Received: 16 March 2021 | Revised: 27 May 2021 | Accepted: 31 July 2021

DOI: 10.1049/ise2.12042

ORIGINAL RESEARCH PAPER

Inhibiting crypto-ransomware on windows platforms through a honeyfile-based approach with R-Locker

José Antonio Gómez-Hernández[1,2] | **Raúl Sánchez-Fernández**[1,2] | **Pedro García-Teodoro**[1,2]

[1]Network Engineering & Security Group, University of Granada, Granada, Andalucía, Spain

[2]CITIC - University of Granada, Granada, Andalucía, Spain

Correspondence

José Antonio Gómez-Hernández, University of Granada, Granada, Andalucía, Spain.
Email: jagomez@ugr.es

Funding information

Ministerio de Economía y Competitividad, Grant/Award Number: TIN2017-83494-R

Abstract

After several years, crypto-ransomware attacks still constitute a principal threat for individuals and organisations worldwide. Despite the fact that a number of solutions are deployed to fight against this plague, one main challenge is that of early reaction, as merely detecting its occurrence can be useless to avoid the pernicious effects of the malware. With this aim, the authors introduced in a previous work a novel anti-ransomware tool for Unix platforms named *R-Locker*. The proposal is supported on a honeyfile-based approach, where 'infinite' trap files are disseminated around the target filesystem for early detection and to effectively block the ransomware action. The authors extend here the tool with three main new contributions. First, *R-Locker* is migrated to Windows platforms, where specific differences exist regarding FIFO handling. Second, the global management of the honeyfiles around the target filesystem is now improved to maximise protection. Finally, blocking suspicious ransomware is (semi)automated through the dynamic use of white-/black-lists. As in the original work for Unix systems, the new Windows version of *R-Locker* shows high effectivity and efficiency in thwarting ransomware action.

1 | INTRODUCTION

As evidenced in the last years, the impact and relevance of threats and security incidents on services and systems are continuously increasing [1, 2]. This is of special interest in the case of mobile environments [3], as the use and deployment of such a kind of platform (smartphones, tablets, and iot related devices) is becoming more and more generalised [4].

The consequences of security attacks can be diverse, as the objectives and motivations of the hackers are varied too [5]. Measured in economic terms or from the perspective of confidence and reputation, the losses provoked by security incidents can be most of the time dramatic for victim users, services and systems [6].

The usual form of perpetrating security attacks is by means of malware, that is malicious software specifically designed to disrupt, damage, or gain unauthorised access to a computer system. Malware has evolved over time from simple parasite routines embedded in legitimate software to really complex, independent, auto-propagable, metaformic, multi-exploit and multi-platform software [7].

As specified in the bulk of the current security threat reports, ransomware is, among others like botnets or bank-related malware, one prominent type of malware nowadays [8, 9]. According to the number of ransomware families and variants appeared [10, 11], also the number of related incidents are continuous since some decades ago. The healthcare industry is a top target for ransomware attacks, but none (either individual or organisations) is free of suffering from the pernicious effects of this form of malware [12]. In particular, the death of a woman motivated by the disruption of patient care by a ransomware attack was reported in a German hospital during last September in 2020.

Moreover, situations like the crisis caused by the COVID-19 extraordinarily spread this problem due to the massive use of technological means by the population, as described in [13].

In consequence with the relevance of the problem, several anti-ransomware solutions have been proposed in the specialised literature [14]. They range from prevention to response countermeasures such as those aimed at trying to recover the encryption keys from the infected system [15]. Regretfully, none of the developed solutions are definitive and, thus, this still continues being a harmful threat for users and organisations all over the world.

A honeyfile-based approach to thwart the action of crypto-ransomware was proposed by the authors in a previous work [16]. It is based on the deployment of a *honeyfile* solution to block the action of ransomware on the host environment in an early stage, without affecting the expected behaviour of the environment either in terms of resource consumption or from the perspective of system interaction. Originally implemented for Unix platforms, the tool, named *R-Locker*, is extended now to Windows platforms. The functionality and conceptual operation is the same, but the implementation varies due to particular aspects of the specific OS regarding FIFOs handling.

In addition to this significant contribution due to the wide use of Windows systems, the new version of *R-Locker* improves the creation, distribution and maintenance of the honeyfiles to dynamically protect the target filesystem over time. Moreover, as a new difference with the original version of *R-Locker*, the effective response to be adopted against suspicious ransomware samples is (semi)automated through the use of dynamic black-/white-lists of well-known harmful/legitimate applications.

The rest of the paper is organised as follows: Section 2 discusses the main proposals available at present to fight against ransomware, focussing on solutions developed from the publication of [16] until now. After that, Section 3 briefly describes the general honeyfile methodology considered by *R-Locker*, and deals with the specific implementation of the tool for Windows platforms. In addition, we detail in the same section some improvements introduced to the tool in comparison with its previous version regarding the dynamic management of the honeyfiles and the adoption of a (semi)automatic response procedure. The overall performance of the tool, both in terms of accuracy and efficiency, is evaluated and discussed in Section 4. Finally, Section 5 summarises the contributions of the work and outlines some future directions.

2 | RELATED WORK

Ransomware is receiving important attention over time by the research community, as the number of related incidents are continuous in the recent years [17, 18]. One of the first known ransomware attacks occurred in 1989 and targetted the healthcare industry [19]. Since then, the evolution in typology and impact experienced by ransomware is exponential, the biggest incidents occurring in more recent years [20, 21]. Most of them refer to the crypto-ransomware form, that is, the encryption (typically RSA or AES) of the information on the infected devices and the subsequent demand of a payment from victims to rescue data.

Some examples of ransomware and related incidents are as follows: *Reveton* is a ransomware type appeared around 2012 that impersonates law enforcement agencies. Known as *police ransomware* or *police trojan*, this malware is notable for showing a notification page purportedly from the victims local law enforcement agency, informing them that they were caught doing an illegal or malicious activity online. Afterwards, *CryptoLocker* infected more than 2,50,000 Windows systems in 2013. Between 2014 and 2016, other variants like *CryptoWall*, *Locky* and *Cerber* were among the most commonly used ransomware types, targetting hundreds of thousands of individuals and organisations. Also in 2015 *TeslaCrypt* appeared, affecting computer games and involving in some cases such as PayPal My Cash cards, and a group known as the Armada Collective carried out a string of attacks against various financial services in countries like Switzerland and Greece.

Based on the previous variants, several incidents occurred in 2016. One of them involved the Hollywood Presbyterian Medical Centre in Los Angeles, and allegedly demanded a ransom of dozen thousand dollars. Also, an Ottawa Hospital was hit by ransomware that impacted more than 9,800 machines. Thanks to diligent backup and recovery processes, the hospital avoided paying ransom. Also, the Kentucky Methodist Hospital and the Chino Valley Medical Centre, and Desert Valley Hospital in California were hit by *Locky* ransomware. In addition to health-related targets, the San Francisco Municipal Transportation Agency fell victim to a ransomware attack by *Mamba* or *HDDCryptor* that disrupted train ticketing and bus management systems. Attackers demanded a whopping 100 Bitcoin ransom (equivalent to about $73,000 at the time), but thanks to the speedy response and comprehensive backup processes, the SFMTA was able to restore its systems within two days.

SamSam is another relevant family of ransomware which appeared in 2015 and affected the Colorado Department of Transportation at the City of Atlanta, as well as numerous health care facilities. *Jigsaw* ransomware appeared in 2016 and was designed to be spread through malicious attachments in spam emails. If the ransom is not paid within one hour, one file is deleted. Following this procedure for each hour without a ransom payment, the amount of files deleted is exponentially increased each time from a few hundred to thousands of files until the computer is wiped after 72 h. Any attempt to reboot the computer or terminate the process will result in 1,000 files being deleted. A further updated version also makes threats to dox the victim by revealing personal information online. Afterwards, in 2017, *WannaCry*, *Petya* and *Bad Rabbit* attacks were famous because they affected hundreds of relevant international companies all over the world.

The *Ryuk* variant hit in 2018 and 2019, whose victims were organisations with little tolerance for downtime, including daily US newspapers and a North Carolina water utility struggling with the aftermath of Hurricane Florence. *Zeppelin* began to appear on the scene in November 2019 and was an evolutionary descendent of the family known as *Vega* or *VegasLocker*, a ransomware-as-a-service offering that wreaked havoc across accounting firms in Russia and Eastern Europe. *Zeppelin* is specifically designed to not execute on computers running in Russia, Ukraine, Belarus, or Kazakhstan. *Sodinokibi*, also known as *REvil*, first emerged in April of 2019 and, like *Zeppelin*, it appeared to be the descendent of another malware family: *GandCrab*. It also had a code that prevented it from executing in Russia and several adjacent countries, as well as Syria, indicating that its origin was in that region.

It is worth to mention that ransomware attacks doubled in number in the last period. Thus, in the first quarter of 2020s financial year, ransomware attacks have dramatically increased due to the lack of cybersecurity measures during home-office working that the COVID-19 pandemic has brought along [22]. This last study also points out that organising auctions on the Internet (generally through the Deep Web) is a tendency in recent ransomware incidents to maximise hackers' earnings. Furthermore, many ransomware families have improved their skills of stealing sensitive data from various sectors such as banking, financial services, governmental services, insurance and manufacturing sectors. In this context, we can mention *NetWalker* ransomware, also known as *Mailto*, one of the most destructive malicious software in the ransomware attacks 2020-2021 list. *NetWalker* uses the network of the victim to encrypt all Windows devices by following two different ways to attack: (a) coronavirus phishing mails and (b) executable files that spread through networks. The appearance of the *Sekhmet* ransomware in June 2020 is also noticeable. It encrypts the files and asks for money to decrypt them. Infected files' extensions are randomly changed such as '.HrUSsw, .WNgh, .NdWfEr'.

In this context, the best way to fight against ransomware is prevention: users' training and education, use of legitimate software, periodical software update, data backups disposal, users' privilege management etc. However, prevention mechanisms do not impede the potential occurrence of malware activity, so that detection schemes should be additionally deployed to protect against ransomware. This way, a holistic taxonomy of countermeasures for ransomware is introduced in [23, 24], where both technical, education based, as well as policy and law-related actions are considered. Focussing on a technical perspective, the detection of ransomware action is usually dealt with according to some well-known methodologies [25–27], each of them with pros and cons:

- *Static*, intended to detect ransomware action before malware runs. This is the case of finding common strings in programs (e.g. 'ransom', 'bitcoin', and 'encrypt') or the use of function calls to encrypt files.
- *Dynamic*, related with the execution of the malware over time. The information accessed in this case is varied: filesystem access (overwriting or removing files, file extension modification), network activity (e.g. DNS requests and C&C communications), and system registry modification, etc.

Based on the above usually recurrent aspects, the authors discuss in [16] several proposals developed in the specialised literature to thwart ransomware action. Since then, some other similar approaches can be found as explained in [28], in particular regarding ransomware detection [29]. For example, Bae et al. introduce in [30] a machine learning approach to detect ransomware based on API sequences expressed in terms of N-gram sequences. Also, Arabo et al. present in [31] a machine learning approach, but in this case DLL APIs calls, usage of system resources (disk, CPU, RAM, network connections) and files opened are considered to conclude or not the action of a ransomware piece.

Instead, the authors in [32] propose an algorithm based on traffic monitoring that can detect ransomware action and prevent further activity over shared documents. Also, [33] presents a traffic-based detector, but in this case the authors design an SDN detection and mitigation framework based on OpenFlow to detect *WannaCry* samples. In a similar line, the authors in [34] introduce a network-based intrusion detection system to detect *Locky* samples, employing two independent classifiers working in parallel on different levels: packet and flow levels. The authors in [35] combine network-related activities with file accesses in an introspection approach to detect crypto-ransomware running on VMs. Considering a more varied activity, Jethva et al. explore and compare in [36] three ML schemes (SVM, logistic regression and random forest) where grouped registry key operations, file entropy and file signature are considered as analysis parameters with detection purposes. More recently, Almomani et al. consider permissions and API packages to detect ransomware in Android platforms making use of machine-learning approaches like Random Forest, Decision Tree, Sequential Minimal Optimisation, and Naive Bayes [37].

Beyond the specific parameters considered in detection, some works are mainly focussed on evaluating novel analysis methodologies. This is the case of [38], where an ensemble-based detection model which incorporates two techniques (incremental bagging and enhanced semi-random subspace selection) is evaluated for ransomware detection. In [39, 40], deep learning techniques are explored. In the first case, to extract the latent representation of a high dimension of collected data to identify malicious behaviours accurately. In the second work, a theoretical model named CRED is developed to accurately define the boundaries of the pre-encryption phase of the attack lifecycle based on ransomware. In [41], a detection method for ransomware by employing a combination of a similarity preserving hashing method called *fuzzy hashing* and a clustering method is applied to detect *WannaCry* samples. Similarly, the authors in [42] propose a fuzzy-import hashing technique, which is the integration of two methods, namely, *fuzzy hashing* and *import hashing*. This integration can offer several benefits such as an improved detection rate by complementing each other when one method cannot detect malware, then the other method can; the generation of fuzzied results for subsequent clustering or classification, as the import hashing result can be easily merged with the fuzzy hashing result.

Although numerous and relevant, none of the available solutions at present are yet definitive. In fact, one principal challenge is that of early detection. That is, although potentially accurate in detection, a valid solution should additionally be as quick as possible. Otherwise, the action regarding the encryption of the system can be completed and, thus, the detection itself will become useless. In this line, the authors in [43] propose an adaptive pre-encryption early detection model which is expected to deal with the population concept drift of crypto-ransomware given the limited amount of data collected during the pre-encryption phase of the attack lifecycle. With such adaptability, the model can maintain up-to-date knowledge about the attack behaviour and identify the polymorphic ransomware that continuously changes its behaviour.

In this overall context, the authors introduce in [16] *R-Locker*, a novel early detection plus reaction anti-ransomware approach based on the deployment of honeyfiles where: (a) Each honey archive deployed is not a 'normal' file but FIFO like, so that a ransomware accessing the trap file will be completely blocked because the OS automatically stops any process reading from an empty (not previously written) FIFO; (b) the honeyfile is connected to a monitor process in such a way that, when accessed, a response procedure is automatically launched aimed to effectively defeat the infection, and (c) the complexity and cost of the solution are really low and do not interfere with the normal operation of the environment.

Provided the good performance exhibited by the original tool for Unix systems, it is now improved in three main aspects as pointed out in [16]:

- It is extended by authors to Windows environments for which FIFO operation varies from that in Unix platforms. Provided the wide use of such types of platforms, the impact of our ransomware solution is expected to be high.
- The creation, distribution and maintenance of the honeyfiles is automated to dynamically handle them over time and, thus, to maximise the filesystem protection.
- Once a suspicious application is blocked by the detection tool, the subsequent countermeasure (e.g. stop and uninstall the application) is (semi)automated through the disposal of dynamic black-/white-lists to avoid relying always the decision on the final user.

From the above section, the rest of the document is devoted to describe and evaluate the new version of *R-Locker*. We shall see that the proposal is effective and efficient, while novel in comparison with others in the literature where the typical parameterisation and monitoring-based detection methodologies previously described are considered [44–46].

3 | A HONEYFILE-BASED ANTI-RANSOMWARE TOOL FOR WINDOWS PLATFORMS

Accepted that the crypto-ransomware operation relies on scanning the infected machine's filesytem to access files and encrypt the contained information, our ransomware detection approach is aimed to satisfy some functional properties and operational requirements (named as *F1–F2*, and *R1–R5* in [16]) regarding effectivity, scalability, transparency, consumption, etc.

3.1 | Honeyfile approach to thwart ransomware action

In [47], we can find the eight steps composing the typical lifecycle of ransomware for the *.NET* framework: (a) infiltrating the host system, (b) gaining execution privileges, (c) creating unique cryptographic keys, (d) enumerating files to encrypt, (e) cyphering files with keys, (f) removing access to original files, (g) protecting keys until payment, and (h) maintaining an extortion channel. From that, any mechanism intended to prevent ransomware from succeeding should be based on stopping its execution in at least one of the previous stages.

Our approach will operate somewhere between fourth and fifth steps. Provided that to encrypt a file it is first needed to read its content, our proposal relies on making use of trap files, or *honeyfiles*, aimed to hold indefinitely the process reading the accessed file. That is, the corresponding read() call will never return to the caller (i.e. the ransomware), so that it will thwart the last goal of the ransomware: file ciphering.

To achieve goals *Fx* and *Rx*, a simple and elegant solution is to make use of an interprocess communication mechanism (IPC) *named FIFO* (*First Input First Output*), or *named pipes* [16]. Some main features of FIFOs are as follows:

- FIFOs are special files which are manipulated with the same set of system calls than regular files: read(), write(), open(), …This will make natural the interaction of ransomware with the honeyfiles.
- A principal property of FIFOs is that when a process reads an empty named pipe, that is, no process has been previ-

ously written on it, the OS automatically blocks the reading process.

A named pipe needs a writing process at the other side. Such a process can control the interactions with the FIFO, so that it is the ideal place to locate the monitor process for the detection tool to be deployed. According to that, [16] shows (see figs 3 and 4 there) the architecture and the operational flow for the proposed ransomware detector. From them: (a) when the user installs the tool, a blocking FIFO is created and one trap file is inserted in every folder around the whole filesystem pointing to the central FIFO; (b) once all the trap files are created and distributed around the filesystem, the monitor process remains waiting at the named pipe; (c) when a process (e.g. ransomware) accesses any of the honeyfiles for reading, it is redirected to the FIFO and becomes blocked; (d) at this moment, the OS wakes the monitor procedure up to launch a process intended to adopt the corresponding countermeasures.

Algorithm 1 outlines the general operation of *R-Locker*.

Algorithm 1 : *R-Locker* related algorithm.

```
Input: null
Output: null
1  Honey file ← create FIFO in blocking mode
2  Trap ←
     list of files with name extensions pdf, jpg, ...
3  begin
       // Create and spread the traps
4      for every folder in filesystem do
5          Create symlink: trap → FIFO;
       // Traps are the reading end, accessed
          by ransomware
6      for (;;) do
7          Create an instance of the detection threat;
8          Connect the thread instance to the writing end of
             FIFO;
9          Treat wait for threads been signalled by FIFO
             reading;
           // Now, blocked threads are waiting
              a ransomware sample fall into
              the trap
10         Awakened thread by OS proceeds with detection
             phase;
```

3.2 | R-locker implementation for windows

Based on the mentioned general operation of *R-Locker*, in this section we describe the specific implementation for Windows platforms, where two main differences exist in handling FIFOs in comparison with Unix systems: (a) System files and pipes have different namespaces, and (b) only one process is allowed to simultaneously read from a FIFO. Therefore, Algorithm 2 is now implemented, where we should remark that the development is made over the Windows application programming interface (Win32API) so that the user only needs to run an executable programme without requiring to modify or configure the OS kernel.

Algorithm 2 : *R-Locker* pseudocode.

```
Input: null
Output: null
1  trap_path      // Symbolic link to honeyfile
2
3  trap_extensions_list ← .pdf .jpg .doc ...
     honeyfile ← \\\\.\\pipe\\trap_name   // FIFO's
     name
4  Function PopulateTraps(trap_path):
5      folders_list ← Generate the diskfolders list
6      for every folder in folders_list do
7          if not exist trap_path then
8              CreateSymbolicLink(..\trap_path.trap_extension,
                 honeyfile)
9  Function InstanceThread(FIFOPathname):
10     hP=CreatePipe(honeyfile,.,PIPE_WAIT,...)
         // Create the FIFO
11     fConnec = ConnectNamedPipe(hP)   // Connect to
         FIFO
12     CreateThread(..., Detection,..)        // Detection
         thread
13 Procedure Detection(hPipe):
14     GetNamePipeClientProcessId(hP, ppid)
15     hProcess=OpenProcess(..., *ppid)
16     program_name ←
         QueryFullProcessImageName(hProcess, ..)
17     if (programis in Whitelist) then
18         continue
19     if (program_name is in Backlist) then
20         Terminate process(hProcess)
21     else
22         if (messageBox == Terminate) then
23             Append program_name to Blacklist
24             TerminateProcess(hProcess)
25 Function PopulateWhiteList():
26     Whitelist ← Generate list of installed applicatios
27 Function Main:
28     PopulateWitelist()
29     PopulateTraps(pipe_name)
30     num_threads ←
         parameter * get number of processors from OS
31     for (;;) do
32         ended_thread = 0
33         for (i=0; i++; i < num_threads) do
34             ThreadArray[i] =
                 CreateThread(.,InstanceThread,Pipe,.)
35         while ended_thread < num_threads - 1 do
36             WaitMultipleObjects(...,ThreadArray,..)
37         if (period) then
38             FolderMonitor =
                 CreateThread(...,PopulateTraps,...)
```

According to the mentioned algorithm, the main() function in *R-Locker* starts the calling function PopulateTraps(), which is responsible for obtaining the folder structure of the filesystem. Then, the tool creates a symbolic link to the named pipe in each of the folders (CreateSymbolicLink system call), so that a tree of direct paths to the detection tool through the central FIFO is deployed around the filesystem.

Nowadays, many ransomware families (e.g. *Cerber*) involve multithread processes to optimise the encryption of the system. In order to deal with such malware variants, *R-Locker* next creates a pool of threads (line 33). To perform multithread operations in Windows, it is necessary to create multiple instances of the same FIFO and to deploy one monitor thread per instance (line 34). The number of threads is determined in an experimental way, so the algorithm declares a parameter to adjust it (line 30). A good estimation of the parameter value is two, because the maximum performance is obtained with two threads per core. This way, while a thread is waiting for reading a file, the other can use the CPU to cypher data. A greater number of threads will not improve performance, as they will compete with each other for the CPU.

Each of the created threads then calls the InstanceThread() function, which is intended to deploy respective instances of the named pipe. At this point, it is noticeable the fact that, as previously mentioned, system files and pipes in Windows have separate namespaces. To solve this inconvenience, we need to use the naming conventions and rules in Windows to name devices like regular files or directories (line 3).

For our main purpose, another interesting property of FIFOs is that the synchronisation of the communications are automatically handled by the OS. This way, when a process reads an empty FIFO (i.e. nobody has written yet on it), the OS blocks the reading process. For that, it is necessary to instruct the OS when creating a synchronous FIFO (flag PIPE_WAIT on CreatePipe() in line 10). After that, the thread gets connected synchronously to the pipe (line 11), just awaiting ransomware to fall into the trap. This way, the monitor process in *R-Locker* will act as a writer to the pipe, while ransomware will act as a reader over the communication channel.

When a reading process accesses a honeyfile, the function Detection() will be executed by the corresponding thread. This function is responsible for determining the identity of the process trying to read the trap and subsequently to put into action the appropriate countermeasure to give response to the incident, according to its real malicious or legitimate nature.

Figure 1 represents the specific flow diagram of *R-Locker* for Windows (*grey box*). As discussed in the previous paragraphs, it is composed of a monitoring process connected to the named pipe. Such a monitor just consists of a writing process to the named pipe, whose action will be successful when a reading process intervenes at the other side. Since our interest is to protect the entire filesystem, we will replicate the trap into multiple directories. Although it is possible to replicate the named pipe itself, the associated management by the master process would become more complex. Therefore, as indicated, we replicate the traps through the use of symbolic links to the same central FIFO. This way, a ransomware sample trying to access the symbolic link in any part on the filesystem will be directed to the (unique) FIFO. On the other hand, as the reading action from a FIFO is blocking in Windows, *R-Locker* will create several threads to manage multithread ransomware.

Let us remark again the low resource consumption expected for *R-Locker*. First, symbolic links do not consume disk space, the named pipe only involving the space to store the metadata in the filesystem since it is not really a file but a device. Second, regarding memory usage, the system only needs space to allocate $2 \times n$ threads, with n equal to the number of system processors. Third, the CPU usage is null when the monitor is awaiting because the operating system puts it in a blocked state and the overall operation relies on a future reading process from the FIFO.

Embedded in the previous operational flow, in the next subsubsections, we present two particular improvements with respect to the original tool introduced in [16] (see Section 4.3 in that study). First, the dynamic management of the folder

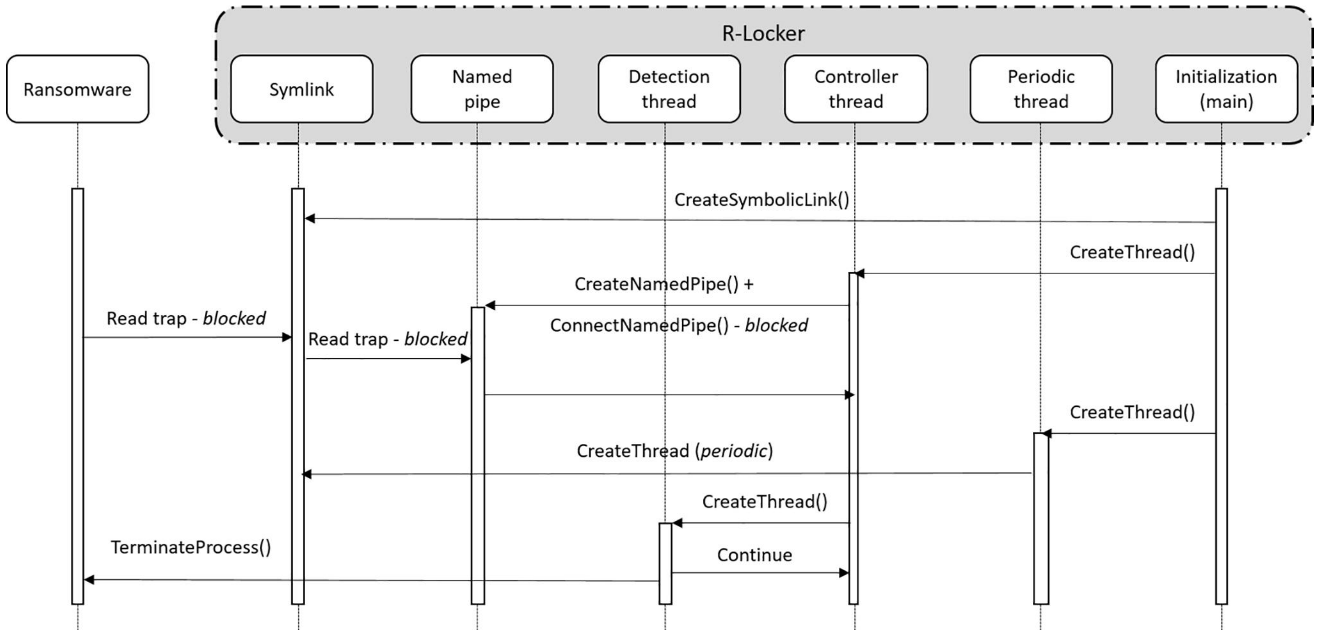


FIGURE 1 Flow diagram of *R-Locker*

structure and the corresponding deployment of honeyfiles, which will provide a better protection of the target filesystem over time. Second, the adoption of subsequent corrective actions once a suspicious ransomware event is detected, where a (semi)automatic procedure is now implemented.

3.2.1 | Dynamic management of the honeyfiles

Beyond the interesting general behaviour of the honeyfiles deployed around the filesystem to protect it, some specific practical aspects must be highlighted about them. One of them is about the particular locations for an effective protection, that is how to select and handle the folders into which to place the traps. In [48], the interaction of some samples of ransomware with the filesystem is analysed, and two important issues are highlighted. First, the order followed by ransomware samples for folder selection. Second, the order of file selection into a folder depending on the filesystem type.

In the first case, the authors in [48] show that there does not exists a unique order for folder selection. Some samples, like *GandCrab* or *TeslaCrypt*, perform an initial folder selection based on depth and then go through the structure following alphabetical order. However, other samples, like *Osiris*, carry out a random selection of the folders. Such a disparate behaviour concludes with the convenience of deploy traps into all the existing folders for complete protection. That is, we need to create a symbolic link to the central FIFO in every folder in the filesystem. This is the purpose of the function PopulateTrap() in the pseudocode of Algorithm 2, which creates such links at the initialisation phase of the tool. In addition, the function will dynamically explore the folder structure of the target filesystem to include symlinks into the new folders appeared over time.

As the size associated to a symbolic link is 0 bytes, no disk space is involved in creating as many symlinks as needed. In addition, due to the fact that all the symbolic links point out to the same central FIFO waiting for a reading process, the only pro-active action that will consume additional resources of the system is the periodic thread checking the existence of new folders around the filesystem. The execution period of each thread is currently tuned to minimise the exposure of new folders at a reasonable computational cost (see Section 4 below).

In respect of the file selection process into a folder, ransomware samples select files according to different criteria. In cases like *GandCrab* and *TeslaCrypt*, Win32 API functions like FindFirstFile() and FindNextFile() [44] are commonly used, which return the files depending on the type of filesystem. For example, in the case of NTFS, the filesystem typology of primary disks of current versions of Microsoft Windows, the entries of a folder will be alphabetically returned. Hence, to stop the ransomware action at the beginning of the reading process, the symbolic link can be named so that it is the first alphabetical entry (e.g. '!\enleadertwodots !'). In other cases, for example *Osiris*, files are first prioritised by extension and then selected in alphabetical order. To address such a situation, we can create multiple links with name '!\enleadertwodots !' and appealing extensions like '.doc', '.pdf', '.jpg', etc. To reinforce transparency from the user perspective, the symlinks can be additionally marked with hidden attribute to be invisible for normal user operation.

3.2.2 | Response after detection

When a reading process accesses a honeyfile, *R-Locker* in its original version launched a notification to allow user the manual adoption of potential subsequent corrective actions. The most feasible way to implement the notification procedure is making use of the general procedure NotifyUser(). This function is responsible for determining the identity of the process trying to read the trap and subsequently notify the user, who will decide the particular action or countermeasure to be taken, if so. The usual reaction mechanism adopted by the moment is to stop and/or kill the supposedly malicious programme.

As an improvement of that manual reaction procedure, the current version of *R-Locker* makes use of a semi-automatic process based on the dynamic management of black-lists and white-lists as follows:

1. The tool creates at the installation time a whitelist with legitimate programs which are permitted to access the filesystem. To do that, *R-Locker* explores the system programme folders, located in *C:\Programme Files* and *C:\Programme Files (x86)*, and incorporates the corresponding executable files into the whitelist.
2. Based on that, when a suspicious programme accesses the honeyfile, the tool will first check the whitelist so that: (a) if the programme is into the list, *R-Locker* does not act in any sense against the process; (b) otherwise, the user will be notified to decide how to proceed.
3. After notification, if the user decides not to act against the incident because she/he considers the programme as legitimate, the system will automatically add it to the whilelist.
4. On the contrary, if the user decides to stop/kill the programme, *R-Locker* will append the name of the programme into the blacklist. This way, the next time the same programme appears and it will be automatically terminated by the tool without requiring the user to intervene.

4 | EXPERIMENTAL EVALUATION

After describing *R-Locker*, this section is devoted to experimentally assess the tool when confronted with real ransomware samples on Windows platforms.

4.1 | Experimental environment

Testing real ransomware samples requires a secure and reusable environment. It is secure to avoid affecting real deployed services and systems and reusable to restore the experimental system to its original state in an efficient and dynamic way in case a test fails and the system becomes affected.

A usual way to test malware samples is to execute and analyse them on a virtual machine (VM) environment. In this case, some considerations must be taken into account:

- First, the VM must be isolated from the host system, while maintaining the access to the internet to allow potential command and control (C&C) communications.
- Second, the VM must be deployed on an isolated network to avoid affecting other systems.
- Third, the VM must be realistic to evade anti-sandbox techniques potentially deployed by malware.
- Finally, the test environment is a proprietary system so we must be licenced.

To accomplish with the above requirements, the experimental environment deployed here is based on *Malboxes* [49] in conjunction with *Vagrant* [50]. Malboxes is an open-source project destined to build Windows virtual machines for malware analysis. Its mission is to create a Vagrant box through the downloading and creation of a VirtualBox-based VM, and its provisioning and configuration. The Vagrant box is the initial state of the tests, allowing virtual machine management in a single workflow that makes the automation of the process easier. The joint use of the two mentioned tools allows creating, destroying or stopping VMs through simple commands.

The Vagrant configuration file helps to configure the resources loaned to the VM, the network configuration, the files to be copied from the host to the VM, and the scripts to be executed on the VM deployed. All those features are needed to automate the overall process. Our experimental setup is configured to create a VM with four CPUs and 4 GB of RAM to disable the shared folder created by Vagrant, to copy the *R-Locker* code into the VM, and to run a PowerShell script to download the crypto-ransomware samples considered.

It should be mentioned that we have not automated completely the deployment of the experimental environment due to some principal reasons. First, *R-Locker* must be executed manually because Vagrant does not have an option to do this automatically. Second, in case the programme analysed is detected as a malicious programme by the system, *Windows Defender* must be manually disabled. Third, some samples need a pre-configuration or dependency resolution to be properly executed.

4.2 | Evaluation samples and results

Once the experimental environment is tuned, it is time to run the tests with real crypto-ransomware samples. For that, some public repositories of malware samples are available. One example of that is [51], but in this case just *pcap* files are available instead of malware samples themselves. Therefore, we have acquired the experimental samples from *TheZoo* repository [52]. The reason is that the malware samples available at this site are classified by families, which is useful with the testing purposes and does not occur with several other well-known datasets such as *VirusShare* or *VirusTotal*.

Table 1 shows the specific ransomware samples tested in our experimentation. They refer to famous ransomware cases recurrently used in worldwide attacks: *Cerber*, *Jigsaw*, and *WannaCray*. Additionally, a proof-of-concept sample named *Generic* [53] is considered, whose basic functionality is similar to the one in the famous ransomware *Cryptolocker*: it encrypts with a RSA-4096 (RSA-OAEP-4096 + SHA256) public key any payload and then uploads it to a remote server. Although some other ransomware samples have been downloaded by authors, they have evidenced not to run and work properly on the deployed environment.

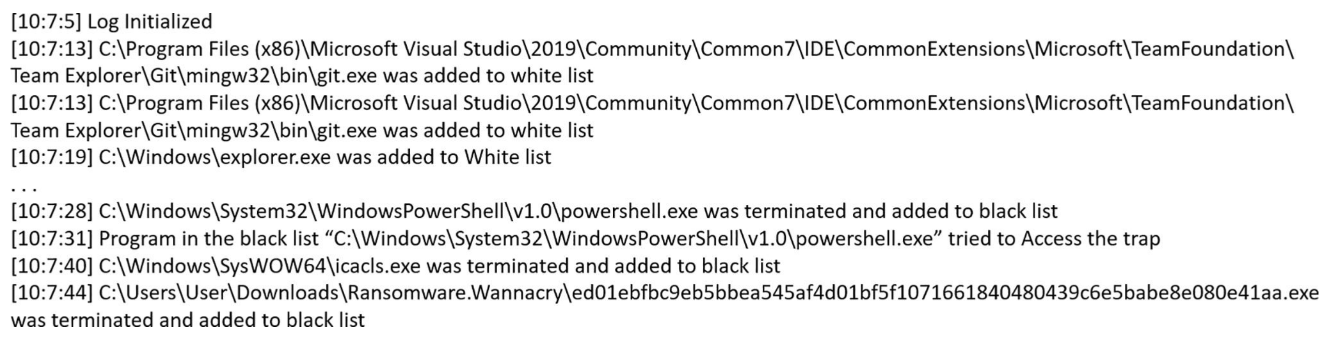
As indicated in Table 1, the samples corresponding to *WannaCry*, *Jigsaw*, and *Generic* have been detected and blocked by *R-Locker*, as expected. Moreover, the affection of the system in all cases is null, that is, no file has been encrypted by the sample thanks to the deployment of honeyfiles around the whole filesystem. Hence, the detection tool succeeds in defeating ransomware, which concludes that the FIFO solution is effective, and the traps deployed cover correctly the filesystem folders affected by the ransomware samples.

However, not all the ransomware samples have been detected by *R-Locker*. In particular, the *Cerber* sample is not detected, which can be justified as follows: The reversing engineering of *Cerber* [54] shows that the authors of the malware defined a minimum file size for candidate files to be encrypted. This way, since symbolic links are of 0 bytes size, the honeyfiles (symbolic links) deployed by *R-Locker* have not been selected by the ransomware for encryption which impedes detecting its operation on the system.

The main operation of *R-Locker* is registered over time into a log file. Figure 2 shows a fragment of an example log. In the first lines, we can see how the whitelist is built from the legitimate programs in the target system (e.g. *explorer.exe, git.exe*). When a programme is detected as a suspicious ransomware sample and it is not included in the whitelist, a notification box like that shown in Figure 3 is generated by *R-Locker*.

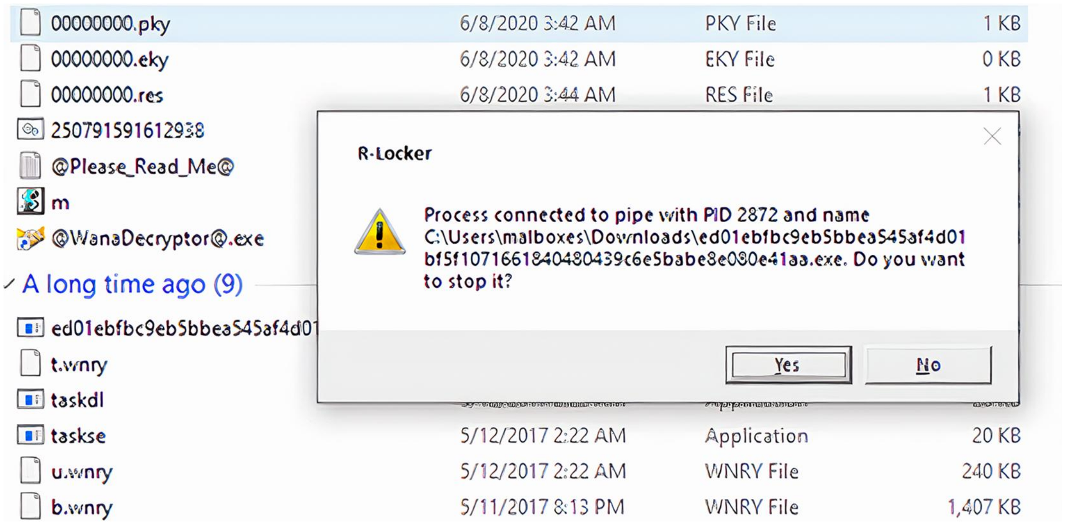
TABLE 1 Crypto-ransomware samples tested

Name	Source	MD5	Detection result
Cerber	TheZoo	8b6bc16fd137c09a08b02bbe1bb7d670	Not detected
Generic	GitHub	Locally compiled	Detected
Jigsaw	TheZoo	3ad6374a3558149d09d74e6af72344e3	Detected
WannaCry	TheZoo	84c82835a5d21bbcf75a61706d8ab549	Detected



```
[10:7:5] Log Initialized
[10:7:13] C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\
Team Explorer\Git\mingw32\bin\git.exe was added to white list
[10:7:13] C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\IDE\CommonExtensions\Microsoft\TeamFoundation\
Team Explorer\Git\mingw32\bin\git.exe was added to white list
[10:7:19] C:\Windows\explorer.exe was added to White list
. . .
[10:7:28] C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe was terminated and added to black list
[10:7:31] Program in the black list "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" tried to Access the trap
[10:7:40] C:\Windows\SysWOW64\icacls.exe was terminated and added to black list
[10:7:44] C:\Users\User\Downloads\Ransomware.Wannacry\ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe
was terminated and added to black list
```

FIGURE 2 Partial content of the log file generated by *R-Locker*



00000000.pky	6/8/2020 3:42 AM	PKY File	1 KB
00000000.eky	6/8/2020 3:42 AM	EKY File	0 KB
00000000.res	6/8/2020 3:44 AM	RES File	1 KB
250791591612938			
@Please_Read_Me@			
m			
@WanaDecryptor@.exe			
A long time ago (9)			
ed01ebfbc9eb5bbea545af4d01			
t.wnry			
taskdl			
taskse	5/12/2017 2:22 AM	Application	20 KB
u.wnry	5/12/2017 2:22 AM	WNRY File	240 KB
b.wnry	5/11/2017 8:13 PM	WNRY File	1,407 KB

R-Locker

Process connected to pipe with PID 2872 and name C:\Users\malboxes\Downloads\ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe. Do you want to stop it?

FIGURE 3 User notification by *R-Locker* for a suspicious ransomware incident

Such a notification informs the user about the process identification of the supposed ransomware and the path where it is launched from, asking the user about the possibility to stop the potentially malicious process.

From the first ellipsis in the log file in Figure 2, we can see how blacklist is handled. First, the execution of *Powershell* will generate a notification, so that the programme is labelled as malicious by the user and, thus, automatically included into the blacklist by *R-Locker*. After that, any further appearance of this programme will be automatically blocked by *R-Locker*. Likewise, the log file also shows the detection of a sample of *WannaCry* as well as an infected version of it, *iCacl.exe*.

In summary, from all the above we can conclude that the overall functional operation of *R-Locker* is as efficient as expected.

With testing and further developing purposes, *R-Locker* for Windows is publicly available for the community at [55].

5 | CONCLUSION AND FUTURE WORK

This work introduces a new version of *R-Locker*, an anti-ransomware solution proposed by the authors in a previous work, for Windows platforms. Like the original version for Unix, it is based on the deployment of honeyfiles around the target filesystem to trap potential malicious processes accessing disk information. However, three main aspects are contributed here to improve the original tool. First, the particular implementation for Windows needs to solve two main differences in handling FIFOs with respect to Unix platforms: (a) separate namespaces for system files and pipes, and (b) allowance of only one simultaneous process reading from a FIFO. Second, as pointed out in the previous work by authors, a novel procedure for the deployment and maintenance of the honeyfiles around the target filesystem is introduced for a better protection over time. Third, also the use of white-/black-lists is considered here to (semi)automatically determine and act against legitimate versus real ransomware processes accessing filesystem, thus reducing the intervention of the final user.

Publicly available at a GitHub repository, after evaluating the tool experimentally, it contributes several benefits: (a) effective operation, (b) unprivileged installation, (c) simplicity with low resources consumption involved, and (d) non-interference with the rest of the system.

As a future work, we guess to advance in two main directions. On the one hand, in defeating ransomware independently of the size of the honeyfiles deployed (to catch samples like that of *Cerber* in Section 4). On the other hand, it is of utmost interest to extend the proposal for mobile platforms, since the current use and consequent risks of such types of devices for the community are notable.

ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish Government-MINECO (Ministerio de Economía y Competitividad) and FEDER funds, through project TIN2017-83494-R.

DATA AVAILABILITY STATEMENT

Data sharing not applicable - no new data generated, or the article describes entirely theoretical research.

ORCID

José Antonio Gómez-Hernández https://orcid.org/0000-0002-8235-7366
Pedro García-Teodoro https://orcid.org/0000-0001-6766-1936

REFERENCES

1. FireEye: M-Trends 2020 (2020). https://content.fireeye.com/m-trends/rpt-m-trends-2020
2. SANS: SANS Top New Attacks and Threat Report (2020). https://www.domaintools.com/content/SANS_Top_New_Attacks_and_Threat_Report_Whitepaper.pdf
3. McAffe: McAffe Mobile Threat Report Q1, 2020. McAffe 2020 (2020). https://www.mcafee.com/content/dam/consumer/en-us/docs/2020-Mobile-Threat-Report.pdf
4. GobalWebIndex: Device GlobalWebIndex's Flagship Report on Device Ownership and Usage (2020). https://www.globalwebindex.com/reports/device
5. Madarie, R.: Hackers' motivations: testing Schwartz's theory of motivational types of values in a sample of hackers. Int. J. Cyber Criminol. 11(1), 78–97 (2017). https://doi.org/10.5281/zenodo.495773
6. OECD: Enhancing the Role of Insurance in Cyber Risk Management (2017). https://www.oecd.org/daf/fin/insurance/Enhancing-the-Role-of-Insurance-in-Cyber-Risk-Management.pdf
7. Razak, M.F.A., et al.: The rise of malware: bibliometric analysis of malware study. J. Netw. Comput. Appl. 75, 58–76 (2016). https://doi.org/10.1016/j.jnca.2016.08.022
8. CheckPoint: Cyber Security Report (2020). https://www.checkpoint.com/downloads/resources/cyber-security-report-2020.pdf
9. Sophos: Sophos 2020 Threat Report (2020). https://www.sophos.com/en-us/medialibrary/pdfs/technical-papers/sophoslabs-uncut-2020-threat-report.pdf
10. Subedi, K.P., Budhathoki, D.R., Dasgupta, D.: Forensic analysis of ransomware families using static and dynamic analysis. IEEE Symposium on Security and Privacy Workshops, pp. 180–185. (2018). https://doi.org/10.1109/SPW.2018.00033
11. Hassan, N.A.: Ransomware families. The most prominent ransomware strain. In Ransomware Revealed, Apress (2019). https://doi.org/10.1007/978-1-4842-4255-1
12. Meland, P.H., Bayoumy, Y.F.F., Sindre, G.: The Ransomware-as-a-service economy within the darknet. Comput. Secur. 92, 1–9 (2020). https://doi.org/10.1016/j.cose.2020.101762
13. McAfee: COVID-19—Malware Makes Hay During a Pandemic. Report. https://www.mcafee.com/blogs/other-blogs/mcafee-labs/covid-19-malware-makes-hay-during-a-pandemic/#_Toc37776295
14. Hassan, N.A.: Ransomware Revealed. A Beginner's Guide to Protecting and Recovering From Ransomware Attacks. Apress (2019). https://doi.org/10.1007/978-1-4842-4255-1
15. Davies, S.R., Macfarlane, R., Buchanan, W.J.: Evaluation of live forensic techniques in ransomware attack mitigation. Forensic Sci. Int.: Digit. Invest. 33, 1–11 (2019). https://doi.org/10.1016/j.fsidi.2020.300979
16. Gómez-Hernández, J.A., et al.: Thwarting ransomware action through a honeyfile-based approach. Comput. Secur. 73, 389–398 (2018). https://doi.org/10.1016/j.cose.2017.11.019
17. Simoiu, C., et al.: I was told to buy a software or lose my computer. I ignored it: a study of ransomware. Proceedings of the Fifteenth USENIX Conference on Usable Privacy and Security, pp. 155–174. (2019)
18. Reshmi, T.R.: Information security breaches due to ransomware attacks—a systematic literature review. Int. J. Inf. Manag. Data Insights. 1(2), 1–10 (2021). https://doi.org/10.1016/j.jjimei.2021.100013
19. Becker: First Known Ransomware Attack in 1989 Also Targeted Healthcare. https://www.beckershospitalreview.com/healthcare-information-technology/first-known-ransomware-attack-in-1989-also-targeted-healthcare.html
20. CSO: Recent Ransomware Attacks Define the Malware's New Age" (2018). https://www.csoonline.com/article/3212260/recent-ransomware-attacks-define-the-malwares-new-age.html
21. DigitalGuardian: A History of Ransomware Attacks: the Biggest and Worst Ransomware Attacks of All Time (2019). https://digitalguardian.com/blog/history-ransomware-attacks-biggest-and-worst-ransomware-attacks-all-time
22. S21Sec: Threat Landscape Report. https://www.s21sec.com/es/threat-landscape-report-es
23. Al-rimy, B.A.S., Maarof, M.A., Shaid, S.Z.M.: Ransomware threat success factors, taxonomy, and countermeasures: a survey and research directions. Comput. Secur. 74, 144–166 (2018). https://doi.org/10.1016/j.cose.2018.01.001
24. Connolly, L.Y., Wall, D.S.: The rise of crypto-ransomware in a changing cybercrime landscape: taxonomising countermeasures. Comput. Secur. 87, 1–18 (2019). https://doi.org/10.1016/j.cose.2019.101568
25. Berrueta, E., et al.: A survey on detection techniques for cryptographic ransomware. IEEE Access. 7, 144925–144944 (2019). https://doi.org/10.1109/ACCESS.2019.2945839
26. Herrera Silva, J.A., et al.: A survey on situational awareness of ransomware attacks-detection and prevention parameters. Rem. Sens. 11(10), 1–20 (2019). https://doi.org/10.3390/rs11101168
27. Bijitha, C.V., Sukumaran, R., Nath, H.V.: A survey on ransomware detection techniques. In: Sahay, S., et al. (eds.) Secure Knowledge Management in Artificial Intelligence Era. SKM 2019. Communications in Computer and Information Science, vol. 1186, pp. 55–68. Springer (2020). https://doi.org/10.1007/978-981-15-3817-9_4
28. Alshaikh, H., Ramadan, N., Hefny, H.A.: Ransomware prevention and mitigation techniques. Int. J. Comput. Appl. 177(40), 31–39 (2020). https://doi.org/10.5120/ijca2020919899
29. Kouliaridis, V., Kambourakis, G.: A comprehensive survey on machine learning techniques for android malware detection. MDPIO Inf. 12(185), 1–12 (2021)
30. Bae, S.I., Lee, G.B., Im, E.G.: Ransomware detection using machine learning algorithms. Concurrency Comput. Pract. Ex. 32, 1–11 (2020). https://doi.org/10.1002/cpe.5422
31. Arabo, A., et al.: Detecting ransomware using process behaviour analysis. Procedia Comput. Sci. 168, 289–296 (2020). https://doi.org/10.1016/j.procs.2020.02.249
32. Morato, D., et al.: Ransomware early detection by the analysis of file sharing traffic. J. Netw. Comput. Appl. 124, 14–32 (2018). https://doi.org/10.1016/j.jnca.2018.09.013
33. Akbanot, M., Vasilakis, V., Logothetis, M.: Ransomware detection and mitigation using software-defined networking: the case of WannaCry. Comput. Electr. Eng. 76, 111–121 (2019). https://doi.org/10.1016/j.compeleceng.2019.03.012
34. Almashhadani, A.O., et al.: A multi-classifier network-based crypto ransomware detection system: a case study of Locky ransomware. IEEE Access. 7, 47053–47067 (2019). https://doi.org/10.1109/ACCESS.2019.2907485

35. Tang, F., et al.: An introspection-based approach to detect crypto ransomware. Comput. Secur. 97, 1–14 (2020). https://doi.org/10.1016/j.cose.2020.101997
36. Jethva, B., et al.: Multilayer ransomware detection using grouped registry key operations, file entropy and file signature monitoring. J. Comput. Secur. 28(3), 337–373 (2020). https://doi.org/10.3233/JCS-191346
37. Almomani, I., AlKhayer, A., Ahmed, M.: An efficient machine learning-based approach for Android v.11 ransomware detection. 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), pp. 1–5. (2021)
38. Al-rimy, B.A.S., Maarof, M.A., Shaid, S.Z.M.: Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. Future. Generat. Comput. Syst. 101, 476–491 (2019). https://doi.org/10.1016/j.future.2019.06.005
39. Al-Hawawreh, M., Sitnikova, E.: Leveraging Deep Learning Models For Ransomware Detection in the Industrial Internet of Things Environment, pp. 1–6. Military Communications and Information Systems Conference (MilCIS) (2019). https://doi.org/10.1109/MilCIS.2019.8930732
40. Alqahtani, A., Gazzan, M., Sheldon, F.T.: A proposed crypto-ransomware early detection (CRED) model using an integrated deep learning and vector space model approach. 10th Annual Computing And Communication Workshop and Conference, pp. 275–279. CCWC (2020). https://doi.org/10.1109/CCWC47524.2020.9031182
41. Naik, N., et al.: Lockout-tagout ransomware: a detection method for ransomware using fuzzy hashing and clustering. IEEE Symposium Series on Computational Intelligence (SSCI), pp. 641–648. (2019). https://doi.org/10.1109/SSCI44817.2019.9003148
42. Naik, N., et al.: Fuzzy-import hashing: a static analysis technique for malware detection. Forensic. Sci. Int.: Digit. Invest. 37, 1–12 (2021)
43. Urooj, U., Maarof, M.A.B., Al-rimy, B.A.S.: A proposed adaptive pre-encryption crypto-ransomware early detection model. 3rd International Cyber Resilience Conference, pp. 1–6. CRC (2021)
44. Hampton, N., Baig, Z., Zeadally, S.: Ransomware behavioural analysis on Windows platforms. Journal of Information Security and Applications. 40, 44–51 (2018). https://doi.org/10.1016/j.jisa.2018.02.008
45. Zuhair, H., Selamat, A.: RANDS: a machine learning-based anti-ransomware tool for Windows platforms. Frontiers in Artificial Intelligence and Applications. 318, 573–587 (2020). https://doi.org/10.3233/FAIA190081
46. Rabadi, D., Teo, S.G.: Advanced windows methods on malware detection and classification. Annual Computer Security Applications Conference (ASAC), pp. 54–68. (2020)
47. Bajpai, P., Enbody, R.: Dissecting .NET Ransomware: Key Generation, Encryption and Operation, vol. 2, pp. 8–14. Network Security (2020). https://doi.org/10.1016/S1353-4858(20)30020-9
48. Grant, L., Parkinson, S.: Identifying file interaction patterns in ransomware behaviour. In: Parkinson, S., Crampton, A., Hill, R. (eds.) Guide to Vulnerability Analysis for Computer Networks and System. An Artificial Intelligent Approach, Computer Communication and Network Series, pp. 317–335. Springer (2018)
49. Malboxes: Malware Analysis Windows Virtual Machines. https://github.com/GoSecure/malboxes
50. Vagrant: Development Environments Made Easy. https://www.vagrantup.com
51. Berrueta, E., et al.: Open repository for the evaluation of ransomware detection tools. IEEE Access. 8, 65658–65669. https://doi.org/10.1109/ACCESS.2020.2984187
52. theZoo: A Live Malware Repository. https://github.com/ytisf/theZoo
53. Mauri, S.: A POC Windows Crypto-Ransomware. https://github.com/mauri870/ransomware
54. Wu, S., Leong, J.: Cerber 5.0.1 Arrives With New Multithreading Method. Fortinet (2016). https://www.fortinet.com/blog/threat-research/cerber-5-0-1-arrives-with-new-multithreading-method
55. Gómez-Hernández, J.A.: R-Locker: Anti-Ransomware Tool for Windows Platforms. https://nesg.ugr.es/r-locker

How to cite this article: Gómez-Hernández, J.A., Sánchez-Fernández, R., García-Teodoro, P.: Inhibiting crypto-ransomware on windows platforms through a honeyfile-based approach with R-Locker. IET Inf. Secur. 1–11 (2021). https://doi.org/10.1049/ise2.12042