Using Reverse Engineering to Face Malware

Utilizando la ingeniería inversa para enfrentar Malware

Utilizando a engenharia reversa para enfrentar malware

Carlos Andrés Sánchez Venegas[1]
Camilo Aguado Bedoya[2]
Daniel Orlando Díaz López[3]
Juan Carlos Camilo García Ruiz[4]

Received: November 15th, 2018
Accepted: February 20th, 2019
Available: May 21th, 2019

How to cite this article:
C. A. Sánchez-Venegas, C. Aguado-Bedoya, D. O. Díaz-López, J. C. García-Ruiz, "Using Reverse Engineering to Face Malware", *Revista Ingeniería Solidaria*, vol. 15, n.° 2, 2019.
DOI: https://doi.org/10.16925/2357-6014.2019.02.02

Artículo de investigación. https://doi.org/10.16925/2357-6014.2019.02.02

[1] Escuela Colombiana de Ingeniería Julio Garavito
ORCID: https://orcid.org/0000-0003-4599-0450

[2] Escuela Colombiana de Ingeniería Julio Garavito
ORCID: https://orcid.org/0000-0001-6255-9828

[3] Escuela Colombiana de Ingeniería Julio Garavito
ORCID: https://orcid.org/0000-0001-7244-2631
Email: daniel.diaz@escuelaing.edu.co

[4] Armada Nacional
ORCID: https://orcid.org/0000-0002-7038-0710

Abstract

This paper is a product of the research Project "Cyber Security Architecture for Incident Management" developed in the Colombian School of Engineering Julio Garavito in the year 2018.

Introduction: Reverse engineering involves deconstructing and extracting knowledge about objects. The use of reverse engineering in malware analysis is extremely useful in understanding the functionalities and purposes of a suspicious sample.

Methods: This paper makes use of Radare which is one of the most popular open source tools for reverse engineering, with the aim of dealing with malware.

Results: A use case related to hacking of anti-sandbox malware is presented, in such a way that it is possible to analyze the behavior of the sample using a sandbox. Additionally, another use case is presented, where an in-depth analysis of a malicious Android application aimed to the audience of a popular event (FIFA World Cup 2018) is developed, making it possible to demonstrate the relevance of reverse engineering techniques in end-user protection strategies.

Conclusions: This paper shows how the results of a reverse engineering process can be integrated with Yara rules, allowing for the detection of malware on the fly, and it also shows an alternative to automatically generating Yara rules through the yarGen generator.

Originality: Use of Open Source reversing solutions by Colombian Law Enforcement Agencies has not been discussed previously, making this paper a notable element toward the modernization of the military forces.

Limitation: Different approaches and perspectives about the limitations in the use of reverse engineering by Law Enforcement Agencies are also shared.

Keywords: Reverse engineering, Radare, Sandboxing, Yara rules, Malware analysis.

Resumen

Este artículo es producto del proyecto de investigación "Cyber Security Architecture for Incident Management" desarrollado en la Escuela Colombiana de Ingeniería Julio Garavito en el año 2018.

Introducción: La ingeniería inversa permite deconstruir y extraer conocimiento de objetos. El uso de la ingeniería inversa en el análisis de malware es extremadamente útil para comprender las funcionalidades y los propósitos de una muestra sospechosa.

Métodos: Este artículo utiliza Radare, la cual es una de las herramientas de código abierto más populares para ingeniería inversa con el objetivo de hacer frente a las amenazas de malware.

Resultados: Se presenta un caso de uso relacionado al análisis de malware anti-sandbox, de forma que sea posible analizar el comportamiento de la muestra utilizando una sandbox. Además, se presenta otro caso de uso en el que se desarrolla un análisis en profundidad de una aplicación maliciosa de Android dirigida a la audiencia de un evento popular (Copa Mundial de la FIFA 2018), que permite demostrar la relevancia de las técnicas de ingeniería inversa en las estrategias de protección al usuario final.

Conclusiones: Este artículo muestra cómo los resultados de un proceso de ingeniería inversa se pueden integrar con reglas Yara, lo que permite detectar malware, y también muestra una alternativa para generar automáticamente reglas Yara a través del generador yarGen.

Originalidad: El uso de soluciones de ingeniería inversa de código abierto por parte de las agencias de seguridad del estado no ha sido discutido anteriormente, lo que hace de este artículo un elemento notable de apoyo hacia la modernización de las fuerzas militares.

Limitación: Se comparten diferentes enfoques y perspectivas sobre las limitaciones en el uso de ingeniería inversa por parte de las agencias de seguridad del estado.

Palabras clave: ingeniería inversa, Radare, Sandboxing, reglas Yara, análisis de Malware.receptores GPS de una sola frecuencia, matriz de cosenos de dirección.

Resumo

Este artigo é produto do projeto de pesquisa "*Cyber Security Architecture for Incident Management*" desenvolvido na Escuela Colombiana de Ingeniería Julio Garavito em 2018.

Introdução: a engenharia reversa permite desconstruir e extrair conhecimento de objetos. O uso da engenharia reversa na análise de malware é extremamente útil para compreender as funcionalidades e os propósitos de uma amostra suspeita.

Métodos: para isso, utiliza-se Radare, que é uma das ferramentas de código aberto mais populares para engenharia reversa com o objetivo de enfrentar as ameaças de malware.

Resultados: apresenta-se um caso de uso relacionado à análise de malware anti-sandbox, de forma que seja possível analisar o comportamento da amostra utilizando uma sandbox. Além disso, apresenta-se outro caso de uso em que se desenvolve uma análise em profundidade de uma aplicação maliciosa de Android dirigida à audiência de um evento popular (Copa do Mundo da FIFA 2018), que permite demonstrar a relevância das técnicas de engenharia reversa nas estratégias de proteção do usuário final.

Conclusões: este artigo mostra como os resultados de um processo de engenharia reversa podem ser integrados com regras Yara, o que permite detectar malware, e também mostra uma alternativa para gerar automaticamente regras Yara por meio do gerador yarGen.

Originalidade: o uso de soluções de engenharia reversa de código aberto por parte das agências de segurança do Estado não tem sido discutido anteriormente, o que torna este estudo um elemento notável de apoio à modernização das forças militares.

Limitação: compartilham-se diferentes abordagens e perspectivas sobre as limitações no uso de engenharia reversa por parte das agências de segurança do Estado.

Palavras-chave: engenharia reversa, Radare, Sandboxing, regras Yara, análise de malware.

1. Introduction

The purpose of this paper is to show the potential of reverse engineering when applied to cybersecurity [1] for the detection of software with malicious or hidden functionalities which can affect a person or company.

Reverse engineering is the process of deconstructing something to discover its architecture and gain knowledge from it [2]. An example to help understand this process is to think of a large building which, after applying reverse engineering, allows for the production of the building plans, type of concrete used in the construction and even information related to who built it. In a software and cybersecurity context, reverse engineering refers to the process of passing from binary code —machine code— to a more understandable language —usually assembly code—, this process is called disassembly.

Reverse engineering is very important in cybersecurity because it supports the examination of samples by means of static analysis, therefore allowing for the

extraction of Indicators of Compromise (IoC). IoCs are typically hashes, compilation dates, import and export functions, file sections, registry keys, hostnames, IPs, emails or even text strings existing in the code, which represent attacker traces. With reverse engineering it is possible to detect, analyze and prevent attacks, which can be empowered with the help of Yara rules [3], which is a language to identify and classify malware samples.

Reverse engineering is also useful in software development because a developer could use it to reverse his own software and discover potential vulnerabilities that had not been considered in the software development process but that an adversary could notice when performing reverse engineering [2].

Another way to analyze files to discover if it is malware or goodware is by means of a sandbox. Sandbox or sandboxing is the process of isolating an application and its execution in a controlled environment —with the help of virtual machines—, so it can be possible to inspect its behavior and actions without really affecting the computer or the network. In this way a sandbox can make a dynamic analysis by looking at the behavior that the file would perform when it is executed. The sandbox executes the file in a totally controlled environment, i.e. an isolated virtual machine, and generates a report where it indicates every movement, contact and altered files that the sample has made. This report will later support the cybersecurity analyst in the identification of the nature of the sample.

Every day malware becomes more sophisticated and incorporates new ways to: avoid being discovered, break into more complex systems and even to avoid being analyzed. In the last case, there is malware that prevents being executed in controlled environments —sandbox— by means of complex techniques, such as validating operative system variables to realize if it is being executed in a controlled environment or checking the internet search history to realize if there is a real user controlling the machine. Other simple techniques that can be used by the malware involve asking for an interaction with the user generating interface dialogues that a sandbox could not respond to and that will pause the execution. In the latter case, it is possible to again see the potential of reverse engineering, which can be used to change the malware code to avoid generating dialogues, once again permitting its analysis in a controlled environment.

Throughout the development of this paper we worked with an open source tool that is currently gaining in popularity called Radare [4] which can be compared with commercial and expensive solutions like IDA Pro [5]. Radare is a very complete solution which provides analysis for the majority of available architectures, and it also includes helpful tools or plugins, e.g. a debugger. A debugger is used to find bugs

e-ISSN 2357-6014 / Vol. 15, n°. 2 / mayo 2019 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

or problems with some given application using features like breakpoints to stop the execution of the program in a desired step. Radare also offers several types of visualization —assembler, hexadecimal— [2].

This paper is composed as follows. Section 2 presents two cases of reverse engineering. Then, Section 3 describes the existing ways to automatically generate Yara rules. Later, Section 4 includes a discussion regarding the current use of reverse engineering by Law Enforcement Agencies that protect critical technological infrastructure —essential governmental services that guarantee the state operation or indispensable services provided to the community by a private company—. Finally, some conclusions and future works are presented.

2. Using reverse engineering to analyze Malware

Radare is a multi-platform open-source framework written in C that was initially designed for file recovery but has evolved towards a tool for reverse engineering and analysis of binary files. It has the support of the GitHub community which offers continual improvements for its functionalities, making it a very complete tool comparable with payment solutions that have the same purpose. Every year the Radare congress (r2con) is held in September in Barcelona (Spain) where the novelties around the tool and its applications are discussed. With this tool it is possible to disassemble —convert from machine to assembly code— and assemble software with many different types of architecture. It has been used a lot in forensics for studying file systems through static analysis and performing malware analysis through dynamic analysis, using an integrated debugger.

This section describes anti-sandbox techniques (Section 2.1) and shows two practical cases of reverse engineering using Radare (Section 2.2 and Section 2.3). In the first case reverse engineering is applied to modify a malware sample, so it can be analyzed by other tools such as a sandboxing tool [6][7]. In the second case, a reverse engineering analysis is done over a mobile application using Radare and some plugins like Androguard.

Although there are several tools that have similar functions to Radare, none of them include as many functionalities as Radare. Some tools are focused on certain types of binary files, like Apktool which is applicable for binaries of Android applications. Apktool can decode resources to nearly original form and rebuild them after making some modifications. On the other hand, Radare is capable of providing the

e-ISSN 2357-6014 / Vol. 15, n°. 2 / mayo 2019 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

same functionalities provided by Apktool and also to emulate an Android environment in order to trace the program execution and identify system events and behaviors. Dex2jar is another common tool used for Android applications, but its main purpose is to read specific type of Android files parsed as Dalvik Executable format (.dex / .odex) which are compiled classes files where all code is written.

Another popular tool used to analyze binary code is OllyDbg, which is specialized in certain types of files used in Microsoft Windows platforms and offers a great debugger functionality, however OllyDbg is outdated as its last release was in 2013. Oppositely, Radare offers debugger functions for binaries belonging to different kinds of platforms (not only Microsoft Windows platforms), supporting even Game Boy applications.

For the purpose of malware analysis, a tool used by many CERT (Computer Emergency Response Teams) worldwide is PeStudio, which can be used to perform a Malware Initial Assessment. PeStudio can discover various types of indicators and classify items without the risk of infection for the user platform. The indicators can be used to perform an initial classification of the malware however it is does not provided enough information to develop a deep malware analysis that include insights regarding the application behavior, a functionality that is important in this type of analysis as will be described in subsection 2.1.

The reason for choosing Radare as the main solution when developing the research presented in this paper was its large range of tools incorporated in one single framework, providing similar functionalities as the ones included in the previously mentioned tools, and the inclusion of additional functions like: i) Assembler / Disassembler, ii) Hex editor in 64bits blocks, iii) Checksum calculator (per block basis), iv) Transparent usage of processes, disks, files, ram, v) Filesystem mounting (fat, ntfs, ext2, among others), vi) Binary file analyzer for Windows, Linux, Mac, Java, Dalvik, vii) Debugger (w32, Linux, Mac, iOS), viii) Binary diff tool, ix) Shellcode creation and editing functionality and x) Multiple scripting language support.

2.1. Analyzing malware with sandbox

The goal of most malware is to be able to produce the highest impact in victims by making as little noise as possible, so it can continue operating and infecting other devices. Due to this behavior sandbox analysis techniques emerged, whose purpose is to deceive the malware so it can perform its functions in a controlled and monitored environment, thus allowing for the analysis of its behavior in the infected machine and its subsequent characterization for future rapid identification [8][9]. Cybercriminals

e-ISSN 2357-6014 / Vol. 15, n°. 2 / mayo 2019 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

have started to use different anti-sandbox techniques to trick sandboxes into believing that the file being analyzed is harmless. What follows is a list of the most common anti-sandbox techniques [10].

2.1.1. INTERACTION WITH THE USER

This is a very common anti-sandbox technique where the execution of the malware is not done until some actions are made by the user, such as the execution of a certain number of mouse clicks, the closing of some pop-ups, the typing of some defined text or the scroll over a text. Regarding this, some sandbox solutions are able to simulate a human behavior by moving the mouse, clicking randomly or scrolling content, however sometimes a more complex interaction is required which cannot be simulated by the sandbox. When the required interactions are not fulfilled the malware does not execute completely and it is marked as inoffensive by the sandbox.

2.1.2. SANDBOX DETECTION

The malware can detect certain processes that are executed only by sandbox solutions. When the malware detects these processes, it pauses its execution or starts inoffensive and fake actions. Similarly, malware can search files that are common in real user devices but not in sandboxes, such as search history files of web browsers, and when not found, the malware infers that it is running in a monitored and controlled environment.

2.1.3. DELAY OF EXECUTION AND EXECUTION AFTER RESTART

Sandboxes usually consider an execution time for a malware analysis, if at the end of this time no suspicious activity is detected, the sample is classified as harmless. This behavior is widely used by cybercriminals since they develop malware that only activates after an initial sleep period. When the malware sleep period is bigger than the sandbox execution time, the malware goes unnoticed through the sandbox. The sandboxes at the end of a malware analysis restores the virtual machine to continue with the analysis of a new sample. This operation is exploited by the malware creators, so they define a malicious process that only starts after the restart of the machine making it difficult for the sandbox to identify the malicious process.

e-ISSN 2357-6014 / Vol. 15, n°. 2 / mayo 2019 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

The previously mentioned methods are commonly used by cybercriminals to deceive sandboxes and go unnoticed, however there may be more methods that have not yet been discovered because of their complexity.

2.2. Case 1: Using Radare to enable malware analysis in a sandbox

Radare is a great tool that allows you to detect the techniques presented in Section 2.1, since by reverse engineering a malware sample it becomes possible to detect these techniques with relative ease. This section will focus on the use of Radare to handle the technique described in section 2.1.1 (Interaction with the user). For this purpose, an application was created which displays a pop-up with a message asking the user to click to continue the execution, simulating what a malware would do to cheat a sandbox [11].

Figure 1. Example of pop-up window
Source: own work

```
root@kali:~/Downloads/c++# ./sample
If this was a malware, your computer would have already beed infected
 Antisandbox Malware Demo
```

Figure 2. Simulation of infection after user interaction
Source: own work

Using Radare the malware sample will be modified so that when executing it does not require user interaction and so can be analyzed by the sandbox solution.

- The first step is to open the sample —binary file— with Radare as shown in Figure 3, where the writing on the file capability is enabled with the argument -Aw (Allow writing) and a complete analysis is requested with the argument -AAAA —Analyze commands, Analyze all functions, Autoname functions, Emulate code—. This will also enable the Radare console.

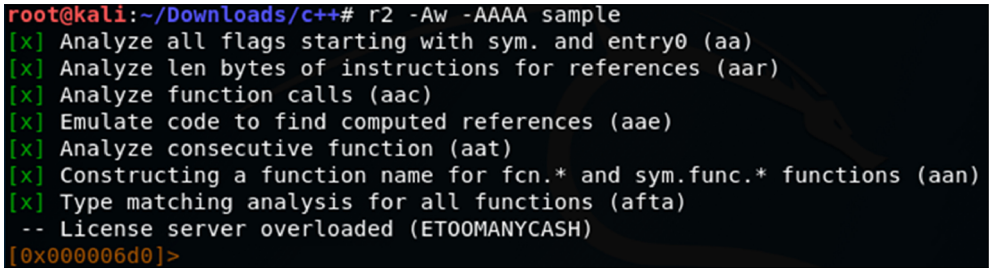

Figure 3. Opening of the program with Radare
Source: own work

- Within the Radare console, the functions used by the sample are listed using the command afl —analyze functions and list them— as shown in Figure 4.

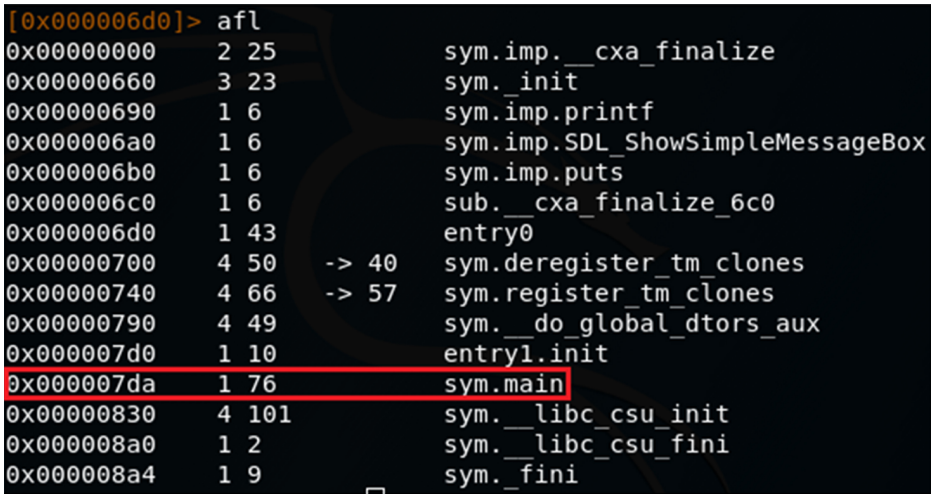

Figure 4. List of functions
Source: own work

e-ISSN 2357-6014 / Vol. 15, n°. 2 / mayo 2019 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

- In the list of functions, normally the main function of the program is easy to find (even if in some cases it is under another name). The sym.main function is highlighted in Figure 4, having the memory location on the left side. Using the command s sym.main (seek address) it is possible to jump to the memory location where this function is stored (Figure 5). The memory position then changes from 0x000006d0 to 0x000007da (the shorter nomenclature 0x7da will be used).

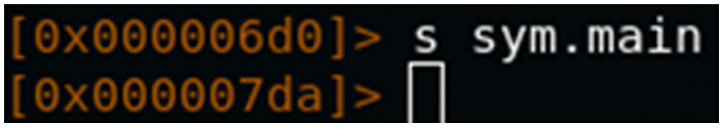

Figure 5. Change in memory location
Source: own work

- Being in the memory location of the main function [12], it is possible to print the assembler code of that function to see the purpose behind it with the command pdf (print disassemble function), as shown in Figure 6. In this figure is possible to see a call to the function that is responsible for displaying the popup window. This call is in memory position 0x7fa and the strings used by the message are in positions 0x7e7 and 0x7ee.

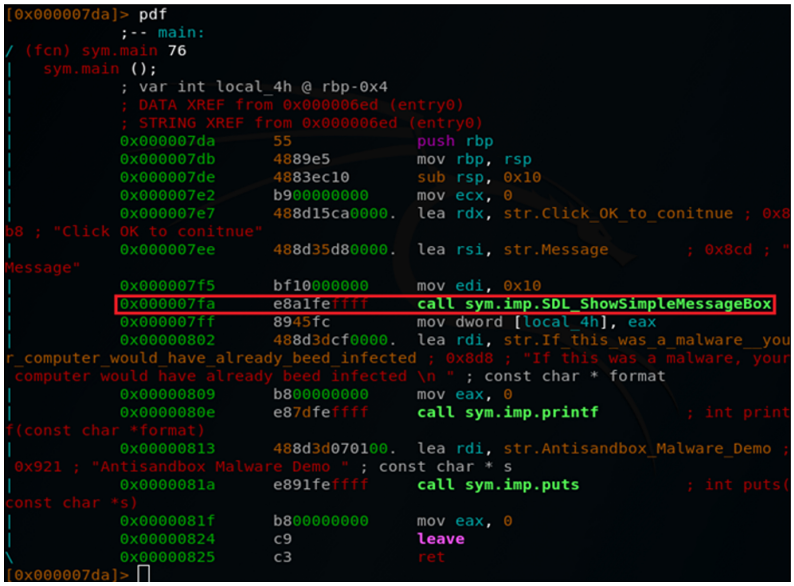

Figure 6. Assembler code of sym.main function
Source: own work

e-ISSN 2357-6014 / Vol. 15, n°. 2 / mayo 2019 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

- Once the memory positions used in the pop-up message have been identified, it is possible to use the command wa —write opcode— to overwrite the mov function located in 0x07e2 with a jmp function, which will serve to skip the call to the pop-up, as is shown in Figure 7 and 8.

```
[0x000007da]> s 0x000007e2
[0x000007e2]> wa jmp 0x000007ff
Written 2 byte(s) (jmp 0x000007ff) = wx eb1b
[0x000007e2]>
```

Figure 7. Overwriting of function mov
Source: own work

```
[0x000007da]> pdf
           ;-- main:
/ (fcn) sym.main 76
|   sym.main ();
|           ; var int local_4h @ rbp-0x4
|           ; DATA XREF from 0x000006ed (entry0)
|           ; STRING XREF from 0x000006ed (entry0)
|           0x000007da      55              push rbp
|           0x000007db      4889e5          mov rbp, rsp
|           0x000007de      4883ec10        sub rsp, 0x10
|       ,=< 0x000007e2      eb1b            jmp 0x7ff
|       |   0x000007e4      0000            add byte [rax], al
|       |   0x000007e6      00488d          add byte [rax - 0x73], cl
|       |   0x000007e9      15ca000000      adc eax, 0xca
|       |   0x000007ee      488d35d80000.   lea rsi, str.Message        ; 0x8cd ; "
Message"
|       |   0x000007f5      bf10000000      mov edi, 0x10
|       |   0x000007fa      e8a1feffff      call sym.imp.SDL_ShowSimpleMessageBox
|       `-> 0x000007ff      8945fc          mov dword [local_4h], eax
|           0x00000802      488d3dcf0000.   lea rdi, str.If_this_was_a_malware__you
r_computer_would_have_already_beed_infected ; 0x8d8 ; "If this was a malware, your
 computer would have already beed infected \n " ; const char * format
|           0x00000809      b800000000      mov eax, 0
|           0x0000080e      e87dfeffff      call sym.imp.printf         ; int print
f(const char *format)
|           0x00000813      488d3d070100.   lea rdi, str.Antisandbox_Malware_Demo ;
 0x921 ; "Antisandbox Malware Demo " ; const char * s
|           0x0000081a      e891feffff      call sym.imp.puts           ; int puts(
const char *s)
|           0x0000081f      b800000000      mov eax, 0
|           0x00000824      c9              leave
\           0x00000825      c3              ret
[0x000007da]>
```

Figure 8. Assembler code of sym.main function after the overwriting
Source: own work

Once this modification on the assembler code is done, the program will not show the pop-up window in the next execution, which will enable the sandbox to make a malware analysis as usual.

2.3. Case 2: Analysis of a malicious Android application

The intent of this section is to show the capacity that Radare has to develop cyber intelligence through the Indicators of Compromise that it can extract from a malware sample.

The first step to be able to analyze a malicious mobile application is to find it, so different android markets currently available on the internet were reviewed in detail. Since Google has greatly improved the security of its application marketplace (Google Play Store) with tools such as Virus Total, it would be a more arduous task to find a malicious mobile application in that market. Additionally, we consider it would be very likely that because of the FIFA World Cup 2018, some cybercriminals would intend to generate a related mobile application that looks innocent and informative but with hidden malicious purposes. So, in order to find a malicious mobile application, another android marketplace with a lower reputation regarding security concerns were reviewed. One of the reviewed marketplaces was Aptoide [13] which is much more permissive in the publication of mobile applications and therefore has a greater probability of hosting malware applications. Aptoide hosts a mobile application (Figure 9) which at first glance looks as a very friendly application, which shows information related to the World Cup in Russia.

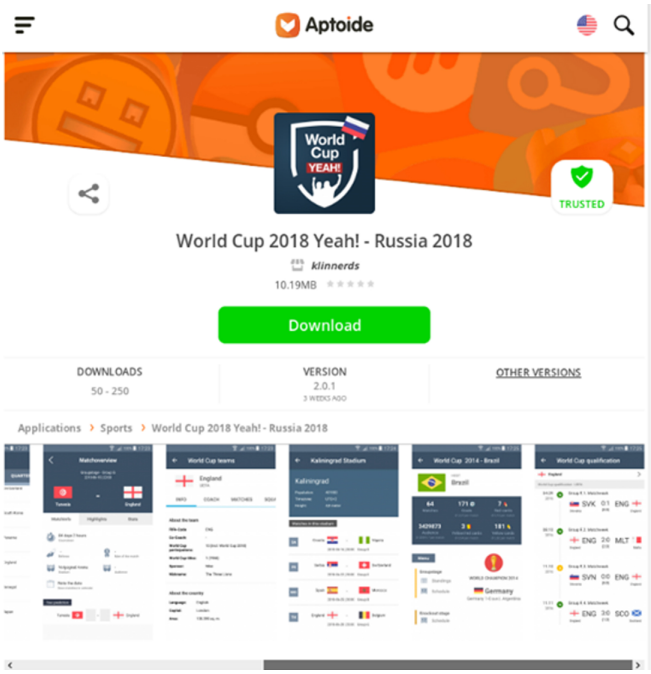

Figure 9. Mobile application found in Aptoide marketplace
Source: [14]

The first step to start reversing the application "World Cup 2018 Yeah! - Russia 2018" is to acquire the .APK of the application —APK is the executable of applications for devices with Android operating system [15]—. The APK is a compressed file that contains several files including the AndroidManifest.xml which manages all the activities, permissions and services required by the application, and the .dex class where all the compiled code of the program is found [16]. The APK can be uncompressed with the unzip Linux command as shown in Figure 10.

Figure 10. Extraction of the content of the APK
Source: own work

When opening the AndroidManifest.xml with a regular editor, e.g. vi Linux command, it will not be possible to obtain meaningful data (Figure 11) because the content is in machine code.

Figure 11. Opening the AndroidManifest.xml with editor vi
Source: own work

To access the content of machine code files, e,g, AndroidManifest.xml, it is necessary to use Radare with Androguard which is a specific tool to perform reverse engineering for Android applications. The command "r2pm -r axml2xml AndroidManifest.xml" is used to access a readable version of the AndroidManifest.xml, as shown in Figure 12. "r2pm" means that a Radare command will be executed with the help of

a plugin, i.e. Androguard, and "-r axml2xml" means that a compiled xml file will be converted to an uncompiled xml Android file.

Figure 12. Radare and Androguard
Source: own work

In the uncompiled AndroidManifest.xml it is possible to see some common permissions related to goodware, e.g. access to internet, but there are also some strange permissions [17] that are typically found in malware like installation of packages, change of Wifi status, control over the camera or management of accounts. The application "World Cup 2018 Yeah! - Russia 2018" have all these suspicious permissions [18] —marked in red box in Figure 13—; a mobile application that offer information about the world cup does not actually need to be able to take pictures, much less change the state of the Wifi, install packages and manage user accounts. It is also possible to see some payment activities through PayPal, which are not common in an application of this type. Analyzing the AndroidManifest file makes it possible to know what an application can do and estimate the impact on the device and the user [19].

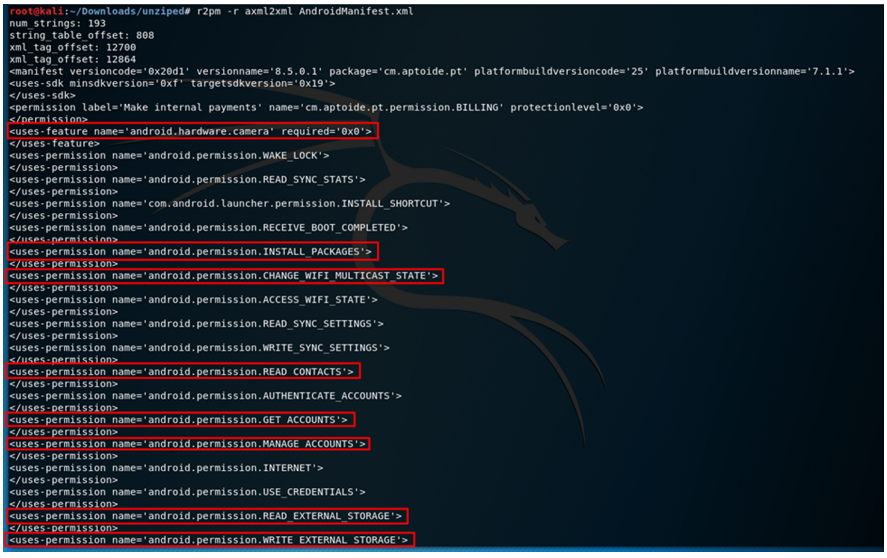

Figure 13. Reverse engineering on AndroidManifest
Source: own work

Radare allows for the analysis of Imports and Exports by means of modules such as rabin2. It also allows for the discovery of any type of hash for a given file through the rahash2 module. Figure 14 shows some Imports related to telephony that are included in the application "World Cup 2018 Yeah! - Russia 2018", which execute some unusual actions like identifying the type of phone, the state of the sim and the telephone operator, amongst others [20].

The rest of the process of collecting Indicators of Compromise is done in the classes.dex file. With the help of the command "rabin2 -qi clases.dex | grep -i -e Telephony " it becomes possible to see all imports related to telephony, where -qi means "show a few data (q) related to imports (i)".

```
root@kali:~/Downloads/unziped# rabin2 -qi classes.dex | grep -i -e Telephony
Landroid/telephony/PhoneNumberFormattingTextWatcher.method.<init>()V
Landroid/telephony/PhoneNumberUtils.method.stripSeparators(Ljava/lang/String;)Ljava/lang/String
Landroid/telephony/TelephonyManager.method.getNetworkCountryIso()Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getNetworkOperator()Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getNetworkOperatorName()Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getPhoneType()I
Landroid/telephony/TelephonyManager.method.getSimCountryIso()Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getSimOperator()Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getSimOperatorName()Ljava/lang/String;
Landroid/telephony/TelephonyManager.method.getSimState()I
root@kali:~/Downloads/unziped#
```

Figure 14. Imports shown by rabin2
Source: own work

As seen previously, inside Radare Shell it is possible to execute commands, change memory position, print functions, change the execution in a specific memory process, etc. One of these commands is izq, that allows the operator to see the strings "burned" in code, this is useful in reverse engineering because it can indicate the installation path of an apk, http or https connections and super user requests, amongst other insights. In the case of the application "World Cup 2018 Yeah! - Russia 2018" it is possible to see how it attempts to gain access to super user commands in order to get new permissions, additional to those found before in the AndroidManifest file (Figure 15).

```
[0x00196c56]> izq ~ /system
0x1a1acd 25 25 /system/app/Superuser.apk
0x1a1ae8 15 15 /system/xbin/su
[0x00196c56]>
```

Figure 15. Strings "burned" in code
Source: own work

e-ISSN 2357-6014 / Vol. 15, n°. 2 / mayo 2019 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

This section shows how Radare generates great value for the malware analysis in the sandbox environment since it can help to eliminate some anti-sandbox functions whose purpose is to deceive these environments and pass themselves off as goodware. Later, an analysis over the mobile application "World Cup 2018 Yeah! - Russia 2018" was developed which is fundamental to generate Indicators of Compromise, so some countermeasures can be implemented, e.g. through the creation of Yara rules, to prevent this application, or similar ones, from getting installed on mobile devices.

3. Countermeasures Designed from Reverse Engineering

Radare is a great tool whose purpose, as already mentioned, is to perform a deep analysis of malware making use of reverse engineering. However, it does not guarantee that the extracted information can be used in the correlation of different samples. To solve this, it is necessary to make use of external tools that facilitate the handling of information and help to identify the presence of a specific malware or malware family.

3.1. Yara

Yara is an open-source tool developed by the creators of VirusTotal [21], which is a service for the analysis of malware samples and URLs. Yara is designed to help analysts to identify and classify malware samples. With this tool it is possible to create rules for a malware family using textual and binary patterns [22].

Yara can be integrated with different programming languages, Python being the most common [23], to extend its use and help automate the search of Yara rules that are related with the malware being analyzed.

Figure 16 shows an example of a Yara rule where if any of the included strings match with the information extracted from the analyzed sample, a silent_banker malware could be identified.

e-ISSN 2357-6014 / Vol. 15, n°. 2 / mayo 2019 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        thread_level = 3
        in_the_wild = true

    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"

    condition:
        $a or $b or $c
}
```

Figure 16. Yara rule
Source: [21]

Radare facilitates the creation of Yara rules, since through reverse engineering it is possible to obtain Indicators of Compromise that are used to build robust and precise rules [24].

3.2. R2Yara

R2Yara is an extension of Yara rules which uses Radare to make a better analysis of the malware sample, since it not only consider strings detected in the sample, but it can use all Radare commands to make more effective malware identification as seen in Figure 17.

e-ISSN 2357-6014 / Vol. 15, n°. 2 / mayo 2019 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

```
rule rule_info
{
condition:
    r2.info.havecode == 1 and
    r2.info.pic == 0 and
    r2.info.canary == 1 and
    r2.info.nx == 1 and
    r2.info.crypto == 0 and
    r2.info.va == 1 and
    r2.info.intrp contains "linux-x86" and
    r2.info.bintype == "elf" and
    r2.info.class contains "ELF64" and
    r2.info.lang == "c" and
    r2.info.arch == "x86" and
    r2.info.bits == 64 and
    r2.info.machine == "AMD x86-64 architecture" and
    r2.info.os == "linux" and
    r2.info.minopsz == 1 and
    r2.info.maxopsz == 16 and
    r2.info.pcalign == 0 and
    r2.info.subsys == "linux" and
    r2.info.endian == "little" and
    r2.info.stripped == 1 and
    r2.info.static == 0 and
    r2.info.linenum == 0 and
    r2.info.lsyms == 0 and
    r2.info.relocs == 0 and
    r2.info.binsz > 100000 and
    r2.info.rpath == "NONE" and
    r2.info.compiled != "Sat Sep 9 11:32:42 2006" and
    r2.info.dbg_file not contains "test" and
    r2.info.guid == ""
}
```

Figure 17. Example of Yara rule using R2yara
Source: own work

3.3. YarGen

YarGen is an open-source tool whose main function is to create Yara rules from the strings that are found in a malware sample. These rules are made by comparing and ignoring strings that appear in goodware. This creation can be done with multiple samples at once from a large database of strings that is downloaded together with the tool, even if it is possible to create an own reference database. The multiple analysis is a great feature since the manual rule generation can be a wasteful and slow process. Thanks to YarGen this generation process is streamlined, however it does not replace it, since it is necessary to clean strings that are not suitable and complement

those generated rules with their own rules created using r2yara or Androguard. Thus, YarGen becomes a great tool for malware analysts increasing their productivity without compromising the quality of the Yara rules.

```
/*
   Yara Rule Set
   Author: yarGen Rule Generator
   Date: 2018-05-10
   Identifier: Malwares
   Reference: https://github.com/Neo23x0/yarGen
*/

/* Rule Set ----------------------------------------------------------------- */

import "pe"

rule Backdoor_MSIL_Tyupkin_c {
   meta:
      description = "Malwares - file Backdoor.MSIL.Tyupkin.c.ViR"
      author = "yarGen Rule Generator"
      reference = "https://github.com/Neo23x0/yarGen"
      date = "2018-05-10"
      hash1 = "16166533c69f2f04110e8b8e9cc45ed2aeaf7850fa68845c64d92ff907dd44f0"
   strings:
      $s1 = "?A0x7d798523.??__E?InitializedPerProcess@CurrentDomain@<CrtImplementationDetails>@@$
$Q2W4State@Progress@2@A@@YMXXZ" fullword ascii /* score: '23.42'*/
      $s2 = "?A0x7d798523.?InitializedPerProcess$initializer$@CurrentDomain@<CrtImplementationDetails>@@$
$Q2P6MXXZA" fullword ascii /* score: '21.00'*/
      $s3 = "??_C@_0CG@OAALEMGB@?6Err?5executing?5WFSExecute?5with?5c@" fullword ascii /* score: '19.00'*/
      $s4 = "/C ping 127.0.0.1 -n 8 & del /F /S /Q " fullword wide /* score: '18.00'*/
      $s5 = "lSystem.Resources.ResourceReader, mscorlib, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089#System.Resources.R" ascii /* score: '18.00'*/
      $s6 = "System.Security.Permissions.SecurityPermissionAttribute, mscorlib, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934" ascii /* score: '18.00'*/
      $s7 = "MSXFS.dll" fullword ascii /* score: '18.00'*/
      $s8 = "??_C@_0ED@JMFNMNIH@ERR?5executing?5WFSGetInfo?$CIWFS_INF@" fullword ascii /* score: '18.00'*/
      $s9 = "ulssm.exe" fullword ascii /* score: '17.00'*/
```

Figure 18. Example of Yara rule generated by yarGen
Source: own work

4. Reverse Engineering as A Tool for Law Enforcement Agencies (Discussion)

"If you control the code, you control the world. This is the future that awaits us" [25] is a quote which describes the daily reality of the theater of operations. The theater of operations can be understood as a space which is no longer just a geophysical site but a real scenario with an audience able to influence what happens on the scene by pointing thumbs up or down, like in the Roman circus. The cyberspace is such a theater which has become the fifth domain of war where different actors, e.g. hacktivists, cyberspies, hackers, governments and organizations, seek to exploit code vulnerabilities to obtain competitive advantages, play the information —or disinformation— war, thieve industrial secrets or simply protest against something.

This reality justifies the fact that one of the most used modalities of attack is through malware development, which is aided by benefits brought about by automation, technification and supercomputers. Specifically, the malware development is

boosted by easy-to-use tools that can be employed by actors to create so much malware that it is estimated that tens of thousands of new variants of malware are being created every hour (a statistic demonstrated by the cybersecurity company SonicWall in its Report of Cyber Threats for 2018). SonicWall has registered 5.99 billion malware attacks for the first semester of 2018, representing an increase of 102 % over the same six-month period of 2017 [26], as seen in Figure 19.

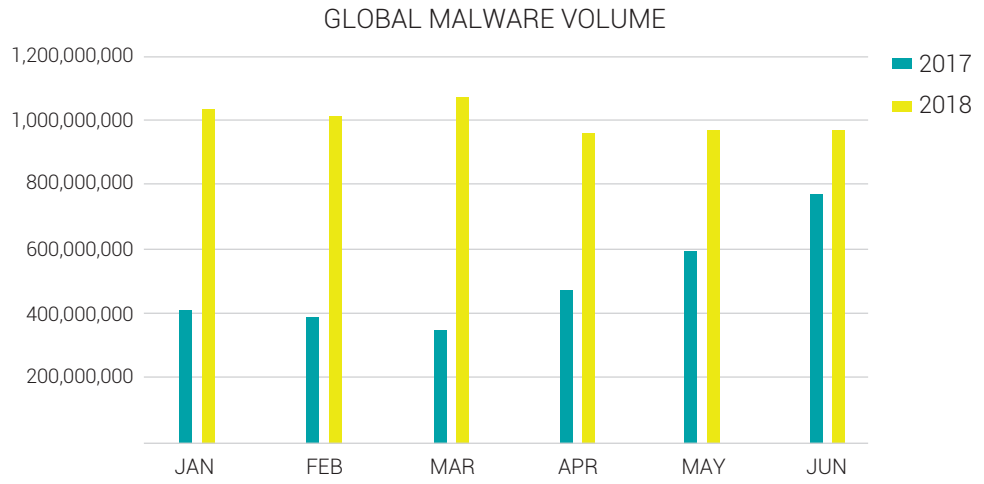

Figure 19. Comparison of malware in the first semester for 2017 and 2018
Source: Report of Cyber Threat 2018, SonicWall.

Considering previous facts, law enforcement agencies have been developing capacities regarding country cyberdefense and citizen cybersecurity, which include study of malware and intelligence of threats. One important case where these cyberdefense and cybersecurity capabilities were tested occurred in May 2017, when different countries in the word suffered a cyberattack due to a ransomware called Wannacry. Wannacry encrypted all information on a computer and asked for a ransom which should be paid in virtual currencies. Wannacry took advantage of a Windows operating system vulnerability to attack and spread itself through the network, making it one of the most dangerous malware for Colombia at that time and placing its national security agencies in a difficult situation. The solution to stop this attack was found by the cybersecurity researcher Marcus Hutchins (@malwaretechblog) who did reverse engineering to the malware using free software tools and found that it was communicating with a domain name that was still not registered on the internet; so Marcus bought the name for $ 10 and stopped the attack.

As seen in Figure 19, malware is a threat that has exponential growth curves instead of linear ones, that is why it is crucial to have open source tools like Radare2 Framework, Androguard, Yara rules and yarGen, which can be applied in an effective way to reveal malware structure and operation. This information regarding the malware possesses great utility because it supports the development of effective countermeasures and extends the knowledge about the origin of the malicious code and its attack vectors [27]. Recovering information from the malware is part of the field named "threat intelligence" or "cyberintelligence". To perform reverse engineering, Law Enforcement Agencies, enterprises and Security Operation Centers have traditionally used commercial solutions from providers like FireEye (Sandbox AX5550), Fortinet, Checkpoint, MacAfee o Hex-Rays (IDA PRO). That is why research such as the one the presented in this paper, where non-commercial tools are used, allow Law Enforcement Agencies to not only lower operating costs (without sacrificing effectiveness compared to commercial options), but also allow analysts to transfer knowledge that allow for the creation and customization of their own tools. The development of cyberdefense and cybersecurity capabilities is definitely supported by the adaptation of new cyber security tools which conduce to reaching a greater level of maturity in the national cybersecurity or cyber defense strategy.

Malware constantly mutates, reinvents and transforms itself with the aim of skipping or violating security controls such as sandboxes as was discussed in Section 2. Many organizations invest large amounts of money licensing sandboxes which, as understood in the security sector, are not fully effective. This is why analysis performed by seasoned cybersecurity or cyber defense analysts can be complemented with tools like Radare to show or demonstrate evasion techniques. Radare can be used for generating a precise analysis of a threat that shows its mechanism within a target system and even allows for the modification of the code so that it can be detected by a sandbox, as seen in Sections 2.2 and 2.3.

In a complementary way, Yara rules may be generated and deployed to security devices working with correlation devices, allowing for the detection and classification of malware in the networks of organizations which in turn, can be uploaded to help desks or international Cybersecurity Emergency Response Teams (CERTs) thereby mitigating damage on a larger scale.

Law Enforcement Agencies can find value in open source tools such as Radare2 in its dependencies and in its Security Operation Centers (SOC), which do not generate excessive expenses, technological dependency to a brand nor payments for annual licenses. Additionally, these tools allow organizations to scale up through generations until they reach the maximum level for a SOC (4th generation), as shown in figure 20

e-ISSN 2357-6014 / Vol. 15, n°. 2 / mayo 2019 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

[28]. SOCs at 4th generation include Threat Intelligence capabilities complemented with Threat Hunting, so that organizations or Law Enforcement Agencies can be more proactive than reactive, have a strategy based on the anticipation of cyberthreats and can fully comply with Colombian cybersecurity and cyber defense missions as defined in document CONPES 3701.

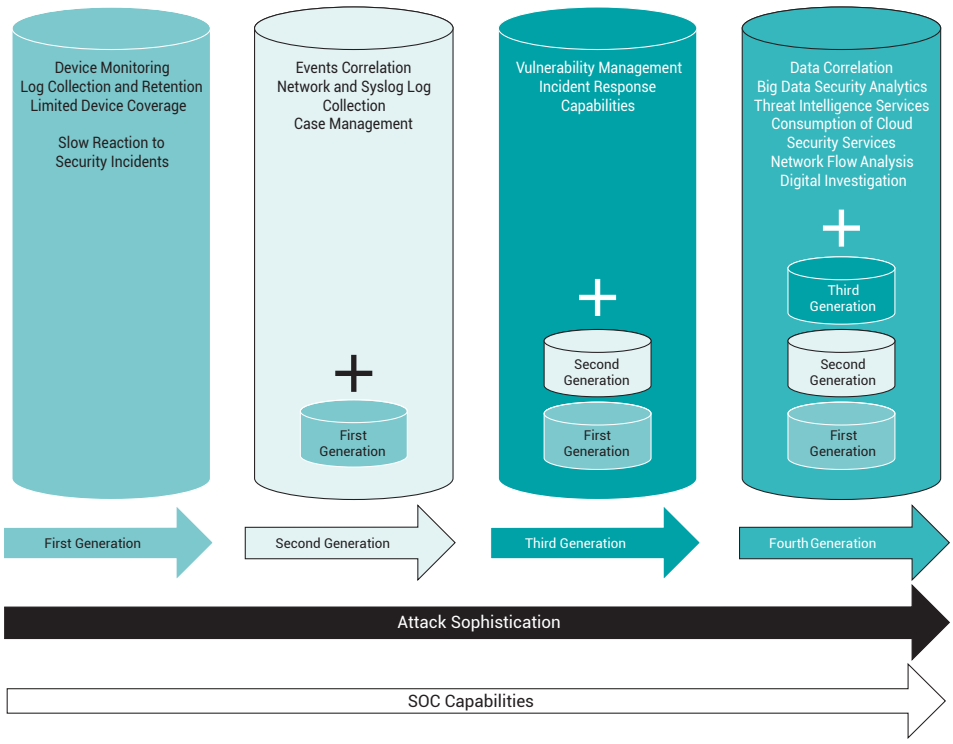

Figure 20. The 4th generations for a SOC
Source: Security Operations Center: Building, Operating, and Maintaining your SOC.

5. Conclusions

As seen throughout the development of this paper, reverse engineering can be useful to identify the real purposes of an application. In the same way, it can support cyber intelligence processes, since it helps to discover how an application is composed and also support the extraction of Indicators of Compromise. Radare is a reverse engineering tool that has gained popularity because of its open source approach but also because of its integration with some very useful modules such as Androguard,

YarGen, and R2Yara. Having said that, Radare and reverse engineering are a great complement for sandboxes and its use is highly recommended to perform static or dynamic analysis. As future work, we plan to research around the improvement in the automatic generation of Yara rules using YarGen. The automatic generation can be supported with context-aware algorithms that consider features around the assets of a specific organization like location, company name, domain names, company roles and specific company vocabulary. In this way it will be possible to generate rules related to addressed malware attacks. An improvement in the own YargGen algorithm in charge of selecting the strings present in goodware can also be required to decrease false positives. Additionally, we plan to work in the development of a mobile application that incorporates a set of automatically updated Yara rules to analyze applications that the user wishes to install. This application would require a version of Radare with Androguard so it can make more advanced reversing over the downloaded binaries.

6. Acknowledgment

This work has been supported partially by the Colombian School of Engineering Julio Garavito (Colombia) through the project "Cyber Security Architecture for Incident Management", funded by the Internal Research Opening 2017.

7. References

[1] M. Sikorski and A. Honig, "Practical Malware Analysis," vol. 53, no. 9. No Starch Press, San Francisco, pp. 650–652, 2012. doi: 10.1016/s1353-4858(12)70109-5

[2] K. Dunham, S. Hartman, J. Morales, M. Quintans, and T. Strazzere, "Android Malware And Analysis." CRC Press, p. 232, 2014.[Online]. Available: https://www.crcpress.com/Android-Malware-and-Analysis/Dunham-Hartman-Quintans-Morales-Strazzere/p/book/9781482252194 doi:10.1201/b17598

[3] J. J. Drake, Z. Lanier, C. Mulliner, P. Oliva, S. A. Ridley, and G. Wicherski, "Android hacker's handbook." John Wiley & Sons, p. 577, 2014. [Online]. Available: https://www.wiley.com/en-co/Android+Hacker%27s+Handbook-p-9781118922255

[4] Radare, "radare/radare2: unix-like reverse engineering framework and commandline tools security." [Online]. Available: https://github.com/radare/radare2.

[5] E. Eilam and E. J. Chikofsky, "Reversing: Secrets of Reverse Engineering." John Wiley & Sons, p. 624, 2011. [Online]. Available: https://www.wiley.com/en-co/Android+Hacker%27s+Handbook-p-9781118922255

[6] A. Singh, "Identifying Malicious code through Reverse Engineering," vol. 44. *Springer Science & Business Media*, p. 198, 2009. [Online]. Available: https://www.springer.com/la/book/9780387098241 doi:10.1007/978-0-387-89468-3

[7] D. Oktavianto and I. Muhardianto, "Cuckoo Malware Analysis." Packt Publishing Ltd, p. 142, 2013. [Online]. Available: https://www.packtpub.com/hardware-and-creative/cuckoo-malware-analysis

[8] C. Elisan, "Advanced Malware Analysis." McGraw Hill Professional, p. 464, 2015. [Online]. Available: https://www.mhprofessional.com/9780071819749-usa-advanced-malware-analysis-group

[9] M. Ligh, A. Case, J. Levy, and Aa. Walters, "The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory," vol. 1. John Wiley & Sons, p. 912, 2014. [Online]. Available: https://www.wiley.com/en-co/The+Art+of+Memory+Forensics%3A+Detecting+Malware+ and+Threats+in+Windows%2C+Linux%2C+and+Mac+Memory-p-9781118824993

[10] D. Regalado, S. Harris, A. Harper, C. Eagle, and J. Ness, "Gray hat hacking: the ethical hacker's handbook." McGraw Hill Professional, p. 577, 2008. [Online]. Available: https://www.mhprofessional.com/9781260108415-usa-gray-hat-hacking-the-ethical-hackers-handbook-fifth-edition-group doi: 10.1036/0071495681

[11] P. Shah, "Security Sandboxing for PC2: Windows Version," California State University, Sacramento, 2017. [Online]. Available: https://csus-dspace.calstate.edu/bitstream/handle/10211.3/190565/SecuritySandboxingForPC2WindowsVersion.pdf?sequence=1

[12] C. Eagle, "The IDA Pro Book." No Starch Press, p. 672, 2011. [Online]. Available: https://nostarch.com/idapro2.htm

[13] Aptoide S.A, *Aptoide | Descarga, encuentra y comparte los mejores juegos y apps para Android.* [Online]. Available: https://es.aptoide.com/.

[14] Klinnerds, "World Cup 2018 Yeah! - Russia 2018 2.2.3 Descargar APK para Android - Aptoide." [Online]. Available: https://world-cup-2018-yeah-russia-2018.es.aptoide.com/

[15] J. Morris, "Hands-On Android UI Development: Design and develop attractive user interfaces for Android applications." Packt Publishing Ltd, p. 348, 2017. [Online]. Available: https://www.packtpub.com/application-development/hands-android-ui-development

[16] N. Elenkov, "Android Security Internals: An In-Depth Guide to Android's Security Architecture." No Starch Press, p. 432, 2014. [Online]. Available: https://nostarch.com/androidsecurity

[17] A. Dubkey and A. Misra, "Android Security: Attacks and Defenses." CRC Press, p. 280, 2016. [Online]. Available: https://www.crcpress.com/Android-Security-Attacks-and-Defenses/Misra-Dubey/p/book/9781439896471

[18] K. Dunham, "Mobile Malware Attacks and Defense." Syngress, p. 440, 2008. [Online]. Available: https://cdn.sonicwall.com/sonicwall.com/media/pdfs/resources/2018-snwl-cyber-threat-report.pdf

[19] K. Mandia, C. Prosise, and M. Pepe, "Incident Response & Computer Forensics." McGraw Hill Professional, p. 624, 2014. [Online]. Available: https://www.mhprofessional.com/9780071798686-usa-incident-response-computer-forensics-third-edition-group

[20] M. Christodorescu, S. Jha, C. Wang, D. Song, and D. Maughan, "Malware Detection." Springer Science & Business Media, p. 312, 2007. [Online]. Available: https://www.springer.com/la/book/9780387327204 doi: 10.1007/978-0-387-44599-1

[21] V. Total, "YARA – VirusTotal." [Online]. Available: https://support.virustotal.com/hc/en-us/articles/115002178945-YARA.

[22] D. Balzarotti, M. Cova, and S. Stolfo, "Research in Attacks, Intrusions, and Defenses," vol. 7462. Springer, p. 400, 2012. doi: 10.1007/978-3-642-33338-5

[23] M. Spreitzenbarth and J. Uhrmann, "Mastering Python Forensics," vol. 21. Packt Publishing Ltd, p. 192, 2015. [Online]. Available: https://www.packtpub.com/networking-and-servers/mastering-python-forensics

[24] J. Six, "Application Security for the Android Platform." O'Reilly Media, p. 97, 2011. [Online]. Available: http://shop.oreilly.com/product/0636920022596.do

[25] M. Goodman, "Future Crimes: Everything Is Connected, Everyone Is Vulnerable and What We Can Do About It." Knopf Doubleday Publishing Group, p. 10100, 2015. [Online]. Available: http://www.futurecrimesbook.com/

[26] T. Intelligence and I. Analysis, "2018 SonicWall Cyber Threat Report," 2018. [Online]. Available: https://cdn.sonicwall.com/sonicwall.com/media/pdfs/resources/2018-snwl-cyber-threat-report.pdf

[27] C. Abad-Aramburu, "Aplicación de metodología de Análisis de Malware al caso de estudio de la Amenaza Avanzada Persistente (APT) 'Octubre Rojo.'" España, p. 2, 2015. [Online]. Available: http://reunir.unir.net/handle/123456789/2841

[28] J. Muniz, G. McIntyre, and N. AlFardan, "Security Operations Center: Building, Operating, and MaintainingyourSOC," vol. 2. CiscoPress, p. 21, 2015. [Online]. Available: http://www.ciscopress.com/store/security-operations-center-building-operating-and-maintaining-9780134052014