Expert Systems With Applications 209 (2022) 118299

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Crypto-ransomware detection using machine learning models in file-sharing network scenarios with encrypted traffic

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

[a] Public University of Navarre, Department of Electrical, Electronic and Communications Engineering, Campus Arrosadia, 31006 Pamplona, Spain
[b] Institute of Smart Cities, Tajonar 22, 31006 Pamplona, Spain

ARTICLE INFO

Keywords:
Crypto-ransomware
File-sharing traffic
Network security

ABSTRACT

Ransomware is considered as a significant threat for home users and enterprises. In corporate scenarios, users' computers usually store only system and program files, while all the documents are accessed from shared servers. In these scenarios, one crypto-ransomware infected host is capable of locking the access to all shared files it has access to, which can be the whole set of files from a workgroup of users. We propose a tool to detect and block crypto-ransomware activity based on file-sharing traffic analysis. The tool monitors the traffic exchanged between the clients and the file servers and using machine learning techniques it searches for patterns in the traffic that betray ransomware actions while reading and overwriting files. This is the first proposal designed to work not only for clear text protocols but also for encrypted file-sharing protocols. We extract features from network traffic that describe the activity opening, closing, and modifying files. The features allow the differentiation between ransomware activity and high activity from benign applications. We train and test the detection model using a large set of more than 70 ransomware binaries from 33 different strains and more than 2,400 h of 'not infected' traffic from real users. The results reveal that the proposed tool can detect all ransomware binaries described, including those not used in the training phase. This paper provides a validation of the algorithm by studying the false positive rate and the amount of information from user files that the ransomware could encrypt before being detected.

1. Introduction

Crypto-ransomware is a type of malware that extorts computer users by encrypting their files and requesting a ransom to recover the file content. In 2016, EUROPOL declared that ransomware was 'the most prominent malware threat [...] for citizens and enterprises alike' (EUROPOL, 2016). Since 2018, crypto-ransomware attacks have been directed at companies in areas such as manufacturing, transportation, telecommunication, finance, public law enforcement, and health services (Cobb, 2018; TrendMicro, 2019). This is done because of the high economic profits that malware developers can gain from each infection. In 2019, the estimated financial damage caused by ransomware attacks in the United States was $7.5 billion (Faghihi & Zulkernine, 2021).

In corporate environments, local computers used by the employees store only system and program files. The valuable company documents are stored in files on shared networked volumes (Intelligence, 2021). This architecture facilitates the implementation of backup policies, and offers sharing capabilities and extended access control. In case of a crypto-ransomware infection, host recovery is a simple procedure, requiring just a disk cloning operation from a clean image of the system and program files. No valuable documents should be lost because they are not stored on the local computer. However, in these volume-shared scenarios, a single infected computer can encrypt all the files it has access to from the shared volumes, thereby creating a highly compromised environment. An independent study of 5000 IT managers across 26 countries (Sophos, 2020) revealed that 65% of ransomware victims lost their data in network-shared volumes.

The proliferation of different strains of ransomware has given impetus to the development of detection tools focused on this type of malware. In a previous survey (Berrueta et al., 2019), we analysed more than 50 different tools, mainly from academia, and some from security companies. The most effective tools are based on monitoring disk-access activities (Continella et al., 2016; Kharraz et al., 2016), although they are always tools installed in the local hosts. In a network file-sharing scenario, disk-access information about operations on valuable

* Corresponding author at: Public University of Navarre, Department of Electrical, Electronic and Communications Engineering, Campus Arrosadia, 31006 Pamplona, Spain.
E-mail addresses: eduardo.berrueta@unavarra.es (E. Berrueta), daniel.morato@unavarra.es (D. Morato), eduardo.magana@unavarra.es (E. Magaña), mikel.izal@unavarra.es (M. Izal).

https://doi.org/10.1016/j.eswa.2022.118299
Received 17 February 2022; Received in revised form 8 July 2022; Accepted 25 July 2022
Available online 30 July 2022

documents can also be obtained from network traffic. This facilitates tool deployment. The network probe does not require any software running on user hosts. The probe can simply monitor file-sharing traffic from a network switch near the document repository.

Traditionally, information contained in network file-access protocol messages is circulated in the clear on Local Area Networks (LANs). However, owing to the popularization of public internet cloud services and the increasing importance of confidentiality in network transactions, these protocols are evolving into their encrypted versions. Therefore, nowadays, a traffic monitor cannot obtain detailed information about the disk-access activities, and the detection tools based on such information do not work as desired. Until today, no tool is capable of ransomware detection based on encrypted network file-sharing traffic, as has been proved in several survey articles (Berrueta et al., 2019; Bijitha et al., 2020; Herrera Silva et al., 2019; McIntosh et al., 2021). The aim of this study is to fill this gap, and detect crypto-ransomware in action (i.e. during the encryption of network-shared files) by analysing features extracted from the encrypted network traffic. No previous proposal has based its detection algorithm on encrypted file-sharing traffic. The large number of features offered by this traffic requires an analysis tool capable of detecting patterns in complex structures. In this study, we train and test three machine learning (ML) models. We evaluate models that could run efficiently at high speed in a network probe. Using deep learning and an adequate set of features, the accuracy of detecting an active ransomware in 30 s reaches 99.8%, with a false positive rate of 0.004%.

The main contributions of this paper are:

- Present a crypto-ransomware detection tool based on the analysis of encrypted network traffic in file-sharing scenarios. This requires extracting and filtering features that can distinguish between benign high activity traffic and ransomware activity. To the best of our knowledge (Ahmed et al., 2021; Berrueta et al., 2019; Bijitha et al., 2020; Faghihi & Zulkernine, 2021; Herrera Silva et al., 2019; McIntosh et al., 2021), no previous proposal has targeted this specific scenario with encrypted traffic.
- Compare the results from three different ML algorithms capable of high speed classification. The comparison uses more than 50 h of crypto-ransomware file-sharing traffic from 67 different ransomware binaries and 50 h of real user traffic from benign applications. The validation is based on ransomware strains not used in the training process (unseen binaries) and 2477 h of real user traffic.
- Validate the detection results for the most popular file sharing protocols in a corporate network. It is not an ad-hoc solution for a single file-sharing protocol.
- Offer not only popular ML evaluation metrics to facilitate comparison with previous and future works but also more scenario-oriented metrics. The latter are for example the elapsed time or the number of bytes encrypted before detection and the number of false alarms triggered.
- We uploaded all the datasets and the optimized and trained ML model to a public repository to facilitate validation and comparison (Berrueta et al., 2021).

The remainder of this paper is structured as follows: Section 2 summarizes the literature on ransomware detection and highlights the unsolved problems tackled by the present proposals. Section 3 provides a detailed explanation of the scenario and the methodology, including the analysis of the datasets used for training, testing and validating the models. Additionally, Section 3 describes the feature and time-sample selection and the metrics that will be used to evaluate the quality of the proposal. Section 4 presents the results of training, optimizing and evaluating the ML models, selecting the one with highest accuracy and showing the existent trade-offs in its design. Section 5 provides a comparison of the results with those provided by other tools in the literature. Finally, Section 6 states the conclusions.

2. Background and related work

Recent surveys (Berrueta et al., 2019; Bijitha et al., 2020; Herrera Silva et al., 2019) have analysed and compared a total of more than 70 different crypto-ransomware detection tools. Ransomware detection tools have traditionally used similar techniques to antivirus tools, such as those based on the static analysis of program binaries before they are run (Hasan & Rahman, 2017; Reddy et al., 2021; Shaukat & Ribeiro, 2018). These techniques offer prevention, however, they are prone to false negatives when new binaries are encountered or obfuscation techniques are used (Chen et al., 2017; Vidyarthi et al., 2019), they cannot cope with the large number of signatures created by the new ransomware-as-a-service offerings (Nieuwenhuizen, 2016), and they are oblivious to file-less ransomware variants (Victor, 2020). Static prevention techniques are being substituted by architectures where indicators are extracted from monitoring the actions taken by any program running at the user's host. These monitored indicators allow the detection of crypto-ransomware activity while it is probably already encrypting files, therefore the most wanted feature is early detection with minimum damage.

Please beware that the work presented in this article is focused on the detection of crypto-ransomware, and not other malware families that ask for a ransom for unlocking a computer (locker-ransomware), or they focus on information exfiltration. On the following we use the terms 'ransomware' and 'crypto-ransomware' interchangeably, meaning always 'crypto-ransomware'.

In this section we present a short review of the different architectures in crypto-ransomware detection tools, centred on the classification of the indicators and on the analysis functions employed. This review shows how the proposal in this article fits in the literature. For further details about each tool, we refer the reader to the above-mentioned survey papers. We present a more critical analysis, and a detailed comparison of the benefits offered by our proposal with respect to previous works in Section 5 of this paper, once the results have been described.

The indicators analysed by the detection tools and describing the activities being taken by a suspect program are most often extracted locally to the host where the suspect program is running (Ahmadian et al., 2015; Alam et al., 2018; Almashhadani et al., 2020; Cabaj & Mazurczyk, 2016; Continella et al., 2016; Hasan & Rahman, 2017; Kharraz et al., 2016; Moussaileb et al., 2018; Scaife et al., 2016; Sgandurra et al., 2016). This strategy offers the highest visibility of malware actions using different types of system call interception techniques (Ahmed et al., 2020; Alam et al., 2018; Chen et al., 2017; Vinayakumar et al., 2017). A less frequent group of indicators can be obtained from network traffic crossing a switching point, usually an Internet access router (Hasan & Rahman, 2017; Lu et al., 2017). This second set of indicators can be obtained without the installation of malware detection software locally to the user's host; they can be obtained using an independent network traffic probe.

The most significant locally-obtained features describe disk activity reading and writing files, as well as the creation of encrypted content on the written data. All this information is easily obtained by intercepting disk access input/output (I/O) operations (Continella et al., 2016; Hasan & Rahman, 2017; Kharraz et al., 2016; Lu et al., 2017; Paik et al., 2016; Scaife et al., 2016; Sgandurra et al., 2016). Complex detection tools add to the previous feature information such as the use of functions from external libraries (searching for encryption libraries) (Continella et al., 2016; Sgandurra et al., 2016) or the identification of directories in which the read or write operations are performed (Ahmadian & Shahriari, 2016; Kharraz et al., 2016; Lu et al., 2017; Moussaileb et al., 2018; Sgandurra et al., 2016).

All these tools require being installed locally on every user host and suffer three serious drawbacks. First, they impose a management burden coming from their installation and frequent update tasks when a large set of hosts is involved (Berrueta et al., 2019; Morato et al., 2018).

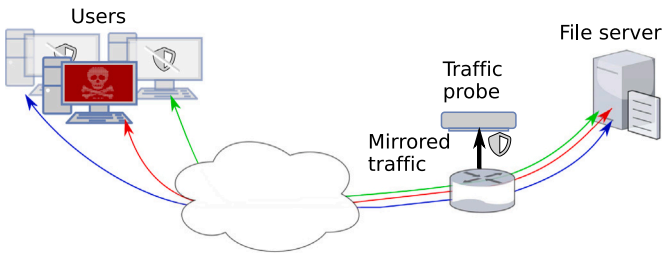

Fig. 1. Monitoring scenario with ransomware detection tool based on network traffic.

Second, locally installed software that analyses system calls consumes CPU cycles at the user host, having a potential impact on computer responsiveness (Continella et al., 2016; Kharraz et al., 2016; Mehnaz et al., 2018; Shaukat & Ribeiro, 2018; Shukla et al., 2016). Third, malware capable of escalating privileges could deactivate the detection tool (Loman, 2019).

All these drawbacks are not present when there is no locally installed tool at the users' hosts, but the indicators are extracted solely from network traffic obtained at a network switching point. The detection tool is then installed on a single independent network probe that receives a copy of the traffic from a large population of users through a port-mirror in a network switch, being capable of monitoring all operations on shared files (Fig. 1). This network probe is an analysis server that can be isolated from common vulnerabilities by being connected to a different network segment. Having no human user, the network probe is not vulnerable to social-engineering attacks, which are a frequent attack vector in ransomware infiltration (Berrueta et al., 2019; Sjouwerman, 2019).

The main drawback of detection tools based solely on network traffic is that they cannot monitor local features, specially local disk access operations. However, in a corporate environment, local files are system files, easy to recover and without real value. The analysis of not only remote file operations but also local disk access operations could improve detection time, but only for crypto-ransomware variants that attack system files before they attack user documents. Moreover, this improvement would come at the expense of an impact on computer responsiveness due to the locally installed analysis tool in every user host.

File access operations on valuable documents are visible from a network traffic probe by monitoring the traffic exchange between the users' hosts and the file servers. Most detection proposals designed based on local interception of disk access system calls can still be applied to a file-sharing scenario because remote-access protocols offer the applications an interface to the shared files using the local file system.

We previously designed and validated a crypto-ransomware detection technique based on file-sharing traffic (Morato et al., 2018). This is the only published work focusing on the corporate network scenario. We analysed the protocol messages in the Server Message block version 2 (SMBv2) protocol, the most common file-sharing protocol in the Microsoft Windows environment for the local area network. The detection tool required the metadata about the disk I/O operations – contained in the file-sharing protocol messages – therefore it could only be used for SMBv2, when the protocol does not send encrypted messages. In this paper we present a different detection technique, still based on network traffic but capable of detecting crypto-ransomware when different file-sharing protocols are employed (not restricted to SMB), even when they use encryption at the protocol level.

Please beware of the distinction between encryption at the protocol level and encryption at the application level. Even if the application is ransomware trying to overwrite a file with encrypted content, a clear-text file sharing protocol will send protocol messages where the type of operation, the file path, the byte position in the path, will all be on-the-clear, even though the content written was encrypted. The algorithm described in Morato et al. (2018) required clear-text protocol messages; however, this is not available in new file-sharing protocols that use encrypted transport.

Up to this point, we have established the convenience of a monitoring architecture based on the indicators obtained from network file-sharing traffic. An analysis function takes these indicators and uses them to accomplish the classification of the suspect program traffic. The complexity of the analysis function varies in the literature. In some cases it only establishes thresholds to the measured metrics (Kharraz et al., 2016; Mbol et al., 2016; Paik et al., 2016), while most often a combined metric from a large number of indicators is built (Moore, 2016; Morato et al., 2018; Scaife et al., 2016). In recent years, machine learning techniques have gained popularity thanks to their ease of use and their capacity of searching for patterns in a large number of features (Ahmadian & Shahriari, 2016; Ahmed et al., 2020; Almashhadani et al., 2019; Arabo et al., 2020; Continella et al., 2016; Hasan & Rahman, 2017; Hirano & Kobayashi, 2019; Lee et al., 2019; Lu et al., 2017; Moussaileb et al., 2018; Reddy et al., 2021; Roy & Chen, 2020; Scaife et al., 2016; Sgandurra et al., 2016; Vinayakumar et al., 2017).

These new solutions take a similar set of input features to previous proposals, from a similar scenario, and they use a new technique for the analysis. They usually compare some ML models, choosing the one offering the best results (Ahmed et al., 2020; Cohen & Nissim, 2018; Hwang et al., 2020), but sometimes the authors select the machine learning model without any comparison and try to improve its effectiveness by parameter tuning of the algorithm (Roy & Chen, 2020) or by changing the feature extraction process (Zhang et al., 2020).

In this work we show that an ML model can be applied to effectively detect crypto-ransomware when using as input to the algorithm features obtained from file-sharing network traffic using an encrypted or unencrypted protocol. We design these features to provide a learnable distinction between traffic form ransomware encrypting files or from benign applications. To the best of our knowledge, no previous work in the literature or industrial tool has solved the problem of detecting crypto-ransomware activity on the base of encrypted file-sharing traffic. We believe it is an interesting scenario that can take advantage of characteristics such as better scalability, due to not requiring installing monitoring software in any host or file-sharing server and, being completely off-path, it does not interfere with user actions.

3. Scenario and methodology

Fig. 1 illustrates a population of users in a corporate LAN, accessing files from a common server. Crypto-ransomware running at one of the hosts can read large amounts of data from the files stored in the server and write the encrypted version of those files in the same server. Crypto-ransomware detection can be accomplished by detecting the file-sharing traffic these read, write, delete, and rename actions that the ransomware performs during its progress.

A network probe can capture and analyse the traffic without any effect on network latency, response time, or computer responsiveness because the probe is not in the traffic path but monitoring a copy of the traffic. Using commodity hardware, traffic rates in the order of tens of gigabits per second can be captured and processed (Julián-Moreno et al., 2018). We will evaluate simple ML models that could sustain these traffic rates without significant degradation in analysis time.

Recent protocols in network file-sharing scenarios are commonly transported over TCP/IP. The most commonly used protocol, both in enterprise and home deployments, is server message block (SMB), particularly its second and third versions (SMBv2 and SMBv3, respectively). Despite the availability of other file-sharing protocols such as network file system (NFS) (Haynes & Noveck, 2015) and Apple filing protocol (AFP) (Apple, 2012), the extended use of the Windows operating system (OS) in corporate environments makes SMB the most

popular protocol because it is the default file-sharing protocol used in this OS. In mixed Windows-UNIX environments, NFS can be (but rarely) the protocol of choice, while in a mixed Windows-macOS environments it is SMB (supported in macOS) and not AFP (not supported in Windows) (NetGear, 2016). Microsoft Windows is also the target of most ransomware attacks, so most binaries run only on this operating system and it will be the subject of analysis in this work, combined with SMB and to a lesser extent with NFS.

Network traffic offers a plethora of metrics: number of TCP connections, bytes transferred, sequence of messages between client and server, packet sizes, inter-packet times, inactivity times, connection durations, and sequences (in time) of any of these, to name a few. We resort to ML techniques, which have been validated in previous works as adequate tools in this type of scenario (Ahmed et al., 2020; Zhang et al., 2020). ML algorithms can analyse these complex data structures, and once they are trained with a complete dataset, they can generalize to different input data. The most adequate set of features used in training the ML model depends on the characteristics of file-sharing traffic. In the following sections we describe the network protocol scenario, the methodology for constructing the feature dataset we used in training and testing, the models that will be evaluated, and the metrics used for this task.

3.1. Protocol behaviour

In this study, we consider a network file-sharing scenario where the server and clients communicate using the SMB protocol. In Morato et al. (2018), we considered such a scenario, although only for version 2 of the protocol (version 1 has been deprecated since 2014 Pyle, 2020). We extend the results to SMBv3 and NFS, which implies significant changes in network traffic. SMBv3 is the default and recommended version since Windows 8 and Windows Server 2012 (NetApp, 2018). This version introduced message encryption and some other changes to the original protocol, making it more secure. The added encryption layer makes it impossible for traffic analysis tools to distinguish messages and different user activities. This version is expected to become the most popularly deployed version of the SMB protocol in enterprises migrating from deprecated versions of the Windows OS (Microsoft, 2020).

SMB is a request–response protocol transported over TCP using a single connection to a well-known server port (value 139 or 445), and with 19 different commands in its second and third versions. Each command corresponds to one action that the client issues over a server file, volume or directory. These commands can be considered similar to the input/output operations (I/O ops) performed locally at the user's computer. Moreover, our previous investigation has revealed that it is possible to detect ransomware in action, based on the commands from the SMBv2 protocol (Morato et al., 2018).

The only difference in protocol messages between SMBv2 and SMBv3 is an added 52-bytes header in the latter, describing the presence of encryption. Owing to this encryption, the command type and parameters such as the file path or the file offset cannot be identified. Tools based on the analysis of these commands cannot successfully detect ransomware in the encrypted version of SMB. Although it is impossible to view the client operations, some features can be extracted from the encrypted network traffic, which could allow accurate identification of the ransomware actions. This study is focused mainly on file sharing using the SMB protocol, however, the procedure employed is extensible to other request–response file-access protocols such as NFS, even with encrypted traffic, as will be proved in Section 4.3.

3.2. Dataset

Two types of samples are used in this study: (i) those from traffic captured while crypto-ransomware was encrypting network-shared files ('infected') and (ii) those from staff office users running benign applications and accessing shared files ('not infected').

The traffic traces for the 'infected' case were obtained from a repository we built and shared publicly in (Berrueta et al., 2020; Berrueta et al., 2022). This repository comprises traffic traces from more than 70 ransomware programs. All these ransomware programs were obtained from Hybrid-Analysis[1] and Malware-Traffic-Analysis.[2] Each binary was executed more than once, generating 150 traffic traces in total. They were captured while the malware was encrypting a large file set shared by an SMB server. We obtained more than 50 h of ransomware activity from these traffic traces.

Table 1 shows a summary of the ransomware strains present in our public repository. We have collected binaries from crypto-ransomware families with different file-access behaviours. There are variants that apply compression to the written data (CTBLocker), variants that batch the file deletion operations (WannaCry), variants that do only partial overwrites of the original files (Shade), or variants operating at low speeds (Revenge), to name a few relevant features. More detail about the binaries and the traffic traces can be found in Berrueta et al. (2020) or Berrueta et al. (2022); we offer statistics, downloadable traffic files, links to the malware binaries, and text files containing all the file-access operations executed.

From these crypto-ransomware execution traffic traces, we separated seven recent families. They are not included in the training set but further used to check whether they are accurately detected or not, even when they are not part of the learning process (we name them as 'unseen' samples). These seven families are: bitPaymer (appearing in November 2019), Shade (November 2019), Sodinokibi (one variant obtained in July 2019 and two in March 2020), Phobos (May 2019), Stop (January 2019 and February 2020), RansomX (June 2020) and Shaofao (August 2020). All those unseen binaries that could run in a Windows 10 environment (5 out of 10) were also run in a scenario using NFS protocol, and employed in the multi-protocol validation of the detection model (Section 4.3).

The 'not infected' samples were obtained from network traffic traces captured in a campus LAN, wherein staff users access files from shared servers. We collected 2527 h of 'not infected' traffic. This represents 316 intervals of 8 working hours. For each 8-h connection, on average, 7640 files were opened, and 91.4 and 294 MB data per connection were read and written, respectively. There are no identifiable personal data in this dataset because we cannot identify the users based on the IP addresses; only the Information Technologies (IT) department has that information and it was not shared with the researchers. No malware infection reports were received from any user during the traffic recollection period.

From these three groups ('infected, 'not infected', and 'unseen'), we extracted samples to create three different datasets. The first dataset: the training and testing dataset, comprises all 50 h of ransomware samples labelled as 'infected', and a selected set of 50 h of user samples labelled as 'not infected'. We used 80% of this first dataset for training the algorithm and the remaining 20% for testing it. During the testing phase, we measured accuracy, F-measure, precision, recall, and phi-coefficient of each model (Almashhadani et al., 2020). These metrics allow an easy comparison with previous and future proposals based also on ML models. Subsequently, we selected the most effective algorithm from a group of ML models that could provide fast evaluation in a network probe.

The second dataset comprises 2477 h of user activity, also labelled as 'not infected', which are not included in the training and testing dataset. This dataset is used for an extended test of the false positive rate, i.e., it is used to measure situations wherein ransomware is falsely detected. This false positive rate determines the usability of the tool

[1] https://www.hybrid-analysis.com.
[2] http://www.malware-traffic-analysis.net.

Table 1
Characteristics of the traffic traces from 26 ransomware families (67 binaries) used in the training phase and 7 ransomware families (10 binaries) not used in the training phase but only in the evaluation phase.

Family	Binaries	Date of first binary	Date of last binary	Total read/write operations	Total traffic trace duration (min)
CryptoFortress	1	March 2015	March 2015	105k/89k	28.7
TeslaCrypt	1	December 2015	December 2015	26k/26k	25
Cerber	13	February 2016	February 2017	746k/721k	292.4
DMALocker	1	February 2016	February 2016	470k/169k	147.2
Locky	10	February 2016	August 2016	1251k/1287k	982
VirLock	3	July 2016	August 2017	415k/560k	273
Bart	1	September 2016	September 2016	163k/441k	37.2
CrypMIC	1	September 2016	September 2016	212k/137k	12.1
CryptFile2	1	November 2016	November 2016	127k/127k	4.48
CryptoMix	4	November 2016	January 2016	373k/372k	125.2
Crysis	11	December 2016	December 2018	155k/236k	1994
CryptoShield	2	January 2017	February 2017	140k/140k	230
CTBLocker	1	January 2017	January 2017	172k/10k	19.2
MRCR	1	January 2017	January 2017	103k/91k	1038
Zeus	1	February 2017	February 2017	124k/75k	12.5
GlobeImposter	1	May 2017	May 2017	1161k/592k	33.5
Jaff	2	June 2017	June 2017	264k/324k	82.3
Aleta	1	July 2017	July 2017	104k/142k	121
Spora	1	May 2017	May 2017	37k/26k	8.9
WannaCry	2	May 2017	May 2017	264k/458k	2466
Sage	2	January 2018	May 2018	329k/370k	57.4
Revenge	1	March 2018	March 2018	60k/84k	21.5
Maktub	1	April 2018	April 2018	16k/10k	3.5
Mole	1	May 2018	May 2018	137k/175k	23.3
Ryuk	1	April 2019	April 2019	98k/130k	10.2
GandCrab	2	May 2019	May 2019	73k/55k	79.7
Unseen ransomware strains					
BitPaymer	1	August 2019	August 2019	121k/128k	121.36
Shade	1	September 2019	September 2019	101k/90k	18.5
Sodinokibi	3	September 2019	March 2020	269k/253k	248
Phobos	1	November 2019	November 2019	43k/66k	106
STOP	2	December 2019	February 2020	76k/70k	34.5
RansomX	1	February 2020	February 2020	78k/44k	14.7
Shaofao	1	March 2020	March 2020	23k/66k	14.7

because a large number of false positives render the tool unusable in a real environment. The absolute number of false positives or the number relative to the period of time (alarms per day or alarms per month) is a better indicator of model quality than the accuracy metric. It provides a better insight on the cost on network administrator working-hours that the false alarms can incur.

Finally, the 'unseen' dataset comprises samples extracted from the newest ransomware traffic traces. Using this dataset, the capability of the tool to detect the activity from unseen crypto-ransomware binaries is measured. Furthermore, we measure the number of files and bytes that the crypto-ransomware encrypts before detection. These metrics are better suited to the ransomware scenario than traditional precision or recall metrics. They provide a scenario-oriented evaluation, instead of an abstract ML-model-oriented evaluation.

Fig. 2 summarizes the process of dataset creation for training and evaluation. Further details about sample filtering are provided in the next sections.

3.3. Feature extraction

Crypto-ransomware programs read and write large amounts of bytes for encrypting user files. While some ransomware strains overwrite the content of the original files, others create new encrypted files before the deletion of the original. In both cases, reading the original and writing the new encrypted one must be performed by ransomware before trying to extort the users. They can compress the output encrypted file, however, they still move large amounts of data to compromise as many files as possible. Crypto-ransomware tries to complete this process as fast as possible to avoid being detected when the user tries to open a file and discovers that it is unreadable. However, the speed of these actions depends on the complexity of the encryption algorithm, the efficiency of its implementation, the hard disk speed or the CPU power. A too-slow encryption hampers the ability of the crypto-ransomware to perform an effective threat because backup policies can be triggered before it can alter a large number of modified and non-backuped files.

Owing to the encryption of file-sharing traffic in recent protocols, a significant amount of metadata about these actions remains hidden from network traffic monitors. These monitors can only record packet endpoint addresses, arrival times, sizes, and direction (from client to server or vice versa). Based on this information, the network probe can only guess the file-access actions performed by the user. These actions could involve opening a file, closing it, obtaining information about it, changing its metadata, writing content to it, or reading content from it. Based on the aforementioned limitation, we define the following three actions:

- Bytes are being written (Fig. 3(a)): We consider a write operation when there is a one-packet response for a large (more than one packet) request. In Fig. 3(a), we name these bytes as 'bytes Client–Server'.
- Bytes are being read (Fig. 3(b)): We consider a read operation when there is a one-packet request and a large response (more than one packet). In Fig. 3(b), we name these bytes as 'bytes Server–Client'.
- Control or short commands (Fig. 3(c)): These commands include operations such as delete, rename, open, and close file. They do not require a large amount of data describing the command; they can usually fit into a single-packet request from client to server and a single-packet response from server to client. Beware that short read or write actions whose data fits into a single packet are indistinguishable from control commands; however, the operating system tends to batch disk access operations to optimize the data flow, making this event unlikely.

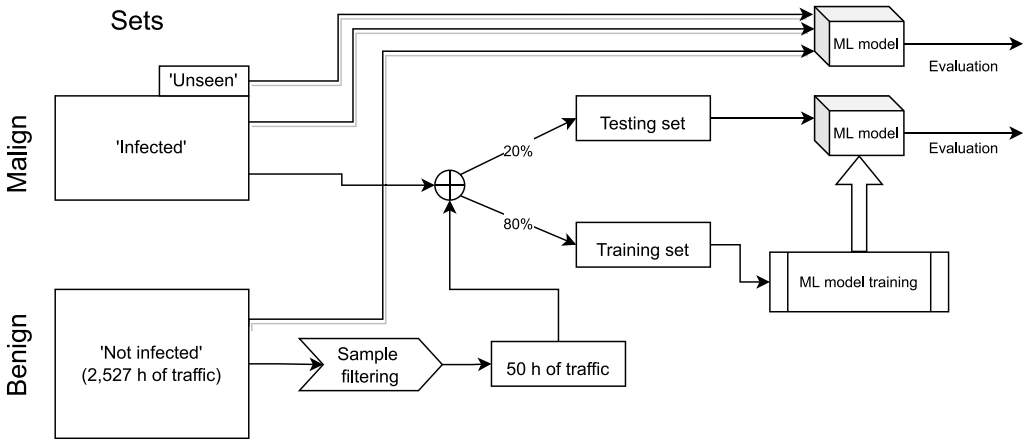

Fig. 2. Training, testing and evaluation process.

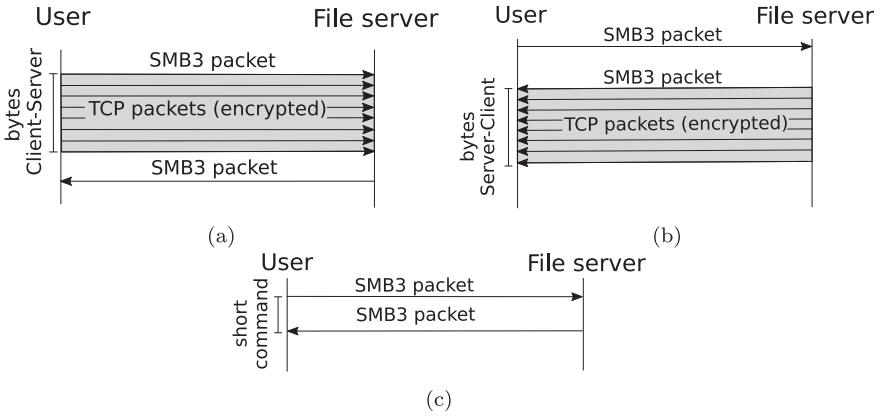

Fig. 3. Write operations (a), Read operations (b), and short commands (c) between client and server.

Crypto-ransomware performs frequent file-access operations, compared to a typical office program. However, in some user behaviours, the number of bytes read and written can be equal or higher than those by crypto-ransomware during a short period of time. One example of these intense behaviours is the duplication of data in the server.

Fig. 4 presents the bytes (a) read and (b) written per second by a ransomware (Cerber) encrypting a shared directory, and by a user while duplicating all the files in the directory. Herein, the user activity is greater than that of the ransomware because the user does not suffer the burden of encrypting the files. It is not easy to distinguish the two cases by only considering the bytes read and written, the inter-arrival packet times or the packet sizes. To this end, we need a feature that can differentiate these extreme cases of similar activity. Fig. 5 presents the number of short commands per second for the same traffic traces. This number is significantly lower for directory duplication. In-depth study of ransomware behaviour (Berrueta et al., 2019) reveals that when ransomware encrypts files, it must delete the original ones and sometimes create extra files in each directory with payment and decryption instructions for the victim. These actions lead to the difference in the number of short commands and enable us to differentiate both cases.

Consequently, similar to the detection techniques run locally on the infected host, crypto-ransomware activity can be recognized from network traffic based on the traffic pattern between the client and server. When network traffic comprises clear-text, the detection tool can also use the file-access action types to assist in differentiating ransomware actions from benign applications (Morato et al., 2018). The analysis tool knows the commands being used (changing file names, deleting or overwriting files) and it can also measure the higher entropy in written encrypted file content. In scenarios where all disk access commands are encrypted, these metadata about user actions are not available, however the aforementioned new feature, based on the number of control or short commands, can still be utilized as a differentiator between both types of actions.

All these features cannot be extracted for a single packet — they are the result of traffic accumulated during a (preferably) short period of time. The larger the analysis window is, the easier it is to distinguish crypto-ransomware from benign applications, owing to the larger variability in disk-access patterns in the latter. However, the larger the analysis window is, the larger the period of time to detect the crypto-ransomware and the number of files lost before it can be blocked.

Analysis of the abovementioned traffic features (number of bytes read, written, or control commands) is performed for each TCP connection between one client and the server. We measure the traffic features in per-second time intervals, and we introduce a temporal window of T seconds to create complete time-samples for the learning process. Hereafter, the term ‘sample’ means the time windows composed by $3 \times T$ features (bytes read, written and short commands) that are input into the classification model. The term ‘samples’ should not be confused with the ransomware executables that are sometimes called ‘samples’ in the literature. We refer to the latter as ‘ransomware binaries’. Fig. 6 illustrates an example for $T = 10$ s, where $N = 3 \times 10 = 30$ features are present in each sample. For each 1-s interval, the traffic probe computes the following.

- Total number of short commands where the response is contained inside the window.

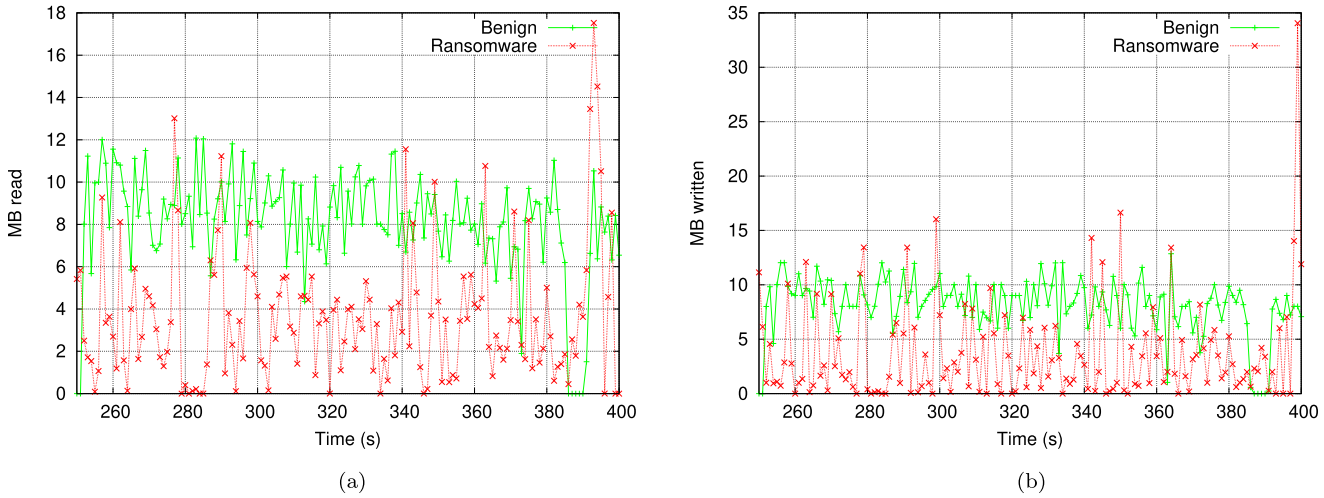

Fig. 4. Bytes (a) read and (b) written per second by ransomware and user.

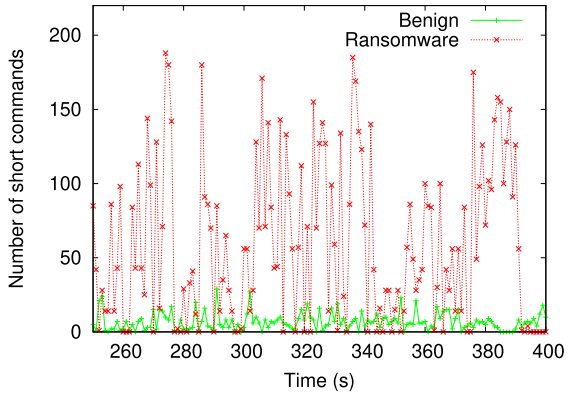

Fig. 5. Short commands per second performed by user and ransomware.

- Total number of data (TCP bytes) in the packets sent from the server to client in read operations.
- Total number of data (TCP bytes) in the packets sent from client to server in write operations.

These features represent the control commands, read actions, and write actions, respectively. The complete sample for the machine learning model comprises these three values for every second in the time window of T seconds.

The next sample is created with traffic out of the window from the previous one. We can slide the time window a small amount of time or as much as the window length T, creating samples without shared data. In the example of Fig. 6, the next sample without shared data will include the features between the 10th and the 20th second; therefore, we generate one sample every T seconds. A new sample with shared data could be built by sliding the windows for example only 1 or 2 s (or even a fraction of a second), creating a new set with data from the 1st to the 11th second or from the 2nd to the 12th second respectively.

Parameter T and the sliding window step are tuneable. The detection accuracy and data loss depend on them. With an increasing T, the samples comprise an increasing number of time-intervals, and the algorithm can learn more complex relations between features. However, detection of ransomware would require more time (especially if it requires more than one sample) and the user will lose more data in the process. With a small sliding step, the new sample does not contain much new information, however the detection algorithm could react earlier to abrupt changes in the traffic pattern. We will evaluate the effect of both strategies during the optimization process.

3.4. Sample filtering

ML models require significant and adequate sets of learning samples for the behaviours that they must classify (Sommer & Paxson, 2010). The different activities (from benign applications and malware) must be accounted for without over-representation of some of them, which could result in ignoring significant patterns.

In the learning process, we used all the ransomware samples available, removing only the inactivity time before the first file was opened for encryption and after the last file was altered by the malware.

Compared to this ransomware activity, network file-sharing traffic from office users is highly intermittent, with large thinking times, for which no significant amount of traffic occurs. Samples from benign users must be selected by considering the higher popularity of time periods for which no traffic exists and being especially cautious by providing enough learning samples where user actions create network traffic.

We studied each of the three features while comparing 'not infected' and 'infected' samples. In Fig. 7, the complementary cumulative distribution functions for the bytes written per second in an 'infected' trace and a 'not infected' one are plotted — the probability $P(bytesWritten \geq x)$ of more than a certain amount of bytes being written in one second. For crypto-ransomware samples, only 20% 1-s intervals contain no written bytes, whereas for user traffic, more than 99% 1-s intervals are inactive intervals in terms of written bytes. A similar situation is revealed for the bytes read. The training of the ML algorithm requires a significant number of 'not infected' samples where the user is active. This number is not proportional to their presence in relation to non-active intervals; they represent a significantly smaller number. A random selection of samples could result in selecting a large set of low activity benign samples and therefore the learning process could ignore rare high-activity benign users and generalize that high activity is always indicative of crypto-ransomware action (Sommer & Paxson, 2010).

To avoid this bias, we include in the training set all the 'not infected' samples where at least one of the 1-s intervals carries at least 5 MB (read or written). We also included all the samples with at least one 1-s interval containing more than 100 short commands. This guarantees a sufficient representation of high-activity benign user intervals. We completed this 'not infected' training set with a random selection of samples from those remaining, up to the 'infected' set size, providing a balanced two-classes training scenario.

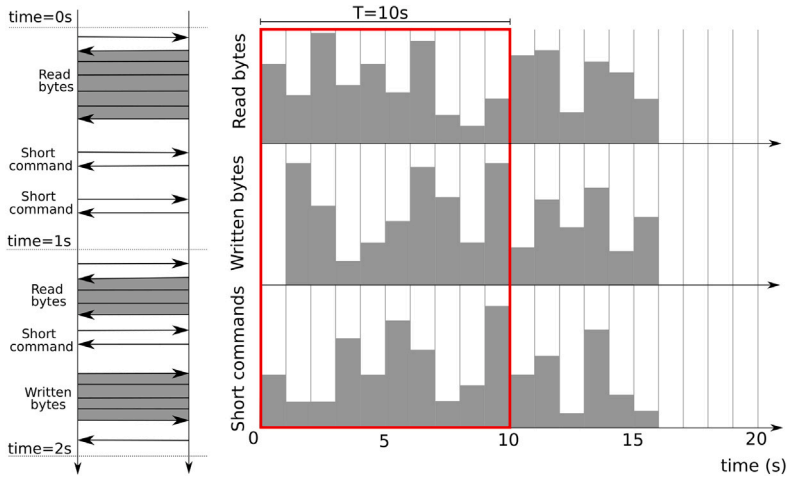

Fig. 6. Example of feature extraction in T seconds.

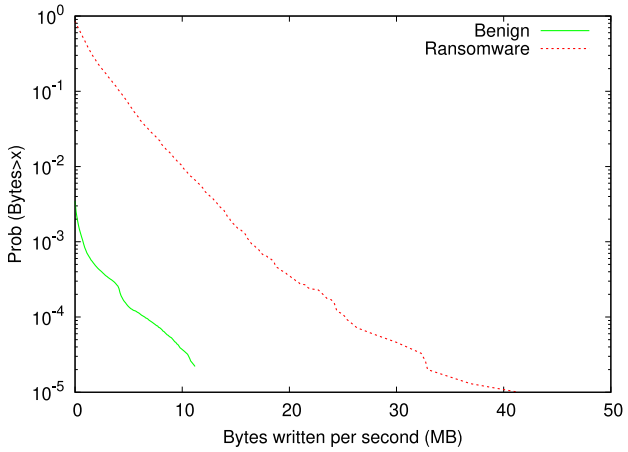

Fig. 7. Complementary cumulative distribution function for the bytes written per second.

3.5. ML models and evaluation metrics

This paper analyses three ML models that could efficiently be implemented in a network probe that is classifying traffic in real-time: decision trees (DTs), tree ensembles (TEs), and neural networks (NNs). DTs are the simplest of these three models; however, they are susceptible to over-fitting. They have been used for ransomware detection in research articles such as Almashhadani et al. (2020), Moussaileb et al. (2018) and Mehnaz et al. (2018). TEs combine DTs and are capable of finding more complex relations between features, although at the expense of a higher model complexity. The authors of Almashhadani et al. (2020) and Lee et al. (2019) analysed the capability of TEs to detect ransomware, achieving high detection rates. Finally, NNs, owing to their flexibility, are popular models in the literature (Agrawal et al., 2019; Almashhadani et al., 2020; Maniath et al., 2017; Roy & Chen, 2020; Shaukat & Ribeiro, 2018). NNs are more complex than DTs or TEs; however, model computation is only required once every T seconds (see Section 3.3) which does not impose a critical speed requirement.

These three models were trained and tested using bigML[3] with the dataset described in Sections 3.2, 3.3, and 3.4. The best model is selected based on several binary classification metrics. These metrics can be derived from the confusion matrix, and they are defined in Eqs. (1) to (5), where TP means 'true positive', FP means 'false positive', TN means 'true negative', and FN 'false negative'.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{1}$$

$$F-measure = 2\times\frac{Precision\times Recall}{Precicion+Recall} \tag{2}$$

$$Precision = \frac{TP}{TP+FP} \tag{3}$$

$$Recall = \frac{TP}{TP+FN} \tag{4}$$

$$Phi\ coefficient = \frac{TP\times TN - FP\times FN}{\sqrt{(TP+FP)\times(TP+FN)\times(TN+FP)\times(TN+FN)}} \tag{5}$$

For computing these metrics, each sample contains N features, covering a time interval of T seconds.

In a crypto-ransomware detection technique based on the dynamic analysis of actions taken by the suspect program, the number of false positives has a more significant effect in the usability of the tool than the number of false negatives. One false negative means that the crypto-ransomware has not been detected in the first T seconds, but it may be detected in the next time window. However, a high number of false positives could reduce user confidence in the detection system, thereby increasing the possibility of the user ignoring a true alarm. Consequently, in the metrics presented, precision is more relevant than recall.

To reduce the number of false alarms and maximize true detections, we tune the value of the time window length parameter (T) in each model (see Section 4.1) The model with the best results is consequently optimized and validated in Section 4.2. On the final evaluation, we have added the following scenario-specific metrics:

- Time for crypto-ransomware detection: The time that the tool takes to detect ransomware. It depends on the window length parameter.
- Data lost before crypto-ransomware detection: The amount of data (measured in megabytes) lost due to encryption. We account for the whole file size even if the crypto-ransomware encrypts only part of the file before it is detected. It is a pessimistic metric, but it is safer to assume that the file may not be recoverable even if only part of it was encrypted.
- Number of working days until a false positive: It is the number of 8-h working days that it is expected to be without a false positive. It is computed based on a dataset containing 309 8-h working days (those not used in the training phase).

[3] https://bigml.com.

Table 2
Test results for DTs.

T. samples (s)	Accuracy (%)	F-measure	Precision (%)	Recall (%)	Phi-coefficient
10	98.9	0.9865	98.5	98.8	0.9779
20	98.8	0.9815	98.1	98.1	0.9725
30	98.7	0.9786	98.4	97.3	0.9697
40	98.8	0.9783	98.6	97.1	0.9704
50	98.8	0.9754	97.9	97.2	0.9671
60	98.8	0.9739	98.5	96.3	0.9661

Table 3
Test results for TEs.

T. samples (s)	Accuracy (%)	F-measure	Precision (%)	Recall (%)	Phi-coefficient
10	99.5	0.9935	99.2	99.5	0.9894
20	99.4	0.9913	99.1	99.2	0.9871
30	99.5	0.9914	99.5	98.8	0.9878
40	99.6	0.9930	99.8	98.8	0.9904
50	99.6	0.9913	99.9	98.4	0.9885
60	99.6	0.9909	99.6	98.6	0.9882

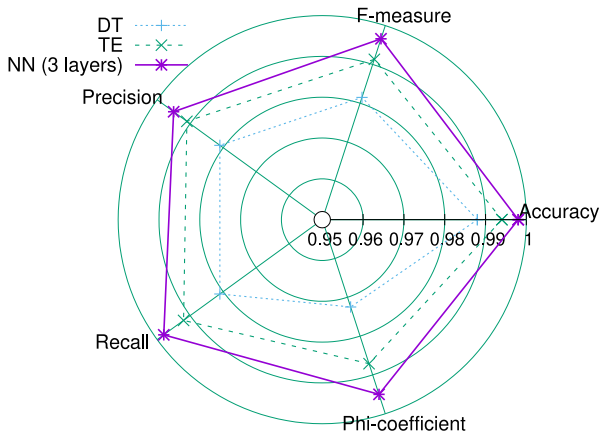

Fig. 8. Comparison of metrics for DT, TE, and NN with T = 20 s.

4. Results of model selection, validation and optimization

We compared three different ML models using the techniques for optimization offered by BigML. These are centred on ML metrics such as those described in Section 3.5. The best model obtained from this broad comparison was studied in detail for specific metrics in our scenario, such as the amount of bytes lost or the time until crypto-ransomware detection.

In this section we present the results from each of these steps.

4.1. ML model selection

We considered six values of interval T, from 10 to 60 s. For each value we prepared the training and testing dataset, trained each of these ML models and computed the classification metrics. These datasets contain a balanced representation of both classes ('infected' and 'not-infected') because they are an 80%–20% random split of a dataset containing 50 h of ransomware traffic and 50 h of benign applications traffic.

The metrics computed for the six values of T are shown in Table 2 (Decision Trees), Table 3 (Tree Ensembles) and Table 4 (Neural Networks). As T increases, the results are expected to improve because a larger number of features allows the model to learn more complex relations. However, using large values of T causes the model to require more time to raise an alarm, allowing the crypto-ransomware to encrypt more files before it is detected. This effect on the detection time is studied in-depth in Section 4.2, after the best model is selected.

All the models achieve high accuracy results (98% or higher). However, the simpler the model is, the less it takes advantage of the larger number of features. Using NNs, both accuracy and precision increase as T becomes larger (larger number of features). Using DTs, both accuracy and precision decrease when the temporal window, T, increases.

Fig. 8 plots the six metrics for the three models, using T = 20 s. To facilitate the comparison, for NNs we plotted only the results with three hidden layers, which offers better results than when using 1 or 2 layers. In Section 4.2, we present a detailed comparison of the other metrics for this model with a different number of hidden layers.

DTs provide the worst results for all the metrics. The difference between DTs and the other models is maintained for other values of T – it is larger for the case of T = 60 s (Table 2). DTs cannot take advantage of all the features offered, unlike TEs and NNs. Therefore, we discard the DT model and focus on TEs and NNs.

NNs provide the best results, with values greater than 99% for all the metrics. Moreover, the phi-coefficient for TEs has the lowest value in the set (98.7%). Although the differences in the NN case are not as large as in the case of DTs, NNs obtain higher values for all the metrics for most of the T values studied (10 to 60 s). In Section 4.2, we discuss the optimization of the NN model studying the effect of the changes in parameter T and the number of hidden layers on the false positive rate and the accuracy results on the unseen crypto-ransomware binaries.

4.2. Neural network model optimization

NNs provide the best results among the models studied; however, they have scope for optimization. We evaluated the true positive and false negative rates using the time until ransomware is detected (true positive) in the 'infected' dataset. We computed the average and maximum time required by the model to raise an alarm for the 150 crypto-ransomware traffic traces available. Each model requires to wait for at least T seconds to raise an alarm because T is the time needed to build a single sample. **All NN models in this section can detect all the crypto-ransomware binaries in our whole dataset** – this includes those binaries not used in the training phase (unseen malware) – however, different configurations require different time to detection. Table 5 lists the results for NNs with one, two, and three hidden layers with different configurations of the time window.

The last column in Table 5 presents the maximum time required to detect a crypto-ransomware in the 'unseen' dataset. These strains were not used in the training phase, as explained in Section 3. This is the most significant result because it reveals the robustness of the system to the ever-changing environment of malware. All the models detect the unseen crypto-ransomware in the time required to build a sample (T seconds) except for two scenarios: (a) when T = 10 s, there are some strains that require two consecutive samples (20 s) to be detected, and (b) for the 1-hidden layer model and T = 60 s, two samples may be required, raising the maximum detection time to 120 s.

The maximum detection time is also high for the training dataset when using the 1 or 2 hidden layer models and $T > 10$ s. The maximum detection time is 160 s when using 2 hidden layers and T = 20 s and it is larger than 15 min (1050 s) when using only 1 hidden layer and T = 50 s. This means that some ransomware strains can run for more than 15 min, encrypting user files, before they are detected. Taking into account only the tool effectiveness to detect ransomware in action we should select a 3-hidden layers model or a 1-hidden layer model but the last case only using T = 10 s. The 3-hidden layers model is complex but has an adequate behaviour for any value of T between 10 and 60 s. The 1-hidden layer model is simpler, but it can take too long to detect some ransomware families, so it should only be used with T = 10 s.

However, we cannot base model optimization only on detection effectiveness. The system will not be usable if it frequently raises an

Table 4
Test results for NNs with 1, 2 and 3 hidden layers.

No. of hidden layers	T. samples (s)	Accuracy (%)	F-measure	Precision (%)	Recall (%)	Phi-coefficient
1	10	99.1	0.9886	99.0	98.7	0.9813
	20	99.1	0.9868	99.2	98.2	0.9804
	30	98.8	0.9789	98.9	96.9	0.9703
	40	99.2	0.9854	98.7	98.4	0.9810
	50	99.3	0.9860	99.3	97.9	0.9813
	60	99.5	0.9896	99.4	98.6	0.9865
2	10	99.1	0.9887	98.9	98.8	0.9816
	20	99.3	0.9888	99.2	98.6	0.9833
	30	98.9	0.9807	98.4	97.8	0.9726
	40	99.2	0.9849	98.6	98.4	0.9793
	50	99.3	0.9853	99.5	97.6	0.9804
	60	99.6	0.9910	99.1	99.1	0.9882
3	10	99.7	0.9959	99.7	99.4	0.9933
	20	99.8	0.9966	99.5	99.8	0.9950
	30	99.8	0.9971	99.6	99.8	0.9959
	40	99.8	0.9962	99.6	99.7	0.9948
	50	99.9	0.9974	99.7	99.7	0.9965
	60	99.9	0.9987	99.7	100	0.9983

Table 5
Validation results for NNs using training binaries or unseen ransomware.

Hidden layers	T (s)	False positives			Average/max detection time (s) (training binaries)	Maximum time to detect any unseen binary (s)
		Number	%	Average number of workdays until a false positive occurs		
1	10	207	0.0230	1.49	10.88/30	20
	20	25	0.0057	12.3	22.62/260	20
	30	9	0.0031	33.6	34.13/630	30
	40	6	0.0027	51.4	41.9/280	40
	50	4	0.0023	75.4	52.46/1050	50
	60	3	0.0021	99.2	62.17/540	120
2	10	386	0.0430	0.8	10.68/40	20
	20	76	0.0174	4	21.37/160	20
	30	26	0.0090	11.5	31.4/180	30
	40	11	0.0051	27.2	42.5/240	40
	50	9	0.0049	34.7	50.7/700	50
	60	6	0.0042	49.6	61.3/300	60
3	10	76	0.0080	4.07	10.2/20	20
	20	35	0.0080	8.84	20.2/40	20
	30	12	0.0041	25.8	30.2/60	30
	40	7	0.0032	43.8	40/40	40
	50	7	0.0040	43.8	50/50	50
	60	4	0.0028	74.8	60/60	60

alarm when benign applications are accessing shared documents. For the evaluation of these false positives, we used the real user samples labelled as 'not infected'. They contain more than 10 months of 8-h working days of traffic. We counted the number of T-seconds-long samples mistakenly classified as 'infected' for each value of T. Furthermore, we calculated the average number of 8-hour working days before raising a false alarm, which has an inverse relationship with the false positive rate. The results of false positives are in the third, fourth and fifth column in Table 5. The number of false positives gets reduced when increasing the number of layers or the time window T. This reduction results in larger periods of time before a false alarm is raised when only benign applications are used.

Looking at the average number of days until a false positive, the 1-hidden layer model obtains the best results for all configurations except for T = 10 s. When T = 10 s, the 1-hidden layer model offers an average of 1.49 days between false alarms; meanwhile, the 3-hidden layer model increases this value to 4.07 days. The best result is obtained using T = 60 s and 1 hidden layer, where only one false positive is expected every 99 8-h working days. However, this behaviour is the result of also not detecting some crypto-ransomware strains early enough. The 1-hidden layer model is conservative in raising an alarm, providing good results of false positive rate at the expense of late crypto-ransomware detections. We had established that this model shows good detection results only when T = 10 s. However, for such a low value of T the 1-hidden layer model does not provide the lowest false positive rate, but it is surpassed by the 3-hidden layers model. Therefore, we discard the 1-hidden layer model as the best option.

The 3-hidden layers model provides a false positive rate lower than the 2-hidden layers model for any configuration and it also shows the best behaviour when T = 10 s. This means that for low T, the 1-hidden layer model shows low maximum detection times, but it does not offer the best result in false positive rate. The best result is offered by the 3-hidden layers model, which also offered good detection results. Although the 3-hidden layers model requires more time for training and classifying each sample, the model is trained only once after its installation; therefore, the time utilized for training is not determinant. The NN receives one sample every T seconds, and NN models using hundreds of neurons can be evaluated in sub-second time on a single-core CPU; therefore, the classification time is also not a cause of concern.

Once we have selected the model with best results, we should select the best value of T. No detection can happen before a single sample is read, which requires the traffic in a time window T. The 3-hidden layers model offers detection in one time window except for T = 10 s that requires a second sample for some crypto-ransomware strains, taking at most 20 s to detect the crypto-ransomware.

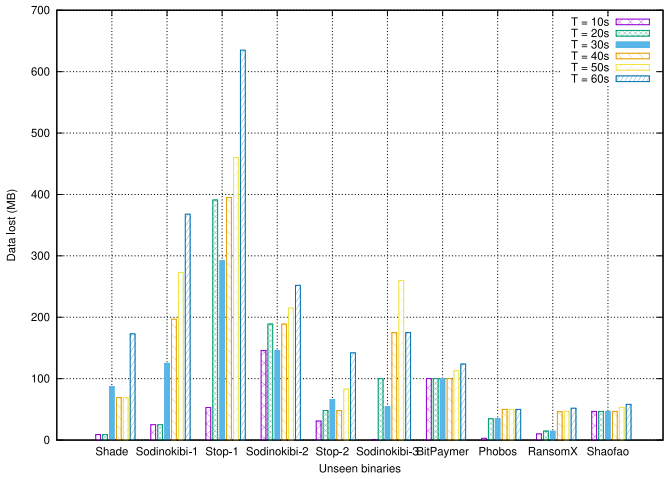

Fig. 9. Data in MBs lost for each unseen binary for different configurations of T (3-hidden layers NN).

Fig. 9 shows the number of bytes encrypted for each unseen binary before successful detection using the 3-hidden-layer model. Owing to the differences between the encryption processes of different ransomware binaries, the number of bytes encrypted can vary greatly among them. Some ransomware strains encrypt files in the alphabetic order, whereas others do it according to size (Berrueta et al., 2020); this is the reason for the differences in the number of bytes encrypted.

For T = 60 s, the average number of bytes lost for unseen ransomware binaries is 202 MB, whereas for T = 10 s, it reduces to 42 MB. Crypto-ransomware such as BitPaymer are considerable insensitive to parameter T, whereas others such as Stop-1 can vary from encrypting less than 100 MB of data when T = 10 s to encrypting more than 6 times this value when T increases by 6 times (T = 60 s). One of the objectives of the tool presented in this paper is quick detection of crypto-ransomware; therefore, we recommend that parameter T should not be greater than 30 s. This allows unseen binaries to encrypt an average of 99 MB of data. By using T = 30 s, a false positive rate of 0.0041% and one expected false alarm only after 25 working days are observed.

Depending on the effect of the alarm on the user, shorter values of T can be configured. If the alarm causes only a warning, it may be acceptable to have one false positive in 4 days (T = 10 s). This would result in less data loss in case of ransomware infection. If the alarm is annoying to the user, larger values must be used for T. Parameter T is tunable through the tool or network administrator, and it depends on the network, user, and file server characteristics.

Once the time-window length T is decided we can evaluate different options in the process of sliding the time window to create the input samples from the network traffic. Table 6 shows the number of megabytes lost for each of the crypto-ransomware binaries in the 'unseen' set. Different values of the sliding step were evaluated, from 1 s (the minimum interval to extract the features) to 30 s (the selected window length). Table 7 contains the number of false positives for each of these configurations.

The data in Table 6 does not show clear evidence about the benefits of shortening the window sliding step in terms of reducing the amount of data lost before detection. The average value of megabytes lost for 5 s of sliding window step is smaller than for 30 s, but less than 20%. Analysing the values for each unseen binary, all except Stop-1, Sodinokibi-1 and RansomX encrypt less data for shorter sliding steps, however, the differences are not very significant. The only exception is Sodinokibi-3, that for less than 10 s window sliding step encrypts 0.27 MB before detection while for 20 s step it encrypts more than 50 MB.

Table 6
Megabytes of data lost before detection of each unseen binary for different sliding steps.

Unseen binary	1 s (MB)	5 s (MB)	10 s (MB)	20 s (MB)	30 s (MB)
Shade	75	40	82	33	87
Sodinokibi-1	167	129	129	129	129
Stop-1	351	365	398	478	328
Sodinokibi-2	142	164	188	188	146
Stop-2	60	59	68	68	68
Sodinokibi-3	0.27	0.27	0.27	54	54
Phobos	12	13	13	14	41
BitPaymer	33	33	51	33	53
RansomX	240	365	406	423	36
Shaofao	44	36	46	51	46
Average (MB)	112	80	138	147	99

Table 7
Number of false positives for 10 months of 8-h working days and each sliding step.

	1 s	5 s	10 s	20 s	30 s
False positives	896	166	49	18	12

In terms of false positives, the data in Table 7 shows a detriment when the sliding step is shortened. When the sliding step is set to 30 s there are 12 false positives in 2477 h of not infected traffic, or an equivalent expected alarm every 25 working days. However, a sliding step set to 1 s results in almost 3 alarms every day. As we have already discussed, a high number of false alarms could disturb the users and result in the detection tool being deactivated, thus we conclude that the detriment in shortening the sliding step overcomes the slight benefits.

4.3. Model validation using different file-sharing protocols

The training, optimization and evaluation process has been carried out using SMBv2 traffic. It is unencrypted traffic, but we have used the features described in Section 3.3, which do not require protocol analysis. The features can be extracted from encrypted traffic such as SMBv3 or from different file-sharing protocols such as NFS (encrypted or not) without protocol analysis. In this section we evaluate the hypothesis that all these protocols behave in a similar manner. They all translate operating system file access primitives into network protocol messages, and they should create very similar traffic patterns. This behaviour should allow the trained model to detect crypto-ransomware activity, even in scenarios using a different file-sharing network protocol.

To test our hypothesis we have run the 'unseen' binaries using SMBv3 and NFS as the file-sharing protocol. We have set a new scenario using a Windows 10 client and an Ubuntu file server sharing the same directory as the one used for training the model. Windows 10 was the required operating system in the client to obtain support for the SMBv3 and NFS configurations, however, the result was that some of the unseen binaries could not run in this environment. Those inactive binaries are marked in Table 8. Each crypto-ransomware binary was run three times: once sharing the directory using SMBv2, another one using SMBv3, and the last one using NFS.

All these binaries, in all file-sharing protocol scenarios, were successfully detected using the 3-layer neural network trained using SMBv2 samples. For each binary and protocol, we calculated the detection time since the beginning of activity. For comparing the results, the time to detection is not very relevant because each crypto-ransomware variant could wait an unknown time from its execution until beginning the encryption of the shared volume. We also calculated the amount of data read and written until being detected by the tool (see Table 8). In the training and optimization experiments we calculated the amount of data encrypted by ransomware, thanks to SMBv2 being not-encrypted. In the SMBv3 case the protocol messages are encrypted and from the network traffic we cannot identify the different disk access actions. We carried an estimation of the amount of lost data based on the amount

Table 8
Time for detection and data read and written by each binary and file-sharing protocol before detection.

	Time for detection (s)			MB read before detection			MB written before detection		
	SMBv2	SMBv3	NFS	SMBv2	SMBv3	NFS	SMBv2	SMBv3	NFS
Shade	226	*N/A*	*N/A*	88.7	*N/A*	*N/A*	88.8	*N/A*	*N/A*
Sodinokibi-1	227	*N/A*	*N/A*	101.5	*N/A*	*N/A*	16.9	*N/A*	*N/A*
Stop-1	1265	1265	1046	15.1	31.1	40.7	13.3	31.1	20
Sodinokibi-2	245	*N/A*	*N/A*	44.2	*N/A*	*N/A*	44.1	*N/A*	*N/A*
Stop-2	916	*N/A*	*N/A*	55.5	*N/A*	*N/A*	55.4	*N/A*	*N/A*
Sodinokibi-3	167	*N/A*	*N/A*	202.5	*N/A*	*N/A*	165.4	*N/A*	*N/A*
bitPaymer	1029	1113	621	128	108	17.6	126.4	101.2	15.5
Phobos	713	752	694	20.2	16	20.8	24.9	20.1	15.7
RansomX	1028	823	716	88.5	21.2	35.4	56.1	13	35.5
Shaofao	694	840	691	16.9	45.7	18.8	21.3	55.8	20.4
Average				76.1	48.9	29.5	61.2	48.1	25.6

of bytes read and written at the TCP layer. We used the same procedure for NFS traffic. Table 8 shows, for each crypto-ransomware binary and protocol, the time before detection, and the bytes read and written up to that time.

The read data can be considered as the original file data read and encrypted by the ransomware, and eventually lost. However, sometimes the ransomware reads the files a second time, increasing this count beyond the real amount of lost data. Some binaries do not read the whole file but only a fraction of it and encrypt only that fraction. The read data before detection is therefore an estimation of the lost data and we cannot generalize it as any kind of bound.

The data written by each variant can be understood as the original data the ransomware has overwritten. As for the case of read data, ransomware frequently writes additional files (e.g., containing payment instructions), actions that will cause more written bytes than the real lost ones. Some ransomware binaries compress the data or only encrypt a fraction of the file content, therefore writing a smaller amount of bytes than those read. The written data is also neither a higher nor a lower bound but an estimation of the amount of lost data.

Taking into account all these aspects on the evaluation of lost data and witnessing the results, the model detected all crypto-ransomware variants in the three scenarios: using SMBv2, SMBv3, or NFS; with a similar precision to using SMBv2 (the training scenario). This is very relevant because the model was never trained using SMBv3 or NFS traffic, but it generalized correctly the traffic patterns from the samples obtained from SMBv2 traffic. In SMBv2 and SMBv3 the amount of data lost is similar (around 50 MB), while in the NFS scenario the amount of read and written data is considerably smaller. Analysing the traffic traces from a crypto-ransomware family and all file-sharing protocols, we noticed that the read and write speed was lower using the NFS protocol, which makes sense because the client is a Windows machine and for mixed environments (Linux and Windows) the recommended protocol is SMB (NetGear, 2016). This lower speed allowed the model to detect the activity before a greater number of files were lost.

In conclusion, the neural network model using T = 30 s can detect ransomware binaries in SMBv2, SMBv3 and NFS scenarios, even when being trained using only traffic from the first one.

5. Discussion and comparison with previous research

Deep learning and other ML models are popular techniques in the literature on malware detection, including ransomware detection, therefore the methodological approach taken in this article is not presented as novel. However, the file-sharing scenario we have described has not been significantly considered in the literature of ransomware detection. Network shared volumes are common practice in the corporate environment, but the literature on ransomware countermeasures has not deeply studied the advantages of detection based on file-sharing traffic. To the best of our knowledge, the work on (Morato et al., 2018) is the only one that takes a similar approach, but it requires unencrypted SMBv2 traffic, which is accepted but not required in this article. Some of the advantages present in this scenario are also present for detection tools based on the analysis of network (not file-sharing) traffic, while some others are the result of analysing file-sharing traffic.

We have organized this discussion first on the advantages of an analysis tool based on file-sharing traffic, compared to other tools based on traffic. Following this discussion, we compare effectiveness and limitations between the deep learning model presented in this article and the results on the previous literature. To conclude the section, we discuss the limitations and our perspectives of future work.

5.1. Advantages and caveats in a passive file-sharing traffic analysis scenario

Antivirus software is most often installed on the users' host. Crypto-ransomware detection tools are a specific case of antivirus tool, therefore they work in a similar manner (see for example Continella et al., 2016; Kharraz et al., 2016). In a file-sharing environment, antivirus or crypto-ransomware detection tools can also be installed at the server, where they can monitor all files written and detect encrypted content. However, in a file-sharing scenario, a large amount of information about user actions can also be obtained from network traffic, which offers the following advantages:

- Unintrusive to the client: Antivirus software running at the client implies being potentially intrusive to the user and consuming computer resources that could affect computer responsiveness. However, local files tend also not to be the critical documents. They are the operating system and main programs files, which in most occasions offer only a remote desktop client environment. Running in an independent hardware (the network monitor), our solution does not affect computer responsiveness.
- Unintrusive to the server: The deployment of a detection tool at the server requires resources at the server and any CPU impact or disk access at the server can translate into performance degradation for all the clients. Again, this problem does not exist for a network-deployed monitor.
- Unintrusive to the traffic: Firewall appliances are deployed in-path, meaning that all traffic analysis adds a packet processing time and in high bit rate traffic scenarios can affect file-sharing protocol performance. We suggest deploying the analysis tool off-path, monitoring a copy of the traffic. To block the traffic from an infected computer, the tool can program access rules in a software defined network control plane, in a firewall existing in-path (without requiring the firewall to run the analysis software), or in the file server, removing the user access privileges.
- Easy updates: A tool, such as the one described in this article, deployed off-path, is a single point to manage and its updates do not affect file-sharing service, compared to deploying it at the clients or the server.
- Malware resistance: A tool deployed off-path is a single point to manage, and it can be easily protected from attacks that

could escalate privileges and deactivate the security software. The network monitor does not require real network access, but only monitoring a copy of the traffic, therefore it cannot receive network attacks. In case of requiring access to a network control plane, it should be connected only to the control network, not being accessible from the production network. It also does not have a console user, therefore it is not susceptible to social-engineering attacks.

Some previous ransomware detection tools in the literature take advantage of information obtained from network traffic. Most of them cannot take advantage of above-mentioned characteristics because they not only use information from network traffic but they also require to be installed at the hosts to capture other fundamental metrics (Hasan & Rahman, 2017; Lu et al., 2017). A shorter list of the literature takes exclusively information from network traffic. They are not based on file-sharing traffic but on DNS traffic or TCP connections to certain IP addresses (Ahmadian et al., 2015; Quinkert et al., 2018).

In Cabaj and Mazurczyk (2016) the tool blocks the access to certain IP addresses after analysing the DNS requests. The malicious addresses must be in a blacklist that should be updated as soon as new ransomware C&C servers appear. The use of domain generation algorithms (DGA) by some crypto-ransomware strains makes this task difficult, and the tool cannot detect ransomware that does not need to contact a C&C server. Thus, the tool presented in Cabaj and Mazurczyk (2016) cannot detect as many strains as the tool presented in this article.

In Ahmadian et al. (2015) the authors use DNS traffic to recognize the use of randomly generated domain names to locate the command and control (C&C) host. This detection technique cannot be applied to crypto-ransomware strains that do not use a DGA and therefore it is less generic than the model presented in this article.

The authors of Almashhadani et al. (2019) analysed the network behaviour of ransomware Locky. The tool's efficiency in detecting other crypto-ransomware strains has not been verified, and the tool could not detect ransomware that does not contact a C&C server. They selected some TCP-level features, in addition to DNS features (domain names or DNS request failures). The TCP features are for example the number of segments with the RST flag active or the number of hypertext transfer protocol (HTTP) POST requests sent by the user. Although they present a thorough behavioural analysis, it is focused on the ransomware Locky.

5.2. Comparison of effectiveness

When comparing crypto-ransomware detection proposals there are three aspects we must take into account and that limit the comparison: the ransomware strains analysed, the metrics used to evaluate performance, and the reproducibility of each technique. We analyse these aspects in the following discussion.

In this article we have presented a detection technique applicable to any crypto-ransomware. In comparison, there is a subset of the literature that describes ransomware detection techniques targeting a single ransomware strain (Ahmadian & Shahriari, 2016; Alam et al., 2018; Feng et al., 2017; Kara & Aydos, 2022; Quinkert et al., 2018). They base the detection on searching for some special behaviour shown by the selected strain and therefore they are hardly generalizable to other ransomware families. Those techniques have not been compared with other families and in most cases they cannot be applied because the mechanisms they search for are not present in different strains. We will not elaborate any further in the comparison with these proposals because they do not apply to a realistic production environment where any ransomware strain can infect a host and it is a well documented problem (McIntosh et al., 2021).

The rest of the literature takes a set of ransomware binaries from some repositories. They can be organized in strains or families (Ahmadian et al., 2015; Scaife et al., 2016; Sgandurra et al., 2016) or they can be a large set that results from searching in a database using ransomware-related keywords (Chen et al., 2017; Kharraz et al., 2016; Lu et al., 2017). In recent years we have witnessed an important increment in new ransomware appearances. Depending on the age of the ransomware detection proposal, it has been tested with a different set of strains. Algorithms described in decade-old papers were designed to detect ransomware binaries that we cannot run nowadays because their C&C servers are down or the DNS names they try to resolve are blocked. Comparison with these papers is problematic because their results cannot be generalized to new ransomware strains and new methods cannot be tested with the families present at their time.

The metrics used to evaluate performance of the detection algorithm are usually centred on the capability to detect the crypto-ransomware and on its erroneous classification of benign applications as ransomware. These two metrics relate to the true positives and the false positives in a binary classification problem. Crypto-ransomware detection can be measured on the basis of whether a ransomware binary is recognized as malware or not; it can also be measured using the time the algorithm requires to classify it as malware, or it can be measured as the amount of data lost before detection. Some of these metrics depend on the environment (computing power, disk files distributions, simultaneous user actions) and there is no consensus on a single metric whose value could be compared in the same circumstances in future works. A similar problem arises when discussing false positive classification, where critical parameters are for example the type or number of benign applications being run, or the user activity pattern. Due to the different metric definitions, any objective comparison of numeric results is not reliable and we will have to consider what each metric is really measuring and in which context.

Finally, some of these handicaps can be solved if previous proposals could be tested against new data. This requires a clear definition of the algorithms, making their implementation reproducible. A second option is publishing all the data used, so new algorithms can be compared in the same scenarios. It is not the objective of this section to enumerate the literature on crypto-ransomware detection which lacks comparability due to unclear algorithm definitions or unpublished data. We just tried to make our best contribution to facilitate future comparisons by making available both the dataset used in this article and the neural network that we obtained after training (Berrueta et al., 2021).

Taking all these aspects into account, we proceed to discuss the results in previous papers having a number of ransomware strains similar to those in this study. The tool we have described requires only a single 30-seconds sample from network traffic to detect more than 99% of the crypto-ransomware samples we tested (150 traffic traces from more than 30 strains appearing in a period of five years). In the remaining cases (less than 20%), 2 samples (60 s) are required. We have divided the existing tools into two groups depending on where they have to be installed.

5.2.1. Locally installed tools

These tools must be installed in each host that has access to the shared files to provide effective protection. They need some features that can only be obtained from the local machine. Therefore, all of them require a higher management burden than a single monitoring probe and present some impact on end-host performance.

Scaife et al. (2016) measured disk-access actions from each running process. Results of testing the described tool using ransomware binaries from 24 different strains revealed 100% detection rate and one in 30 benign applications causing a false positive (applications selected due to their similarities with ransomware behaviour). On average, 10 files were encrypted before malware detection, although it depends on the file-size and on the ransomware strain. In the case of our tool, 51 files were lost on average in a worst-case scenario of 10 unseen ransomware binaries. Our ransomware dataset was collected between 2015 and 2020, whereas the tool in Scaife et al. (2016) was trained and evaluated using ransomware binaries that appeared up to 2016, and it has not

been tested with more recent ransomware strains such as WannaCry, notPetya, or Phobos.

Sgandurra et al. (2016) developed an ML-based tool using a selected set of features obtained either statically from the binary program or dynamically while it is running. They tested their tool using ransomware binaries from 11 different strains that appeared before 2016, and 942 manually executed user applications. They achieved a detection rate of 96.3% and false positive rate of 1.6%. We have presented a solution that improves the results in terms of detection and false positive rate, using a more modern set of ransomware strains and an unrestricted set of benign applications run from an uncontrolled population.

Zuhair et al. (2020) proposed a hybrid ML model that combined Naive Bayes and Decision Trees, and used 10 static and 14 dynamic features. The model classified unseen ransomware samples into their ancestral families. The authors used 35.000 ransomware versions from 14 ransomware strains, adding 500 versions of malware and 500 benign applications, obtaining 97% accuracy in the classification. Although they used a large corpus of malware binary versions, our tool improves the results in terms of accuracy using a higher number of ransomware strains for evaluation.

Mehnaz et al. (2018) described a software solution named RWGuard. The authors trained a Random Forest classifier based on the malware disk access operations, and adding the monitoring of deployed decoy files they achieved a detection rate of 100% and a false positive rate of 0.9%. The proposal can detect ransomware before the encryption of 10 files. This solution, however, introduces significant latency in I/O operations and overloads the user machine. With our tool, the results concerning false positive rate (0.004%) were improved, and there was no added latency.

Continella et al. (2016) computed the number of different types of file access operations in time intervals of various sizes. They created a multi-tier structure of sets of Random Forest classifiers, using each set to observe the data from a different scale. They added file recovery capabilities by maintaining a copy of every file being edited while the classifier is validating the operations on it. The detection and false positive rates were 97.7% and 0.038% respectively. Similar to Scaife et al. (2016), the ransomware dataset covered only ransomware strains that appeared until 2016. We improved the detection results with a more modern experimental set. We can also offer file recovery using the network-based analysis probe but only in case of the file-sharing protocol being not encrypted. Non-encrypted protocols are for example versions 1 and 2 of SMB, and we showed a proof-of-concept of the file-recovery feature in a previous publication (Berrueta et al., 2018).

Moussaileb et al. (2018) monitored the file system traversal paths and velocity of the analysed programs. These features were the input to multiple classifiers (k-nearest neighbours, DTs and random forests) and the final output was the result of a majority vote among the classifiers. They obtained a detection rate of 99.35% and a false positive rate of less than 1%. The detection was based on ransomware file traversal behaviour that could easily change in future strains, therefore becoming undetectable. In comparison, our trained ML model detected all the modern ransomware strains that were not used in the training phase and is not restricted to any specific file-traversal behaviour; it is based on the file reading and writing ransomware activity, which is unavoidable to encrypt the files.

Ramesh and Menen (2020) monitored the modification of the entropy of file contents, the use of system resources, the implementation of persistence to system reboots and the possibility of lateral movements of the malware. These behaviours were used in a finite state machine (FSM) to observe whether the malware actions took the FSM to a state matching an infection or not. The authors evaluated their model using 1500 benign apps and 475 ransomware binaries, reaching 99.5% accuracy. However, there were 9 non-detected binaries, while our model could detect all ransomware binaries, either used in the training phase or not.

Arabo et al. (2020) monitored CPU, RAM usage and disk-access operations. They combined weighted average computations on the time series of some of these metrics with a machine learning technique for those where it was more suitable. They did not describe the ML technique used, and they analysed only 7 ransomware binaries. They did not perform zero-day evaluation of the solution (i.e., the evaluation using ransomware unseen during the training process), while we clearly described the learning phase in our model, using ten times the number of ransomware binaries, and we validated the model using 10 unseen binaries from 7 different families.

Al-rimy et al. (2019) analysed the dynamic call to library functions from 15 ransomware families before any call to cryptography-related procedures (pre-encryption phase). After pre-processing the extracted features, the authors trained 7 different ML techniques and used a majority voting strategy to obtain the classification. The system reached a 91.9% accuracy with 1.85% of false positive rate. We improved both figures, validating the results with a larger training set of ransomware families and unseen families.

Zhang et al. (2020) extracted the instruction machine codes or operation codes (opcodes) by disassembling the program file under analysis. They created features based on sequences of the opcodes, and used a combination of self-attention convolutional neural networks and bi-directional self-attention network to classify these sequences. They achieved an 89.5% accuracy, 87.5% precision, 87.6% recall and 87.3% F1-measure. Their main limitation is that they performed static analysis of the binaries, which is prone to false negatives, while our architecture is based on dynamic analysis. They also used only 8 different ransomware families in their study.

Ahmed et al. (2021) used 292 ransomware binaries (both crypto-ransomware and locker-ransomwares) from 67 ransomware families. They analysed the system events generated by the programs; events such as reading or writing files, launching processes or loading and unloading dynamic libraries. They detected high correlation between some of these events, and used them to train a Support Vector Machine (SVM) and a Bidirectional Encoder Representation for Transformers (BERT). They obtained a classification accuracy of 99.52%, precision 99.41%, recall 99.63%, and F1-measure 99.52%. Their main limitations are the resource consumption in the user machine, and its tendency to cause false positives. Our solution does not consume any resources at the users' computers, and using neural networks with 3 hidden layers we obtain better results in all four metrics.

Roy and Chen (2020) developed a solution based on events captured in hosts. Analysing them when there was a ransomware in action and when there was not, they could detect abnormal events that would indicate the presence of ransomware. They compared five deep learning models and achieved 99.87% detection accuracy using a BiLSTM-FC (Bi-directional Long Short Term Memory with a Fully Connected layer) and 17 different ransomware strains, obtaining 0 false positives. No experiments with unseen samples were performed in this study; therefore, the solution could have problems in detecting ransomware strains that are not present in the training phase. They overcame the resources' consumption problem by sending the logs to a server for the analysis.

5.2.2. Network based tools

These tools do not require their installation in the user's machine because they do not need any information monitored locally. The main advantages compared to the locally installed group is that they do not consume resources in the user's machine, and they cannot be deactivated by ransomware that escalates privileges in the infected host. However, they do not have access to local host information, which can hamper their capabilities.

Chadha and Kumar (2017) detected whether the names in DNS requests were generated by a domain generation algorithm or not. They compared supervised and unsupervised ML algorithms, obtaining an optimum configuration with a detection rate of 85% and false positive

rate less than 10%. They trained the algorithms with 101 domain names from 3 ransomware variants and evaluated them with 30 domain names not used in the training phase (zero-day scenario). Ransomware detection based on DNS traffic fails for ransomware strains that do not need to contact an external server or they establish the connection after the encryption process, such as CTBLocker, DMALocker or Crysis.

Almashhadani et al. (2019), similar to Chadha and Kumar (2017), detected ransomware by analysing the DNS requests. They added extra information from HTTP and TCP protocols, and based the classification process on per-packet and per-stream measures. They compared the results from different ML algorithms, including DTs and TEs. They achieved a detection rate of 97.8% and a false positive rate of 0.04%. Compared to Chadha and Kumar (2017) the authors added some extra characteristics, however, they were extracted from the traffic between the ransomware and its C&C server. Thus it fails for ransomware families that do not need to contact external servers.

Morato et al. (2018) described a tool that is most similar to the one presented in this paper. It detects crypto-ransomware by analysing traffic from SMBv2. It achieves 100% detection rate and 1 out of 10 billion false positives. However, owing to the features it extracts from the traffic, it is not applicable to an SMBv3 scenario, wherein the file-access commands are encrypted. We have generalized the scenario with a new tool, capable of crypto-ransomware detection in encrypted file-sharing scenarios or using different file sharing protocols. The encryption, or protocol-agnostic restriction, limits the available information in network traffic, causing the false positive rate to increase compared to Morato et al. (2018). However, the tool can still detect the unseen crypto-ransomware binaries downloaded from Berrueta et al. (2020), losing an average of only 37 MB of data before detection.

5.3. Limitations and future work

We have shown that, compared to relevant papers in the literature, the detection technique presented in this paper obtains similar or better ransomware detection results than those previously achieved. Also, our tool deployment scenario, based on file-sharing network traffic analysis, is clearly advantageous from a management perspective, avoiding any slow down in user host performance. However, it is not a fair comparison if we do not highlight the caveats still present in our proposal. This is the task we undertake in this section, offering as much a critical analysis as possible.

We have described a crypto-ransomware detection technique based on file-sharing traffic from Microsoft Windows desktop (or laptop) computers that are the most frequent potential target of ransomware infections. We did not consider mobile operating systems. We believe a file-sharing scenario is unlikely for these systems.

The architecture described is only applicable to scenarios where the important files are stored in a file server. This is the common scenario for an enterprise deployment, where we have placed our focus. It is a less useful tool in the home environment.

The detection algorithm was evaluated using 33 crypto-ransomware families, but file-less ransomware cases could not be included because we could not reproduce those infection scenarios. However, as long as their objective is locking access to files, they are expected to present a behaviour similar to the one described for the ransomware families considered in this work.

Only crypto-ransomware is being detected, and not other descendants of a generic ransomware category, such as malware doing only data exfiltration.

We have described a static solution, trained using crypto-ransomware strains from 2015 up to 2020 that we expect to be valid when new families appear, however, nothing in the analysis supports this affirmation. We pretend to create better adaptive training methodologies where new ransomware strains can be incorporated into the ML model and evaluate the improvement or deterioration of the results.

6. Conclusions

In this paper, we have established a deep learning model capable of detecting crypto-ransomware while the malware is reading and writing files in a network-shared volume. No previous study has targeted this scenario, which is a very common scenario in corporate networks. The input set of features to the model described not only the intensity of file access activity, but also the number of files accessed, through a novel feature named as the number of short commands. These commands are recognized in encrypted traffic and in both SMB and NFS traffic by the exchanged sequence of small packets. They serve as a differentiator between crypto-ransomware and benign application activity.

We validated the tool using more than 70 crypto-ransomware binaries acting in a file-sharing scenario using encrypted or unencrypted protocols. The tool works with a copy of the traffic obtained from a network switch; therefore, it does not affect user activity.

We explained the feature extraction and sample reduction processes before the selection of the best ML model. The comparison between decision trees, tree ensembles, and neural networks reveals that neural networks provide the best results using 3 hidden layers of neurons. The validation reveals that the model has a false positive rate of 0.004% with more than 2400 h of real user traffic. It can detect all ransomware binaries used in the training phase in an average time of 30.2 s. It detects 100% of a set of 10 crypto-ransomware binaries not used in the training phase, losing only an average of 99 MB of user data before detection.

The time window length is a tuneable parameter in the feature extraction process, and it must be configured depending on the scenario and user and server characteristics. The best trade-off in the results was obtained with a time window of 30 s. Using larger values of time window, the ransomware encrypts a significant number of bytes (more than 100 MB of data on average). Shorter time windows result in a higher number of false positives, which could annoy the network administrator and make the tool useless. Finally, we compared the model in this paper with other crypto-ransomware detection tools in the literature. Despite the novelty of the scenario that hinders comparison, the tool improves most of the results found in the literature.

CRediT authorship contribution statement

Eduardo Berrueta: Methodology, Software, Validation, Investigation, Resources, Visualization, Writing – original draft, Writing – review & editing. **Daniel Morato:** Conceptualization, Methodology, Software, Formal analysis, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Eduardo Magaña:** Conceptualization, Writing – original draft, Funding acquisition. **Mikel Izal:** Methodology, Validation, Writing – original draft, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data is available at (Berrueta et al., 2021).

Acknowledgements

Funding

This work was supported by Spanish Ministry of Science and Innovation through project PID2019-104451RB-C22/AEI/10.13039/501100011033. Open access funding provided by Universidad Pública de Navarra.

References

Agrawal, R., Stokes, J. W., Selvaraj, K., & Marinescu, M. (2019). Attention in recurrent neural networks for ransomware detection. In *ICASSP 2019 - 2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 3222–3226).

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). http://dx.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)* (pp. 79–84). IEEE.

Ahmed, M. E., Kim, H., Camtepe, S., & Nepal, S. (2021). Peeler: Profiling kernel-level events to detect ransomware. In E. Bertino, H. Shulman, & M. Waidner (Eds.), *Computer security – ESORICS 2021* (pp. 240–260). Cham: Springer International Publishing.

Ahmed, Y. A., Koçer, B., Huda, S., Al-rimy, B. A. S., & Hassan, M. M. (2020). A system call refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection. *Journal of Network and Computer Applications*, *167*, Article 102753.

Al-rimy, B. A. S., Maarof, M. A., & Shaid, S. Z. M. (2019). Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. *Future Generation Computer Systems*, *101*, 476–491.

Alam, M., Bhattacharya, S., Mukhopadhyay, D., & Chattopadhyay, A. (2018). RAPPER: Ransomware prevention via performance counters. URL http://arxiv.org/abs/1802.03909.

Almashhadani, A. O., Kaiiali, M., Carlin, D., & Sezer, S. (2020). MaldomDetector: A system for detecting algorithmically generated domain names with machine learning. *Computers & Security*, *93*, Article 101787. http://dx.doi.org/10.1016/j.cose.2020.101787.

Almashhadani, A. O., Kaiiali, M., Sezer, S., & O'Kane, P. (2019). A multi-classifier network-based crypto ransomware detection system: A case study of locky ransomware. *IEEE Access*, *7*, 47053–47067.

Apple (2012). Apple filing protocol programming guide. URL https://developer.apple.com/library/archive/documentation/Networking/Conceptual/AFP/Concepts/Concepts.html Last access: August 2022.

Arabo, A., Dijoux, R., Poulain, T., & Chevalier, G. (2020). Detecting ransomware using process behavior analysis. *Procedia Computer Science*, *168*, 289–296. http://dx.doi.org/10.1016/j.procs.2020.02.249.

Berrueta, E., Morato, D., Magaña, E., & Izal, M. (2018). Ransomware encrypted your files but you restored them from network traffic. In *2018 2nd Cyber security in networking conference (CSNet)* (pp. 1–7).

Berrueta, E., Morato, D., Magaña, E., & Izal, M. (2019). A survey on detection techniques for cryptographic ransomware. *IEEE Access*, *7*, 144925–144944. http://dx.doi.org/10.1109/ACCESS.2019.2945839.

Berrueta, E., Morato, D., Magaña, E., & Izal, M. (2020). Open repository for the evaluation of ransomware detection tools. *IEEE Access*, *8*, 65658–65669. http://dx.doi.org/10.1109/ACCESS.2020.2984187.

Berrueta, E., Morato, D., Magaña, E., & Izal, M. (2021). Ransomware and user samples for training and validating ML models. http://dx.doi.org/10.17632/yhg5wk39kf.1.

Berrueta, E., Morato, D., Magaña, E., & Izal, M. (2022). Ransomware PCAP repository. URL http://dataset.tlm.unavarra.es/ransomware/ Last Access: June 2022.

Bijitha, C. V., Sukumaran, R., & Nath, H. V. (2020). A survey on ransomware detection techniques. In S. K. Sahay, N. Goel, V. Patil, & M. Jadliwala (Eds.), *Secure knowledge management in artificial intelligence era* (pp. 55–68). Singapore: Springer Singapore.

Cabaj, K., & Mazurczyk, W. (2016). Using software-defined networking for ransomware mitigation: the case of cryptowall. *IEEE Network*, *30*(6), 14–20.

Chadha, S., & Kumar, U. (2017). Ransomware: Let's fight back!. In *2017 International conference on computing, communication and automation (ICCCA)* (pp. 925–930). IEEE.

Chen, Z.-G., Kang, H.-S., Yin, S.-N., & Kim, S.-R. (2017). Automatic ransomware detection and analysis based on dynamic API calls flow graph. In *Proceedings of the international conference on research in adaptive and convergent systems* (pp. 196–201).

Cobb, S. (2018). Ransomware vs printing press? US newspapers face "foreign cyber-attack". URL https://www.welivesecurity.com/2018/12/31/ransomware-printing-press-newspapers/ Last access: August 2022.

Cohen, A., & Nissim, N. (2018). Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory. *Expert Systems with Applications*, *102*, 158–178. http://dx.doi.org/10.1016/j.eswa.2018.02.039.

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, http://dx.doi.org/10.1145/2991079.2991110.

EUROPOL (2016). *Internet organised crime thread assessment (IOCTA) 2016*: *Technical Report*, Europol - European Police Office, http://dx.doi.org/10.2813/275589.

Faghihi, F., & Zulkernine, M. (2021). RansomCare: Data-centric detection and mitigation against smartphone crypto-ransomware. *Computer Networks*, *191*, Article 108011. http://dx.doi.org/10.1016/j.comnet.2021.108011.

Feng, Y., Liu, C., & Liu, B. (2017). Poster: A new approach to detecting ransomware with deception. In *38th IEEE symposium on security and privacy*.

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). IEEE.

Haynes, T., & Noveck, D. (2015). *Network file system (NFS) version 4 protocol*: *RFC 7530 RFC Editor*, URL https://tools.ietf.org/html/rfc7530.

Herrera Silva, J. A., Barona Lopez, L. I., Valdivieso Caraguay, A. L., & Hernandez-Alvarez, M. (2019). A survey on situational awareness of ransomware attacks—Detection and prevention parameters. *Remote Sensing*, *11*(10), http://dx.doi.org/10.3390/rs11101168.

Hirano, M., & Kobayashi, R. (2019). Machine learning based ransomware detection using storage access patterns obtained from live-forensic hypervisor. In *2019 Sixth international conference on internet of things: systems, management and security (IOTSMS)* (pp. 1–6). IEEE.

Hwang, J., Kim, J., Lee, S., & Kim, K. (2020). Two-stage ransomware detection using dynamic analysis and machine learning techniques. *Wireless Personal Communications*, *112*(4), 2597–2609.

Intelligence, M. (2021). Global network attached storage (nas) market - growth, trends, COVID-19 Impact, and forecasts (2021 - 2026). URL https://www.mordorintelligence.com/industry-reports/network-attached-storage-nas-market, Last access: August 2022.

Julián-Moreno, G., Leira, R., de Vergara, J. E. L., Gómez-Arribas, F. J., & González, I. (2018). *On the feasibility of 40 Gbps network data capture and retention with general purpose hardware*. New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3167132.3167238.

Kara, I., & Aydos, M. (2022). The rise of ransomware: Forensic analysis for windows based ransomware attacks. *Expert Systems with Applications*, *190*, Article 116198. http://dx.doi.org/10.1016/j.eswa.2021.116198.

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*.

Lee, K., Lee, S., & Yim, K. (2019). Machine learning based file entropy analysis for ransomware detection in backup systems. *IEEE Access*, *7*, 110205–110215. http://dx.doi.org/10.1109/ACCESS.2019.2931136.

Loman, M. (2019). How the most damaging ransomware evades IT security. URL https://news.sophos.com/en-us/2019/11/14/how-the-most-damaging-ransomware-evades-it-security/, Last access: August 2022.

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). IEEE.

Maniath, S., Ashok, A., Poornachandran, P., Sujadevi, V. G., Sankar A.U., P., & Jan, S. (2017). Deep learning LSTM based ransomware detection. In *2017 Recent developments in control, automation power engineering (RDCAPE)* (pp. 442–446).

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* (pp. 532–541). Springer.

McIntosh, T., Kayes, A. S. M., Chen, Y.-P. P., Ng, A., & Watters, P. (2021). Ransomware mitigation in the modern era: A comprehensive review, research challenges, and future directions. *ACM Computing Surveys*, *54*(9), http://dx.doi.org/10.1145/3479393.

Mehnaz, S., Mudgerikar, A., & Bertino, E. (2018). RWGuard: A real-time detection system against cryptographic ransomware. In M. Bailey, T. Holz, M. Stamatogiannakis, & S. Ioannidis (Eds.), *Research in attacks, intrusions, and defenses* (pp. 114–136). Cham: Springer International Publishing.

Microsoft (2020). Windows 7 support ended on January 14, 2020. URL https://support.microsoft.com/en-us/help/4057281/windows-7-support-ended-on-january-14-2020 Last Access: August 2022.

Moore, C. (2016). Detecting ransomware with honeypot techniques. In *2016 Cybersecurity and cyberforensics conference (CCC)* (pp. 77–81). IEEE.

Morato, D., Berrueta, E., na, E. M., & Izal, M. (2018). Ransomware early detection by the analysis of file sharing traffic. *Journal of Network and Computer Applications*, *124*, 14–32. http://dx.doi.org/10.1016/j.jnca.2018.09.013.

Moussaileb, R., Bouget, B., Palisse, A., Le Bouder, H., Cuppens, N., & Lanet, J.-L. (2018). Ransomware's early mitigation mechanisms. In *Proceedings of the 13th international conference on availability, reliability and security* (p. 2). ACM.

NetApp (2018). What is the default negotiated SMB version with various versions of data ONTAP and windows clients?. URL https://kb.netapp.com/Advice_and_Troubleshooting/Data_Storage_Software/ONTAP_OS/What_is_the_default_negotiated_SMB_version_with_various_versions_of_Data_ONTAP_and_Windows_clients, Last access: August 2022.

NetGear (2016). Recommendation for using AFP and SMB in the same environment. URL https://kb.netgear.com/29765/Recommendation-for-using-AFP-and-SMB-in-the-same-environment, Last access: August 2022.

Nieuwenhuizen, D. (2016). *A behavioural-based approach to ransomware detection. MWR Labs whitepaper*: *Technical Report*, MWR Labs, URL https://info.varonis.com/hubfs/docs/whitepapers/en/Varonis-Ransomware-Whitepaper-Netapp.pdf, Last access: August 2022.

Paik, J.-Y., Shin, K., & Cho, E.-S. (2016). Poster: Self-defensible storage devices based on flash memory against ransomware. In *Proceedings of IEEE symposium on security and privacy*.

Pyle, N. (2020). Stop using SMB1. URL https://techcommunity.microsoft.com/t5/storage-at-microsoft/stop-using-smb1/ba-p/425858 Last Access: August 2022.

Quinkert, F., Holz, T., Hossain, K. S. M. T., Ferrara, E., & Lerman, K. (2018). RAPTOR: Ransomware attack predicTOR. URL http://arxiv.org/abs/1803.01598.

Ramesh, G., & Menen, A. (2020). Automated dynamic approach for detecting ransomware using finite-state machine. *Decision Support Systems*, *138*, Article 113400.

Reddy, B. V., Krishna, G. J., Ravi, V., & Dasgupta, D. (2021). Machine learning and feature selection based ransomware detection using hexacodes. In V. Bhateja, S.-L. Peng, S. C. Satapathy, & Y.-D. Zhang (Eds.), *Evolution in computational intelligence* (pp. 583–597). Singapore: Springer Singapore.

Roy, K. C., & Chen, Q. (2020). DeepRan: Attention-based BiLSTM and CRF for ransomware early detection and classifcation. *Information Systems Frontiers*, 1–17.

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). http://dx.doi.org/10.1109/ICDCS.2016.46.

Sgandurra, D., Muñoz González, L., Mohsen, R., & Lupu, E. C. (2016). Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. arXiv preprint arXiv:1609.03020.

Shaukat, S. K., & Ribeiro, V. J. (2018). RansomWall: A layered defense system against cryptographic ransomware attacks using machine learning. In *2018 10th International conference on communication systems & networks (COMSNETS)* (pp. 356–363). IEEE.

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* (pp. 1784–1786).

Sjouwerman, S. (2019). Ransomware's impact highlights the threat of social engineering. URL https://blog.knowbe4.com/ransomwares-impact-highlights-the-threat-of-social-engineering Last access: August 2022.

Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy* (pp. 305–316). IEEE.

Sophos (2020). *The state of ransomware 2020*: *Technical Report*, Sophos, URL https://www.sophos.com/en-us/medialibrary/Gated-Assets/white-papers/sophos-the-state-of-ransomware-2020-wp.pdf, Last access: August 2022.

TrendMicro (2019). Report: Huge increase in ransomware attacks on businesses. URL https://www.trendmicro.com/vinfo/us/security/news/threat-landscape/report-huge-increase-in-ransomware-attacks-on-businesses Last access: August 2022.

Victor, K. (2020). Reflective loading runs netwalker fileless ransomware. URL https://www.trendmicro.com/en_us/research/20/e/netwalker-fileless-ransomware-injected-via-reflective-loading.html Last access: August 2022.

Vidyarthi, D., Kumar, C., Rakshit, S., & Chansarkar, S. (2019). Static malware analysis to identify ransomware properties. *International Journal of Computer Science Issues (IJCSI)*, *16*(3), 10–17.

Vinayakumar, R., Soman, K., Velan, 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). IEEE.

Zhang, B., Xiao, W., Xiao, X., Sangaiah, A. K., Zhang, W., & Zhang, J. (2020). Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes. *Future Generation Computer Systems*, *110*, 708–720.

Zuhair, H., Selamat, A., & Krejcar, O. (2020). A multi-tier streaming analytics model of 0-day ransomware detection using machine learning. *Applied Sciences*, *10*(9), 3210.