide low values of $V_{thres}(N, T)$ which, as will be shown later, offer fast ransomware detection.

Fig. 4b shows the value of $V_{thres}(N, T)$ fixing the value of T. When $N > 7$ the value of $V_{thres}(N, T)$ is always below 3 Mb/s, for any value of T. These rates are below those expected for read or write operation on any disk array in an office environment.

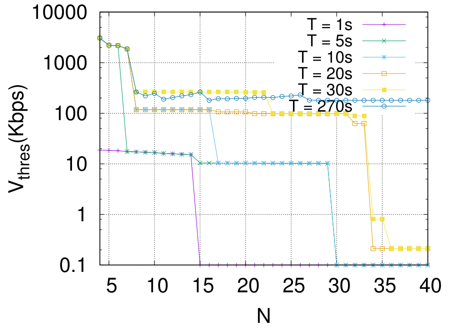
We check the algorithm detection capabilities using the ransomware traces presented in section 3. For every tuple $(N, T, V_{thres}(N, T))$ and every ransomware traffic trace we run the algorithm from the beginning of ransomware action. When the alarm is triggered we measure the amount of files that were lost and the time elapsed until that point. We reset the algorithm on that point and let it continue running with the rest of the trace until another alarm is raised. With this procedure we simulate a situation where the detection algorithm could start running in almost any point of time when the ransomware is already active. With these data we can obtain the number of cases where the minimum number of files N is lost or how much time elapsed on average until ransomware detection.

Fig. 5 shows the percentage of cases where exactly N files were lost before ransomware detection, using a configuration $(N, T, V_{thres}(N, T))$ and the ransomware traffic traces. Having just N deletion events is the best expected result, as the alarm cannot be raised before that amount of files is lost. We have marked using a darker colour in Fig. 5 the configuration cases for which no ransomware sample escapes undetected (100% true positive detection). The worst detection results are obtained when simultaneously a small value of T and a large value of N are configured (lower-left corner in Fig. 5). In this range not all ransomware samples are detected and there is a drop to less than 50% of the cases where we lose only N files before ransomware detection (if it is detected at all). The best results are obtained for large values of both T and N.

For small values of N, Fig. 4b shows that the configured V_{thres} for no false positives is large (tens of Mb/s). This V_{thres} takes to more cases where the ransomware does not reach that throughput with the mini-



(a) Fixing N



(b) Fixing T

Fig. 4. Threshold values of V that assure no false alarms with training trace *Campus0*.

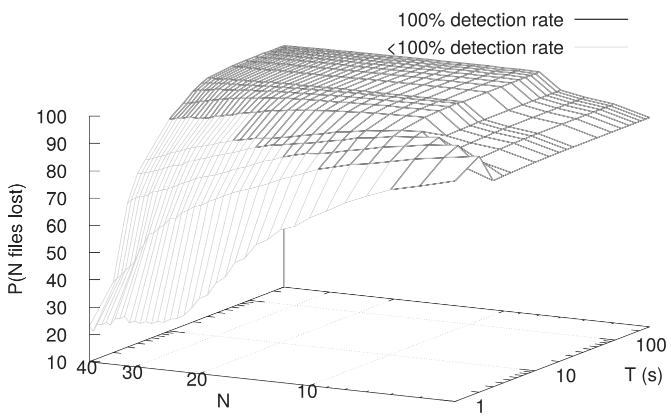


Fig. 5. Percentage of cases where only N files are lost with an algorithm configuration with no false positives with the training traffic trace.

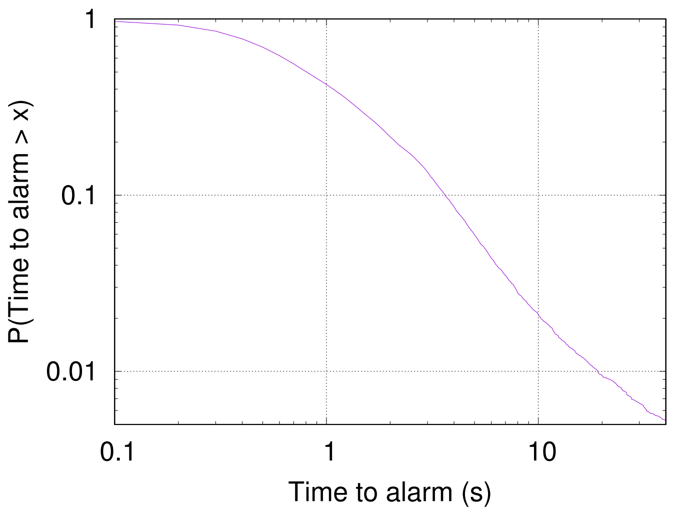


Fig. 6. Complementary cumulative distribution function for the time until the ransomware is detected ($T = 20\,\text{s}, N = 10, V_{thres} = 107\,\text{Kb/s}$).

mum number of files lost, but requires more time to trigger the alarm. For $N \geq 7$ and $T \geq 20\,\text{s}$, the probability of detection with the minimum number of files lost is above 90%, getting above 98% for $N \geq 10$. 100% detection is achieved with larger values of lost files.

After this training process we select N = 10 files and T = 20 s as the reference configuration values for REDFISH1.

N files must be lost before the ransomware is detected. However, this event can take place in a very short time. Fig. 6 shows the complementary cumulative distribution function for the time until the ransomware is detected with a configuration of N = 10 files, T = 20 s and $V_{thres} = 107\,\text{Kb/s}$. For 99% of the cases it takes less than 20 s before exactly 10 files are lost and the alarm is triggered. For 90% of the cases it takes less than 4 s to detect ransomware action. The alarm can be triggered before the time interval T is expired, as long as enough files are destroyed and the disk activity reading and writing is above the threshold.

4.5.2. Parameters added in REDFISH2

All the available ransomware samples are detected using REDFISH1 procedure. However, for the case of ransomware that violates the same-path condition we tune the parameters (N^*, T^*, V^*_{thres}) in the extra security check implemented in REDFISH2. The recommended values for (N, T, V_{thres}) in REDFISH2 are the same as the ones for REDFISH1.

Fig. 7 shows the pairs of (N^*, T^*) that result in no false positives for the training trace when $V^*_{thres} = 0$. For example, using $T^* = 20\,\text{s}$ and $N^* \geq 40$ there are no false positives. The global speed $V^*[k]$ is sensitive to any disk operation from benign software so it is normal that we obtain $N^* > N$. The smallest values of N^* are obtained for the smallest T^*. It is not reasonable to use a value of T^* smaller than T, therefore, $T^* = T = 20\,\text{s}$ is the minimum or optimum value for T. Using $T^* = 20\,\text{s}$ we only require $N^* = 40$ in order to achieve 0 false positives. However, this is only guaranteed for the training trace. Taking into consideration that $V^*[k]$ is very sensitive to normal user actions, it is advisable to select a larger value of N^*. We select to enlarge 25% the minimum value, obtaining $N^* = 50$. We provide in a different section a sensitivity analysis on this parameter. Using $T^* = 20\,\text{s}$ and $N^* = 50$ there is no need for a value of V^*_{thres} larger than 0, however, we select $V^*_{thres} = V_{thres}$ in order to remove possible false positives. The value selected for V_{thres} is small and it will be shown that it does not block the detection of any ransomware.

With this set of parameters, we run REDFISH2 using the ransomware samples in Table 2, but ignoring the path where each read or write operation takes places. This way, only the extra rules in REDFISH2 will be able to detect the ransomware. This simulates a worst case scenario where disk operations from ransomware take place in completely unrelated filesystem paths from each other. The results show 100% detection rate (all ransomware samples are detected). 93.18% of the alarms raised when the 50th file was deleted. For 95% of the cases the alarm was triggered before 63 files were lost and for 99% of the alarms, less than 104 files were lost.

4.6. Analytical model for ransomware detection

In order to measure the effectiveness of the detection proposal we provide an analytical evaluation of both the probability of ransomware detection with minimum file losses (true positives) and the probability

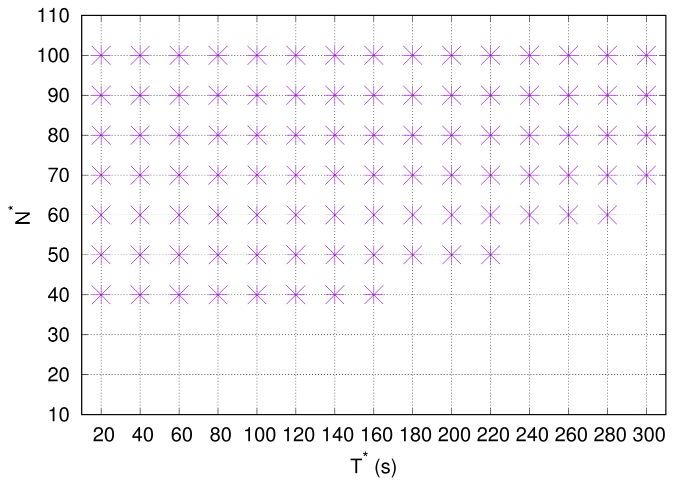


Fig. 7. Combinations of $(N^*, T^*, 0)$ that result in no false positives in REDFISH2 for the training trace.

of false alarms (false positives).

The detection of ransomware is based on two events that must take place simultaneously:

- Event e_T: At least N deletion events take place in less than T seconds.
- Event e_{rw}: The data rate of bytes read&written between first and last deletion event is at least V_{thres}.

If both events happen when the traffic is from normal user activity, we say that a false positive detection took place. If not both events happen when the traffic contains ransomware action, then the algorithm misses an early detection. It can still detect the ransomware but it will take more than N lost files. Obviously we want any of these incidents to be very rare.

Let $\mathbb{P}(e_T)$ be the probability of event e_T from an initial deletion event and $\mathbb{P}(e_{rw} \mid e_T)$ be the probability of V exceeding V_{thres} in the time period when e_T takes place. The alarm rises when both events e_T and e_{rw} take place. This corresponds to probability $\mathbb{P}(e_T \cap e_{rw})$. As an approximation we suppose that both events e_T and e_{rw} are independent, and as an upper bound we take a worst case of $\mathbb{P}(e_{rw}) = 1$. We will later prove that this last hypothesis is reasonable. Then, as shown in equation (12), we can approximate the target probability by the probability of e_T.

$$\mathbb{P}(e_T \cap e_{rw}) = \mathbb{P}(e_T)\mathbb{P}(e_{rw} \mid e_T) = \mathbb{P}(e_T)\mathbb{P}(e_{rw}) \leq \mathbb{P}(e_T) \tag{12}$$

In order for e_T event to take place, at least N files must be deleted in less than T seconds. On the following sections we provide an estimation of the probability of detecting the ransomware right when the N-th file is lost by offering and analytical model for $\mathbb{P}(e_T)$. We also estimate the false alarm probability for a large population of users. This last result is based on REDFISH algorithm and a model for normal user behaviour.

4.6.1. *Probability of ransomware fast detection with minimum file losses*

Ransomware reads a file, writes its encrypted content and removes the original file. The last step could be omitted if the encrypted version overwrites the original file, however the original unencrypted file content is always lost. The files are encrypted sequentially, so a time series of deletion events $\{\tau_k\}$ is created, where the time between these events $\{t_k\}$ is the result of the time it takes to read, encrypt and write each file.

During each time interval t_k the ransomware reads, encrypts and writes a file; therefore this time must correlate to the file sizes, as larger files take more time to transfer from the network file share to the infected host and back.

We use the model described in section 3 for the distribution of file sizes in a typical user disk. File k has size s_k and all the s_k are i.i.d. random variables. The files are distributed across the directory structure and the ransomware typically does some kind of deep breadth first directory tree traversal (Scaife et al., 2016). Each s_k is a combination of a lognormal random variable and a Pareto tail. We approximate the random variable with a lognormal, as it generates more than 99.99% of the values (Agrawal et al., 2009). Equation (13) defines the probability density function (PDF) for the lognormal random variable, based on two parameters μ and σ.

$$f_{LN}(x) = \frac{1}{\sigma x \sqrt{2\pi}} e^{\frac{-(\ln x - \mu)^2}{2\sigma^2}} \tag{13}$$

We estimate the t_k random variable based on the file size to transfer and a constant transfer speed s. The resulting $t_k = s_k/s$ are i.i.d. random variables following a lognormal distribution. The parameter s depends on the transfer speed between the user host and the filer. This parameter is influenced by the available network capacity, user host processing speed, filer file access speed, number of simultaneous users and SMB and TCP protocol parameters (windows sizes, delay acknowledgements, congestion avoidance algorithms, etc.). File access speeds in nowadays enterprise networks are in the range of hundreds of megabits per second. Lower transfer speeds would seriously hamper file sharing users and therefore are infrequent in an enterprise-grade local area network. The proposed detection algorithm could behave worse for lower speeds. Hence, we assume a very conservative worst case range from 10 Mb/s to 100 Mb/s transfer speed per user, while typical enterprise-class network sharing appliances provide several 10 Gb/s network links (Isilon All-Flash Scale-Out NAS Storage; QNAP).

In this scenario, $\Delta_N \tau_k$ is the sum of N lognormal i.i.d. random variables. Its PDF is the convolution of N lognormal PDFs. There is no known closed form for this convolution. The PDF can also be computed from the product of the characteristic function of N lognormal random variables. However, the characteristic function for the lognormal distribution is also not known. The research literature offers many published works with different approximations to the distribution of the sum of N lognormal random variables (Zhang and Song, 2008; Beaulieu and Rajwani, 2004; Nie and Chen, 2007; Lam and Le-Ngoc, 2006). We have selected the systematic procedure offered in Zhang and Song (2008), based on the Pearson's family of functions.

Based on the lognormal parameters recommended from Agrawal et al. (2009), the PDF for the selected distribution is shown in equation (14) (Pearson type-IV), where the parameters u, d, m, and v are computed from the lognormal sample and k is a scaling factor for a total cumulative probability of 1. For brevity, we show only the results from the procedure, and we refer the reader to (Zhang and Song, 2008) for more details.

$$f_{IV}(x) = k\left(1 + \frac{(x+u)^2}{d^2}\right)^{-m} e^{-v\tan^{-1}\left(\frac{x+u}{d}\right)} \tag{14}$$

Fig. 8 shows the probability of not detecting the ransomware when the tenth file is lost, obtained from the above-mentioned approximation. This is the probability that the sum of the N = 10 lognormal random variables results in a value above T = 20 s. The probability is below 0.01 for transfer speeds larger than 12.7 Mb/s. For a network with 100 Mb/s transfer speeds between host and filer the probability of not detecting the ransomware when the tenth file is lost is below 0.001 (0.1%).

We must highlight that this last result does not mean that in 0.1% of the cases the ransomware would not be detected. It only means that it will not be detected when the tenth file is lost but **it will be detected later**. For example, in case the probability of missing the ransomware in the 10th deletion event is 0.001 then the probability of missing it also when the 20th file is lost is at most $1 - (0.001)^2 = 0.999999$, or more than "five-nines", which is a typical quality requirement in *telecom* networks.

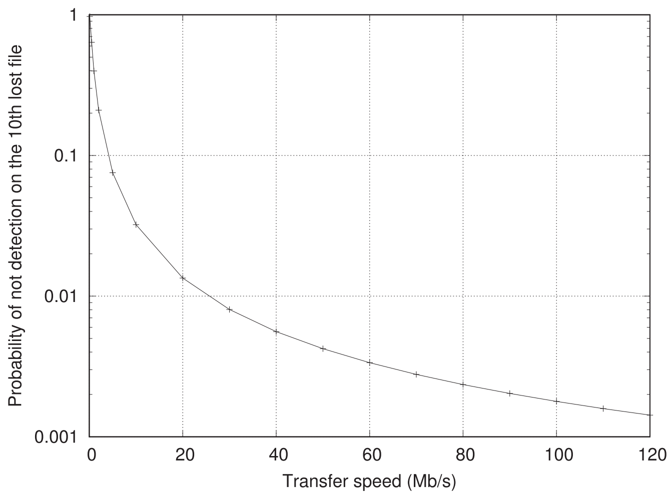


Fig. 8. Analytical results for the probability of fast ransomware detection.

4.6.2. *Estimation of* $\mathbb{P}(e_{rw})$

We have used a worst-case bound for $\mathbb{P}(e_{rw}) = 1$. We show now a better approximation using a similar methodology to the previous section.

Let s (constant) be the transfer speed and s_k be the i.i.d. lognormal random variables describing the file sizes (with lognormal parameters μ and σ). When a program opens a file, it sends a CREATE Request SMB message to the file server. Once the response is received, it can send READ Requests in order to read the file content and later WRITE Requests in order to write the encrypted version of the file. The amount of bytes transferred is $2s_k$. These bytes are transferred at speed s, therefore it takes $2s_k/s$ seconds. Between the CREATE Request and the time when the file starts being received there is a gap due to search times at the file server hard disk (Fig. 9). If the disk is a mechanical drive we can expect access times in the range of several milliseconds. Let t_e the extra time in the whole operation, when no data is being transferred.

When N files are transferred following this procedure we can compute and average transfer speed as shown in equation (15). The numerator is the amount of bytes in the file sizes because REDFISH uses the minimum value between bytes read and written. The denominator takes into account both operations of read and write in the time interval, and an overhead due to N files affected. The event e_{rw} is the event of obtaining an average measured speed above $V_{thres} = 107$ Kb/s when $N = 10$ files are deleted. Therefore $\mathbb{P}(e_{rw}) = \mathbb{P}(v_e > V_{thres})$ and we only need to compute this last value.

$$v_e = \frac{\sum_{k=1}^{N} s_k}{Nt_e + \sum_{k=1}^{N} \frac{2s_k}{s}} \tag{15}$$

We use the definition of v_e and obtain a result that depends on NV_{thres}/s (equation (16)).

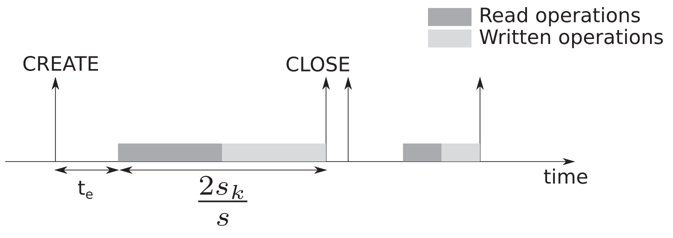


Fig. 9. Waiting times in file transfers.

$$\mathbb{P}(v_e > V_{thres}) = \mathbb{P}\left(\frac{\sum_{k=1}^{N} s_k}{Nt_e + \sum_{k=1}^{N} \frac{2s_k}{s}} > V_{thres}\right)$$

$$= \mathbb{P}\left(\sum_{k=1}^{N} s_k > \frac{Nt_e V_{thres}}{1 + N\frac{2V_{thres}}{s}}\right) \tag{16}$$

The value of s is in the range of several megabits per second, as it correspond to the file transfer speed. We have selected V_{thres} at least an order of magnitude smaller (107 kb/s). Therefore, we are interested in computing $\mathbb{P}(v_e)$ in a range of values for which we can safely say that $2NV_{thres} << s$, therefore $2NV_{thres}/s << 1$ and equation (16) is just approximately equation (17). For $V_{thres} = 107$ Kb/s and $N = 10$ this requires $s >> 2$ Mb/s. We can therefore assume this approximation for transfer speeds above tens of megabits per second.

$$\mathbb{P}(v_e > V_{thres}) \approx \mathbb{P}\left(\sum_{k=1}^{N} s_k > Nt_e V_{thres}\right) \tag{17}$$

Computing this last probability is straightforward. Based on a similar procedure to previous section, s_k are the i.i.d lognormal random variables used for modeling file sizes, therefore $\sum_{k=1}^{N} s_k$ can be approximated using the Pearson type-IV (equation (14)). From our experiments, the value of t_e is in the range of a few tens of milliseconds, however, it can depend on the response times from the hard disk in the server. For $t_e = 50$ ms (which can be considered large) the result is $\mathbb{P}(v_e > V_{thres}) \approx 0.9942$ and for $t_e = 10$ ms the result is $\mathbb{P}(v_e > V_{thres}) \approx 0.999948$. We can therefore assume that for transfer speeds above tens of megabits per second, $\mathbb{P}(e_{rw}) \approx 1$.

4.6.3. *Model for false detection probability*

False positives take place for normal user activity (no ransomware is active) when normal user deletion events cluster close together. In order to estimate $\mathbb{P}(e_T)$ we require a model for the normal user deletion event times.

We take the experimental distribution for the time between deletion events, t_k, from the data in the training trace *Campus0*. Fig. 10 shows the experimental cumulative distribution and a mean-squared-error fit using a Weibull distribution.

A Weibull random variable presents the cumulative distribution function shown in equation (18), where α and β are the distribution parameters.

$$\mathbb{P}(t_k > x) = 1 - e^{-\left(\frac{x}{\beta}\right)^{\alpha}}, x \geq 0 \tag{18}$$

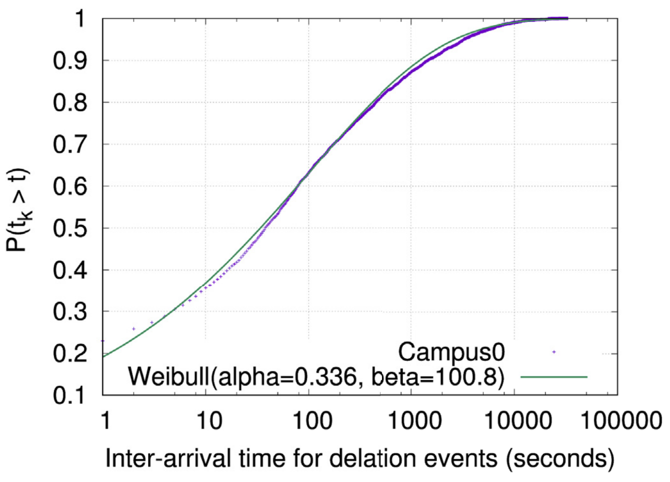


Fig. 10. Cumulative distribution function for the time between events (*Campus0* traffic trace) and a fit using a Weibull distribution.

Based on (Yilmaz and Alouini, 2009) we can calculate the distribution for $\Delta_N\tau_k$ as the sum of N i.i.d. Weibull random variables. We approximate the result using equation (19) from Johnson (1960).

$$\mathbb{P}(e_T) = \mathbb{P}(\Delta_N\tau_k < T) = \mathbb{P}\left(\sum_{i=k-N+1}^{k-1} t_i < T\right)$$

$$\approx 1 - e^{-\left(\frac{p}{\beta}T\right)^\alpha} \sum_{k=0}^{N-2} \frac{1}{k!}\left(\frac{c}{\beta}T\right)^{\alpha k} = 1 - \frac{\Gamma\left(N-1,\left(\frac{c}{\beta}T\right)^\alpha\right)}{\Gamma(N-1)} \tag{19}$$

$\Gamma(n)$ is the gamma function, $\Gamma(n,x)$ is the incomplete gamma function and c is computed using equation (20).

$$c = \frac{\Gamma\left(N-1+\frac{1}{\alpha}\right)}{(N-1)!\,\Gamma\left(1+\frac{1}{\alpha}\right)} \tag{20}$$

The mean-squared-error fit shown in Fig. 10 provides estimated parameter values of $\alpha = 0.336$ and $\beta = 100.8$ for the Weibull distribution. The resulting value of c is 17.82 and $\mathbb{P}(e_T) = 0.000032$. This is the probability of a false alarm for N = 10 consecutive deletion events, measured as the probability that 10 deletion events take place in less than 20 s. For a single user there could exist several opportunities along a single day when the alarm could be raised, simply because he could delete more than N files and each group of N events provides an alarm opportunity. If a single user deletes on average M files in a single day, we can estimate a maximum number of alarm opportunities $opts = \min\{0, \lfloor M-N+1 \rfloor\}$. Assuming independent events, the probability of one or more false alarms raising in a single day for one user (p_{user}) is just the result of equation (21).

$$p_{user} = \mathbb{P}(\text{alarms for single user 1 day} \geq 1) = 1 - (1-\mathbb{P}(e_T))^{opts} \tag{21}$$

We have estimated M from the behaviour of more than 300 users in the training trace *Campus0*. The result is $p_{user} = 0.00022$. Then, for a population of 300 independent users the probability of at least one false alarm during a whole day would be $p_{day} = 0.065$ or 6.5%. Therefore, the expected time until the first false alarm would be around 15 days.

5. Experimental results

Parameter tuning for ransomware detection has been accomplished using the whole set of ransomware samples presented in Table 2. For N = 10 files lost in a time interval of T = 20 s, using $V_{thres} = 107$ Kb/s, we obtained 100% ransomware detection with REDFISH1, with 99% of the cases where the ransomware is detected immediately when the tenth file is lost. In 99% of the cases the alarm raises less than 20 s after ransomware action begins. These results have been obtained aggregating the behaviour from all the ransomware samples. However, there could be ransomware families offering better results at the expense of other families that could present worse detection rates. In order to detect this possible behaviour we analyse in this section each family separately.

Even though no false positives are possible for the training trace they could appear in other days or other network scenarios, due to changes in user behaviour. In this section we check the results using other traces in the same environment and also using a large traffic trace from a real private business scenario.

5.1. Detection of different ransomware families with REDFISH1

Fig. 11 shows, for each ransomware family, the percentage of cases where the ransomware is detected with exactly N files lost, for a range of values around N = 10. For each value of N, a different V_{thres} is configured, according to the results from Fig. 3.

Most families present a slow improvement when N increases, reaching optimal detection with percentages above 90% for $N > 10$. However, a few families show a steep improvement on that vicinity. For

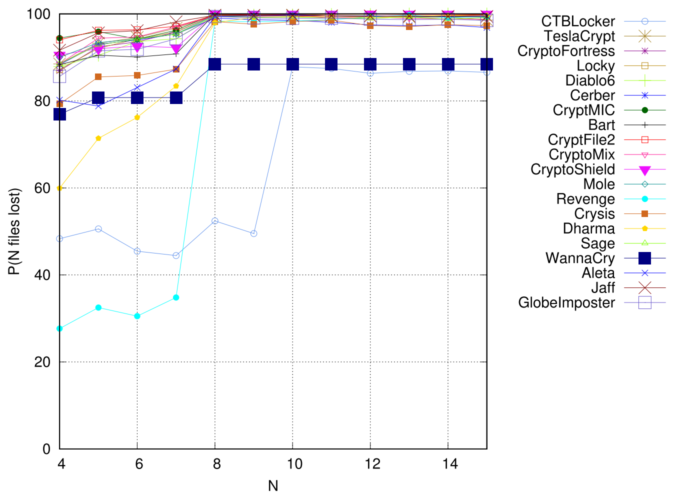


Fig. 11. Percentage of cases where only N files are lost for T = 20 s and each ransomware family.

$N < 10$, "Revenge" and "CTBLocker" families offer only 50% of cases where the ransomware is detected with only N files lost. For $N \geq 10$ they raise to 85% for "CTBLocker" and above 95% for "Revenge".

"WannaCry", for T = 20 s, presents a different behaviour, with low sensitivity to changes in parameter N in the range from 4 to 16 files. The encryption-and-deletion sequence for this family of ransomware is slightly different from the others. Instead of encrypting and then deleting each file, it batches encryption and deletion events. After encrypting a batch of around 200 files, it deletes all of them before starting a new batch. The result is percentages of fast detection for a configuration with $N > 10$ slightly above 85%.

Increasing T to 120 s we can observe in Fig. 12 an improvement in "WannaCry" detection, getting to percentages above 95%. This is at the expense of slightly worse results for "CTBLocker". For "CTBLocker", using T = 20 s and N between 4 and 16 files lost, only in 70% of the cases it is detected with exactly N files lost.

From Figs. 11 and 12 it is not clear whether in the rest of the cases the ransomware is not detected, it is detected after a few more files are lost or after many more files are lost. To clarify this point we show in Fig. 13 the maximum number of files lost in 99% of the alarms. For most families, in 99% of the cases the ransomware is detected before 2xN files are lost. For "CTBLocker", 99% of the alarms raise before 80

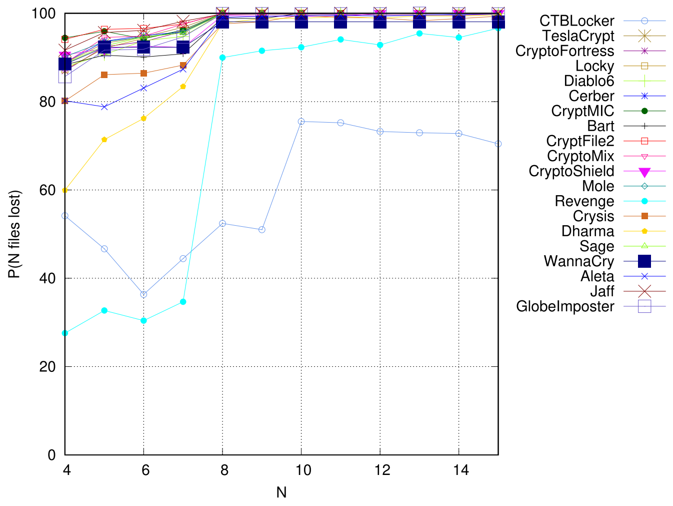


Fig. 12. Percentage of cases where only N files are lost for T = 120 s and each ransomware family.

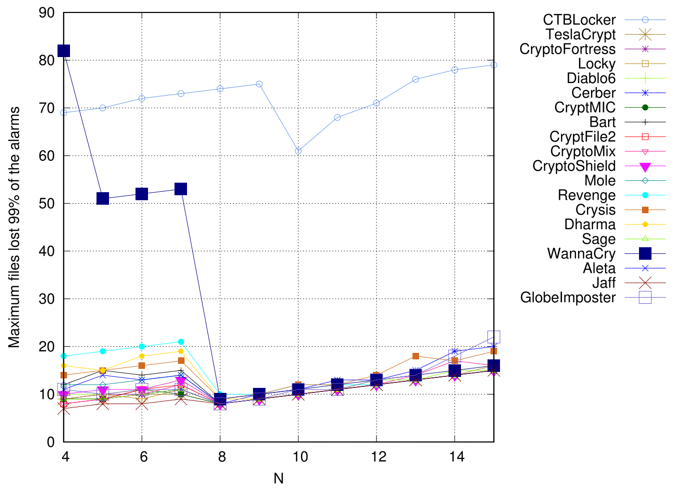


Fig. 13. Maximum number of files lost for 99% of the alarms. T = 20 s.

files are lost, with an average of around 20 files (the average is not shown in the figure). Therefore, the results are still good in absolute numbers. We will show in section 6 that those files are recoverable from the traffic trace.

The explanation for the results from “CTBLocker” comes from its special behaviour encrypting files. It does not only encrypt but it also compresses the file. Disk-write activity is therefore much lower than read activity and when the read&write speed threshold is high it results in harder detection. This diversity in the behaviour of different ransomware families serves as an additional validation of the algorithm effectiveness.

5.2. Validation of REDFISH1 on different user scenarios

As shown in section 4, the combination of parameters T = 20 s, N = 10 and V_{thres} = 107 Kb/s provides the best configuration using small values. However, the results could be very sensitive to small changes in user activity. We test the results using different traces and a small set of parameters, in order to detect whether for other scenarios they are a good choice or not.

Table 5 shows, for each traffic trace, the number of false positives for each configuration. It also shows the value of parameter N that results in no false positives. The number of false positives is very small, even for the *Private* traffic trace. We must highlight that this last trace contains the activity from 4882 users, accessing a total of 1677 different SMB servers during a whole day (24 h). It even includes any late-night automatic processes that could access the shared volumes.

For the *CampusX* traffic traces there is at most one false positive per day. The results are as good for T = 120 s as for T = 20 s. For the trace *Campus1* the larger time interval T = 120 s allows a false alarm to appear that does not exist for T = 20 s. In *Campus6* the contrary happens and an alarm that raises for T = 20 s is not present for T = 120 s. This is probably due to the different V_{thres} for each configuration.

For the *Private* traffic trace there are at most 3 false alarms in the whole day (using T = 120 s). They are due to:

- Database file modifications: Files from database software are usually overwritten when changes are applied. If several database files are modified by the same user in a short time period, an alarm could raise. This happens only once in the whole day because a user modifies 10 database files in less than 2 min.
- Modifications to several Microsoft Excel files in a short time: This is probably due to files linked among themselves. It happens twice in the whole day, both cases for the same user.

These 3 false alarms are the only alarms during a whole day. During this same day the users open collectively more than 40 million files (Table 1). This an extremely low rate of false positives as only about 1 in 10 million opened files triggers a false alarm.

For T = 120 s and $N > 12$ there are no false positives in the *Private* traffic trace. For the *CampusX* traffic traces this is achieved using a different configuration for each day. In the worst case we can guarantee that using N = 61 files lost **there are no false positives any day**.

5.3. Validation of REDFISH2

REDFISH2 extends the logic in REDFISH1 and obtains the same detection results for all the available traces. It presents an advantage only for still non existing ransomware behaviour. We have carried a sensitivity analysis varying the parameters exclusive to REDFISH2 and adapting the available ransomware samples for not triggering the alarm in the part of the algorithm that is shared with REDFISH1. This analysis shows whether REDFISH2 could detect these samples or not, how many files would be lost and the number of extra false positive alarms that could arise.

Table 6 shows the number of false positives for each of the samples in the two different network scenarios. It also shows (column “N lost”) the percentage of alarms that are triggered using the ransomware samples right when the N^*th file is lost. We have varied the parameter N^*, keeping the values of T^* and V^*_{thres}. For all the selected values we achieve 100% detection rate, i.e. all the ransomware samples are detected using exclusively REDFISH2 second set of rules. Even the traffic trace *Private* (containing the traffic from 4882 users) does not show any extra false positive result. The percentage of alarms that stop the ransomware when exactly N^* files are lost is lower the larger the value of N^*. This means that a very large N^* is not advised, as both the alarm cannot be triggered with less than N^* files lost and the number of cases with this optimal results gets reduced the larger the value of N^*.

We can state that REDFISH2 contains a low cost modification of the logic in REDFISH1, keeping its good results and adding only a small

Table 5
False positive events for different configurations in REDFISH1.

T	N	V_{thres} (Kb/s)	Number of false positives						
			Campus1	Campus2	Campus3	Campus4	Campus5	Campus6	Private
20	10	107	0	1	0	0	0	1	2
20	12	107	0	1	0	0	0	1	2
20	14	107	0	1	0	0	0	1	1
20	16	107	0	1	0	0	0	1	0
20	21	107	0	0	0	0	0	1	0
20	61	107	0	0	0	0	0	0	0
120	10	263	1	1	0	0	0	0	3
120	12	263	1	1	0	0	0	0	1
120	14	263	1	1	0	0	0	0	0
120	21	263	1	0	0	0	0	0	0
120	61	263	0	0	0	0	0	0	0

Table 6
False positive events and ransomware detection rates for different configurations in REDFISH2.

N^*	Number of false positives ($T^* = 20$ s, $V^*_{thres} = 107$ Kb/s)							N^* lost	N^* (99%)
	Campus1	Campus2	Campus3	Campus4	Campus5	Campus6	Private		
50	0	1	0	0	0	1	0	93.18%	104
60	0	1	0	0	0	1	0	90.43%	175
70	0	1	0	0	0	0	0	89.6%	213
80	0	1	0	0	0	0	0	87.07%	291

risk of extra false alarms with a proper configuration. As no existing ransomware presents a behaviour that escapes REDFISH1 detection, the alarms from REDFISH2 could be reduced to a "*Warning*" category, with a reduced impact of any false positive.

5.4. Validation of analytical models

In section 4.6.1 we provided an approximated model for the probability of ransomware detection when the Nth file is lost. The result depends on the distribution of file sizes accessed from the shared volume and the transfer speed between each client and the filer.

We have configured a testbed similar to the one described in section 3.2.2, which was used to obtain the ransomware samples from Table 2. We have improved the computing hardware in order to support a third virtual machine, acting as an Ethernet bridge between the infected client and the filer. In the bridge we configured a rate limiting traffic policy in order to modify the transfer speed. Using this testbed we ran a sample of "Cerber", still active, varying the transfer speed and obtaining the detection results from REDFISH1 using the optimal parameters.

Fig. 14 shows the analytical results (the same as shown in Fig. 8) compared to the experimental ones. Both curves are reasonable close and the analytical one provides a worst case bound, as the curve is always above the experimental one. The analytical model takes some approximations for small values of transfer speed, however, it still seems valid for values as small as 2 Mb/s. The maximum rate limiting values we could safely configure in the experimental testbed were around 50 Mb/s. We must highlight that it is a rate per user. Therefore, in a real deployment where for example 300 users reach 50 Mb/s each one, we are loading the filer with 15 Gb/s. Therefore, the validation range is reasonable.

In section 4.6.3 we provided an approximated model for the probability of false positives. It required a model for normal user behaviour, that we obtained from the traffic traces. In order to validate this result we have shown that in the seven traffic traces we have available, only a

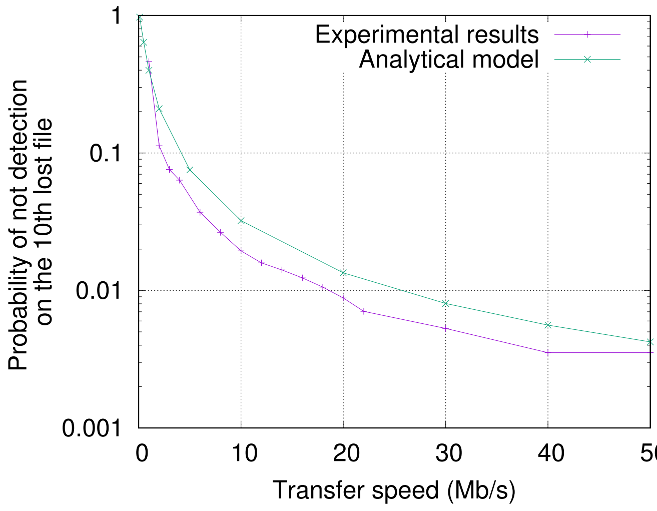


Fig. 14. Analytical results for the probability of fast ransomware detection.

few false positives appear. The available data for validation is short, as each full day of traffic provides only one sample. The analytical result we have offered was computed for a population of about 300 users. As the traffic trace *Private* contains the traffic from 4882 users we can assume that it is equivalent to approximately 16 one-day traces from 300 users each. In this set of 16 traces we obtain 2 false positive results. With these results we test the null hypothesis that false positive detection is a Bernoulli trial with probability $p = 0.065$. An observed sample of 2 false positives out of 16 experiments has a statistical significance (or p-value) of $\alpha = P(Binomial(16, 0.065) \geq 2) = 0.27$. Thus with a confidence level of 73%, we can accept the hypothesis that the analytical model is correct. This confidence interval could be improved with a larger number of traffic traces or longer traces (several days long).

6. Discussion

6.1. Comparison to previous works

As explained in section 4, there are no methods in the literature directly applicable to the file sharing scenario. All of them, either require information not available in the network traffic or require expensive computation in a scenario with several gigabits per second of traffic from a large population.

In order to compare the success and failure results from REDFISH to previous proposals, we show in Table 7 the most relevant performance results. We have focused on only a few metrics, as all of them are difficult to compare due to differences in the scenario. For example, many ransomware samples become inactive after their command and control servers are blocked, so the comparison against the same samples is usually impossible. Also, not all the papers offer clear data about all these metrics.

The percentage of ransomware samples detected is the most clear result. It is in a range from 90% to 100%. However, as shown in the sixth column, the algorithms have been tested with very different number of ransomware families. (Alam et al.,; Continella et al., 2016; Kharraz and Kirda, 2017) offer 100% detection rate, but they have tested with 1, 11 and 29 different ransomware families respectively, which offers very different degrees of reliability. REDFISH has been tested against 19 different ransomware families, available since 2014. It has been designed for 100% detection rate, which it achieves.

The metric of false positives is based on experiments with normal users. Different populations of users and applications are recorded in each paper, so comparisons are again difficult. In general, the percentage of false alarms for the whole population of clean measurements is offered. In this scenario, REDFISH offers 0 or 1 false positive result in traces with more than 300 users for a whole work-day. In a trace containing thousands of users in a corporate environment for a whole day it reported only 2 false alarms. No previous work has validated false positive results against such large scenarios.

The number of files lost before ransomware detection varies among different algorithms. The best results are usually offered by those architectures where the lost file can be recovered from a backup. REDFISH is in this category.

Finally, most of the previous works are based on anti-malware software installed on the user computer, so they incur in CPU or disk usage.

Table 7
Comparison to other methods in the literature.

Method	% detected	% False positives	Files lost	Overhead	Samples/families
N. Scaife et al. (2016)	–	–	10	10 ms	492/14
A. Kharraz et al. (2015)	–	–	–	–	1359/15
D. Sgandurra et al.	93.3%	1.6%	–	–	582/11
F. Mbol et al. (2016)	>90%	0.05%–0.25%	–	–	1/1
A. Continella et al. (2016)	100%	0.038%	Recovered	1.8–3.8 ×	383/11
M. Shukla et al. (2016)	–	–	<20	$1-2\%$	(unknown)
A. Kharraz and Kirda (2017)	100%	0.8%	5	$7-9\%$	677/29
R. Vinayakumar et al. (2017)	98%	<1%	–	–	755/7
M. Alam et al.	100%	≈0%	Recovered	–	1/1
M. M. Ahmadian and Shahriari (2016)	98%	Varies	–	–	8/(unknown)
Y. Feng and Liu (2017)	100%	–	0	“low”	1/1
K. Cabaj and Mazurczyk (2016)	–	–	–	–	332/1
M. M. Ahmadian et al. (2015)	100%	0%	0	–	20/14
F. Quinkert et al.	96%	–	0	–	1/1
A. Kharraz et al. (2016)	96.3%	0%	–	–	2121
H. Kim et al. (2017)	–	–	Recovered	–	–
E. Kolodenker et al. (2017)	–	–	Recovered	–	107/20
T. Lu et al. (2017)	95%	$7-8\%$	–	–	1000/(unknown)
M. M. Hasan and Rahman (2017)	97%	3%	–	–	360/20
REDFISH	100%	1 out of 15 days	Recovered	0%	54/19

The metrics vary among papers but on this respect we must highlight that REDFISH adds no load to these computers, as it runs in an isolated network appliance.

6.2. Security, performance and file recovery

REDFISH ransomware detection system does not add any delay to user traffic, as the analysis probe works off-path, receiving a copy of the traffic. Ransomware detection takes a few seconds and in 99% of the cases it can detect the ransomware before 10 files are lost. When the alarm is triggered the probe can use SDN control mechanisms (e.g. OpenFlow) to configure blocking rules for the user traffic, denying any further access to the filer. The alarm can be collected in a Network Management System (NMS) and trigger the adequate administrative actions in order to remove the ransomware and allow further access to the disk array. All the actions related to the detection and blocking procedures can be taken remotely, accessing the network probe configuration, without any host access.

No software must be installed in any user host. No modification is needed in the filer or the network disk array. Only the analysis probe must be added to the network, receiving a copy from the traffic to the filer network port and allowing access to the SDN enabled switch in order to block traffic from infected users. Also, no malware that infects the hosts can uninstall the detection software as it runs in a separate machine (the analysis probe). This probe is configured in the management network, not accessible from local or remote hosts.

We have evaluated the false positive probability and it is extremely low after a correct parameter configuration (one in ten million opened files). Although we suggest blocking any further access to the filer from the suspected host, different options are possible. If the filer allows privilege modification or it resorts to some kind of external database system for this purpose, then a read-only mode could be configured when a user is suspected to be infected. This is a less intrusive mode for the case of the extremely rare false positives, while still stopping ransomware action.

The *Private* traffic trace contains more than 1Tbyte of traffic and it reaches more than 400 Mb/s of sustained traffic. Our prototype implementation of the algorithm is capable of analysing the trace in less than 1/32th of its duration, using a single CPU core in an Intel i5-4690 running at 3.5 GHz. This means that more than 10 Gb/s of traffic can be processed in near real-time. In contrast to other tools (Continella et al., 2016; Kharraz and Kirda, 2017; Kharraz et al., 2016; Shukla et al., 2016; Sgandurra et al.,), it does not analyse the content of the read and written files, therefore much higher speeds are achievable. High traffic network scenarios (tens of gigabits per second) can be supported and even higher traffic could be analysed including more CPU cores. No CPU cycles are taken from any user host, compared to locally installed antivirus software.

Common network traffic analysis software and hardware like HPCAP (Moreno et al., 2014), FlowScope (Emmerich et al., 2017), DPDK (Intel) and EndaceProbe (Introducing EndaceProbe) can store the traffic in hard disk while sustaining the analysis of 10 Gb/s links. They usually take a circular buffer approach and delete the stored traffic after a certain period or when the available disk space is low. Using REDFISH, when ransomware is detected the traffic traces for that user could be marked for not deletion. Afterwards, a network administrator could recover the lost files from the stored traffic. This is possible because the ransomware must read the file before creating the encrypted version. Once the infected host is located, the lost files can be recovered from the SMB READ commands. This is what a plugin for Wireshark already does (Deck, 2015). Therefore, even if some files were lost before the ransomware was detected they can be recovered from the traffic stored at the network probe. The recovered files are not out-of-date backups but the exact file version that the ransomware encrypted and removed. Therefore, 0 lost files can be achieved.

6.3. Robustness analysis

The design of REDFISH is based on intrinsic characteristics to any ransomware. It is based on their reading and writing activity, as well as the act of deleting files or overwriting content. Ransomware cannot accomplish its goal without any of these three actions. However, it could try to appear as a benign application, reducing its activity below the thresholds that distinguish benign from malign software actions. We provide in this section a critic analysis of any strategy that we could imagine where a ransomware, having perfect knowledge about the algorithm, could avoid detection.

REDFISH establishes a threshold on the read&write activity. REDFISH1 quickly recognises ransomware that overwrites the original file or creates the encrypted version in the same filesystem path. REDFISH2 is insensitive to changes in file path while it is kept in the same filer. However, if ransomware keeps its activity below V_{thres} it would remain undetected. The problem in this ransomware strategy is that the recommended value for V_{thres} is very small, around 100 kb/s. If the malware keeps its activity below this threshold then it cannot destroy so much data before a new periodic backup takes place. For example, if nightly backups are implemented then the maximum amount of file data that the ransomware could read, encrypt and destroy would be around

1 GByte (24 h of activity at 100 kb/s). It must also be smart enough to destroy recently modified files first, or else it could be destroying files that have not been modified since the previous backup. This strategy also offers the users more time to notice the ransomware, simply by human detection of encrypted files. This weakness exists but the strategy to use it reduces greatly the impact the ransomware could have.

The ransomware could overwrite only a fraction of the whole file. There are already samples that follow this behaviour, although all of them are currently detected by REDFISH. For some file formats it is enough with a small modification in the file to render it unusable. If the amount of bytes overwritten was small, the ransomware could destroy many files while keeping its read&write footprint below the threshold. REDFISH2 can reduce its second threshold V^*_{thres} to 0, increasing only slightly the number of false positives but forcing this type of ransomware to keep not only their disk access speed below the threshold but also the number of files they delete. However, using $T^* = 20$ s and $N^* = 50$ the ransomware could remove almost 150 files per minute and stay undetected if it modifies only a few bytes from each file. The effectiveness of this strategy depends on whether the files are really rendered unrecoverable by small modifications, which would depend on the file types in the shared volume. For example, it would be easy to recover any data not overwritten from a plain text file, therefore, again the effectiveness in this strategy is low.

If the user computer has access to more than one shared volume, from different filers, file operations to one volume and to another would be carried in different TCP connections transporting SMB commands. REDFISH, as it has been described, analyses each TCP connection independently. The ransomware could read and delete files from one volume while writing the encrypted versions to a different volume, trying to void increasing the read&write speed as it is measured. This would certainly work but the scenario where it could be applied is quite restrictive (several filers) and REDFISH could be easily extended in order to consider any SMB session from the same IP address (which identifies the user).

Finally, the ransomware could read and delete files from the network shared volume while keeping the encrypted versions local to the infected host. This behaviour could still be detected by REDFISH2 by using $V^*_{thres} = 0$. Writing in the same volume as the original file is a safer strategy for the ransomware, as the encrypted files occupy mostly the same as the original files, so they fit in the volume. However, moving them to a different disk (which also happens in the previous attack proposed) has the risk of filling the destination disk, therefore not being able to continue operation.

Although we cannot prove that the strategy described in REDFISH will work for any possible ransomware behaviour, we are confident that any devised strategy to avoid detection will come at a cost of reduced ransomware effectiveness, therefore the implementation of REDFISH provides a clear advantage. The capability of file recovery from the stored network traffic, alone, already provides benefits from the deployment of the analysis probe.

7. Conclusions

We have described and analysed REDFISH, a detection algorithm for strains of ransomware that encrypt files in network shared volumes. The algorithm works with a copy of the traffic, without any effect on normal user activity. It detects the ransomware based on its basic behaviour of reading, writing and removing files.

We have shown how the algorithm parameters can be tuned in order to obtain 100% ransomware detection with 19 different ransomware families. We have shown by experimentation and analytical modeling that in more than 99% of the cases the ransomware is detected before 10 files are deleted. These files can be recovered from the traffic that the network probe is storing, therefore a no-losses scenario is achieved.

False positives (triggering the alarm without real ransomware action) are extremely rare: only one false alarm for 10 million files opened, read and written in a real business scenario with more than 4800 users working for a whole day and accessing more than 1500 network shared volumes. The results are consistent with the analytical model provided for the false positive rate.

The algorithm can be implemented in 10 Gb/s network traffic probes, using a low number of CPU cores, without any impact on host CPUs because no software is installed in them. The whole detection system is out of the production network and it cannot be attacked by any form of malware that could deactivate it.

Acknowledgements

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

References

Agrawal, N., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H., 2009. Generating realistic impressions for file-system benchmarking. ACM Trans. Storage 5 (4), 1–30, https://doi.org/10.1145/1629080.1629086.

Ahmadian, M.M., Shahriari, H.R., 2016. 2entfox: a framework for high survivable ransomwares detection. In: 2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC), pp. 79–84, https://doi.org/10.1109/ISCISC.2016.7736455.

Ahmadian, M.M., Shahriari, H.R., Ghaffarian, S.M., 2015. Connection-monitor & connection-breaker: a novel approach for prevention and detection of high survivable ransomwares. In: 2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC). IEEE, pp. 79–84, https://doi.org/10.1109/iscisc.2015.7387902.

Alam, M., Bhattacharya, S., Mukhopadhyay, D., Chattopadhyay, A., RAPPER: ransomware prevention via performance counters, CoRR abs/1802.03909. http://arxiv.org/abs/1802.03909.

Beaulieu, N., Rajwani, F., 2004. Highly accurate simple closed-form approximations to lognormal sum distributions and densities. IEEE Commun. Lett. 8 (12), 709–711, https://doi.org/10.1109/lcomm.2004.837657.

Cabaj, K., Mazurczyk, W., 2016. Using software-defined networking for ransomware mitigation: the case of CryptoWall. IEEE Netw. 30 (6), 14–20, https://doi.org/10.1109/MNET.2016.1600110NM.

Canto, J., Dacier, M., Kirda, E., Leita, C., 2008. Large scale malware collection: lessons learned. In: Proceedings of the 27th International Symposium on Reliable Distributed Systems (SRDS).

Cisco Systems, 2016. 2016 Midyear Cybersecurity Report. Tech. rep. Cisco Systems, https://www.cisco.com/c/dam/m/en_ca/never-better/assets/files/midyear-security-report-2016.pdf.

Continella, A., Guagnelli, A., Zingaro, G., Pasquale, G.D., Barenghi, A., Zanero, S., Maggi, F., 2016. ShieldFS: a self-healing, ransomware-aware filesystem. In: Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC 16. ACM Press, https://doi.org/10.1145/2991079.2991110.

Crowe, J., WannaCry ransomware statistics: The numbers behind the outbreak. https://blog.barkly.com/wannacry-ransomware-statistics-2017. (Accessed 9 January 2018).

CryptoStopper, 2017. WhatchPoint's Solution for Ransomware Detection. https://www.watchpointdata.com/cryptostopper/ (Accessed January 2018).

Deck, S., December 2015. Extracting Files from Network Packet Captures. Tech. rep. SANS Institute, https://www.sans.org/reading-room/whitepapers/tools/extracting-files-network-packet-captures-36562.

Defeat Ransomware with Varonis and NetApp, Tech. rep., Varonis. https://labs.mwrinfosecurity.com/assets/resourceFiles/mwri-behavioural-ransomware-detection-2017-04-5.pdf.

Emm, D., Unuchek, R., Garnaeva, M., Liskin, A., Makrushin, D., Sinitsyn, F., 2016. IT Threat Evolution in Q3 2016. Tech. rep. Kaspersky Labs, https://securelist.com/files/2016/11/KL_Q3_Malware_Report_ENG.pdf.

Emmerich, P., Pudelko, M., Gallenmuller, S., Carle, G., 2017. FlowScope: efficient packet capture and storage in 100 Gbit/s networks. In: 2017 IFIP Networking Conference (IFIP Networking) and Workshops. IEEE, https://doi.org/10.23919/ifipnetworking.2017.8264852.

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

Eurostat Statistics Explained. Digital economy and society statistics - enterprises. http://ec.europa.eu/eurostat/statistics-explained/index.php/Digital_economy_and_society_statistics_-_enterprises. (Accessed 9 January 2018).

Feng, B. L. Yun, Liu, Chaoge, 2017. Poster: a new approach to detecting ransomware with deception. In: 38th IEEE Symposium on Security and Privacy. https://www.ieee-security.org/TC/SP2017/poster-abstracts/IEEE-SP17_Posters_paper_26.pdf.

Hasan, M.M., Rahman, M.M., 2017. Ranshunt: a support vector machines based ransomware analysis framework with integrated feature set. In: 2017 20th International Conference of Computer and Information Technology (ICCIT), pp. 1–7, https://doi.org/10.1109/ICCITECHN.2017.8281835.

Hybrid Analysis. https://www.hybrid-analysis.com/. (Accessed January 2018).

IANA. Service name and transport protocol port number registry. http://www.iana.org/assignments/port-numbers. (Accessed January 2018).

Intel. Data plane development kit. http://dpdk.org. (Acessed January 2016).

Introducing EndaceProbe. Tech. rep., Endace. https://www.endace.com/introducing-endaceprobe.pdf.
Isilon All-Flash Scale-Out NAS Storage. Isilon F800 Specification Sheet. Tech. rep., DELL EMC. https://www.supernaeyeglass.com/ransomware-defender.
Johnson, L., 1960. Gmr Reliability Manual. Tech. rep. General Motors Research Labs.
Kaspersky Security Bulletin, 2017. Overall Statistics for 2016. Tech. rep. Kaspersky Labs, https://kasperskycontenthub.com/securelist/files/2016/12/Kaspersky_Security_Bulletin_2016_Review_ENG.pdf.
Kharraz, A., Kirda, E., 2017. Redemption: real-time protection against ransomware at end-hosts. In: Research in Attacks, Intrusions, and Defenses. Springer International Publishing, pp. 98–119, https://doi.org/10.1007/978-3-319-66332-6_5.
Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., Kirda, E., 2015. Cutting the gordian knot: a look under the hood of ransomware attacks. In: Detection of Intrusions and Malware, and Vulnerability Assessment. Springer International Publishing, pp. 3–24, https://doi.org/10.1007/978-3-319-20550-2_1.
Kharraz, A., Arshad, S., Mulliner, C., Robertson, W.K., Kirda, E., 2016. Unveil: a large-scale, automated approach to detecting ransomware. In: USENIX Security Symposium.
Kim, H., Yoo, D., Kang, J.S., Yeom, Y., 2017. Dynamic ransomware protection using deterministic random bit generator. In: 2017 IEEE Conference on Application, Information and Network Security (AINS), pp. 64–68, https://doi.org/10.1109/AINS.2017.8270426.
Kolodenker, E., Koch, W., Stringhini, G., Egele, M., 2017. PayBreak: defense against cryptographic ransomware. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security - ASIA CCS - 17. ACM Press, https://doi.org/10.1145/3052973.3053035.
Lam, C.-L., Le-Ngoc, T., 2006. Estimation of typical sum of lognormal random variables using log shifted gamma approximation. IEEE Commun. Lett. 10 (4), 234–235, https://doi.org/10.1109/lcomm.2006.1613731.
Lee, M., Mercer, W., Rascagneres, P., Williams, C., 2017. Player 3 Has Entered the Game: Say Hello to ‘WannaCry’. http://blog.talosintelligence.com/2017/05/wannacry.html. (Accessed 9 February 2018).
Liao, H.-J., Lin, C.-H.R., Lin, Y.-C., Tung, K.-Y., 2013. Intrusion detection system: a comprehensive review. J. Netw. Comput. Appl. 36 (1), 16–24, https://doi.org/10.1016/j.jnca.2012.09.004.
Lin, P.-C., Lee, J.-H., 2013. Re-examining the performance bottleneck in a NIDS with detailed profiling. J. Netw. Comput. Appl. 36 (2), 768–780, https://doi.org/10.1016/j.jnca.2012.12.009.
Liviu Arsene, A.G., January 2016. Ransomware, a Victims Perspective. Tech. rep. Bitdefender, http://www.bitdefender.com/media/materials/white-papers/en/Bitdefender_Ransomware_A_Victim_Perspective.pdf.
Lu, T., Zhang, L., Wang, S., Gong, Q., 2017. Ransomware detection based on v-detector negative selection algorithm. In: 2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), pp. 531–536, https://doi.org/10.1109/SPAC.2017.8304335.
Malware Traffic Analysis. http://www.malware-traffic-analysis.net. (Accessed January 2018).
Mathieu Rosemain, J.D., Le Guernigou, Yann. Renault, Nissan European operations deal with global cyber attack. http://www.autonews.com/article/20170513/OEM01/170519882/renault-nissan-european-operations-deal-with-global-cyber-attack. (Accessed 31 January 2018).
Mbol, F., Robert, J.-M., Sadighian, A., 2016. An efficient approach to detect torrentlocker ransomware in computer systems. In: International Conference on Cryptology and Network Security. Springer, pp. 532–541.
Microsoft Corporation. File system behavior in the Microsoft Windows environment. Tech. rep., Microsoft Corporation. http://download.microsoft.com/download/4/3/8/43889780-8d45-4b2e-9d3a-c696a890309f/File%20System%20Behavior%20Overview.pdf.
Microsoft Corporation. Server Message Block (SMB) Protocol versions 2 and 3. https://msdn.microsoft.com/en-us/library/cc246482.aspx.
Moreno, V., Rio, P.M.S.D., Ramos, J., Dorado, J.L.G., Gonzalez, I., Arribas, F.J.G., Aracil, J., 2014. 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, https://doi.org/10.1109/hpcc.2014.81.
Nie, H., Chen, S., 2007. Lognormal sum approximation with type IV Pearson Distribution. IEEE Commun. Lett. 11 (10), 790–792, https://doi.org/10.1109/lcomm.2007.070842.
Nieuwenhuizen, D., 2016. A Behavioural-based Approach to Ransomware Detection. MWR Labs Whitepaper, Tech. rep. MWR Labs, https://info.varonis.com/hubfs/docs/whitepapers/en/Varonis-Ransomware-Whitepaper-Netapp.pdf.
Osterman Research, Inc, August 2016. Understanding the Depth of the Global Ransomware Problem. Tech. rep. Osterman Research, Inc., https://www.malwarebytes.com/pdf/white-papers/UnderstandingTheDepthOfRansomwareIntheUS.pdf.
QNAP, TVS-EC2480U-SAS-RP R2. https://www.qnap.com/en/product/tvs-ec2480u-sas-rp. (Accessed January 2018).
Quinkert, F., Holz, T., Hossain, K.S.M.T., Ferrara, E., Lerman, K., RAPTOR: ransomware attack predictor, CoRR abs/1803.01598. http://arxiv.org/abs/1803.01598.
Sari, A., 2015. A review of anomaly detection systems in cloud networks and survey of cloud security measures in cloud storage applications. J. Inf. Secur. 06 (02), 142–154, https://doi.org/10.4236/jis.2015.62015.
Scaife, N., Carter, H., Traynor, P., Butler, K.R.B., 2016. Cryptolock (and drop it): stopping ransomware attacks on user data. In: 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), pp. 303–312, https://doi.org/10.1109/ICDCS.2016.46.
Selvaraj, K., Florio, E., Lelli, A., Ganacharya, T., 2017. WannaCrypt Ransomware Worm Targets Out-of-date Systems. https://blogs.technet.microsoft.com/mmpc/2017/05/12/wannacrypt-ransomware-worm-targets-out-of-date-systems/. (Accessed 9 February 2018).
Sgandurra, D., Muñoz-González, L., Mohsen, R., Lupu, E.C., Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. CoRR abs/1609.03020. http://arxiv.org/abs/1609.03020.
Shukla, M., Mondal, S., Lodha, S., 2016. 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, https://doi.org/10.1145/2976749.2989051.
Sjouwerman, S., 2017. New Ransomware CryptoFortress Encrypts Unmapped Network Shares. https://blog.knowbe4.com/new-ransomware-cryptofortress-encrypts-unmapped-network-shares. (Accessed 9 February 2018).
Sophos Intercept X, 2017. https://www.sophos.com/en/products/intercept-x.aspx. (Accessed 29 June 2017).
Symantec Corporation, 2016. Ransomware and Businesses 2016. Tech. rep. Symantec Corporation, http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/ISTR2016_Ransomware_and_Businesses.pdf.
Umbrella, 2016. The Cisco's Software to Increase Protection against Ransomwares. Tech. rep. Cisco Systems, https://www.cisco.com/c/dam/m/hr_hr/training-events/2017/cisco-connect/pdf/Introducing_Cisco_Umbrella_for_cloud_based_threat_protection.pdf.
Vidal, J.M., Orozco, A.L.S., Villalba, L.J.G., 2017. Alert correlation framework for malware detection by anomaly-based packet payload analysis. J. Netw. Comput. Appl. 97 (1), 11–22, https://doi.org/10.1016/j.jnca.2017.08.010.
Vinayakumar, R., Soman, K.P., Velan, K.K.S., Ganorkar, S., 2017. Evaluating shallow and deep networks for ransomware detection and classification. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 259–265, https://doi.org/10.1109/ICACCI.2017.8125850.
Yilmaz, F., Alouini, M.-S., 2009. Sum of Weibull variates and performance of diversity systems. In: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing Connecting the World Wirelessly - IWCMC'09. ACM Press, https://doi.org/10.1145/1582379.1582434.
Zhang, Q., Song, S., 2008. A systematic procedure for accurately approximating lognormal-sum distributions. IEEE Trans. Veh. Technol. 57 (1), 663–666, https://doi.org/10.1109/tvt.2007.905611.

Daniel Morato received the M.Sc. degree in Telecommunication Engineering and the Ph.D. degree from the Public University of Navarre, Spain. During 2002 he was a visiting post-doctoral fellow at the Electrical Engineering and Computer Sciences Department, University of California, Berkeley. Since 2006 he has been working at the Department of Automatics and Computing, Public University of Navarre, as an associate professor. His research interests include high-speed networks, performance and traffic analysis of Internet services and network monitoring.

Eduardo Berrueta graduated on Telecommunication Engineering in 2016 from the Public University of Navarre (UPNA), Spain. Previously he attended the University of Turin for completing his thesis on Software Defined Networking. During 2016 he held a scholarship on the Automatics and Computing department. Since October 2017 he is a research assistant for the Telecommunications, Networks and Services Research Group at UPNA while he completes an M.Sc. in Telecommunication Engineering.

Eduardo Magaña received his M.Sc. and Ph.D. degrees in Telecommunications Engineering from Public University of Navarra, Pamplona, Spain, in 1998 and 2001, respectively. Since 2005, he is associate professor at Public University of Navarra. During 2002 he was a postdoctoral visiting research fellow at the Department of Electrical Engineering and Computer Science, University of California, Berkeley. His main research interests are network monitoring, traffic analysis and performance evaluation of communication networks.

Mikel Izal received his M.Sc. and Ph.D. degrees in telecommunication engineering in 1997 and 2002 respectively. In 2003 he worked as a scientific visitant at Institute Eurecom, Sophia-Antipolis, France, performing measures in network tomography and peer-to-peer systems. Since then, he has been with the Department of Automatics and Computing of the Public University of Navarre where he is an Associate Professor. His research interests include traffic analysis, network tomography, high speed next generation networks and peer to peer systems.