A behavioural in-depth analysis of ransomware infection

Yassine Lemmou1✉ | Jean-Louis Lanet2✉ | El Mamoun Souidi1✉

1Mohammed V University in Rabat, Faculty of Sciences, Rabat, Morocco
2INRIA, LHS-PFC, Rennes, France

Correspondence
Yassine Lemmou, Mohammed V University in Rabat, Faculty of Sciences, LabMIASI, BP 1014 RP, Rabat 10000, Morocco.
Email: yassine.lemmoo@gmail.com

Abstract
Ransomware is a type of malware that has spread rapidly over the last 4 years, causing significant damage, especially in Windows environments. It is designed to encrypt or block victim’s data, including documents, backups, and databases, unless a ransom is paid. In this study, the authors present the results of their research on Windows crypto-ransomware during the last 3 years by exploring and discussing the relevant ransomware behaviours. The results of this study can be used to identify or to detect the ransomware. Indeed, these behaviours were extracted from in-depth manual analysis of more than 20 ransomware families, including the known and the recent families. In addition, some extracted behaviours were automatically searched for more than 200 different ransomware collected during 2019.

1 | INTRODUCTION

Malware detection or identification is essentially based on statistical analysis; in other words, it looks for signatures. This detection/identification consists to process a given file like a sequence of bits out of any execution context. Indeed, it searches according to one or several databases for a number of bits extracted from the given file. When this file arrives at a specified machine and even before this file is written in the drive (if real-time scanning is enabled), the traditional anti-virus scans this file searching for signatures. If a signature is found, the file will be deleted or quarantined. The signature can be unique for a specified file or for multiple different files.1

To evade a particular signature, a single malware can generate multiple variants that vary in their static properties using packers or by altering a character in its content. This is a disadvantage for the anti-virus signatures. Unlike this category of anti-virus, behavioural or host-based indicators are also used to detect malware. These indicators focus on what the malware is doing on the target machine and not on the characteristics of the malware as a file. Several studies have been published on this topic like the study in [1], which classifies malware based on their application programming interface (API) calls and behavioural analysis.

Recently, some anti-virus products have applied machine learning to detect malware. Although signatures can find the exact match with the searched pattern, the machine learning models search for the closest matches with the searched pattern. Therefore, it can catch more variants of a given malware according to a single pattern. Many papers have been published in this regard demonstrating the effect of applying machine learning on malware detection, for example, the work of Suleiman et al. [2] on Android malware detection.

In this research on ransomware, we found many ransomware in VirusTotal (VT) not detected by any anti-virus engine, for example, some variants of FTCode ransomware. The first analysis of a variant of this ransomware in VT was performed at 2019-09-30T20:59:31 (UTC) and not detected by any anti-virus engine. Nine days later, we found another variant of this ransomware detected only by two anti-virus engines. Therefore, compared to other methods (host-based indicators and machine learning models) that involve complex programmes and more resources, we suggest that static anti-virus engines have shown their limit on malware detection, especially on ransomware.

Generally, the single objective of a ransomware on a target machine is encrypting files. Therefore, most ransomware keeps some common behaviours. Each ransomware detection or prevention must be based on monitoring several behaviours to detect the ransomware without consuming many resources on the machine. For this reason, we present this article as a continuation and a complementary to the existing works on ransomware. Indeed, our motivation is to present to the research community a set of more than 20 ransomware behaviours to detect, identify the ransomware, and limit its
damage. The extracted behaviours are divided into the following three categories: pre-encryption behaviours, post-encryption behaviours, and behaviours related to the process of files encryption. These behaviours can be involved or used in the discussed methods of the previous papers on monitoring file system or network activities, using machine learning or other methods to detect/identify the ransomware. They can also be used jointly with the proposed behaviours in these papers to increase the effectiveness of their tools/ideas.

In this article, the discussed behaviours were extracted by a manual complete analysis of more than 20 ransomware of different families. Then, some extracted behaviours are discussed according to the results of an automatic analysis of more than 217 different collected ransomware during 2019 using the new version of Malware-o-Matic (MoM) [3] that we implemented. Table A1 shows the manually analysed families. These families were chosen depending on the needs of our laboratory on analysing some specified ransomware. The samples were fully analysed using all malware analysis steps (static, dynamic, and reverse engineering) and according to some works on ransomware detection or prevention. In other words, we did not analyse these families to decrypt the encrypted files, but we analysed them to extract the useful behaviours to detect the ransomware. Then, according to these collected behaviours, we present and discuss a collection of known tools of ransomware detection, which we found that the majority of these tools should increase their effectiveness on the detection of ransomware.

The term “ransomware” covers two types of malware: locker ransomware and crypto-ransomware. The first type of ransomware denies or blocks access to files by locking the whole screen of the device. Indeed, this class does not encrypt any files on this device. Today, this class is not prevalent on Windows and it is only seen on Android platform. Some notable examples of this class are Reveton, Winlock and Urausy on Windows, Android/Koler.BoStr, and Android Locker on Android. The second type of ransomware encrypts important data on a target machine. Several Windows crypto-ransomware are cited in this article. On Android, we can cite Simple Locker ransomware that encrypts files on the device and its SD card or DoubleLocker that encrypts files and changes the PIN of the device to a random number.

Several studies like [4] mention that the AIDS Information Trojan, called PC Cyborg, is the first created ransomware around 1989. Moreover, this topic of ransomware has been investigated only during the last 25 years. For example, the work of Adam Young and Moti Yung on cryptoviral extortion [5] in 1996 and the work of Alexandre Gazet that published the first ransomware analysis in 2010 [6] or other works on cryptovirus like [7]. During our research works on ransomware, we found that the ransomware is not generally a cryptovirus. In other words, the ransomware is not a virus as the virus is defined in computer virology by any self-reproducing program. The behaviour of self-reproduction is discussed in the following section.

The frequency of papers that discussed the ransomware in a technical or a theoretical side has increased since the end of 2015. Many ideas were discussed to detect the ransomware or to limit its damage. In 2015, the work of Kharraz et al. [8] suggested some mechanisms to detect the ransomware by monitoring abnormal file system activities. The next year, they published a continuation of their work [9] in which they presented a dynamic analysis system called UNVEIL to detect the ransomware. In the same year, Continella et al. [10] proposed another solution to detect the ransomware named Shield-FS. This solution tries to make a difference between I/O file system requests of ransomware and benign applications. The results presented in these two papers are promising, but until the date of writing this paper, UNVEIL and Shield-FS are not available for tests and downloads. Contrary to these two tools, Scaf et al. [11] made their tool CryptoDrop available for download. Using a set of monitoring ransomware behaviours, this tool can alert users during suspicious file activity. Data Aware Defense (DaD) [3] is another tool to detect the ransomware available for download. This tool monitors the file system activity using the chi-squared test instead of Shannon entropy that is used in CryptoDrop. During our research, we tested CryptoDrop, DaD, and other ransomware detection tools on some ransomware. We present our conclusions on these tools in the fifth section. Other works like [12] by Palisse et al. or [13] by Eugen et al. were based on monitoring the used algorithms to encrypt files. The first intercept calls made to Microsoft Cryptographic API. The second proposes PayBreak that observes the use of symmetric session keys of known libraries, then hold them for files decryption.

Other works on ransomware detection/classification or data recovery were also published. For example, Zhang et al. [14] proposed a static analysis framework based on N-gram opcodes with deep learning to classify ransomware. Lee et al. [15] applied some machine learning models to recover the original files from a backup system by classifying encrypted files based on file entropy analysis. However, the work of Mbol et al. [16] on TorrentLocker ransomware shows that the Shannon entropy cannot distinguish JPEG compression. Moreover, during our research, we found a version of Cerber that does not encrypt all the contents of the target files, which negatively affects the ransomware detection solution based only on entropy.

On other platforms like Android, several works on the detection of Android malware mainly focused on permission [17], static [18], and dynamic analysis [19]. On Android ransomware, we can cite the proposed method by Sanggeun et al. [20] to dynamically monitor memory usage, process usage, and read/write accesses to the file systems. Amirhossein et al. [21] proposed DNA-Droid, a detection framework that evaluates the APK samples using static and dynamic analyses. Their tool utilises deep neural network to distinguish between ransomware and benign programs. Another two-layer detection framework was proposed by Ferrante et al. [22]. The first layer is based on static analysis to extract opcode frequency. The second layer is based on dynamic analysis to create a set of ransomware behaviours like CPU/memory/network usage and system calls as an input of some machine learning models.

This article is organised as follows: Section 2 describes the extracted behaviours before files encryption, Section 3
discusses some behaviours related to the ransomware encryption, Section 4 presents other ransomware features that can be used to detect the ransomware, Section 5 discusses some ransomware detection/prevention techniques and tools, and Section 6 provides conclusion. The organization of this article is similar to how several ransomware make their infection on a target machine. But, it does not mean that all ransomware follow the same organization. For example, we can find some discussed behaviours in the second section that are performed at the end of infection and so on.

2 | RANSOMWARE PRE-ENCRIPTION

2.1 | Self-reproduction and overinfection

Self-reproduction and overinfection have been considered as two fundamental notions in computer virology theory since the first papers in this field like the Cohen definition in 1987 [23]. The first notion is the ability of a program to reproduce or to copy itself in another location, and the second notion is any infection or execution that follows the first infection/execution of the program. Generally, the ransomware is not a cryptovirus. Many ransomware do not copy their codes in other directories like AppData or Temp directories. For example, we found that FTCode, StopDjvu, PrincessLocker, and some versions of Cerber ransomware are not viruses. Instead they limit themselves to a simple infection according to the Adleman definition [24]. On the other hand, the 12 ransomware manually analysed in this study are self-reproducible programs. Table 1 shows these ransomware and the location of their copies. The copy of some ransomware like TeslaCrypt and Searab ends the infection of the executed ransomware and encrypts the target files, while other ransomware makes their copies looking for persistence on the target machine for a long time.

The results are not different for the ransomware that were analysed automatically. Indeed, we found at least 52 ransomware (25% of the automatically analysed ransomware) that copy their codes in other locations at the beginning of their infections. This indicates that this behaviour can be used with other ransomware behaviours to detect the ransomware. Especially, the case of an unknown process that starts its execution with a copy of its code in a suspicious location like %AppData% directory and runs this copy.

The process of self-reproduction was also seen on Android malware. The research security on this field found an Android worm (spreading device-to-device) in February 2018. This worm (called ADB.miner) replicates its code over Android devices using the opened port ADB debugging interface. Since worms are a class of viruses, this worm can be a virus. The behaviour of self-reproduction needs more investigations and discussions on the Android platform to answer the following questions:

- Is there any self-reproducing Android ransomware that replicates its code in other locations?
- If it exists, can this copy terminate the infection of the main program? In other words, can it encrypt the files instead of the installed application?

Effective malware tries to check if the current infection is the first infection on the target machine. Filiol [24] mentioned that running a malware without managing the overinfection can be fatal, especially in the case of appending viral code to a target program. In this case and related to the number of executions of this malware, the same viral code will be added multiple times to the target program. Most ransomware try to not be executed twice at the same time and/or not encrypt an

---

Table 1: Self-reproducing ransomware

| Ransomware       | Identical copy | Location of the copy       | Remarks                  |
|------------------|----------------|---------------------------|--------------------------|
| Nemty            | -              | -                         | Multi-step infection chain |
| Searab           | Yes            | %AppData%                 | Copy ends the infection  |
| LockerGoga       | Yes            | Temp                      | Copy ends the infection  |
| GandCrab         | Yes            | AppData\-Microsoft\-      | -                        |
| Dharma           | Yes            | System32 and others      | Two analysed versions    |
| CryptoLocker     | Yes            | %AppData%                 | Copy ends the infection  |
| TeslaCrypt       | Yes            | Windows                   | Copy ends the infection  |
| Spora            | Yes            | root directory (C:)       | -                        |
| Alpha            | Yes            | %appdata%                 | Copy ends the infection  |
| Diamond          | No             | %appdata%\-winmgr322.exe   | Copy ends the infection  |
| WannaCry         | -              | %appdata%\-winmg\-32.exe   | Multi-step infection chain |
| StopDjvu         | Yes            | AppData\-Local            | -                        |

---

2 https://blog.netlab.360.com/adb-miner-more-information-en/.
already encrypted file. Other ransomware coordinates all their infections on the target machine with one victim identity, then they demand one ransom to pay. Lemmou et al. [25] presented another side of managing the overinfection. It is the ability of a ransomware to infect multiple connected machines with a single victim identity and demand one ransom to pay for all the machines.

To not run twice at the same time and to ensure that only one instance of the ransomware is running, ransomware or generally malware call CreateMutex or OpenMutex API to create or to open a mutex. This call returns an error if the called mutex exists, which means that this malware is running on the machine. During our research on ransomware, most analysed ransomware uses a mutex. Recently, we discovered that the Napoleon version of Blind ransomware does not manage its multiple runnings at the same time. Running this ransomware two times makes it easily detected by some anti-virus engines. Moreover, these multiple executions cause multiple file-sharing violations during the file encryption, which means that some files are encrypted by the first execution and other files by the second execution.

On the other hand, some ransomware like LockerGoga, Scarab, and WannaCry create the mutex using a hardcoded name. Others, like GandCrab and Kraken, use a combination of values taken from the target machine to create the name of the mutex. The detection of these ransomware is possible looking for the used mutex name or the used combination to create the mutex name. Table 2 shows the used mutex names by some ransomware.

Most ransomware labels the encrypted files with an extension different to the target extensions. For example, BTCW are ransomware labels the encrypted files with the extension .payd ay, and it does not encrypt any file labelled with this extension. Other ransomware like Spora does not add extension to the encrypted files. These ransomware adds some specified values to the content of the encrypted files and they check these added values for each file before encryption in their infections on the same machine. In this case, these ransomware generates several similar ReadFile and WriteFile (e.g. the same buffer size) to check and label the encrypted files without new extension. This behaviour and the behaviour of adding extension are two behaviours to detect the ransomware.

The ransomware can infect the same machine several times with one victim identity. Then, it demands one ransom to pay. Indeed, to make this coordination between the infections, some ransomware like Cerber and Spora manage their multiple infections on the same machine using some files in specified directories. GandCrab and other ransomware communicate with their C&C and others like Scarab use some registry keys/values. An interesting example is TeslaCrypt [26]. This ransomware uses some registry keys/values to keep the identity of the victim and the encryption configuration. These registry keys/values allow TeslaCrypt (after each reboot using the run registry key) to encrypt the new created files after each reboot with the same identity. In the same way, Nemty ransomware uses three registry values in the registry key HKCU/Software/NEMTY to coordinate all its infections on a target machine. Figure 1 shows these registry values with their content. Any anti-virus engine can stop this ransomware before files encryption looking for the registry key NEMTY. This remark can be generalised on other ransomware by looking for some known or suspicious added registry keys/value.

The same behaviour was performed by SLocker Android ransomware. When this ransomware is installed, it uses SharedPreferences APIs to check if the target device was infected. SharedPreferences APIs are used to store a small collection of key values. Figure 2 shows that SLocker retrieves and holds the content of the preferences file named XH and then checks if the key bah is empty using getstring (String key and String defValue). This method retrieves a string value from the preferences. If the key bah is empty, SLocker will generate a random number and store it in the bah value.

### Table 2 Some ransomware mutex names

| Ransomware         | Mutex name                                                                 |
|--------------------|-----------------------------------------------------------------------------|
| PrincessLocker     | h0jUpevqHA (Evolution version)                                              |
| Kraken             | Microsoft-Kraken-[ComputerName]                                            |
| GandCrab           | Global\pc_group = [Pc_Group]&ransom_id = [Victim_id]                        |
core module, and the both processes have the same path. As mentioned earlier, this technique can be used to replace the memory content of a legitimate process. Therefore, the malicious process will have the path of the legitimate process. We can conclude that any malware or ransomware detection approach that trusts on the running processes based only on their paths can fail to detect the malware/ransomware that use this technique.

The installation process means that the ransomware or the malware builds a tree of files and folders from its code to run itself. This behaviour is mostly seen in the Python ransomware, for example, Striked ransomware. As shown in Figure 4, this ransomware builds its tree files and directories in %temp%
\_MEIXXXX directory. These files are used by this ransomware to encrypt the files. This behaviour can be used to detect the Python ransomware that put several files in temporary directories like the Temp directory.

To remain on the infected machine for a long time, malware or ransomware authors use various persistence techniques. These techniques allow the malware to continue its infection after machine restarts, reboots, or log-offs. This is achieved using various methods, for example, registry keys, Winlogon Registry, Task Scheduler, Startup directories, AppInit_DLLs, and others. Many ransomware use persistence to encrypt any new created file after the first infection. Others use these techniques to display their ransom notes. Table 3 shows some persistence mechanisms of some manually analysed ransomware.

Windows services also provide another method for persistence. The service can start automatically and may not show up in the task manager as a running process. This behaviour was seen in WannaCry ransomware. Indeed, this ransomware creates two services to ensure persistence mssecsvc.exe (worm component) and tasksche.exe
(ransomware loader). Monitoring the persistence techniques can be used to detect the ransomware before encryption. Table 3 shows various ransomware that ensure their persistence before encryption. Moreover, some ransomware name the used registry keys/values to ensure persistence with suspicious names, for example, WannaCry, TeslaCrypt, and GandCrab. We think that monitoring the used names (suspicious or random names) in some registry keys like Run and RunOnce can be combined with the other malicious ransomware behaviours to detect the ransomware.

Several ransomware use the registry keys/values to store some data like the identity of the victim, the used public key, and other parameters of infection. For example, Alpha ransomware adds in addition to the Run registry key, another registry key in HKCU named Alpha containing a registry value named CryptoKey. TeslaCrypt adds the registry key xxxsys in HKCU\Software\ containing a value key named ID. We do not cover all created or modified registry keys by the analysed ransomware. Our idea is to mention that monitoring the used names in the registry keys (e.g. using the names of the known ransomware or using random terms) can be used to detect the ransomware.

Windows ransomware try to persist within the target machine. Android ransomware are not an exception. For example, some Android ransomware request administrator privileges (BIND_DEVI-CE_ADMIN), which make difficult to uninstall these applications from the device. Another persistence method is widely used by Android ransomware is RECEIVE_BOOT_COMPLETED. Jing et al. [34] found that from 2721 Android ransomware collected, 2489 (91.4%) ransomware request RECEIVE_BOOT_COMPLETED permission, which allows them to be started when the device starts up.

### 2.3 Network activities

Monitoring the network traffic to detect the ransomware activity before encryption can fail for several ransomware. Some papers like [28] proposed a ransomware network alert algorithm to distinguish ransomware network communication from benign traffic. Their proposition cannot be valid for most ransomware, especially for recent ransomware that do not communicate with outside before encryption. Indeed, the ransomware can be classified using their network activities into three classes:

1. Ransomware that contact their C&C before encryption. In this case, we can detect these ransomware before encryption if they generate suspicious network activities (e.g. contacting some blacklisted addresses). Some approaches were proposed in this way. For example, Cabaj et al. [29] proposed a software-defined networking (SDN) to detect the ransomware. Their solution is based on the network communication of CryptoWall and Locky ransomware families. In the same way, we found that Cerber
ransomware sends multiple User Datagram Protocol (UDP) requests containing encrypted data (25 bytes) before encryption. In order to hide the address of its C&C, Cerber sends these requests to more than 550 hosts in four different countries. Table 4 shows the UDP destination addresses and their owners. Cerber can be detected using its network activities. Indeed, it keeps the same destination port and the same size of data sent. Moreover, the same data were sent to multiple destinations in a short time.

Other ransomware communicates with their C&C before encryption, for example, some versions of GandCrab, TeslaCrypt, and PrincessLocker [30]. The first versions of GandCrab do not encrypt the files if they do not receive a response from the C&C. TeslaCrypt communicates with its C&C before encryption, but it does not encrypt the files despite no response from its C&C. The three ransomware can be detected before encryption using some Intrusion Detection System rules. For example, we suggest for the evolution version of PrincessLocker, the signature-based Snort rule in Figure 5 to alert the administrator for any machine being infected by this ransomware. Indeed, this rule catches any UDP request to the port 6901 of an external host and has the form of the regular expression mentioned in the query pcre.

2. To submit the encryption parameters, some ransomware contact their C&C during encrypting the files (Striked ransomware) or at the end of encryption. Most ransomware choose the last option to communicate with their C&C. For example, Nemty ransomware installs TOR browser and sends the configuration data to its C&C using the TOR proxy at the end of its infection.

3. Other ransomware need some user actions to communicate with the C&C. All Dharma versions, Blind, BTCWare, and LockerGoga ransomware suggest the victims to contact the owners of these ransomware by email to recover their files. Generally, the used emails are different in each version. Another example was seen in Spora ransomware that sends the encryption parameters after the victim click on the button Authorization in the HTML ransom note file.

Figure 6 shows an example of submitted data by Spora (encrypted and encoded base64) to its C&C.

Some ransomware verifies before encryption if the target machine is connecting to outside. This behaviour was seen in CryptoLocker, Kraken, and two versions of PrincessLocker ransomware. Table 5 shows the contacted domain names by these ransomware.

PrincessLocker did not encrypt the files if it did not receive a response from myexternalip.com. On the other hand, some variants of WannaCry did not encrypt the files if they had a response from a contacted domain. WannaCry was stopped accidentally after registering the contacted domain iuserfsod-p9fjaposdfjghosurijaerwrgwea.com by a security researcher known by the name @MalwareTechBlog. Table 5 proves that the identification of the external IP address of a target machine or a target network can be seen in some cases as a suspicious activity. This remark was generalised by Yaneza [31] on all malware. Indeed, he found many malware families used some online services like icahnzip, myexternalip, myipaddress, ipinfo, and others to map and identify their targets. Moreover, his tests show that the malware that uses the most multiple external IP look-ups is the ransomware. As he said, there is no suspicious intent to determine the external IP address of a machine or a network. But, it may be correct to monitor these multiple look-ups. In our context of ransomware detection, we suggest that monitoring the look-ups (recurrent or not) can be combined with other monitored ransomware behaviours or used as a secondary behaviour to detect the ransomware. Finally, we found that any ransomware detection approach based only on network activities can have an effect on a few number of ransomware, but it cannot be used to detect most ransomware.

Like Windows malware/ransomware, the use of the generated network traffic by Android malware was proposed in some papers like [32,33] to detect these applications. The first paper proposes to cluster Android malware using the generated HTTP network traffic to create signatures for detection. The second paper designs a machine learning approach to identify malicious Android applications by analysing their network activities. Android ransomware keeps the same characteristics discussed above about the network traffic of Windows ransomware. Some Android ransomware like Android/Filecoder.C do not communicate to outside before encryption. On the other hand, Trend Micro4 mentioned that some Android crypto-ransomware use HTTP/S, XMPP or TOR to communicate with their C&C server before encryption. Therefore, any solution based only on monitoring the network activity of these applications can fail to detect these ransomware. The behaviour of using the network activity can be used jointly with other behaviours or features to detect Android ransomware. Especially, when the Android application needs network permissions. Indeed, INTERNET permission is

---

**TABLE 4** Cerber UDP destination addresses

| Destination | Port | Owner          |
|-------------|------|----------------|
| 1.11.32.0-32 | 6892 | South Korea    |
| 55.15.15.0-31| 6892 | United States  |
| 194.165.16.0-254 | 6892 | Russian Federation |
| 194.165.17.0-254 | 6892 | Russian Federation |

---

**FIGURE 5** Alert Snort rule for First POST request of PrincessEvolution

---

3https://twitter.com/MalwareTechBlog/status/863187104716685312

4https://blog.trendmicro.com/trendlabs-security-intelligence/androidmobile-ransomware-evolution/
widely requested in both Android ransomware and other Android malware [34].

Looking for other Android ransomware behaviours, we found that Android/FILECODER.c ransomware links itself via SMS message to all contacts of the victim before encryption. This behaviour is interesting to detect this ransomware and other ransomware that distribute themselves using this method before their damage on the Android device. Especially, if they request SEND_SMS permission, which is in the top 20 requested permissions by Android ransomware [34].

2.4 Evasion techniques

Like malware, ransomware authors have developed various anti-detection or anti-analysis techniques. During our analysis on ransomware, we found that ransomware authors use sometimes anti-detection, anti-virtual machine, anti-debugging, and/or anti-disassembly techniques to delay or prevent analysis and detection of the ransomware. We will not cover the two last evasion techniques because they are more related to anti-reverse engineering techniques.

Some analysed ransomware attempt to detect if they are running inside a virtual machine to act differently. Figure 7 shows the searched processes by Kraken ransomware to detect if it is running inside a VirtualBox, VMware, Parallels Desktop, or other virtualisation applications. Other ransomware like Alpha ransomware verify in addition to the existence of some registry keys to check the existence of visualization software.

Other ransomware disables or kills some detection or monitoring processes on the target machine. For example:

- CryptoLocker ransomware attempts to disable Windows Defender using the windows registry value DisableAntiSpyware in HKLM\SOFTWARE\Policies\Microsoft\Windows Defender. Moreover, it terminates the running process of Process Explorer.
- TeslaCrypt ransomware creates a new thread to terminate these processes: TaskMgr.exe, ProcExp.exe, cmd.exe, msconfig.exe, and regedit.exe. This ransomware uses EnumProcesses, OpenProcess, and GetProcAddressFile, then TerminateProcess to stop these processes.
- Alpha ransomware lists all running process searching to terminate TaskMgr.exe process. Moreover, it adds the registry value DisableTaskMrg in HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System to disable Windows Task Manager.
- Kraken ransomware terminates the process Appcheck.exe.
- AppCheck is a ransomware detection tool of CheckMl company. It also checks the opened windows on the machine searching for window titles like Wireshark, Ethernet, NETSTAT, SysAnalyzer, and others.
- LockerGoga tries to kill several Antivirus products and disable some security tools using Taskkill commands. Among the killed services, we found some services from Sophos, Symantec, Acronis, Avast, and others.

In the same way, several ransomware includes in their code a list of benign programs to kill them in order to unlock the files that are locked by these programs. This behaviour was seen in Kraken, Nemy, WannaCry, Scarab, LockerGoga, and Dhara. The last in its cmd version tries to terminate six processes: 1c77.exe, 1c8.exe, postgres.exe, outlook.exe, mysqld-nt.exe, and sqlserver.exe. The last process is the most targeted by the ransomware. Dhara uses CreateToolhelp32Snapshot to create a snapshot of the running processes, and Process32First and Process32Next to browse the processes. Then, TerminateProcess to terminate the target processes. The same behaviour was performed by Android ransomware. Compared to other Android malware [34] or to benign Android programs [35], several Android ransomware request GET_TASKS and KILL_BACKGROUND_PROCESSES permissions. The first is used to get information about running processes, and the second could be used to kill the running processes on the device, including anti-virus processes. This suspicious behaviour that kills several benign processes is performed by the ransomware before encryption. We think that the behaviour of killing several trusted programs or enumerating the running processes must alert the ransomware detector for a detailed analysis to spot the following suspicious behaviours like a call to the API Crypto.

For the above-mentioned reasons, we modified the locations of the Python directory and the used scripts to run and analyse the ransomware in the new version of MoM. Moreover, we changed the name of the Python process to avoid killing it by ransomware. Some ransomware like Crypt need some user interactions to start their infections on the machine. For this reason, we simulated some user interactions during the automatic analysis. Despite these changes, some ransomware

| Ransomware   | Domain name       | Details                                      |
|--------------|-------------------|----------------------------------------------|
| CryptoLocker | ipinfo.asio       | It encrypts the files even though             |
| Kraken       | ipinfo.asio/json  | It does not receive a response.              |
| PrincessLockr| myexternalip.com  | No response, then no encryption              |

FIGURE 6 Submitted data by Spora ransomware
avoided running on our platform. For example, the version seen in June 2019 of SadComputer. This ransomware showed an alert message with the No button as a default choice. Clicking automatically on the No button ended the execution of this ransomware without encryption.

3 Encryption Process

We can summarise the ransomware encryption process by the following steps. The ransomware browses through directories to find relevant files (e.g. searching target extensions) and encrypts them. Then, it adds to the encrypted files an extension different to its original extension. In this part, we discuss the behaviour of searching and encrypting the target files, then we present some remarks on the encrypted files.

3.1 Target files

The ransomware starts searching the target files by checking the available disk drives on the target machine. Some of them infect all the available disk drives by alphabetical order, but others infect only the root disk drive (C:) or only some specified directories. Among the 225 automatically analysed ransomware, we found 193 working ransomware: 153 ransomware target the available drives on the target machine, 23 ransomware target the Desktop directory of the current user, the remaining target other locations like Documents or Pictures of the current user. Table 6 shows the first target location by the automatically analysed ransomware.

The used API functions to check the available disk drives are as follows:

- GetLogicalDrives to retrieve a bitmask representing the available disk drives on the target machine.
- GetDriveTypeW to determine the type of disk drive (removable, fixed, CD-ROM, RAM disk, or network drive). The most targeted drives are the fixed drives followed by the removable and the network drives. Table 7 shows the infected drives by some manually analysed ransomware.
- GetDriveFreeSpaceW to retrieve information about the target drive, including the free space of the drive.

This combination of API functions is performed by several ransomware. We cannot suppose this behaviour as a malicious behaviour to detect the ransomware. But, it is more interesting to explore this behaviour, looking for differences between ransomware and benign programs.

![Image](Image)

**FIGURE 7** The searched virtualisation processes by Kraken ransomware

| First target location | Number of ransomware |
|-----------------------|----------------------|
| Available drives      | 153                  |
| Desktop of the current user | 23              |
| Documents of the current user | 5            |
| User directory of the current user | 1          |
| Users directory      | 2                    |
| Pictures of the current user | 2           |
| Links of the current user | 2           |
| Musics of the current user | 2           |
| Unknown locations    | 3                    |

**TABLE 6** First target locations by the automatically analysed ransomware

| Ransomware                  | Targeted drives                  |
|-----------------------------|----------------------------------|
| PrincessLocker (Evolution version) | Network and fixed drives         |
| Locky (Lukitus version)     | Fixed and removable drives       |
| Blind (Napoleon version)    | Network and fixed drives         |
| Diamond and Dharma          | All available drives              |

Cerber ransomware in its Lukitus version targets only the fixed and the remote drives. It calls GetLogicalDrives followed by GetDriveTypeW. Then, it creates a thread for each drive to enumerate all its content searching for target files. Enumerate all target files, then encrypt them is a known ransomware behaviour. We discovered this behaviour in the first version of PrincessLocker that enumerates all the target files on the machine and stores them in its memory before encryption. After the first post to its C&C, PrincessLocker encrypts the stored files in its memory by order: the first enumerated file is the first encrypted. According to this behaviour, the ransomware can enumerate all the target files, then encrypt them randomly to avoid encrypting the decoy files first. Ransomware detection using decoy files was proposed by Jeonghwan et al. [36]. This method is based to put some monitored files (decoy files) in some locations to be firstly encrypted related to the ransomware directories traversal order. Decoy files are discussed in the fifth section.

Using several threads to search and encrypt the files is another behaviour performed by the ransomware. Indeed, the automatic analysis on the 193 working ransomware shows that at least 63 ransomware use several threads to encrypt the files. The interest of this behaviour is to ensure faster and more
damage before victims discover they are under ransomware attack. The ransomware detector must be aware of the multi-threading ransomware. Especially, the ones that use several threads for reading data from a target file, writing encrypted data, and renaming/deleting this target file.

File system traversal is a malicious behaviour performed by most ransomware. Indeed, the ransomware targets the files for encryption by depth-first or breadth-first search using FindFirstFile and FindNextFile APIs. This behaviour was discussed by Routa et al. [37]. They found that the per-thread file system traversal is sufficient to highlight the malicious ransomware activity. Some ransomware targets some specified files using some API calls like SHGetSpecialFolderPathW that retrieve the path of a special directory. Among the manually analysed ransomware that targets some specified directories (only these directories or in addition to the root directory), we found Alpha, Diamond, Striked, and GandCrab. Therefore, we think that monitoring file system traversal is not sufficient to detect the ransomware. It can be added as another monitored behaviour to detect the ransomware. Especially, in the case of ransomware that enumerate all target files on the machine before encryption.

The ransomware avoids encrypting some directories or some files on the target machine. In addition to a target extensions list, most ransomware have an avoided directories/files list to avoid them. Generally, these directories/files are avoided to ensure the normal running of the target machine. Table 8 shows some avoided directories/files by some ransomware.

The most avoided directory by ransomware is Windows directory. We can suggest to hide some valuable data in this directory to not be encrypted by ransomware, but not all ransomware. Indeed, during our automatic analysis, we found that OnyxLocker ransomware encrypted our hidden files in Windows directory. However, this suggestion needs more investigation to answer this question: can we hide our files in the Windows directory to not be encrypted by ransomware?

The same behaviour was observed on Android ransomware. Indeed, most Android ransomware avoid encrypting system files of the target device. Others focus only on some specified files like the Download files and Pictures. Lucy Android ransomware first tries to retrieve all directories on the device. If it fails, it searches for/storage directory. If this search also fails, Lucy looks for the/sdcard directory. Android/Filecoder.C excludes files in directories that their names contain tmp, temp or cache. Identifying the common target directories and the common avoided directories by Android ransomware needs more investigations. The first can be used to put decoy files to detect these applications and the second can be used to hide valuable data on the device.

Some Windows ransomware do not use the full path of the avoided directories to not encrypt their content. Indeed, they compare only the name of each directory in the machine to the avoided directories. In this case, they do not encrypt the files in the directories that have the same names of the avoided directories. For example, PrincessLocker (the evolution version) does not encrypt the content of any directory named Windows. In contrary, Dharma (emb version) ignores the Windows directory in the root path (C:) and encrypts the content of any directory named Windows in other directories.

### 3.2 Encryption process

After searching the target files, the ransomware starts encrypting these files using some libraries like API Crypto, Crypto++, and .NET Crypto. Other ransomware like TeslaCrypt uses unknown libraries to encrypt the files. Lemmou et al. [38] mentioned that the developers of GandCrab used the AES256 algorithm from TrueCrypt.\(^5\)

Ransomware file encryption is divided into three steps: read data, encrypt this data, then write the encrypted data. Using some tools like Process Monitor to monitor the read/write operations of the manually analysed ransomware, we found three classes of ransomware:

1. Ransomware that tries to read the content of the target files by one read (usually using ReadFile API). This class tries to reduce the number of the generated read/write operations.

\(^5\)https://github.com/AuditProject/truecrypt-verified-mirrors/.
This behaviour was performed by TeslaCrypt, CryptoLocker, and other ransomware.

2. Ransomware that reads the content of parts (read a block of data, encrypt it, then write it etc.). An example was seen in the evolution version of PrincessLocker that reads the target file, then creates a new file containing the encrypted data. The size of the used buffers in the read/write operations is 128 bytes.

3. Ransomware that reads a fixed length from the beginning/end of the target file. Then, they read/write all the content using the first or the second item. This behaviour was seen in Spora ransomware [39]. Indeed, this ransomware starts the encryption of any target file using at least two ReadFile (with two fixed buffer sizes) from the end of this file. Two ReadFile are used by Spora to check if the target file was encrypted or not.

On the other hand, most ransomware destruct the original files by one of the following techniques:6

- They write the encrypted data into the original file, then rename it (optional).
- They save the encrypted data into a new file and delete or overwrite (move) the original file. In this case, the ransomware can use two streams to read and write data. This behaviour was also performed by Android ransomware. Figure 8 shows the case of Android/Filecoder.C ransomware that saves the encrypted data into a new file (using the extension.enc) and deletes the original file.
- They move the original file in other location (e.g. %temp%). They write the encrypted data, then they move the file back to its original location.

The generated read/write operations or the file system requests were proposed in some solutions like ShieldFI's [10] to detect the ransomware. As mentioned in the introduction, this solution is based on the difference between I/O file system requests of ransomware and benign applications. In our case, most of the analysed ransomware generates many similar file system requests different to the generated file system requests of benign applications. Table 9 shows some examples of these file system requests.

Some ransomware delete the target files after encryption. In this case, the deleted files can be recovered using any files recovery solution. Other ransomware rename the target/encrypted file. Moreover, some ransomware like TeslaCrypt create the files that receive the encrypted data with the added extension before encryption. Spora and other ransomware generate similar WriteFile requests with the same offset/length for each target file. These requests are used to store some information like the representation of the used key to encrypt the file. Generally, the similar requests of ReadFile/WriteFile with the same offset/length, rename/delete the target files, or create new files with suspicious or unknown names/extensions can be used to detect the ransomware or limit its damage to few encrypted files.

On the other hand, we can summarise the data encryption of a target file using the Microsoft API Crypto by the following scenario:

1. The ransomware calls GetFileAttributesW and SetFileAttributesW to retrieve the file system attributes and set the file to removal or backup file. By these calls, the ransomware can avoid any access to the file by other running applications.
2. The ransomware generates some random values (e.g. using the API function CryptGenRandom) to create the used key for data encryption or/and the other encryption parameters like the initialisation vector. These values are encrypted by an embedded RSA public key inside the binary or received from the C&C. The encrypted key/

---

6Recently (November 2019), a new technique called RIPlace was disclosed by some security researchers of Nyotron company: https://www.nyotron.com/collateral/RIPlace-report_compressed3.pdf.
parameters are added to the encrypted content of the target file after encryption.

3. The ransomware calls ReadFile to read data, then it encrypts this data using the API function CryptEncrypt. Finally, the encrypted data are added to the same file or to a new created file.

As mentioned in some papers like [12], monitoring the API crypto can be used to detect the ransomware or used to retrieve the used encryption keys as proposed in [13]. From our point of view, we think that monitoring the AP Crypto functions, especially CryptEncrypt, must be combined with other malicious behaviours. For example, a ransomware detector can predict that the behaviour of terminating multiple benign applications followed by a call to some API Crypto functions is a ransomware activity.

### 3.3 Encrypted files

The encrypted content of a target file is different from its original content. Moreover, an encrypted file can have a different name, extension and data structure from its original form. In this part, we present some differences between the encrypted files and the original files before encryption. These differences can help the ransomware detector or the identifier to make differences between an encrypted file and an unencrypted file.

The first difference is the difference between the read data and the written encrypted data. This behaviour is used by some tools like CryptoDrop [11] and DaD [3]. The first uses Shanon entropy that provides high entropy on the encrypted and compressed data. The second uses chi-square ($\chi^2$) that measures how closely a set of numbers follows a particular distribution. The entropy was proposed in [40] and used in the real-time detection system RansomProber [34] to detect Android ransomware. As mentioned in the introduction, the entropy cannot distinguish JPEG compression. For this reason, we propose the use of the $\chi^2$ instead of entropy to increase the effectiveness of these works.

On the other hand, many ransomware add to the encrypted files an extension to make them different from the other files. Some ransomware try to alert the user that his files were encrypted by adding the email address of the attacker, the victim identity, and other messages. As shown in Table 10, the ransomware can add to the name of the encrypted files a random generated extension, known extension .mp3 that was used by TeslaCrypt in its version 3.1. Others add some combinations of the attacker email address, the victim identity, and other information. In contrary to these ransomware, some ransomware like Spora do not add any extension to the encrypted files.

The behaviour adding a new extension to each target file is given in Figure 9. Indeed, the ransomware reads multiple file types, but it writes only a single type. This behaviour was mentioned in [11] by the name file type funnelling. It is a behaviour that occurs when a running application reads several files of different types as it writes. This behaviour is also seen in benign applications like the compression applications that embed several file types (documents, pictures and videos) but will only write a single file type if the compressed file. This is the reason that CryptoDrop [11] includes this behaviour as a secondary indicator to detect the ransomware. As seen in Figure 8, the same behaviour was seen in Android ransomware which make the idea of using file type funnelling jointly with other behaviours valid to detect Android ransomware.

Other ransomware add a new extension to the encrypted files and modify/encode/encrypt the names of these files. This behaviour was seen in Kraken ransomware. Indeed, this ransomware renames the encrypted files in the form 00000001-Lock.onion. The number $i$ in the name is incremented for each encrypted target file. The original name is encrypted and added to the encrypted content of the file. Generally, this behaviour is another ransomware behaviour in which the ransomware reads several files of different names, but will only writes a single file type with a specified name format. In the case of Kraken, the detector must be aware of any process reads, for example, three files of different names (the same type or different) in a short time, but writes only three files in this form (especially, the first three written files $i \in \{1, 2, 3\}$):

$$00000001i - \text{Lock.onion} \quad 0 < i$$

The same idea can be used on the ransomware that encode the names using some algorithms like base64. Moreover, we can check the similarity of the names of these files after and before encryption to catch the activity of some ransomware. This idea cannot be generalised on all ransomware. Indeed, as mentioned in Table 10, there are some ransomware that do not change the name or the extension of the encrypted files.

The encrypted files by some ransomware have a well-defined structure. This structure depends on the hidden data by the ransomware inside these files. For example, the file markers that identify the ransomware family/version and other information about the infection like the initialization.
vector and/or the fingerprint of the used key to encrypt the file. As shown in Figure 10, GandCrab ransomware in its version 5 adds 16 numbers at the end of each encrypted file. These numbers are checked by this ransomware in its next infections (overinfection). Indeed, it does not encrypt any file containing these numbers. The behaviour of laying the encrypted files with the same internal structure can be added as another monitored behaviour to detect the ransomware. The same for Android ransomware. This behaviour is like the two previous discussed behaviours (file type funnelling and files renaming). Indeed, it transforms all encrypted files from different internal structures to a single structure using the same file markers.

On the other hand, most ransomware (Android and Windows) and especially the recent ransomware generate for each target file a new key and/or a new initialization vector to encrypt the file. This remark can be used, for example, by the sandboxes to identify and differentiate the ransomware activity from other benign activities like the compression activity. Indeed, two similar files encrypted by the same ransomware will have two different content. In contrary, two similar files will have the same compressed contents.

The size of the encrypted files can be added as another behaviour to detect the ransomware. Indeed, we checked automatically the size of the encrypted files of the 193 working ransomware. We found three cases:

1. Encrypted files have a size multiple of 16 bytes added to a value (cte).

\[
\text{Size}_{\text{encrypted}} \mod 16 = \text{cte}
\]

The encrypted files by some ransomware have a size multiple of 16 bytes (\(\text{cte} = 0\)).

2. Encrypted files have a size that matches to the sum of their size before encryption and a fixed value.

\[
\text{Size}_{\text{encrypted}} = \text{Size}_{\text{original}} + \text{cte}
\]

In this case, the encrypted files can have the same size before encryption (\(\text{cte} = 0\)).

3. Other cases do not match the previous two items.

Table 11 shows the number of ransomware and some examples for each case. The ransomware detector can monitor the size of the created files on the machine. It must be aware if a process creates in a short time three or four successive files that have a size matches to one of two first items.

4 | OTHER RANSOMWARE FEATURES

4.1 | Ransom note files

After encrypting the files, the ransomware shows a ransom note to its victim to inform him that his files were encrypted. These files instruct the victim on how to pay the ransom and recover the files. Table 12 shows the used ransom files by three manually analysed ransomware.

Generally, the ransomware makes a predefined algorithm to put its ransom notes. For example, the lukitus version of Locky ransomware puts the second ransom note (Table 12) in any target directory containing target files. Then, it puts the first ransom note in the Desktop directory of the current user. Dharma adds the txt ransom note in the root of each drive. Then, it adds the hta file in the Desktop of the current user and the Startup directory.

Viewing the used algorithms to put the ransom files by the manually analysed ransomware, we can define three classes of ransomware:

- The first class is the class of ransomware that put their ransom files before encryption. Some ransomware of this class add the ransom files in the Desktop directory of the current user before the other directories. Moreover, using the automatic analysis on the 193 working ransomware, we found at least 59 ransomware make their ransom files before encryption.
- The second class is the class of ransomware that put their ransom files during encryption.
- The last class is the class of ransomware that put their ransom files at the end of encryption.

We suggest two new methods using the ransom note files to detect the ransomware. These two methods can be used jointly with other ransomware behaviours to increase the effectiveness of the ransomware detection tools. Moreover, we think that our suggestion can detect the ransomware of the first class without
any encryption and limit the encryption damage of the second class. Our suggestion is based on monitoring the names and the content of the ransom files. Indeed, the names and the content of the ransom files are noteworthy compared to the name and the content of benign files. Table 13 presents the most used terms in the content of some ransom files.

There are other ransomware behaviours related to ransom files. The behaviour of adding the same files in several directories in a short time can be used jointly with the other behaviours to detect the ransomware. Especially, if the ransom files have the same name. This behaviour was seen for example in PrincessLocker that adds three files with the same name (html, txt, and url) in any target directory. TeslaCrypt also makes three files with the same name in the target directories. Other ransomware like Striked and Blind make only one file in the target directories. To make the ransom files in the target directories, the ransomware generates many similar operations that can be used to detect it:

- Most ransomware check if the ransom file is already in the target directory by a call to CreateFile containing the name of the ransom file. Let suppose that the ransomware puts more than one ransom file. In this case, it checks the three files in each target directory which means three similar CreateFile for each target directory.
- Other ransomware like Bulba and Protected (two ransomware seen on July 2019) have a weak implementation to generate the ransom files. Indeed, they try to open their ransom files by CreateFile before encrypting each target file.
- To write the same content (the same size) of data in the ransom files, the ransomware generates similar WriteFiles with the same data and buffer size. The used terms in the ransom files are limited to few terms like your, file, encryption, and others. These terms can be used to detect any activity related to the ransom notes.

Some recent ransomware like GandCrb, Sodinokibi, WannaCry, CryptoLocker, Alpha, and Cerber replace the Desktop wallpaper, mostly with a scary image containing a ransom message. Generally, this behaviour is performed by setting the registry key HKCU\Control Panel\Desktop\Wallpaper. We think that monitoring this registry key or the desktop wallpaper change can be used with the other behaviours as a secondary indicator to detect the ransomware. Especially, the case of ransomware that change the wallpaper before encryption.

Like Windows ransomware, Android ransomware notify victims that their files were encrypted and request a ransom to pay. This notification is generally done using text messages [21]. Therefore, performing a linguistic analysis like Latent Semantic Analysis (LSA) on the content of these messages can reveal another behaviour-based extortion. Our current research on LSA applied on the content and the names of the ransom files (Windows ransomware) shows encouraging results to discriminate between user files and ransom files. Moreover, several Android ransomware show the victims messages and images related to pornographic material and police activities [21]. Therefore, the LSA similarity between these messages is high compared to the content of other files. We present an overview to detect Android Ransomware using LSA in Section 5.
4.2 | Logging in file and shadow copies deletion

Some ransomware like CryptoLocker log the full path of the encrypted files in a specified files. Figure 11 shows how CryptoLocker uses an html file named log.html in %Appdata% directory to log the encrypted files. This file is used by its hta ransom file to show to the victim the list of its encrypted files. The behaviour of logging in a file, checking another file, and loading the same DLLs before the encryption of each target file are suspicious behaviours that can be used to detect the ransomware. As shown in Figure 11, the behaviour of logging the encrypted files generates several file system operations that can be added to the other operations of reading data and writing the encrypted content. For example, CryptoLocker is a ransomware that adds its ransom notes before encryption. It writes the encrypted data into the original file and renames it. Then, it logs this file in the html file.

Several ransomware uses the windows command vssadmin to delete shadow copies. Shadow copies create backup/snap-shots of computer files/volumes. Using the command vssadmin delete shadows/all/quiet, the ransomware deletes all available shadow copies on the target machine. Other commands are used jointly with the previous command. For example, whadmin DELETE SYSTEMSTATEBACKUP to delete system state backups, wmic SHADOWCOPY DELETE to delete shadow copies or using the bededit/set {default} recovery enabled No command to disable automatic repair. Several ransomware use these commands at the beginning of infection before the files encryption, for example, the case of Blind, Scarab, and BTCWare ransomware. The last ransomware calls the vssadmin command to delete the shadow copies before and after encryption. Other ransomware like Spora ransomware delete the shadow copies after encryption, which makes monitoring only this behaviour useless, but it can be used jointly with the other behaviours.

5 | RANSOMWARE DETECTION AND PREVENTION

In this section, we present some ransomware detection tools and our tests using these tools on some analysed ransomware. We present an overview on the new version of MoM that we used to replace the first version. We discuss the killing switch, the feature that stops a specified ransomware on a target machine. Finally, we present an overview to detect Android ransomware using LSA.

5.1 | Ransomware detectors

The ransomware detection or prevention is the subject of several academic studies and anti-virus companies. As mentioned in the introduction, there are many academic projects on ransomware detection or prevention, but only few projects are available for downloads and tests. Moreover, the anti-virus companies make some solutions to detect only the ransomware as an added solution or an extension to their known solutions. Table 14 presents some known solutions to detect the ransomware. The techniques used to detect the ransomware can be divided into three techniques: behavioural, decoy file, and signature detection. The last is used jointly with the behavioural analysis in Kaspersky Anti-Ransomware. We present only some chosen tests because the subject of this part is not to find the best tool to detect the ransomware but to present some issues of the evaluated versions of these tools.

- CryptoDrop [11] is an academic tool to detect the ransomware. As shown in Figure 12, this tool monitors only some specified locations like the users directories. We tested this tool on Kraken, Dharma, Blind, and Spora ransomware. It detects these ransomware after encrypting only few files in the monitored directories. The encrypted files were recovered correctly: Kraken (one encrypted file), Dharma (four encrypted files), Spora (one encrypted file), and Blind (eight encrypted files).

- CyberReasonRansomFree is a tool of CyberReason company. This tool creates two directories (decoy directories) to be first targeted by the ransomware that uses the normal order or the inverse order to encrypt the files. The chosen location by this tool are the root drive (C\), Users directory, and Documents directory of the current user. This tool tries to monitor the modifications (e.g. write/deletion) in the content of the created directories. These decoy directories contain 10 files
Some ransomware detection tools

| Detector              | Used version | Used technique           |
|-----------------------|--------------|--------------------------|
| CryptoDrop            | 1.5.353      | Behavioural detection    |
| CyberReasonRansomFree | 2.4.2.0      | Decoy files              |
| Kaspersky Anti-Ransomware | 3.0.0.738 | KSN and system watcher   |
| DaD                   | -            | Behavioural detection    |
| Padvish               | 1.5.108.619  | Decoy files              |
| AntiRansomV3          | -            | Decoy files              |
| MalwareBytes Anti-ransomware | 3.0.0   | Behavioural detection    |
| Acronis Ransomware Protection | 1700 | Behavioural detection    |
| RansomBuster          | 12.0         | No decoy files seen      |
| Hitman Pro Alert      | 3.7.3        | Behavioural detection    |

Abbreviations: DaD, Data Aware Defense; KSN, Kaspersky Security Network.

Some monitored directories by CryptoDrop

- General
  - DropSafe
  - Rules

- Protect Location:
  - $Recycle.Bin
  - Desktop

- Enable Self-Protect
- Alert on bulk file modification

with different names and extensions: sql, ooxml, xls, mdb, rtf, txt, pem, xlsx, doc, and jpeg. We tested this tool on Dharma, Kraken, Blind, TeslaCrypt, Spora, PrincessLocker, and CryptoLocker. This tool was able to detect the ransomware after encrypting some of its files and those directories and other files in other locations. For example, the files in $Recycle.Bin and Desktop of the current user.

- DaD [3] is an academic tool to detect the ransomware. We tested the first version of DaD on Dharma, Kraken, GandCrab, and PrincessEvolution. This tool does not use decoy files and monitors all files on the machine. Contrary to CryptoDrop that monitors the files within some specified directories. DaD cannot detect Dharma ransomware. It detects Kraken after encrypting 130 files. GandCrab ransomware was detected after encrypting 30 files in $Recycle. Bin directory. Two files were encrypted before the detection of PrincessEvolution ransomware.

Recently, we were invited to test the forthcoming version of DaD. We chose 60 arbitrary ransomware including some known families like GandCrab, StopDjvu, GlobImposter, and Dharma families. This version was able to detect all these ransomware without any problem except six ransomware. Moreover, by adding new monitored behaviours (ransom notes with and size of files), this tool was able to detect all evaluated ransomware. All results of these experience will be published in our future papers.

- Kaspersky anti-ransomware uses two techniques to detect the ransomware. KSN\(^6\) for signatures scan and System Watcher for a behavioural detection. We found some differences between using this tool offline and with an authorised connection to outside. We tested this tool on GandCrab ransomware using the offline mode. It detects this ransomware after encrypting 25 files. On the other hand, it was able to detect this ransomware without encrypting any file using the online mode.

- Acronis Ransomware Protection is a ransomware detection tool of Acronis company. We tested this tool on Dharma, Kraken, PrincessEvolution, and GandCrab. This tool was able to detect all these ransomware with some issues on recovering the encrypted files. For example, only 11 files were correctly recovered among the 27 encrypted files by Kraken ransomware.

Generally, the known issues of the ransomware detection tools can be divided into four general issues:

\(^{6}\)Kaspersky Security Network.
The detection of ransomware. Indeed, some tools have some internal issues to detect some specified ransomware.

Maybe the monitored behaviours in these tools do not cover all major behaviours of the ransomware, or the solution is poorly implemented to detect some ransomware. In our evaluations, several ransomware were not detected by many ransomware detection tools.

Some tools monitor only the malicious behaviours in some specified directories. We think that the damage of ransomware cannot be limited only to some chosen directories by the user. What about the files, configurations, and backups of the installed programs on the machine?

Generally, using decoy files to detect/identify the ransomware is not a bad idea. Moreover, it can be applied to detect/identify Android ransomware. Especially, inside directories that contain valuable information like personal account/contact information (e.g., /data/system/accounts.db), databases like the SMS database or other directories like /sdcard/Pictures, /sdcard/Downloads. We think that using decoy files to detect the ransomware on Android devices can have more effect than using it in Windows machines. To ensure maximum efficiency of decoy files, it is necessary (on Windows and Android environments):

1. At least one extension of the used decoy files is targeted by all ransomware. The same remark is valid for Android devices. Indeed, some ransomware avoid encrypting some extensions like Android/Filecoder.C. Android ransomware that excludes zip and rar files.
2. Some ransomware check the size of target files before encryption. Any file has a size less/greater than a specified size is avoided. This behaviour was seen in Android/SLocker or Android/Filecoder.C that avoids large files (over 50 MB) and small files (under 150 KB). Whether on Windows or Android, the decoy files must be generated with different sizes.
3. The used decoy files will be the firsts encrypted before the user files. This can be impossible in the case of ransomware that browse the target files, then encrypt them randomly.
4. The number of used decoy files is sufficient to detect and stop all running processes of the ransomware without any user file encryption.
5. The decoy files are generated differently for each machine (Windows or Android) restart to avoid their detection by the ransomware. In this case, the detector must avoid the randomly generated names/content of these decoy files.
6. The decoy files must be generated from the user files to not alert any attacker trying to execute the ransomware manually. It can use some attractive names like Password, backup, keys, wallet, secret, confidential, pictures and other names for these decoy files/folders.

Recover the encrypted files. Despite that some tools detect the ransomware, there are some issues to recover all encrypted files before detection.

5.2 Decoy files

MoM is a bare-metal ransomware analysis platform [12]. This platform is made of a single master server and several slave machines. The whole system is on a dedicated network connected to outside. MoM was made for two modes of experiments: the first downloads the ransomware samples and executes them to determine the active samples. The second evaluates DaD to detect the active samples marked in the first mode. The first version of MoM uses some decoy files in the current user directory (C:\users\user) to identify the active samples. The new version of MoM is more performed compared to the first version. Indeed, we added more decoy files in different locations to ensure that these files will be encrypted before the other files, regardless of the used order by the ransomware to encrypt the files (Windows-1252 or vice versa). The chosen locations to put decoy files/folders were: %homedrive%, %homepath%, user directory, Desktop, Pictures, Document of the current user, and any available drive different to %homedrive%. The objective of this part is to present how decoy files is effective to identify the active samples.

As discussed in the introduction, we collected until the date of writing this paper (the end of November) 217 different samples. MoM was able to identify correctly 182 working ransomware and 30 not working ransomware. MoM cannot identify automatically eight ransomware for some issues like the used time to evaluate the ransomware or the need to some specified mouse moves/clicks before running the sample. In summary, the new version of MoM ensures that all the working evaluated ransomware start with an event (move, delete, write) on decoy files/folders. Only five ransomware do not start their encryption on a decoy file. Table 15 presents these ransomware and the location of the first encrypted file.

As shown in Table 15, the ransomware can avoid encrypting decoy files or encrypt them after encrypting other user files. For example, AnteFrigus targets only the D:, E:, F:, G:, and H: drives. Avest ransomware has a weird encryption algorithm to encrypt the files. We conclude that using only decoy files to detect the ransomware fails to detect several ransomware before encrypting the user files. On the other hand, decoy files can be used jointly with the other discussed behaviour to detect the ransomware.

5.3 Kill switch and vaccine

A kill switch, also known as an emergency off, is a switch, button or any mechanism used to immediately turn off a device, system, or program. In the ransomware area, this term was mostly used after the WannaCry ransomware attack. As mentioned previously, this ransomware was stopped using a kill switch by registering the ransomware contacted domain by the security researcher @MalwareTechBlog. The ransomware vaccine is a kill switch on the target machine. This term is used by some ransomware researchers in the case if the ransomware
can be stopped on the current machine without encrypting the files.

GandCrab ransomware is the most known ransomware in 2018. During its activity, its authors tried in each version to improve the code of this ransomware (with some mistakes). These mistakes allowed us to make a vaccine to stop some versions (v4.x and v5.x) of this ransomware without encrypting the files. GandCrab has some issues in its code, especially in checking if the target machine was already infected. Indeed, GandCrab checks two registry values (named public and private) before encryption in the registry keys HKCU\Software \Wow6432Node\keys_data\data and HKCU\Software \Wow6432Node\keys_data\data. In the version v5.x, Gandcrab used the registry key _keys_data instead of keys_data. GandCrab does not encrypt files in the following cases:

- public registry value does not exist.
- public registry value exists and private registry value does not exist.
- public registry value exists and private registry value containing data greater than or equal to 2564 bytes.

In conclusion, the user can avoid encrypting its files by GandCrab (v4.x and v5.x) using, for example, an empty registry value named public in the previous registry keys. In the same way, the user can avoid encrypting its files by Kraken ransomware using an added registry value named WordLoad containing 1 in HKCU\Console. PrincessEvolution ransomware ends its execution and does not encrypt files if a file named McGEZan.VDE exists in %AppData% directory.

## 5.4 About android ransomware detection

Several works on the detection of Android ransomware mainly focused on permission, API calls, and static and dynamic analyses. RanDroid [41] is an example. It is an automated Android framework to detect ransomware using a linguistic and structural similarity analysis on the content of these applications. Checking similarities between Windows ransomware and benign applications is another way to detect ransomware on Windows. During our research on Windows ransomware, we applied LSA to check similarities between extracted dumps memory of several ransomware including the known families like GandCrab, Xorist, and Cerber and dumps memory of benign applications like Firefox, Winzip, and WinRAR. The results of this experience are promising to be published in our future work. This method was also used to check similarities between the content of user's files and the content of ransom note files that alert the user about its encrypted files.

LSA is an unsupervised analysis that attempts to solve the problem of comparing the meanings or the concepts behind the words by mapping both words and documents (in our case, the content of the APK files) into an algebraic space and doing the comparison in this space. Indeed, LSA performs an analysis of co-occurrence across the corpus of documents using singular value decomposition to reduce the dimensionality of this space. This reduction has the effect of preserving the strongest semantic information in the text and throws away the noise. Then, LSA identifies the related terms (e.g. using the cosine similarity) and put them under one topic without any prior knowledge of their real categories. This reference [42] presents more details about LSA.

Due to its success in the area of source-code plagiarism detection [43], spam filtering [44], and the notable results during our experiences, we suggest in this article the use of LSA to detect or identify android ransomware. We can apply LSA on the extracted permissions from the manifest (AndroidManifest.xml) that contains a list of all required permissions by the Android application. Such permissions like the previously discussed (BIND_DEVICE_ADMIN, GET_TASKS, KILL_BACKGROUND_PROCESSES, RECEIVE_BOOT_COMPLETED), the permissions that prevent the Android device from going to sleep like WAKE_LOCK, or the other permissions that allow the ransomware to read/write in the external device (READ_EXTERNAL_STORAGE, WRITE_EXTERNAL_STORAGE) are widely requested by Android ransomware compared to benign applications. Therefore, applying LSA between a set of extracted permissions from benign applications and a set of ransomware permissions will give low similarities between the two sets. These low similarities make the detection or identification of any new evaluated ransomware possible due to its high similarity with the permissions of the second set. Mainly, LSA is a method to extract similarities between documents. Therefore, it can be easily applied on the extracted text from the XML layout files and classes.dex. In the same way, the LSA method can be used to check similarities between extracted API methods from a given decompiled APK file and extracted API methods of benign programs/ransomware.

### TABLE 15 The unidentified ransomware by MoMv2 and the location of their first target file

| Sample       | Date     | Location of first encrypted file               |
|--------------|----------|-----------------------------------------------|
| AnteFrigus   | Nov. 2019| It targets the files in D:, E:, F:, G:, H:, and I: drives. |
| Unidentified | May 2019 | Links directory of the current user.          |
| Crown        | July 2019| Links directory of the current user.          |
| Avest        | Sep. 2019| Weird encryption. First encrypted file is not a decoy file. |
| Dodger       | June 2019| Music directory of the current user.          |
In this work, we tried to explore and discuss all relevant abnormal ransomware behaviours. Our research works on ransomware led to classify the ransomware behaviours into three categories:

- the pre-encryption behaviours that are generally performed by the ransomware before encryption. It can include the self-reproduction, managing the overinfection, process installation, persistent/evasion mechanisms, network activities, and others.
- the behaviours that can be monitored during files encryption, these behaviours can include the ones of file system exploration to find files to encrypt or the ones of the encryption process (read, write, delete, move, rename, extension change etc.).
- the post-encryption behaviours that are performed by the ransomware at the end of its infection. These behaviours can include showing the ransom notes and deleting the shadow copies of the target machine.

Several ransomware follow this classification to infect the target machine but, this classification is not respected by all ransomware. Indeed, some ransomware copy their code at the end of infection to ensure persistence. Others explore all target files or make their ransom note file before encryption and not during the files encryption.

The majority of the discussed behaviours in this article can be monitored using the file system activities, which make them easily used in the new or the available tools of ransomware detection. For this reason, to increase the effectiveness of DaD ransomware detection tool, the monitored behaviours of its next version will include other behaviours like the names or the content of the ransom note files and the size or the extension of encrypted files. Future work will investigate the robustness of this combination to detect the ransomware.

The discussed behaviours in this study are mainly focused on Windows crypto-ransomware. But, similar behaviours can be found on other platforms like Android as we mentioned in some parts of this article. Indeed, similar behaviours are performed by Android ransomware and other behaviours need more investigation to confirm them on this platform.

**ORCID**

Yassine Lemmou  
https://orcid.org/0000-0002-3182-3275

Jean-Louis Lanet  
https://orcid.org/0000-0002-4751-3941

El Mamoun Souidi  
https://orcid.org/0000-0003-1716-3204

**REFERENCES**

1. Abdurrahman, P., Tanikut, A.: Malware classification based on API calls and behaviour analysis. IET Inf. Secur. 12(2), 107–117 (2018)
2. Suleiman, Y.Y., Sakir, S., Igor, M.: High accuracy android malware detection using ensemble learning. IET Inf. Secur. 9(6), 313–320 (2015)
3. Palisse, A., et al.: Data aware Defense (DaD): towards a generic and practical ransomware countermeasure. In: Proceeding of 22nd Nordic Conference, NordSec 2017, pp. 192–208. Tartu (2017)
4. Philip, K., Sakir, S., Domhnall, C.: Evolution of ransomware. IET Nerw. 7(5), 321–327 (2018). https://doi.org/10.1049/iet-net.2017.0207
5. Young, A., Yung, M.: Cryptovirology: extortion-based security threats and countermeasures. In: Proceedings 1996 IEEE Symposium on Security and Privacy, pp. 129–140. (1996)
6. Gazet, A.: Comparative analysis of various ransomware variants. J Comput. Virol. 6(1), 77–90 (2010)
7. Young, A. L.: Building a crypto-virus using Microsoft’s Cryptographic API. In: Proceedings of Information Security, 8th International Conference, ISC 2005, vol. 389–401. Singapore (2005)
8. Kharraz, A., et al.: Cutting the Gordian knot: a look under the hood of ransomware attacks. In: Proceeding of 12th International Conference, DIMVA 2015, vol. 3, pp. 3–24. Milan (2015)
9. Amin, Kharraz, et al.: UNVEIL: a large-scale. In: Automated Approach to Detecting Ransomware. 25th USENIX Security Symposium, vol. 16. USENIX Security, Austin, TX (2016)
10. Continella, A., et al.: ShieldFS: a self-healing, ransomware-aware file system. In: Proceedings of the 32nd Annual Conference on Computer security Applications, ACSAC 2016, pp. 336–347. Los Angeles, CA (2016)
11. Scaife, N., et al.: CryptoLock (and Drop it): stopping ransomware attacks on user Data. In: Proceeding of International Conference on Distributed Computing systems, ICDCS, pp. 303–312. Nara (2016)
12. Palisse, A., et al.: Ransomware and the legacy Crypto API. In: Risks and Security of Internet and Systems – 11th International Conference, CRiSIS 2016, vol. 5–7. Roosevelt (2016)
13. Kolodenker, E., et al.: PayBreak: Defense against cryptographic ransomware. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017. Abu Dhabi (2017)
14. Zhang, B., et al.: Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes. Future Gener. Comput. Syst. 110, 708–720 (2020)
15. Lee, K., Lee, S., Yim, K.: Machine learning based file entropy analysis for ransomware detection in backup systems. IEEE Access. 7, 110205–110215 (2019)
16. Mbol, F., Robert, J.M., Sadighian, A.: An efficient approach to detect Torrent-locker ransomware in computer systems. In: International Conference on Cryptology and Network Security, pp. 532–541. Springer (2016)
17. Wang, W., et al.: Exploring permission-induced risk in android applications for malicious application Detection. IEEE Trans. Inforn. Forensics Secu. 9(11), 1869–1882 (2014)
18. Enck, W., et al.: A study of Android application security. Paper presented at Proceedings of the 26th USENIX Security symposium.
19. Lok, K., Heng, Y.: DroidScope: seamlessly reconstructing the OS and Dalvik semantic views for Dynamic android malware analysis. Paper presented at Proceedings of the 21st USENIX Security Symposium
20. Sangueun, S., Bongjoon, K., Sangjun, L.: The effective ransomware prevention technique using process monitoring on android platform. Mobile Infor. Syst. 2016, 1–9 (2016)
21. Gharib, A., Ghorbani, A.: DNA-Droid: a real-time android ransomware detection framework. In: Proceedings of Network and System Security – 11th International Conference (NSS 2017), vol. 10394, pp. 184–198. Helsinki (2017)
22. Alberto, F., et al.: Extinguishing ransomware - a hybrid approach to android ransomware detection. In: Proceeding of Foundations and Practice of Security – 10th International Symposium (FPS 2017), vol. 10723, pp. 242–258. Nancy (2017)
23. Cohen, F.: Computer viruses, Ph.D Thesis University of Southern California (1986)
24. Eric, F.: Computer Viruses: From Theory to Applications. Springer (2005)
25. Lemmou, Y., El Mamoun, S.: Overinfection in ransomware. In: Proceeding of Security in Computing and Communications – 6th International Symposium, pp. 504–519. SCC, Bangalore (2018)
26. Lemmou, Y., El Mamoun S.: Infection, self-reproduction and over-infection in ransomware: the case of TeslaCrypt. In: Proceedings of 2018 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2018, pp. 1–8. Glasgow (2018)
27. Monnappa, K.A.: Learning Malware Analysis: Explore the Concepts, Tools, and Techniques to Analyze and Investigate Windows Malware. Packt Publishing Ltd (2018)

28. Routa, M., et al.: Ransomware network traffic analysis for pre-encryption alert. In: Foundations and Practice of Security – 12th International Symposium (FPS 2019), vol. 12056, pp. 20–38. Toulouse (2019)

29. Cabaj, K., Gregoreczyk, M., Mazureczyk, W: Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics. CoRR. vol. abs/1611, 08294 (2016)

30. Lemmou, Y., Souidi, E.M.: PrincessLocker analysis. In: Proceeding of 2017 International Conference on Cyber security and Protection of Digital Services, 1–10. (2017)

31. Yaneya, Jay: Do Random IP Lookups Mean Anything? SANS (2018). https://www.sans.org/reading-room/whitepapers/detection/paper/38405

32. Marco, A., et al.: Clustering android malware families by http traffic. In: 10th International Conference on Malicious and Unwanted software (MALWARE), pp. 128–135. Fajardo (2015)

33. Zhang, H., Yao, D., Ramakrishnan, N.: Causality-based sensemaking of network traffic for android application security. In: Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security (AISec ’16), pp. 47–58. New York (2016)

34. Chen, J., et al.: Uncovering the face of android ransomware: characterization and real-time Detection. IEEE Trans. Inf. Forensics Secur. 13(5). (2018)

35. Shweta, S., Rakesh, K., Krishna: Ransom/Analytic: the evolution and investigation of android ransomware. In: Proceedings of International Conference on IoT Inclusive Life (ICIIL 2019), vol. 116. NITTTR Chandigarh. (2019)

36. Lee, J., Lee, J., Hong, J.: How to make efficient Decoy files for ransomware Detection? In: Proceedings of the International Conference on Research in Adaptive and Convergent Systems. RACS, Krakow (2017)

37. Moussaileb, R., et al.: Ransomware’s early mitigation mechanisms. In: Proceedings of the 13th International Conference on Availability, Reliability and Security. ARES, Germany (2018)

38. Lemmou, Y., Souidi, E.M.: Inside GandCrab ransomware. In: Proceedings of Cryptology and Network Security - 17th International Conference, CANS 2018, vol. 11124, pp. 154–174. Naples (2018)

39. Lemmou, Y., Souidi, E.M.: An overview on Spora ransomware. In: Proceedings of Security in Computing and Communications – 5th International Symposium, SSCC 2017, Manipal, 259–275. (2017)

40. Cazzuone, A., Martinelli, F., Mercaldo, F.: A novel structural-entropy-based classification technique for supporting android ransomware detection and analysis. In: IEEE International Conference on Fuzzy Systems, pp. 1–7. (2018)

41. Abdulrahman, A., et al. RanDroid: structural similarity approach for Detecting ransomware applications in android platform. In: IEEE International Conference on Electro/Information Technology, EIT 2018, pp. 892–897. Rochester, MI (2018)

42. Anandarajan, M., Hill, C., Nolan, T.: Practical Text Analytics. Springer International Publishing (2019)

43. Cosma, G., Joy, M.: An approach to source-code plagiarism Detection and investigation using latent semantic analysis. IEEE Trans. Comput. 61(3), 379–394 (2012)

44. Qian, F., et al.: A case for unsupervised-learning-based spam filtering. Paper presented at SIGMETRICS ’10, (2010)
## APPENDIX

| Ransomware   | Version                          | First sub. in VT |
|--------------|----------------------------------|------------------|
| Dharma       | Two versions: .cmb and .bip      | Aug./May 2018    |
| Kraken       | . onion version                  | Oct. 2018        |
| CryptoLocker | . ecc version                    | Apr. 2015        |
| PrincessLocker| Three versions                   | Nov. 2016 and Aug. 2018 |
| TeslaCrypt   | v3, the .mp3 extension           | Feb. 2016        |
| Spora        | First version                    | Feb. 2017        |
| Striked      | -                                | July 2017        |
| Alpha        | . encrypt version                | Apr. 2016        |
| Diamond      | -                                | Aug. 2017        |
| Cerber       | NSIS version.                    | Mar. 2017        |
| Locky        | lakitis version                  | Aug. 2017        |
| BTCWare      | payday version                   | Oct. 2017        |
| Blind        | Napoleon version                 | Dec. 2017        |
| GandCrab     | v1.0-1, all v2 ver., v3, v4 and v4.1 | During 2018     |
| StopDjvu     | Multiple versions                | Jan. 2019        |
| LockerGoga   | -                                | Jan. 2019        |
| WannaCry     | -                                | May 2017         |
| Nemty        | -                                | Sep. 2019        |
| Sodinokibibi | -                                | Aug. 2019        |
| FTCode       | -                                | Nov. 2019        |
| Scarab       | -                                | Jan. 2018        |
| StopDjvu     | puma version                     | Nov. 2018        |

Abbreviation: VT, virus total.