R-Locker: Thwarting Ransomware Action through a Honeyfile-based Approach

J.A. Gómez-Hernández, L. Álvarez-González, P. García-Teodoro

Network Engineering & Security Group (https://nesg.ugr.es)
CITIC - University of Granada
Email: jagomez@ugr.es, luciaalvarez@correo.ugr.es, pgteodor@ugr.es

Abstract

Ransomware has become a pandemic nowadays. Although some proposals exist to fight against this increasing type of extorsion, most of them are prevention like and rely on the assumption that early detection is not so effective once the victim is infected. This paper presents a novel approach intended not just to early detect ransomware but to completly thwart its action. For that, a set of *honeyfiles* are deployed around the target environment in order to catch the ransomware. Instead of being normal archives, honeyfiles are FIFO like, so that the ransomware is blocked once it starts reading the file. In addition to frustrate its action, our honeyfile solution is able to automatically launch countermeasures to solve the infection. Moreover, as it does not require previous training or knowledge, the approach allows fighting against unknown, zero-day ransomware related attacks. As a proof of concept, we have developed the approach for Unix platforms. The tool, named *R-Locker*, shows excellent performance both from the perspective of its accuracy as well as in terms of complexity and resource consumption. In addition, it has no special needs or privileges and does not affect the normal operation of the overall environment.

Keywords: Ransomware, Detection, Honeyfile, System security

1. Introduction

The integrity and confidence of the Internet is increasingly compromised by a vast of cybercriminal actions. As pointed out in [1], malware-as-a-service (MaaS) and related variants (Hacking-as-a-Service/HaaS, Crimeware-as-a-Service/CaaS, Fraud-as-a-service/FaaS) have gained popularity in providing

an attacker with access to exploits, botnets and other types of malware to get money in an illegal and easy way. In fact, although cyberattacks can be launched by experts, attackers with low level of expertise can acquire their own malware by means of well-known exploit kits (EKs) like Angler, Magnitude, Rig, and Nuclear, among others [2].

In addition to other destructive types of malware, ransomware constitutes at present a pandemic that affects both individuals and organizations all over the world [3]. The increase of ransomware in number of families and variants in the last years is exponential (see Figure 1). To give some recent figures about this problem, consider the case of WannaCry occurred last May 2017, and the more recent one (although similar) of Petya. Supported on a vulnerability of the SMB service for Windows systems [4], the former affected more than 200,000 machines in over 150 countries in just one day. Among several other companies, we can mention the infection of Spain's Telefónica, parts of Britain's National Health Service, FedEx, Renault and Deutsche Bank [5].

There exist two main types of ransomware: *locker* and *crypto*. In the first case, the access to the victim's device is blocked by generally locking the display or the keyboard. Instead, crypto-ransomware blocks the access to the information on the device by ciphering the victim's files and documents. In both cases, an economical ransom is subsequently demanded to the victim to restore the normal access to the kidnapped device/archives.

Locker-ransomware can be easily dismantled through various system restore techniques and tools. However, crypto-ransomware is much more destructive in general as current encryption techniques (*e.g.*, AES and RSA) are almost impossible to be reverted. And this despite some proposals have been developed to overcome this situation [6, 7, 8]. From this perspective, prevention is the first and most recommended method to fight against crypto-ransomware at present [9]. For that, user's training and education, use of legitimate sofware, periodical software update, data backups, or users' privilege management are well-known recommended best practices.

Since prevention mechanisms do not guarantee the occurence of malware activity, as in any other malware related context also detection schemes should be deployed to protect against ransomware. With this purpose, detectors of filesystem activities, API calls, registry access, C&C communications or encryption procedures are developed by researchers. However, the actual effectivity of such detection schemes must rely on their capability for a very early detection. Otherwise, detection could be useless once the system is

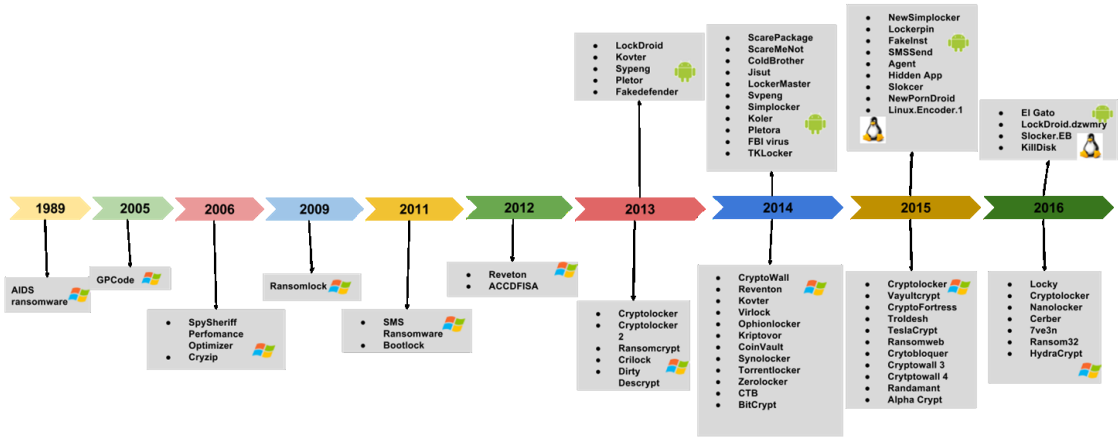

Figure 1: Ransomware families and variants.

affected by the ransomware action.

We contribute here a novel integrated detection plus reaction approach for crypto-ransomware based on the use of special *honeyfiles* with three main benefits: *i)* the ransom operation is completely blocked when the trap file is accessed, *ii)* countermeasures are automatically launched to solve the infection, and *iii)* the complexity and overhead involved in the solution are really low. The approach is implemented and evaluated here for Unix platforms through a tool named *R-Locker*, it standing for (in what tries to be in some sense a word game about ransomware and its effect) *ransomware locking*.

The rest of the paper is organized as follows. Section 2 presents main efforts in the field of fighting against crypto-ransomware. In this context, we introduce in Section 3 a novel methodology based on the use of special honeyfiles to thwart the ciphering action of a ransomware sample. The specific implementation for Linux platforms, R-Locker, is described in this section too. The general performance of the tool, both in terms of accuracy and efficiency, is evaluated and discussed in Section 4. Finally, Section 5 summarizes the contributions of the paper and outlines some further work.

2. Related work

Assuming that we are affected by ransomware, should we to pay or not? The recommended answer to this question is clear [10, 11]: *Paying the ransom*

does not solve the problem because there is not warranty neither to recover the data nor to suffer again the extorsion to continue paying! This way, instead of paying the ransom it is recommended to clean/replace the machine/s and restore the data affected by ransomware from data backups. As an example of this consider the case of MedStar, a non-profit group that manages a number of hospitals in the Baltimore and Washington area. A Samsam ransomware attack in March 2016 requested MedStar 45 Bitcoins for restoring the encrypted files. Fortunately, MedStar did not pay the ransom since it had a backup of the encrypted information [12]. Beyond recovering the data, it is also recommended to notify the authorities about the incident [9].

From the above, and as it has been mentioned in the previous section, the adoption of prevention schemes is highly encouraged to solve ransomware infections. However, they do not completely prevent the appearance of ransomware. As a consequence, a number of proposals are also available to detect this malicious behavior. The most common detection methodology is signature-based, so that an alarm is triggered if a certain well-known 'pattern' is observed. As an example, McAfee affirms to make use of more than 8 million ransomware signatures in its solutions [13]. A complementary detection methodology to signature-based is that of anomaly detection. In this case, the activity of the target environment is monitored in search of non-expected suspicious events [14].

According to the usual operation carried out by ransomware, common specific events refer to filesystem activity[1] [15, 16, 17]:

- Increasing number of files with well-known extensions like, let's say, *.locky*.
- Modificacion of specific files like *PIPE\lsarpc*, *\Device\Ip* or *system.pif*, among others.
- Execution of special commands like *vssadmin*, used to clean all volume shadow copies thus making restore of the system impossible.
- Suspicious access to MFT (Master File Table), mainly aimed at modifying and/or deleting the original files under attack in a very short period of time.

[1]System activity can be monitored with tools like SSDT (System Device Descriptor Table) and IRP (I/O Request Packets).

- Modification of the MBR (Master Boot Record), to prevent the system from loading the valid boot code by replacing it with the ransom message to be displayed.

A real tool developed to detect crypto-ransomware based on filesystem activity is UNVEIL [18]. Firstly, it randomly generates a realistic user environment by generating a set of documents and adding them to the sandbox filesystem. Then, during execution UNVEIL extracts features from I/O requests such as the type of request (*e.g.*, open, read, write) and the entropy of the data buffer if present. These events are then matched against a set of I/O access pattern signatures as an evidence that the sample is in fact ransomware.

Other ransomware-specific events are related to API calls. For example, a significant number of locker-ransomware samples use functions like *CreateDesktop* to lock the victims desktop by creating a new one and making it persistent. Moreover, disabling some keyboard shortcuts (*e.g.*, Windows key+Tab) will prevent the victim bypassing blocking. In the case of crypto-ransomware (*e.g.*, *CryptoWall*), the use of standard system functions like *CryptEncrypt* is common to encrypt files. Regretfully, this can be easily bypassed by attackers through the development of their own cryptosystems.

It is also usual that ransomware samples modify system registry values to specify some valuable configuration for the attacker's purposes. Among others [16], this is the case of *HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run* for the automatic execution of programs at the session start.

Additional relevant events from the perspective of detecting crypto-ransomware operation concern network activity. For example, ransomware usually encrypts files using an AES-256 encryption key which is encrypted using a 1024/2048-bit RSA public key. The RSA key is remotely generated on an external C&C server once the compromised device sends a POST to the server. This way, if the ransomware cannot connect to the C&C server, the malicious action will not be performed. In this line, authors in [19] propose the use of dynamic blacklisting of the proxy servers to detect malicious C&C communications in *CryptoWall* ransomware, a clone of *CryptoLocker* (`https://en.wikipedia.org/wiki/CryptoLocker`).

Taking into account all of the above, works like [20] propose complex multi-source monitoring systems aimed at detecting ransomware infection. Such a kind of systems also exist for mobile platforms, like HelDroid [21]. In

this case, three types of events are analyzed:

- Dynamically allocated strings derived from the execution of the malware sample in a sandbox. This will allow to detect potential key words like 'threat', 'law', 'porn', 'ransom', 'payment', etc.
- Execution of flows originating from functions that access the storage and ends into functions that write encrypted content.
- Execution paths containing certain heuristics about locking strategies.

Although some authors have developed further detection methods, like the use of formal methods to identify malware's code intructions in [22], filesystem monitoring is a recurrent method to detect crypto-ransomware infection. In this line, authors in [23] present ShieldFS, a system that monitors the low-level filesystem activity to update a set of adaptive models that profile the system activity over time. Whenever one or more processes violate these models, their operations are deemed malicious and the side effects on the filesystem are transparently rolled back. Authors in [24] propose CryptoDrop, an early-warning detection system based on three primary file access indicators: file type changes, similarity measurement and Shannon entropy.

A filesystem activity-based real detection tool available either for Windows and Linux is Cryptostalker (`https://github.com/unixist/randumb#cryptostalker-example`), which monitors a specific folder and generates and advertisement when more than a certain number of files into it are modified within a given time interval.

Despite the detection efficacy of the previous techniques are argued to be adequate (take into account that no definitive solution exists for the problem yet), further proposals advocate to get earlier detection through honeypot like techniques [25]. A real tool in this line for Windows platforms is *Anti Ransom* [26]. More recently, and as a consequence of the effect of WannaCry and Petya on Windows systems, Microsoft has introduced a control folder access to prevent data from ransomware and other malicious apps and threats in Windows 10 (`https://gbhackers.com/microsoft-introduced-a-control-folder-access-to-prevent-data-from-ransomware-and-other-malicious-apps-and-threats-in-windows-10-insider-release/`).

However, the real capability of this kind of approach is very limited because the existence of *witness* files may be not enough to detect and stop in time the action of the ransomware sample. In this context, we contribute

here a novel *honeyfile*-based methodology to fight against crypto-ransomware. Particularized in the tool named *R-Locker* for Unix platforms, it presents three principal benefits in preserving the data of the target system:

- The honey archive deployed is not a 'normal' file but FIFO like, so that a ransomware accessing the trap file will be completely blocked.
- The honeyfile is connected to a process in such a way that, when accessed, a response procedure is automatically launched to effectively defeat the infection.
- The complexity and cost of the solution are really low and do not interfere with the normal operation of the environment.

In the rest of the document the proposal will be properly described and evaluated.

3. A novel honeyfile-based approach to thwart crypto-ransomware action

In this section, we first introduce a general functional methodology aimed at thwarting crypto-ransomware action. It should be lightweight while accurate and efficient in defeating the threat. Based on this, and as a proof of concept, we will subsequently describe *R-Locker*, a specific implementation of the proposed methodology for Linux platforms.

3.1. Honeyfile-based approach for ransomware solution

Crypto-ransomware operation relies on scanning the infected machine's filesytem to find files, either indiscriminately or selectively according to specific file extensions (pdf, doc, jpg, etc.), and access them to encrypt the information. Based on this general behavior, we propose as a novel anti-ransomware solution to create a *honeyfile* intended to serve as a trap for the malware. Such a proposal will present the following main features and benefits, **F**={F1, F2}:

F1. The ransomware sample will be definitively blocked when accessing the honeyfile, so that the rest of the system will remain undamaged.

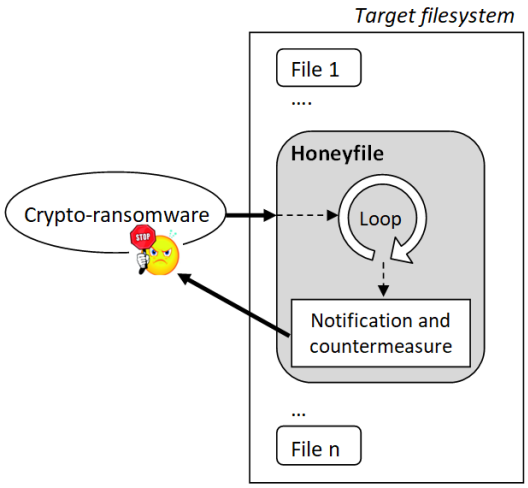

Figure 2: Honeyfile's functional methodology.

F2. In addition to locking the ransomware, the malicious event should be properly notified and/or a countermeasure automatically deployed to solve the threat.

The above methodology corresponds to the functional architecture shwon in Figure 2. Such an operational procedure is conceptual and should be independent on the specific target platform or OS considered (Windows, Unix, iOS, ...). In addition to the previous desired anti-ransom operation, some other demands should be satisfied in order to get a scalable, usable and, as such, valid solution for real environments. Some principal requirements, $\mathbf{R}$={R1-R5}, in this line are as follows:

R1. *Effectivity*, so that the harmful action of the ransomware over the system is null or as minimum as possible. Otherwise, the solution might not be such.

R2. *Low consumption*, both from the perspective of computation, memory usage and storage. Otherwise, it could not be scalable.

R3. *Plainness*, without necessarily requiring special privileges for installation and execution. Otherwise, it will not be used by end users.

R4. *Transparency*, from the perspective of the normal operation of the overall environment; that is, the solution should not affect the rest of applications and services. Otherwise, unexpected behaviors and malfunctions can appear.

R5. *Simplicity*, so that no additional complex procedures are necessary. Although not mandatory, this is an appealing additional property for a real system.

Developing a general solution accomplishing the previous main benefits **F** and requirements **R** is not a trivial task. However, as a proof of concept which can be further extended to other environments, we devise in the next subsection a specific implementation for Unix systems which accomplishes all of them. It is supported on the usage of *named pipes* or FIFOs, and is as described in the next.

3.2. R-Locker: Implementation for Unix platforms

As stated, we will discuss here the implementation of the proposed methodology for Unix platforms, in particular for Linux systems. For that, we first describe the detection solution itself, and then how it should be deployed in a target system to be properly protected.

In order to achieve the principal feature or benefit for the honeyfile solution, F1 (*i.e.*, blocking the ransomware when accessing it), while accomplishing the fixed requirements (in particular, R3 and R4 related to no special privileges and non-interaction with the rest of the system), we though of making use of an 'infinite archive' to distract the malware and thwart its action. For instance, we can create a link to the `zero` device (`/dev/zero`), which is an infinite source of zeros. Regretfully, although this solution may work it is not effective for a main reason: a high disk space is needed for the trap file. That is, requirement R2 would be thus unaccomplished.

We can also simulate an infinite file by modifying reading function related libraries. However, some problems will arise in this case: (a) we don't know which of these (on the other hand diverse) libraries will be used by the ransomware, and (b) the techniques to modify the functions are usually complex (*e.g.*, binary instrumentation [27]). As a consequence, R4 and R5 would be unaccomplished.

A simple and elegant solution to achieve our goals, both F1 and F2, while satisfying all the requirements **R** established for the solution, is to make use

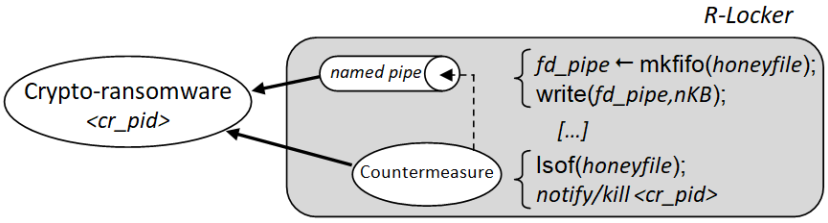

Figure 3: R-Locker's anatomy.

of *named pipes* or FIFOs. A FIFO is a pipe with a name into the filesystem, and with two very interesting and useful properties for our purpose due to such a dual nature [28]:

- As an entry in the filesystem, a FIFO is (in some sense) a regular file accessible as usual by applications and services.
- As a pipe, a FIFO also implies a communication channel between two processes, a reader and a writer. At this point, it is important to mention that the synchronization between the reader and the writer is automatically managed by the kernel. This way, if a process tries to read from an empty channel or that contains less information than the expected one, the kernel will block the reading process until the writting one inserts more bytes into it.

From the above, our specific anti-ransomware proposal for Unix systems consists of the development of an application, named *R-Locker* because of its objective (*i.e.*, to lock ransomware), as follows (see Figure 3):

1. It first creates a named pipe by using the function `mkfifo()`. Such a file will be our central honeyfile or trap file.
2. Second, a few bytes are written on it. The bytes should be different from `EndOfFile` bytes and the number of them will depend on the specific system, although typical low values are about 3 KB.
3. Symbol *[...]* in Figure 3 stands for a sleeping stage. That is, since no reading pair process is accessing the trap file at the beginning, the abovementioned writting process is blocked by the system ... At this point, the trap is ready and awaiting for a prey!

4. After that, when an external process (the supossed ransomware) accesses the honeyfile and starts reading it, the flow is automatically unblocked and a countermeasure executed as follows:
    - First, the pid/s of the application/s accessing the (honey)file is determined through `/proc/<pid>/fd` or the command `lsof`.
    - Second, the user is properly notified in order to, if so, kill the corresponding process (<*cr_pid*> for ransomware in Figure 3) and even uninstall it from the system.

In summary, we can see in Figure 4 the overall operational flow of R-Locker. After implementing it, its installation is very simple:

1. After executing the associated bin, a main screen like that shown in Figure 5 will appear, where 'Start service' initiates the tool and 'Stop service' finishes it.
2. In the first case, a central unique trap file is created as described above.
3. Aimed at maximizing the coverage of the solution while minimizing the resource consumption, we deploy around the filesystem a set of logical links pointing to the central trap file. They will act as a distributed honeyfile solution for the whole filesystem (Figure 6(a)).
4. Instead of starting the service, through the button 'Stop service' both the honeyfile and the logical links will be removed, and the process/application stopped.

With respect to the abovementioned logical links acting as distributed honeyfiles, it is important to remark that:

1. With early detection purposes, the links are included as the first entry into the corresponding folder.
2. The use of 'attractive' names and/or extensions (*e.g., personalvideo.mpg*) may be interesting to capture, if so, the attention of potential selective ransomware.
3. The links can be distributed all around the filesystem, including kernel related folders, or just covering the user's filesystem (Figure 6(b)). In the former case, special privileges will be required during installation.

Once the global methodology is described and how it is particularized for Linux systems, we shall evaluate the approach in a real scenario in the next section.

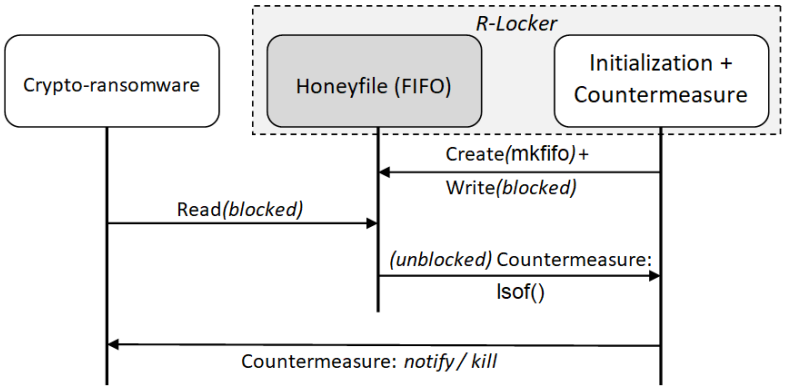

Figure 4: R-Locker's operational flow.

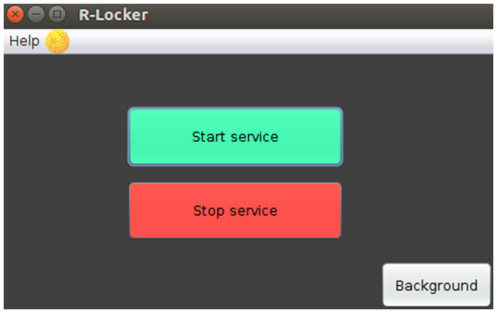

Figure 5: R-Locker's main screen.

4. Experimental evaluation

After describing the anti-ransomware solution, this section is devoted to evaluate the performance of R-Locker in a real environment with real samples. With this aim, an experimental scenario is first described. After that, some experiments are carried out to obtain relevant operational figures in terms of detection accuracy, resource consumption and overall impact on the target system. Finally, main conclusions and some other principal issues are discussed.

4.1. Scenario and crypto-ransomware samples

The scenario deployed to carry out our ransomware detection experiments is a simple end machine running Ubuntu 16.04 64 bits with Internet access.

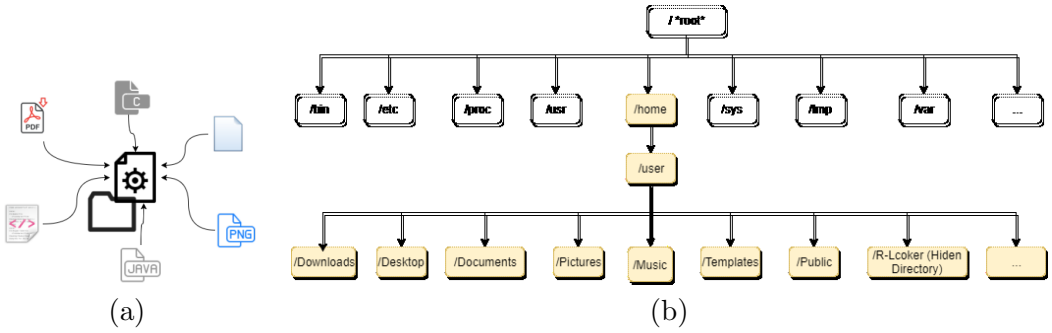

Figure 6: Deployment of honeyfiles around the filesystem as logical links to a (central unique) R-Locker's trap file: conceptual links (a), and user's filesystem */home/user* in our specifc scenario (b).

No special services are installed on it except a standard Apache HTTP server with user's personal information.

Regarding crypto-ransomware samples for evaluation purposes, three are the cases here considered:

- *C1*: As a first approximation to the problem, we have used the Linux GPG Suite (`https://gpgtools.org/`) to develop a generic ransomware sample devised to encrypt the user's filesystem in a systematic way.

- *C2*: As a proof of concept for Linux platforms, Bash-Ransomware (`https://github.com/SubtleScope/bash-ransomware`) is an open ransomware tool. It is based on OpenSSL to encrypt files and its operation is similar to *Cryptowall*, a well-known and very harmful crypto-ransomware appeared in later 2013 against Windows systems.

- *C3*: More specific than the previous ones, and motivated by their well-known impact on end systems, we have also considered *Linux.Encoder.1* (`https://vms.drweb.com/virus/?i=7704004&lng=en`) and *WannaCry* (`https://github.com/aguinet/wannakey`) as ransomware samples for our experimentation. Linux.Encoder.1 (also known as *ELF/Filecoder.A* and *Trojan.Linux.Ransom.A*) was the first specific ransomware trojan for Linux platforms and has affected thousands of users all around the world from its apparition in later 2015. More recent and with a higher impact, WannaCry is specifically developed against Windows systems, so that we will execute it on our Linux machine through Wine^{HQ} (`https://www.winehq.org/`) software.

4.2. Results and R-Locker performance

Under normal operation of the environment, and with R-Locker installed and running, we executed each of the ransomware samples considered in cases C1 to C3. In all of them, the behavior of R-Locker was as expected regarding the desired features: the sample was inmediately blocked (F1) and subsequently notified for its removal (F2). From the perspective of the operational requirements R1-R5, R-Locker performed as follows:

R1. *Effectivity*: Mainly because of the distribution of links to the central honeyfile around the user's filesystem, as well as their inclusion as the first entry in every folder, each of the ransomware samples has been blocked and detected/notified inmediately (see Figure 7 as an example). That is, the detection accuracy is 100% and the damage caused by the ransomware on the system null.

R2. *Low consumption*: Thanks to the FIFO design, R-Locker just involves about 2KB of disk storage for the honeyfile (the logical links do not require physical space but only an *inode* entry), no more than 1.000 lines of code to implement the complete solution (just around 100 for the basic functionality), and negligible computational cost to capture and detect the ransomware.

R3. *Plainness*: R-Locker can be installed and executed without requiring special privileges or permissions. In this respect, the only (obvious) limitation is that just a part of the complete filesystem (that corresponding to the user that installs R-Locker) is protected (see Figure 6(b)).

R4. *Transparency*: To check potential undesired behaviors and malfunctions, we have also accessed the disposed links through legitimate programs: Adobe Acrobat for *pdf* extensions, Microsoft Word for *doc* files, etc. They all aborted (without consequences) the reading process on the file due to 'format error'. In summary, R-Locker has demonstrated to achieve the desired goal without affecting the normal operation of the rest of the system. However, further discussion about this topic is addressed in Section 4.3.

R5. *Simplicity*: R-Locker is demonstrated to be simple and autonomous, no additional processes being necessary to complement its functionality.

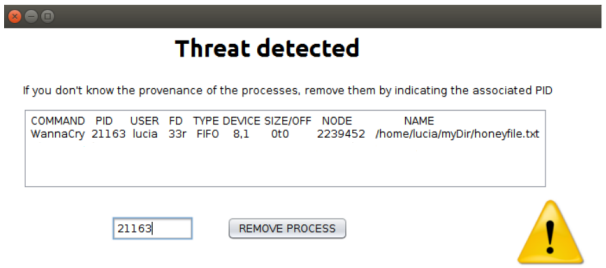

Figure 7: Threat detection performed in R-Locker.

In comparison with some other detection tools tested in our experimentation (*e.g.*, Cryptostalker –see Section 2–), R-Locker has shown much better performance in several aspects. This way, although the ransom was detected and advertised by such alternative tools, the effect suffered by the system was more severe that that with our approach. In fact, a number of files were encrypted before the tool even advertised the infection. In addition, a continuous (and thus consuming) monitoring process over time was necessary to determine the occurrence of the potential crypto-ransomware attack.

4.3. Further discussion

As a consequence of the previous analysis, we must necessarily conclude the goodness of R-Locker from a number of aspects. In addition, we must remark that the approach just does rely on the fact that a ransomware will access files to encrypt information, and not on more complex behaviors or previous knowledge. This way, our solution is able to defeat zero-day ransomware attacks. Indeed, the detection experiments carried out above correspond to zero-day/unknown attacks from the perspective of R-Locker, as they have not been previously observed nor used to train the tool.

Despite all of the above, some additional discussion is needed for a successful deployment of the approach in real environments.

As already mentioned, the disposal and distribution of the logical links to the central honeyfile is primordial to protect the whole system. This way, such a task should be carefully addressed for the specific target system in order to maximize the chance to defeat attacks. For that, questions like *"Where should we locate the links, just into the root folder or into all in the structure? What extensions and names to use?"*, among others, should be

properly addressed.

Also related to the links, it is convenient to monitor them in order to determine the potential occurrence of changes in their distribution. If so, the situation should be corrected. As a particular case, we should take care of the potential removal of the central trap file. In our case, we have tried to minimize this risk by creating it as a hidden file ('.' prefix in Linux) into the specific folder where the application R-Locker is installed. However, a monitoring process to detect its potential removal is still necessary. In particular, we have used *inotify* for that, which allows to monitor events in the filesystem [28].

We are aware that our defense can be partially bypassed by accessing in a random way the files into a given folder. In the worst case, the sample will be blocked after encrypting all the files in the folder. To avoid this, more honey links might be deployed per individual folder. Maybe a more critical procedure to bypass the defense is to study the honeyfile's metadata in order to corroborate its correspondence with the associated extension (pdf, jpg, ...), which is not respected in our case. However, the use of logical links offuscates the specific type of file they point to and, thus, complicates such a verification to be performed by the ransomware, which favors our solution.

There is still an additional aspect about R-Locker that deserves a brief discussion. In its current state, killing and/or uninstalling the suspicious malware relies on the end user, who, if so, is required to introduce the associated detected PID and confirm the subsequent action. To improve the usability of the proposal, the user's intervention can be replaced or complemented by the (semi)automatic use of pre-defined black-/white- lists of well-known harmful/legitimate applications (*e.g.*, system commands like `copy` or applications used to generate data backups).

Despite all of the previous aspects are relevant for new better versions of R-Locker, we firmly think that none of them devaluate the promising performance achieved by the approach in its current state. In particular, if we take into consideration that none of the current available solutions are definitive to solve this pressing social problem.

5. Conclusions and future work

A general methodology intended to thwart harmful crypto-ransomware action is introduced here. It is based on the deployment of a honeyfile structure to block the ransom when it accesses a trap file, thus allowing to preserve

the rest of the system. Moreover, while the ransom is blocked, it would be desirable to automatically launch a countermeasure intended to eradicate the problem from the environment. As a proof of concept, we have implemented the methodology for Linux platforms by making use of named pipes or FIFOs. The resulting tool is named R-Locker.

We have demonstrated by means of experimentation the good behavior of our approach from a numbers of perspectives, which include detection accuracy and operation efficiency. This way, R-Locker achieves the detection objectives established with low resources consumption and without affecting the normal operation of the system. We have also discussed some practical aspects mainly related to the deployment and maintenance of the honeyfile structure around the filesystem, in order to maximize the success of the method in real environments.

As further work, we are working on improving our current implementation in some of the aspects mentioned in Section 4.3. In addition, the development and evaluation of the general methodology for other platforms is also desirable. In particular, for Windows and Android because of their general acceptance among users and companies and their high affectation by malware at present. Although the general honeyfile solution is applicable to both types of platforms (*i.e.*, FIFOs are accepted in both cases), some specific aspects should be carefully addressed in future works. For example, named pipes are not included into the normal filesystem space in Windows, and the use of regular logical links is not allowed in Android.

Acknowledgement

This work has been partially supported by Spanish Government-MINECO (Ministerio de Economía y Competitividad) and FEDER funds, through project TIN2014-60346-R.

References

[1] NCCIC: "Malware Trends. October 2016". *Homeland Security Report*, 2016. Available at https://ics-cert.us-cert.gov/sites/default/files/documents/NCCIC_ICS-CERT_AAL_Malware_Trends_Paper_S508C.pdf.

[2] J.C Chen, B. Li: "Evolution of Exploit Kits. Exploring Past Trends and Current Improvements". Trend Micro Inc., 2015. Available at https://www.trendmicro.de/cloud-content/us/pdfs/security-intelligence/white-papers/wp-evolution-of-exploit-kits.pdf

[3] P. García-Teodoro, M. Robles-Carrillo: "Ransomware: Technical Aspects and Legal Issues". *IEEE Communications Magazine*, in press, 2017.

[4] Symantec: "Ransom.Wannacry"'. Report, 2017. Available at https://www.symantec.com/security_response/writeup.jsp?docid=2017-051310-3522-99

[5] Wikipedia: "WannaCry Ransomware Attack", May 2017. Available at https://en.wikipedia.org/wiki/WannaCry_ransomware_attack

[6] M. Weckstén, J. Frick, A. Sjostrom, E. Jarpe: "A Novel Method for Recovery from Crypto Ransomware Infections". *2nd IEEE International Conference on Computer and Communications*, pp. 1354-1358, 2016.

[7] A. Palisse, H. Bouder, J.L. Lanet, C. Guernic, A. Legacy: "Ransomware and the Legacy Crypto API". *International Conference on Risks and Security of Internet and Systems (CRiSIS)*, pp. 11-28, 2017.

[8] E. Kolodenker, W. Koch, G. Stringhini, M. Egele: "PayBreak : Defense Against Cryptographic Ransomware". *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pp. 599-611, 2017.

[9] FBI: "Incidents of Ransomware on the Rise. Protect Yourself and Your Organization". Report, 2016. Available at https://www.fbi.gov/news/stories/incidents-of-ransomware-on-the-rise

[10] Bitdefender: "Ransomware. A Victims Perspective. A Study on US and European Internet Users". Report, 2016. Available at http://www.bitdefender.com/media/materials/white-papers/en/Bitdefender_Ransomware_A_Victim_Perspective.pdf

[11] C. Everett: "Ransomware: To Pay or Not To Pay?". *Computer Fraud & Security*, vol. 4, pp. 8-12, 2016.

[12] S. Gallager: "Maryland Hospital: Ransomware Success Wasn't IT Department's Fault". *Ars Technica*, 2016. Available at https://arstechnica.com/security/2016/04/maryland-hospital-group-denies-ignored-warnings-allowed-ransomware-attack/

[13] McAffee: "How to Protect Against Ransomware". Report, 2016. Available at https://www.mcafee.com/us/resources/solution-briefs/sb-how-to-protect-against-ransomware.pdf

[14] P. García, J.E. Díaz-Verdejo, G. Maciá, E. Vázquez: "Anomaly-based Network Intrusion Detection: Techniques, Systems and Challenges". *Computers & Security*, vol. 28; pp. 18-28, 2009.

[15] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, E. Kirda: "Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks". *12th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, pp. 1-20, 2015.

[16] Monika, P. Zavarsky, D. Lindskog: "Experimental Analysis of Ransomware on Windows and Android Platforms: Evolution and Characterization". *2nd International Workshop on Future Information Security, Privacy & Forensics for Complex Systems*, pp. 465-472, 2016.

[17] R. Brewer: "Ransomware Attacks: Detection, Prevention and Cure". *Network Security*, pp. 5-9, 2016.

[18] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, E. Kirda: "UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware". *Proceedings of the USENIX Security Symposium*, pp. 757-772, 2016.

[19] K. Cabaj, W. Mazurczyk: "Using Software-Defined Networking for Ransomware Mitigation: The Case of CryptoWall". *IEEE Network*, pp. 14-20, 2016.

[20] J.K, Lee, S.Y. Moon, J.H. Park: "CloudRPS: A Cloud Analysis based Enhanced Ransomware Prevention System". *Journal of Supercomputing*, pp. 1-20, 2016.

[21] N. Andronio, S. Zanero, F. Maggi: "HelDroid: Dissecting and Detecting Mobile Ransomware". *18th International Symposium on Research in Attacks, Intrusions and Defenses*, pp. 382-404, 2015.

[22] F. Mercaldo, V. Nardone, A. Santone, C.A. Visaggio: "Ransomware Steals Your Phone. Formal Methods Rescue It". *International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE)*, pp. 212-221, 2016.

[23] A. Continella, A. Guagnelli, G. Zingaro, G. Pasquale, A. Barenghi, S. Zanero, F. Maggi: "ShieldFS: A Self-healing, Ransomware-aware Filesystem". *32nd Annual Conference on Computer Security Applications*, pp. 336-347, 2016.

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

[25] C. Moore: "Detecting Ransomware with Honeypot Techniques". *Cybersecurity and Cyberforensics Conference*, pp. 77-81, 2016.

[26] J. Yago: "Security Projects: Anti Ransom". Available at http://www.security-projects.com/?Anti_Ransom.

[27] M.A. Laurenzano, M.M. Tikir, L. Carrington, A. Snavely: "PEBIL: Efficient Static Binary Instrumentation for Linux". *IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*, pp. 175-183, 2010.

[28] M. Kerrisk, *The Linux Programming Interface*, No Starch Press, 2010.