applied sciences

MDPI

Article

Systematic Approach to Malware Analysis (SAMA)

Javier Bermejo Higuera [1], Carlos Abad Aramburu [1], Juan-Ramón Bermejo Higuera [1], Miguel Angel Sicilia Urban [2] and Juan Antonio Sicilia Montalvo [1,*]

[1] Escuela Superior de Ingeniería y Tecnología, Universidad Internacional de La Rioja, Avda. de La Paz 137, 26006 Logroño, La Rioja, Spain; javier.bermejo@unir.net (J.B.H.); carlosaa@gmail.com (C.A.A.); juanramon.bermejo@unir.net (J.-R.B.H.)

[2] Departamento de Ciencias de la Computación de la Escuela Politécnica Superior de la Universidad de Alcalá de Henares, Ctra. Madrid-Barcelona, Km. 33,600 Alcalá de Henares (Madrid), Spain; msicilia@uah.es

* Correspondence: juanantonio.sicilia@unir.net

Received: 3 January 2020; Accepted: 13 February 2020; Published: 17 February 2020

Featured Application: The systematic and methodological process of analysis described in this document will provide a complete understanding of the life cycle of a malware specimen in terms of its behavior, operation, interaction with the environment, methods of concealment and obfuscation, system updates, and communications.

Abstract: Malware threats pose new challenges to analytic and reverse engineering tasks. It is needed for a systematic approach to that analysis, in an attempt to fully uncover their underlying attack vectors and techniques and find commonalities between them. In this paper, a method of malware analysis is described, together with a report of its application to the case of Flame and Red October. The method has also been used by different analysts to analyze other malware threats like 'Stuxnet', 'Dark Comet', 'Poison Ivy', 'Locky', 'Careto', and 'Sofacy Carberp'. The method presented in this work is a systematic and methodological process of analysis, whose main objective is the acquisition of knowledge as well as to gain a full understanding of a particular malware. Using the proposed method to analyze two well-known malware as 'Flame' and 'Red October' will help to understand the added value of the method.

Keywords: malware analysis; malware sample; Flame; Red October; sandbox; behavioral analysis; code analysis

1. Introduction

'Stuxnet' was the first malware that received worldwide attention for being able of causing physical damage in an industrial infrastructure. However, there are also a significant number of other known malware that can be grouped together as a recent and particular phenomenon known as Advanced Persistent Threats (APT), that deserves separate attention from other kinds of malware. 'Flame', as a part of this group, belongs to a family of complex malware that also includes 'Stuxnet', 'Duqu', and 'Gauss'. It was intended to attack particular types of networks and infrastructures using different techniques to persist during long periods of time [1]. On the other hand, another complex malware that was deployed mainly in Eastern European and Central Asian countries was 'Red October'. It was intended to attack government organizations and diplomatic services. Its operation focuses on information exfiltration in order to improve the intelligence of attackers.

WannaCry, Locky, etc. are other malware, called 'ransomware' and developed to steal money. Today there are many specimens of malware, as mentioned above, that use multiple attack techniques and vectors and are conducted stealthily to avoid detection, in order to retain unnoticed control over targeted systems for long periods of time [2]. On the other hand, in recent decades with the new

educational technologies and the growth of the Massive Open Online Course (MOOCs) [3–8] there are new opportunities for this kind of malware to 'catch' new objectives and personal data which can be really important.

In this paper, we describe a new systematic approach or methodology of malware analysis and its evaluation (case studies) by applying it to two of the most complex and well-known malwares that have emerged so far: Red October and Flame. It provides a structured analysis process and the definition of the steps to be followed, the objectives to be achieved and the specific techniques and tools to be used. It proposes a four stages implementation in order to that will help the analyst to work securely to produce a repeatable and objective result, to cooperate with other analysts. In the end, using this method will provide a better understanding of the analyzed malware without forgetting the importance of using the right information in the right moment enhancing the analyst effectiveness.

Regarding the toolset that is needed for any malware analysis, it must be confirmed that the proposed method is independent of any specific tool or technic allowing an easy and cost-effective adaptation to malware analyst resources. Main characteristics of the presented method to analyze malware are as follows:

- It does not depend on the complexity of the malware to be analyze
- It is independent of the tools used making easier its adoption
- In enhances the importance of the malware classification as one of the first steps to be done
- It considers the integration of any existing information of the malware as an input for the analysis
- It defines a specific sequence for the analysis based on making the analysis of code, before the dynamic or behavioral analysis
- It includes a systematic feedback loop to retake the process when malware complexity requires to do it, as well as to improve the current analysis feeding previous stages with the obtained results of later ones

The rest of this paper is structured as follows. Section 2 surveys our current knowledge on the types of malware analysis techniques and available tools. Then, Sections 3 and 4 discuss the design of a methodological framework to integrate such techniques in a coherent and systematic way. The detailed results of applying that method to the analysis of the malware Red October and Flame are provided in Section 5. Finally, conclusions and future research guidelines are included in Section 6.

2. Background

Today, malicious code, hereinafter referred to as malware, has evolved into one of the most important dangers and threats that affects the security of information technology (IT) systems. The degree of the technique complexity and the level of knowledge needed to analyze malware is proportional to the level of sophistication of the sample. The main benefits of malware analysis are the following [2,9]:

- To assess the malware detection capability of organizations' protection systems
- To evaluate the damage caused by the intrusion and malware actions
- To discover other machines that have been affected by the same malware
- To identify the vulnerability that was exploited by the malware, to obtain the software update that mitigates it, if available
- To obtain enough data and information to implement the appropriate defense mechanisms to mitigate and neutralize the damage caused by the malware, including firewall rules, host and network intrusion detection systems, firewall, and antivirus
- To determine the level of malware sophistication and complexity
- If it is possible, to determine the origin of an attack and to identify the intruder and malware developer

In the analysis of malware, a series of methods and techniques are required for any analysis. They will help analysts to understand the risks, threats, and intentions associated with the malware. Thus, the obtained information may be used to reconfigure the organization's defenses in order to detect it and being protected against it.

The first and oldest analysis method used in analysis of malware is observation. Jiang, in his doctoral thesis [10], use this method to compare the states of a system before and after verified malware activity. Data is achieved by the mere observation of a machine for clues modifications of registers, files or creation of new ones as, for example, 'enbiei.exe' activity which indicates 'Blaster Worm' malware. Theerthagiri, in the article "Reverting Malware, Confirms the Following: A Detection Intelligence with In-Depth Security Analysis" [11], provides a review of the most relevant malware analysis techniques. Specifically, three specific and differentiated lines of work are identified, such as the following:

- Static Code Analysis: technique that collects as much information as possible of a binary without actually executing it. To do it, disassembly and reverse engineering techniques are used. This latter technique of analysis includes others more specific as string analysis, reverse engineering, packaging, obfuscation, restricted execution environments, etc. A tool to perform this type of analysis is IDA PRO [12] or GHIDRA [13].
- Dynamic code analysis: the analysis is basically done through a type of tools called 'debuggers' that are associated with the program code under analysis to take full control of them, allowing the possibility of executing it code line by code line. Some popular debuggers for malware analysis are OllyDbg [14], ImmunityDbg [15], and WinDbg [16].
- Dynamic or behavioral analysis: It analyzes the binary or specimen malware while it is running focusing on the supervision and monitorization of the malware interaction with the environment. The aim is to look for changes in the infected machine where it is running, network traffic and potential external communications with command and control servers. It can be done manually in a virtual or physical environment using different tools that collect the data of its interaction with the machine infected or automatically by "Sandbox" toolset [17].

The Figure 1 summarizes the main methods and techniques for the analysis of malware that we have identified as a framework. As it has been mentioned in the previous paragraphs, all of them present a series of advantages and disadvantages that mean that malware analysis cannot be carried out using a single method or technique.

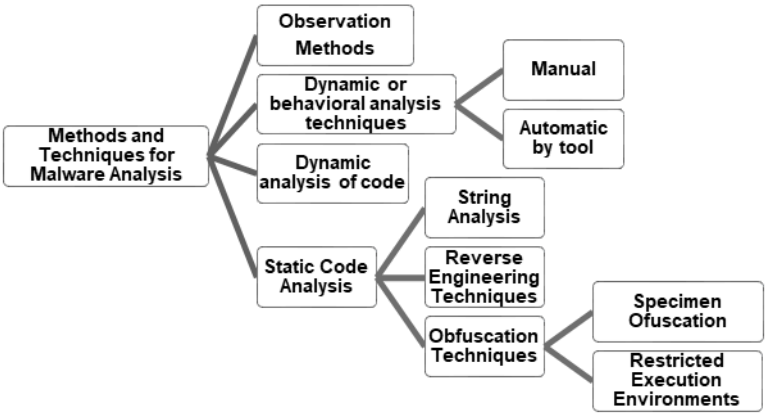

Figure 1. Methods and techniques for malware analysis.

Once the different types and methods of malware analysis have been shown, the following paragraphs present various considerations reflected in different scientific research articles.

Malware is constantly evolving by developing obfuscation techniques using metamorphic and polymorphic techniques to frustrate static code analysis [2]. Moser et al. [18] proposed an obfuscation approach that demonstrates that static analysis techniques alone might no longer be sufficient to analyze

complex specimen of malware. Also, in [19], the "Problems of Static Malware Analysis Approaches", they stated that sometimes the source code of malware samples is not readily available. As the analysis of binaries is complex and it requires high knowledge of reverse engineering, it is probable that the disassembly of binaries of the malware presents ambiguous results. Given the problems indicated for this type of analysis, this article demonstrates that it must be complemented with other types of analysis in order to maximize its effectiveness.

Derivative of the cost of resources and time of a manual analysis, some methodologies to automate the process have been developed, such as the one that is presented in this paper. Its implementation is done via virtual and isolated run environments known as "Sandbox" [11]. Basically, it is a supervised and controlled execution environment of malicious code so that it cannot do any harm to the real system that hosts the virtual environment. It is usually implemented through a virtual environment where the processor, memory, and file system are simulated.

Given the complexity of carrying out manual analysis, the need to automate analysis arises, in principle, due to the high cost in human resources and time. However, this automation can only be carried out, nowadays, in behavioral analysis using virtualization environments. This does not constitute a global solution of malware analysis. Nevertheless, it can be used as an element of support, taking into consideration some inconveniences that are presented below.

The main purpose of an automated dynamic malware analysis [20] is to support the analysts to understand the malware behavior and to obtain data, effectively, in order to facilitate reverse engineering to design effective countermeasures.

Furthermore, in the [9], it is indicated that using 'Sandbox' presents important drawbacks, for example, the inconvenience of running the malware and lose information due to the lack of registering all events because its complexity. To address this limitation, it is always necessary to perform dynamic code analysis.

There are also different types of tools that perform this type of analysis (some for free and some for a fee), like CWSandbox [21], Norman SandBox [22], Cuckoo [23], TTAnalyzer [24], Anlyz [25], Malwr [26] y ThreatExpert [27], or Cobra [19]. In [19], a table with a detailed analysis of the different Sandbox solutions is included.

All of them record the actions taken by the malware under analysis automatically, but they have the drawback of analyzing only a single execution path, so they could ignore relevant behavior of the malware. (i.e., malware logic behavior under certain conditions like system date). According to Gandotra et al. [2], "The virtual environment in which malwares are executed is different from the real one and the malwares may perform in different ways resulting in artificial behavior rather than the exact one".

Another problem that has to be faced using this type of tools and performing dynamic analysis and behavioral techniques, are the methods of malware evasion. It means that, in order to prevent the analysis of the malware, it can detect the presence of an instrumented environment of analysis forcing malware to hide or inhibit its malicious behavior [19]. It is possible to use the Pafish [28] tool that allows the analyst to test if the environment for the analysis is properly implemented, thanks to its capability to detect sandboxes and analysis environments in the same way as malware do.

An additional important aspect of a malware analysis methodology is to identify the tools that are used in the different types of malware analysis. As it has been said before, there are plenty of tools both free and paid available.

So as to create an effective laboratory for malware analysis, a study of these tools is necessary. It means that depending on the machines that will be used and the type of analysis that will be performed, a specific toolset must be chosen. The specific toolset shown in this paper is a result of a study based on the needs of each phase of the methodology and their functionality taking into consideration the ease of use and the capabilities they provide. The Table 1 shows a classification of tools based on its functionality.

Table 1. Malware analysis tools

Malware Analysis Tool
Identification and classification of binary
Antivirus engines
String analysis
Analysis of changes in host machine
Traffic analysis and simulation of network services
Analysis disk images
Memory analysis
Analysis of binary files
Dynamic code analysis
Static code analysis

It was found quite complex to identify a toolkit which could be seen as a standard. The wide variety of tools, techniques and methods for analyzing malware, the disadvantages of the dynamic analysis automatization tools, sandbox, and the increasing complexity of malware have pushed the development of many tools. This situation forced us to study around one hundred of tools in order to select the most adequate ones to be used in the in the present article.

It has been confirmed that having a method independent of technical tools may be useful to adapt it to the complexity that different types of malware may present [29]. Beyond technical tools or individual types of analysis, it seems that having a methodology that allows us to respond systematically to the complexity that malware presents today could be necessary. It is therefore necessary to develop systematic and methodological analysis process in order to provide a set of procedures, methods, and techniques with their associated tools to help the analyst to gain an understanding and complete information malware under study. In this way, by applying this methodological process, "systematize the analysis of malware" could be possible as well as the contribution to enhance the prevention by identifying malware thanks to the results obtained after applying the methodology.

Without a high level structured approach, it is known that it is not only difficult to know which tool must be used, but also when, where an in which order should be used. That is why the need of a methodology is justified, because it provides the basis for conducting a systematic and methodological analysis process for malware analysis.

In addition, establishing this methodology can allow the distribution of the malware analysis phases among different groups of analysts, laboratories or computer, each one with its own strengths, knowledge, and specific resources for the analysis. Using this methodology could also allow the sharing of the result with other groups of analysts able to understand it (if they know the methodology that was used).

3. Related Work

At present, we have identified only one malware analysis methodology [30]. It presents a comprehensive methodology for malware analysis that includes a structured analysis process. It also defines the steps to follow to achieve the objectives and the techniques and tools used in each phase.

It proposes four logical phases which will help the analyst to produce a repeatable and objective output in order to obtain a better understanding of the analyzed malware. These phases are as follows:

1. Detection
2. Isolation and extraction
3. Behavioral analysis
4. Code analysis

However, we do not consider it very appropriate, since there are other procedures for obtaining malware that are more effective because the malware has already been compressed or it can even be obtained from another organization that has already extracted it with its own forensic procedures

assuring always the chain of custody. It also proposes its application to the judicial sector, which is only a particular case of the one described above.

This methodology includes the 'detection' phase first. This implies high complexity due to the need of very different ways to get the malware including complex and expensive defense systems such as antivirus, intrusion detection systems (IDS), firewall, or security information and event management (SIEM). It means that the difficulty to systematize and standardize is very high, so it is the inclusion of this phase.

The 'isolation and extraction' phase is focus on the rootkit malware extraction process. Nowadays, more dangerous and complex malware such as Advanced Persistent Threat (APT) or Ransomware, do not work in the same way that old fashion malwares making this phase invalid for these types of specimen. The conclusion is that this phase could not respond adequately to the complexity of the current malware. The main goal of these tasks is the collection, analysis, and preservation of data from an IT machine as part of forensic processes, to make it admissible in a court of law [31,32].

For the reasons mentioned above, it is considered not necessary to include the two previous phases in the methodology set out in this article. Apart from the aforementioned, a number of important differences are also identified:

- One of the main differences with the methodology proposed in this paper is the execution order of the phases, performing the 'code analysis' phase first, before the 'dynamic or behavioral analysis' phase. That is why the information gathered during 'code analysis' is usually very useful for the following phase as passwords, installation commands, orders of command and control, possible execution paths, etc.
- MARE does not include the technique of 'dynamic analysis of code' by means of a debugger. This technique is very important to understand how malware works.
- It has not been published its application to a specific case study of a malware in order to test it and be validated. However, the methodology advanced has been applied to eight cases, two of them being included in this article.

The Figure 2 represents MARE and SAMA comparison with the differences indicated in previous paragraphs.

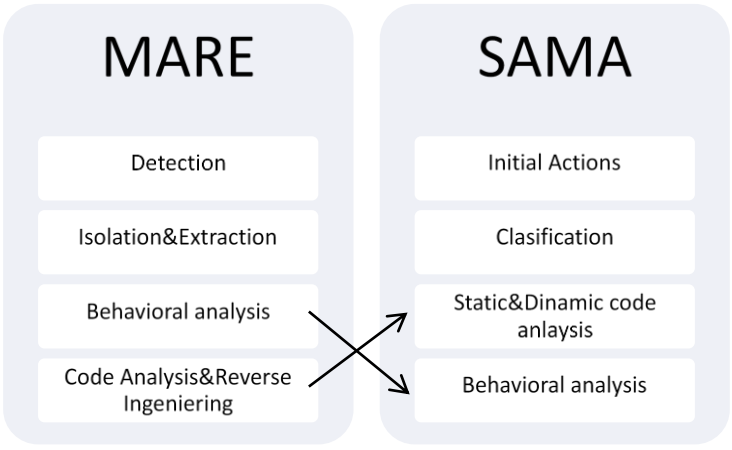

Figure 2. Differences between the methodologies MARE and SAMA.

4. Method Proposal

The aim of the proposed method is to provide an effective guidance during any detailed analysis of complex malware. The outcome of such a method will be valuable information that will allow the analyst to develop effective countermeasures as well as to extend our knowledge about the origin of malware and attack vectors used during the campaign.

Conducting a systematic, repeatable, and methodological process during the analysis is very important in order to facilitate the acquisition of knowledge about a particular malware. Then the main objective is to analyze and get a full understanding of it, in terms of its operation, its identification and the ways of removing the threat. The objectives and specific information to be obtained in the process of applying a data analysis methodology are listed in the Table 2.

Table 2. Objectives malware analysis methodology. Adapted from REMUX Lenny Zeltser distribution.

Objectives Malware Analysis	
Analysis Summary	• Key observations • Recommendations • Limitations • Date report and authors
Classification	• Filename • File size • File type • Hash file • Identifications antivirus
Features	• Infection capabilities • Capacity for self-preservation. • Diffusion mechanisms • Ability to data leakage • Iteration with the remote attacker
Dependencies	• Supported operating systems • Libs required • Configuration files • Script and executable files • URL's
Conclusions code analysis and behavior	• Behavior analysis • Static code analysis • Dynamic code analysis • Analysis of memory
Support Elements	• Logs • Screenshots • Function lists • String
Incident Recommendations	• Indicators • Steps to eradicate

The main structure of the methodology with its main phases is depicted in the following figure with the workflow of its application.

As it has been highlighted before, the 'code analysis' phase is performed before the 'behavioral analysis' since the data obtained in the first step can be used to improve the second, obtaining more and better data.

SAMA methodology includes a feedback loop between the final stage of 'behavioral analysis' phase and 'classification phase'. This feature would be useful when malware under analysis uses more than one files or specimens when it is running. General flow chart of the methodology and its associated processes are as follows on Figure 3:

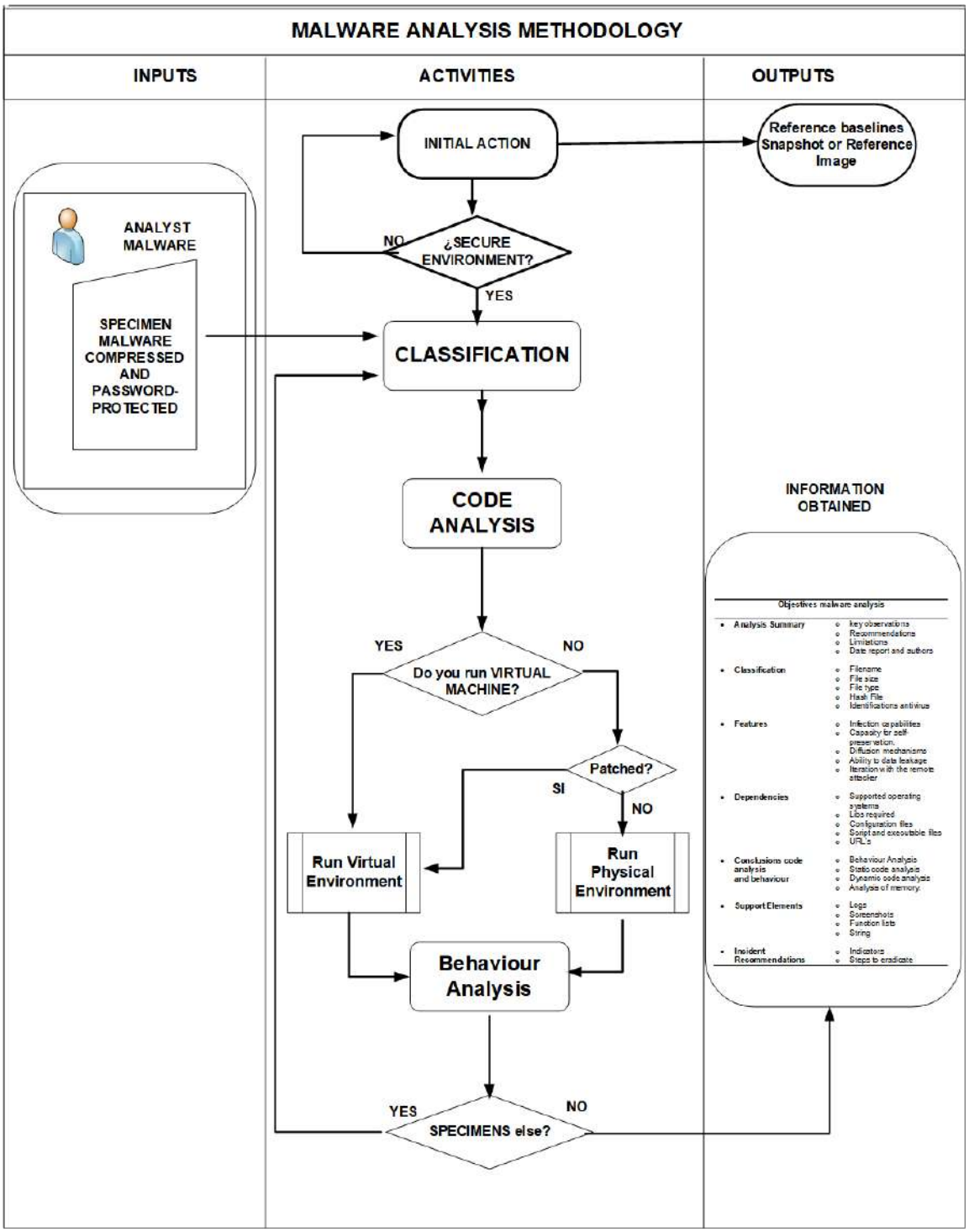

Figure 3. Diagram of the malware analysis methodology.

The process begins with a series of actions with the aim of starting the analysis of malware in a clean state without any possible infection. Next, in the 'classification' stage, the malware compressed and protected by password is the input. During this phase the code analysis is performed.

In the last stage, 'behavioral analysis', malware security measures to avoid its execution on a virtual environment are checked. To do so, using IDA Pro tool the magic code of VMware can be searched. If it is founded, the laboratory must be patched to allow running the analysis in the virtual environment. If it fails, the last option is to use physical environment to do the analysis.

Subsequently, the dynamic behavioral analysis is performed. If the malware is complex, it could generate more specimens during its execution, forcing the analyst to go back to the 'classification' for each of the new samples or specimens created. It means that depending on the complexity of the malware, SAMA could be applied at least one time for each piece of malware generated.

Once the full process is finished, all the information listed in Table 2 will have been provided. The phases or steps proposed in this methodology are detailed below:

Table 2 should be moved here, is it fine with you?

- Initial Actions. It consists of performing a series of actions mainly to obtain a record of the configuration of the machines involved in the analysis, with the purpose of obtaining a reference enabling us to compare the state of the same before and after running the malware under study. More specifically, in the next flow chart of the steps, it is as follows on Figure 4:
  - Preserving integrity by deactivating services: System recovery and automatic updates
  - Preservation of integrity by baseline generation of system configuration: snapshot (VMWare) of victim's virtual machines, monitor and services; snapshot units C: (Systracer)
  - Guarantee of the integrity of the generated samples: MD5 hash of the snapshot of the C units: by means of the tools "Md5sum" and verification with "WinMD5".

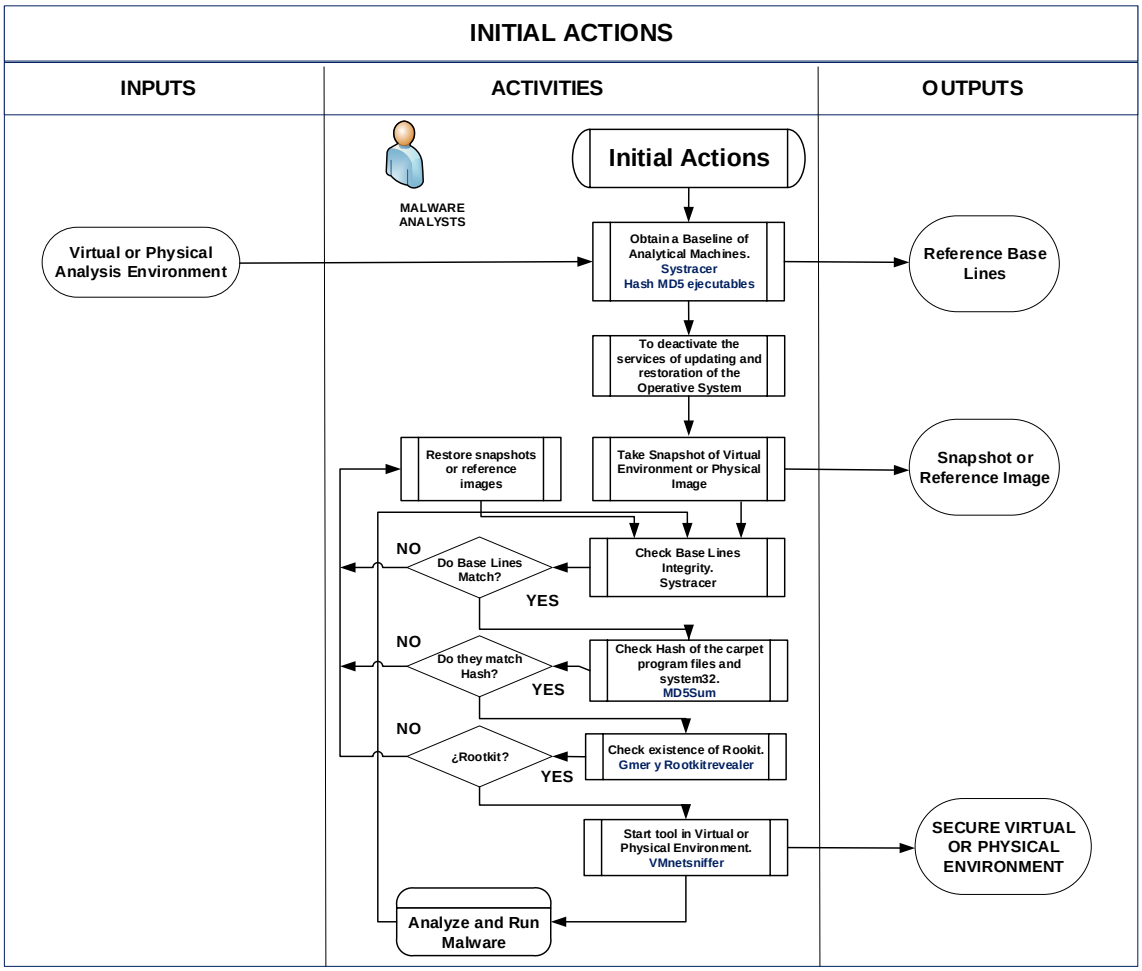

Figure 4. Diagram of the initial actions step.

The tools selected to these phases are Systracer, MD5Sum, and Gmer.

- Classification. It is to examine the malware executable file, without accessing your code, to confirm it is some kind of malware and to obtain information in open sources, like the Internet,

of its functionality that aid to realization of the following phases of analysis. It even enables the realization simple network firms, should be necessary for reasons of urgency in protecting systems. The tasks executed in this phase are:

- Transfer the malware
- Identify malware
- Check family of malware
- Finding information about the malware in open source
- Analysis of strings of malware binary
- Analyze techniques used in malware obfuscation
- Analyze file format

More specifically, in the next flow chart, the steps are as follows on Figure 5:

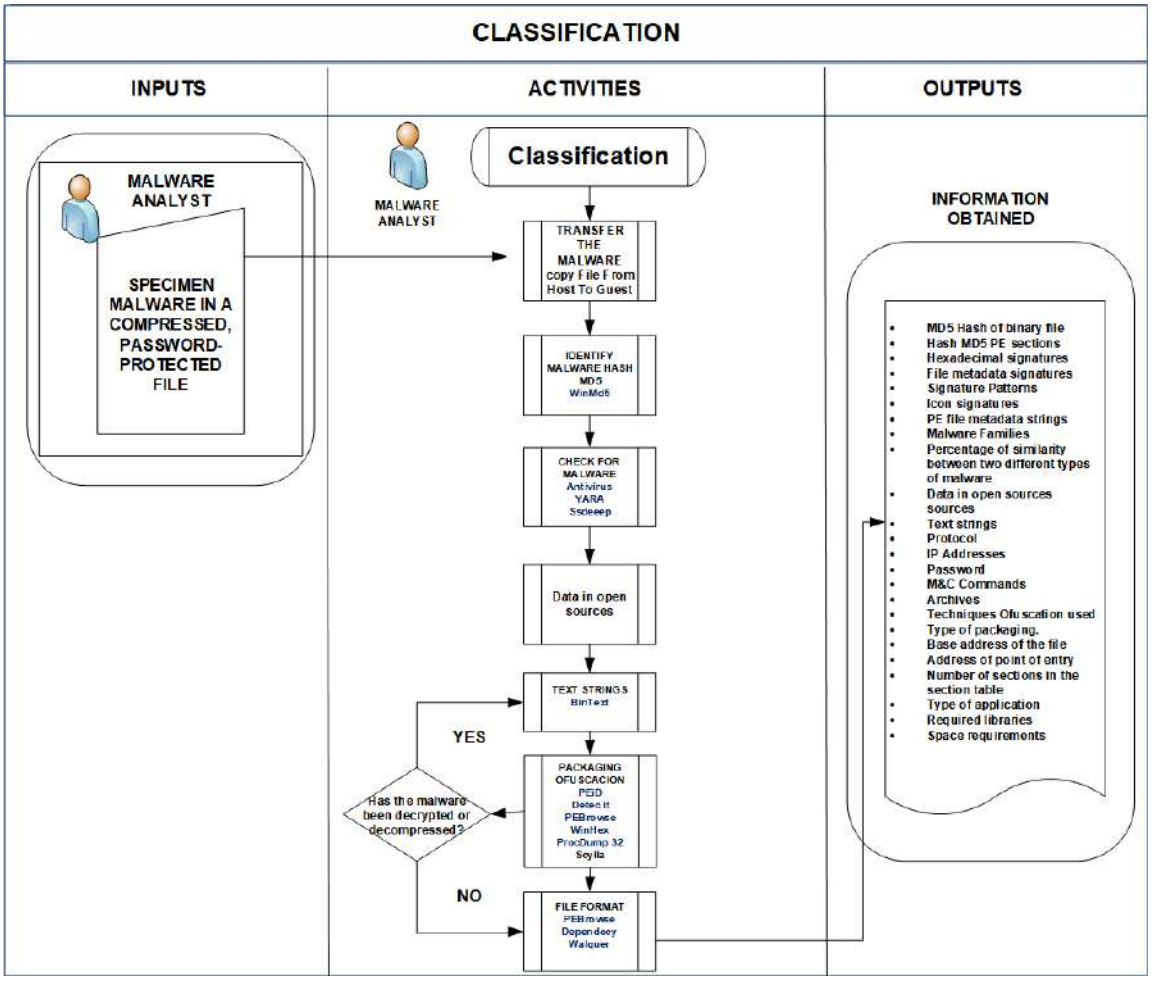

Figure 5. Diagram of the classification step.

The objectives and specific information to be obtained in this step will be shown in the Table 3.

Table 3. Information to obtain in classification step.

Information to Obtain in Classification Step		
MD5 Hash of binary file	Text strings	Number of sections in the section table
Hash MD5 PE sections	Protocol	Type of application
Hexadecimal signatures	IP Addresses	Required libraries
File metadata signatures	Password	Space requirements
Signature Patterns	M & C Commands	
Icon signatures	Archives	
PE file metadata strings	Techniques obfuscation used	
Malware families	Type of packaging	
Percentage of similarity between two different types of malware	Base address of the file	
Data in open sources	Address of point of entry	

In case of the malware being obfuscated, the process to follow would be (see Figure 6):

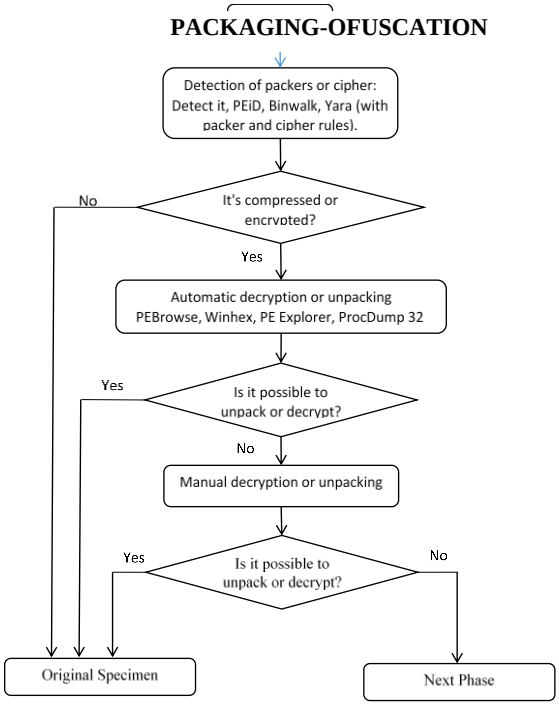

Figure 6. Unpacking and decryption process.

The tools selected to these phases are Winmd5, Ssdeep, Bintext, String, PEiD, Detect It, Scylla, Yara (with packer and cipher rules), Binwalk, PEBrowse, Winhex, PE Explorer, ProcDump 32, and Dependecy Walquer.

- Code Analysis. It consists of a static and dynamic analysis of the malware's code, in assembly language, performing a detailed browsing process through it in order to get a better understanding of its functionality. It is a complex process that provides information for carrying out the next phase. It requires a reverse engineering analyst to find hidden features that involve new execution paths, which could be tried by the malware during the next phase to interact with the user. These discoveries will help to understand better its purpose and its functionality. In the next flow chart, the detail of the analysis steps is shown on Figure 7:

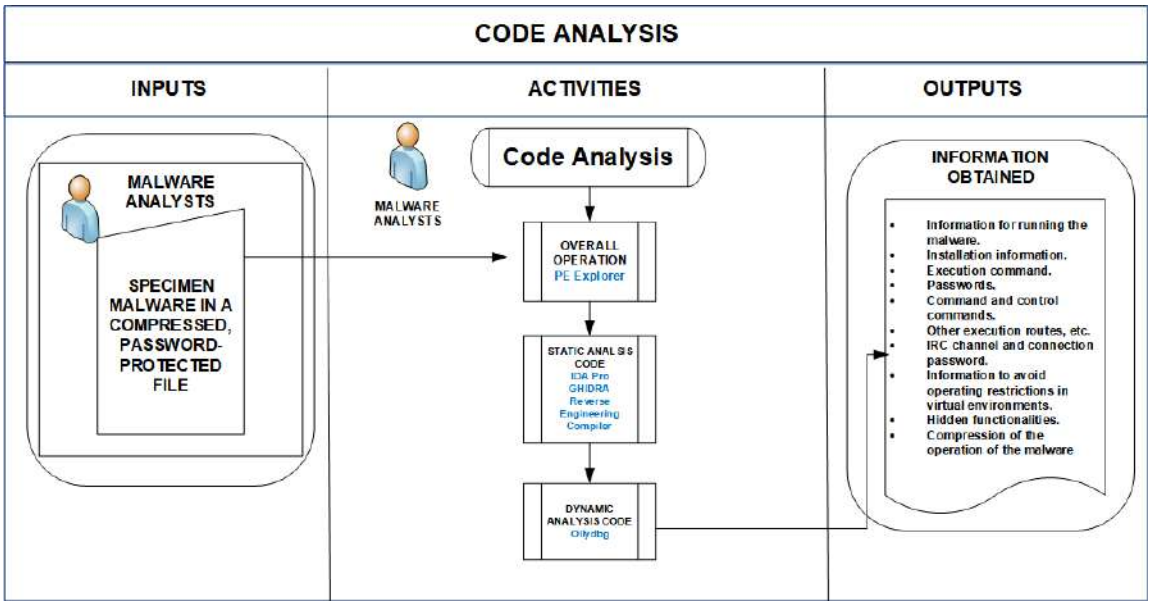

Figure 7. Diagram of the code analysis step.

After finishing the dynamic code analysis stage, the system will be returned to its original uninfected state because using a debugger means the execution of malicious code in the victim machine.

The objectives and specific information to be obtained in this step will be shown in the Table 4:

Table 4. Information to obtain in classification step.

Information to Obtain in Code Analysis Step
Information for running the malware.
Installation information.
Execution command.
Passwords.
Command and control commands.
Other execution routes, etc.
IRC channel and connection password.
Information to avoid operating restrictions in virtual environments.
Hidden functionalities.
Compression of the operation of the malware.

The tools selected to perform this phase are PE Explorer, IDA Pro, Ghidra, Reverse Engineering Compiler, and Ollydbg.

- Behavioral Analysis (See Figure 8). It consists in analyzing the implementation of malware in a lab that simulates a real environment, in order to observe malware behavior, network traffic generated and thus, the actions that are taken on the target system (registry modifications, files, etc.).

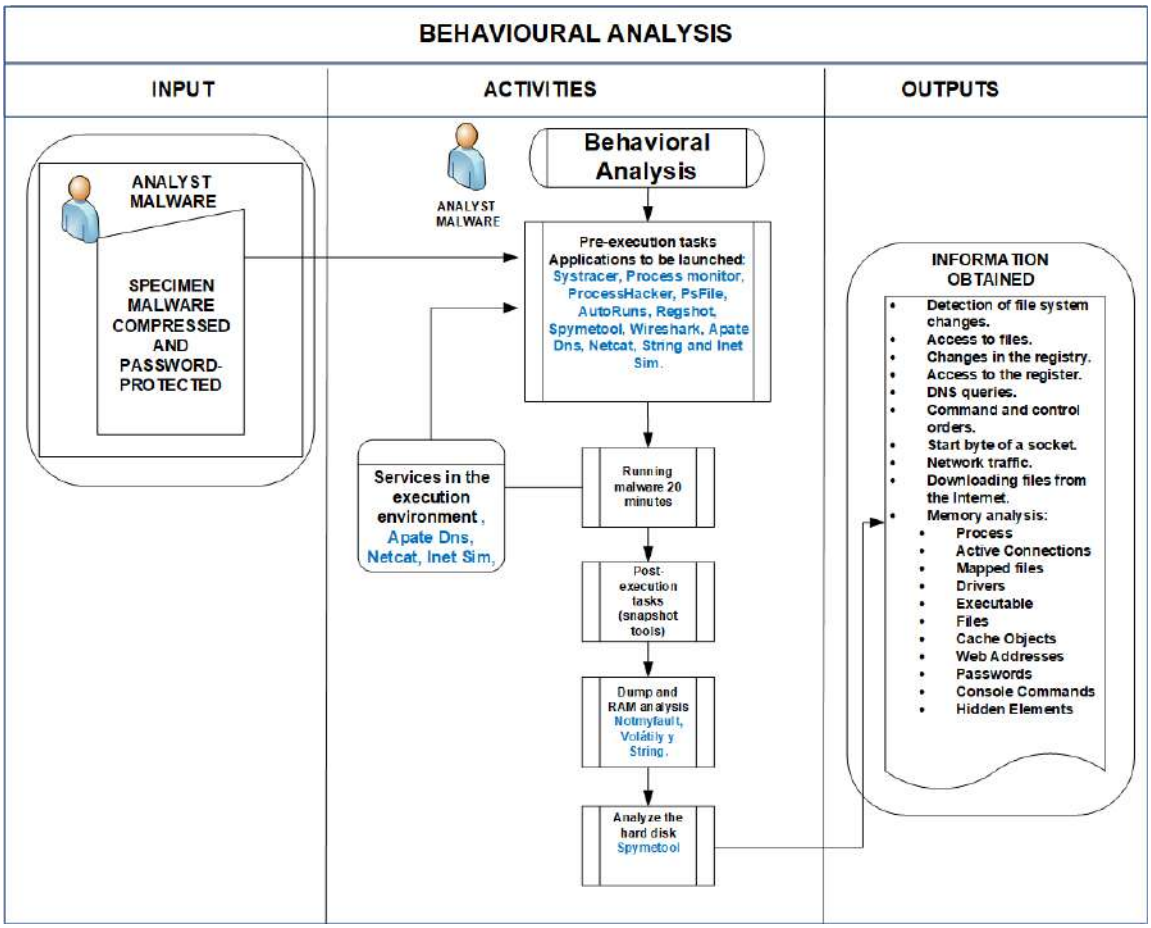

Figure 8. Diagram of the behavioral analysis step.

The objectives and specific information to be obtained in this step will be shown in the Table 5.

Table 5. Information to obtain in behavioral analysis step

Information to obtain in Behavioral Analysis Step	
Detection of file system changes	Memory analysis:
Access to files	• Process
Changes in the registry	• Active Connections
Access to the register	• Mapped files
DNS queries	• Drivers
Command and control orders	• Executable
Start byte of a socket	• Files
Network traffic	• Cache Objects
Downloading files from the Internet	• Web Addresses • Text strings • Passwords

The tools selected to this phase are Systracer, Process monitor, ProcessHacker, PsFile, AutoRuns, TCPview, Regshot, Spymetool, Wireshark, Apate Dns, Netcat NotMyfault, Volatility, String, and Inet Sim.

This phase is divided in five stages. First, one consists in taking a snapshot of the system to compare their condition before and after the execution of the malware, using specific tools. This helps to identify files that have been added or modified in the system and to understand what changes have occurred after the malware execution.

In this phase, it is needed to analyze the network traffic between machines in order to detect the interaction between the malware and command and control trough a simulated channel. After launching the different malware functionalities, all the captured traffic will allow the analyst to perform a more detailed analysis of the malware.

Added to the other analyses it is the RAM memory check that could be done with the tools volatility or memorize as follows on Figure 9:

Figure 9. Memory analysis.

The proposed methodology is intended as a basis for conducting a systematic and methodological process of analyzing complex and flexible malware that may require slight variations given the large number of existing types of malicious codes and the used technologies in its development, which may apply to specific or granular parts of the code. However, this situation does not change the general structure thereof.

5. Study Cases

The research shown in this article corresponds to the category 'experimental development' (systematic work based on existing knowledge, derived from research and/or practical experience, aimed at the production of new materials, products, devices, processes, systems and services, or the improvement of existing ones) [33], since it is a scientific work oriented to obtain a new methodology or systematic process of malware analysis.

Likewise, taking as a reference the article "Research Methods in Computer Science", by Serge Demeyer [34], and taking into account that the article falls within the field of Computer Science, the research method used is the dominant one within this type of science, are the so-called 'case studies': useful to "investigate a contemporary phenomenon within its real context, when the boundaries between the phenomenon and the context are not clearly evident [35].

This is why two case studies of the application of the malware analysis methodology proposed to the Red October and Flame malware have been included in the article. The first one is presented in a more complete way to reflect in detail the application of the methodology and a second one more focused on the presentation of the result of the analysis.

The reason for choosing these two malware has been because they are considered the two most complex to date, one of Russian and one of Anglo-Saxon descent. This requires an organizational approach to allocating resources in order to be effective in their use. The proposed methodology provides a systematic process of analysis and reverse engineering of complex pieces of malware, using techniques and methods that include, for example, the re-engineering of a complete malware lifecycle analysis. This means that analysis of its behavior; stealth methods, obfuscation, updates, and communications are included.

The validity of the applicability of the methodology used as a reference is confirmed by consistency between the results obtained using it and the ones obtained from expert sources [36,37] respectively, demonstrating its success.

5.1. Red October Analysis

This section presents the results and experience gained from the application of the methodology presented in this article, to the analysis of the malware Red October.

The main objective of this malware is the theft of classified information on systems with Windows OS and external devices like BlackBerry and using spear-phishing campaigns to spread the malware.

Given its complexity, it has been considered to be developed by a skillful and resourceful organization, which fall within the category of 'Advanced Persistent Threats'.

The following objectives were pursued:

- To test the practical applicability of the presented methodology
- Identify future areas of development, to guarantee the evolution and adaptability of the methodology to the evolution of malware
- Reinforce the recognition of the importance of the development and evolution of methodologies that can effectively counteract the development of malware and serve to generate transferable knowledge
- To broaden the existing knowledge base on the evolution of malware development, a clear end to the design and use of malware analysis methodologies

First, the laboratory environment is prepared for malware analysis Red October. The complete laboratory is based on the article "Malware Analysis. Environment Designs" by A. Sanabria [38], adapting it to the size of the experiment.

After defining the laboratory and the malware analysis tools, it is identified that the methodology is independent of both technological aspects, allowing each analyst or team of analysts to adapt the methodology to the real needs and existing capabilities. The architecture of the working environment defined for testing the methodology is presented below in the Figure 10:

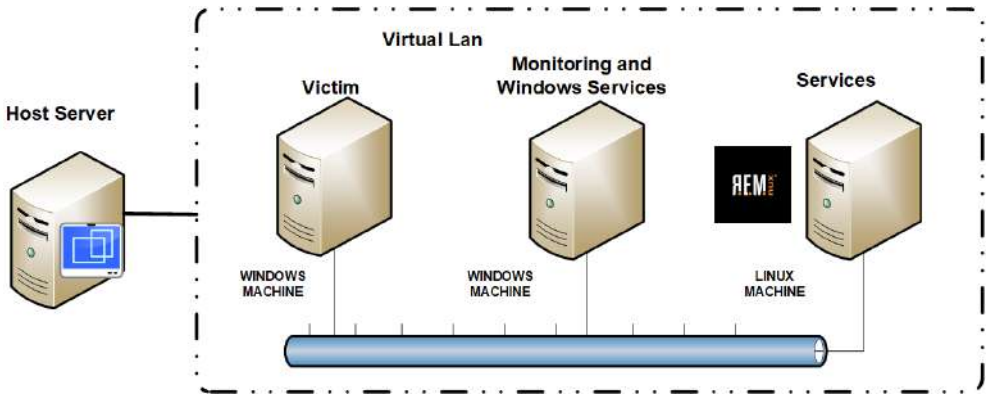

Figure 10. Working environment.

- Host Server. To host the virtual environment to support the working environment.
- Victim. Its objective will be to constitute the platform where the malware will be executed and monitored in order to study its behavior. Built on a Windows machine.
- Monitoring and Windows services. Its objective is to record and monitor network traffic produced by malware, provide specific Windows system services such as SMB and another like DNS, FTP, and WEB.
- Services. The objective of this server is to provide services to malware in its interaction with the environment, such as HTTP, DHCP, Chat, IRC Server, FTP, DNS, and SMTP, to simulate its execution environment. Built on Sans Institute REMNUX malware analysis distribution. It has installed tools such as Inetsim and Netcat to simulate services for malware.

Secondly, the working tools for malware analysis must be selected. In this case, the main choice is to use freeware tools instead of licensed ones. It is important to highlight the independence of the methodology with any specific tools. This fact is demonstrated by the following table, which shows that the tools used in this case are a subset of the total tools identified in Section 4.

The tools shown in the Table 6 are those, which were considered necessary to perform the analysis. The importance of this characteristic lies in the optimization of the working tools in each analysis,

without altering the effectiveness of the methodology, as well as being ready to the update process of the set of tools according to their evolution.

Table 6. Tools used in the analysis

Victim		Monitor and Windows Services	Services
WinMD5	PEViewer	Md5sum	Netcat
Md5sum	Process Monitor	WinMD5	HTTP Apache
Strings	RegShot	Strings	Aplicación FTP
BinText.	PeStudio	PEiD	Fake DNS
Avira	VMMap	Dependency Walker	Inetsim
Process Explorer	Systracer	PEBrowse	AVG
PEiD	DiskPulse	Ollydbg	F-prot
Dependency Walker	GMER	PE Explorer	Binwalk
Ollydbg	Winpmem	Wireshark	Volatility
PE Explorer	Wireshark		Foremost
IDA Pro			

The following are the most relevant findings from the point of view of both the analysis of the selected malware and the use of the methodology.

5.1.1. Malware Classification

- Transfer. The malware is transferred via encrypted USB and decompressed:
  - red_oct.document.exploit–(exploit)
  - red_oct.bin.drop–(dropper)
- Identification (see Table 7). Obtaining the HASH MD5 signature of the files that theoretically contain the malware:

Table 7. Malware Identification

File Name	File Size	MD5
red_oct.document.exploit	547 KB	bb2f6240402f765a9d0d650b79cd2560
red_oct.bin.drop	521 KB	598ce670b5e4be42966dbfae18188792

- Check for malware (see Table 8). Tested using online tools (virus total) and installable application Avira. Considering that Red October was detected in 2012, the results obtained from the verification phase confirm that we are analyzing a piece of malware.

Table 8. Antivirus detection

Entity	Detection
Avast	XLS:CVE-2009-3129 [Expl]
Kaspersky	Trojan-Dropper.MSWord.Agent.ga
TrendMicro	TROJ_OLEXP.B
Fortinet	MSExcel/CVE_2009_3129.A!exploit
McAfee-GW-Edition	Heuristic.BehavesLike.Exploit.X97.CodeExec.O
Microsoft	Exploit:Win32/CVE-2009-3129
Ad-Aware	Exploit.CVE-2009-3129.Gen
BitDefender	Exploit.CVE-2009-3129.Gen
McAfee	Exploit-MSExcel.ac
AVG	Dropper.Generic_c.NZR
Symantec	Backdoor.Rocra
ClamAV	NO
Panda	NO

- Open Source (OSINT). Derived from the previous steps, we are able to conduct an open source search very effectively. During this part of the analysis, high relevance information is obtained for the following phases of the methodology. The most important ones are listed below.

    - Identification of the level of complexity of the malware in analysis. It is confirmed that it is an APT, which allows us to prepare and be aware that it is a highly complex malware.
    - Identification of the vulnerabilities used by APT to infect victims. In particular, five potential attack vectors are identified:

        - CVE-2009-3129: XLS files (MS Excel)
        - CVE-2010-3333 DOC files (MS Word)
        - CVE-2012-0158 DOC files (MS Word)
        - CVE-2011-3544 Java-related vulnerability
        - CVE-2008-4250 Vulnerability to infect other computers on the network

    - It is confirmed that the malware does not make use of any 0-Day vulnerability, thus taking advantage of systems not patched correctly or not updated.
    - Key files used in the infection process are identified:

        - %TEMP%MSC.BAT
        - %ProgramFiles%WINDOWS NTLHAFD.GCP (This name may vary)
        - %ProgramFiles%WINDOWS NTSVCHOST.EXE

    - The code of a critical file 'MSC.BAT' is identified. It is also verified that a command is executed to change the reference character table for coding which means that can be a correlation with the use of Cyrillic characters in its origin.
    - MSC.BAT file has the following contents:

```
chcp 1251
Repeat
attrib -a -s -h -r "%DROPPER_FILE%"
of "%DROPPER_FILE%"
if exist "%DROPPER_FILE%" goto Repeat
of "%TEMP%msc.bat"
```

    - It is identified that the malware does not interact if it is not able to establish an outside connection, in particular with the command and control (C & C).
    - It is identified that once the back door is established, an exchange of information is initiated allowing the malware to install specific modules in order to develop a precisely attack adapted to the victim.

It is true that in this case, a sample of malware detected in 2012 is being analyzed, and this allows access to a lot of information that allows for being more effective during the perform of the analysis. However, it is considered that this is a great advantage of the methodology being tested. Both from the point of view of efficiency and effectiveness in the use of resources, and the possibility of increasing the degree of knowledge about malware by understanding what has already been discovered, allowing the analyst to focus on new aspects.

- Strings. 'Bintext' and 'Strings' tools are used to look for text strings in the code that can be helpful to the analysis. This part of the analysis yields a lot of information that needed to be analyzed in detail. The most relevant results obtained are presented below:

    - Reference identification to the Excel.exe application

- Identification of TLS functions that can be used to avoid debuggers as well as a reference to the function 'IsdebuggerPresent'
- Identification of strings related to encoding 'Encodepointer' and 'DecodePointer'
- Sequences that may be related to memory write attempts and buffer overflow attacks (!%s)
- Identification of references to specific websites (SamLab.ws)

- Obfuscation techniques (see Figure 11). Thanks to the previous phase, it was identified that the file 'red_oct.document.exploit' had almost no ASCII chains that is a potential indicator of being an obfuscated program. The 'PEiD' tool is used, but is not capable of producing any valid results.

The same test is carried out with the file 'red_oct.bin.drop' and interesting results are obtained. An entropy value of 6.62 is detected, which suggests that obfuscation techniques may be being used.

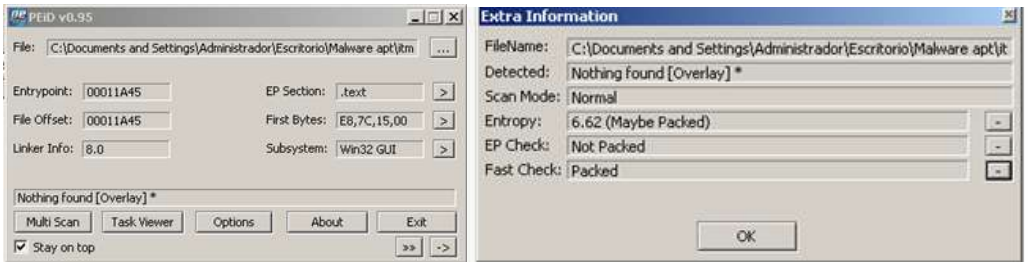

Figure 11. PEiD tool results.

According to the methodology, at this point it would be necessary to continue with the classification of the obfuscation techniques used, through a manual process of the packaging analysis and potential encryption of the file 'red_oct.document.exploit'.

Bearing in mind that the object of this analysis is aimed at verifying the applicability of the methodology presented, and getting empirical proof of the benefits it provides, it was decided not to develop the tasks of the obfuscation techniques analysis of used through manual unpacking and decryption because they require significant resources.

In this way, during the following stages of the application of the methodology, the work will focus on the file 'red_oct.bin.drop'.

- Format and structure. We are in the last step of the classification stage and it is about analyzing the structure of the file to identify the use and the relationships between libraries, versions, dates and entry sections that may suggest that the code under analysis is a malware.

During the practical exercise, the tools 'Dependency Walker' and 'PEBrowser' were used. Due to the dependence of the analysis with the tools used, in this step only the archive 'red_oct.bin.drop' could be analyzed since the other file was not recognized as a PE file, limiting our analysis. Thanks to the extensive identification of modules and dependencies, it is possible to identify the following relevant aspects:

- Identification of the use of the 'CRYPT32.DLL' library and the 'CryptStringToBinayW' function that indicate the need to process encrypted code at runtime
- Identification of the use of the 'MPR.DLL' (multi provider router) module related to the establishment of different connections and the identification of key changes and session starts
- Identification of the use of the 'SHELL32.DLL' library, used to open different Windows components

Finally, the risk of the archive is analyzed using the 'PEStudio' tool, which allows the identification of indicators of suspicion of the code being analyzed. The analysis performed gives a result of 17 risk indicators with the highest severity, highlighting aspects such as:

- ‘OVERLAY’ function that allows to embed code, so once it is in execution time, malware can use additional memory to the one reserved for that program

5.1.2. Code Analysis

At this stage of the analysis, the code is analyzed using debugging tools. In this case, according to the methodology that is being used, the dynamic analysis was carried out first and then the static analysis.

As a conclusion within this same phase, it is experimentally proved that thanks to performing first the dynamic analysis, it is possible to know with greater accuracy how the code is developed. It will allow the analyst to optimize the static analysis and its effectiveness. Below are two aspects that confirm the above statement:

- Identification of the use of obfuscation techniques that increase the complexity of static code analysis
- Identification of additional malware protection mechanisms such as the need for external connection to be able to be fully executed.

The results obtained during this phase are presented below.

- Functional check. The ‘red_oct.bin.drop’ file is checked using the ‘PE Explorer’ tool. However, this tool cannot unpack the file, although it provides us with the following information.
  - Different entry point (00411A45) to the one shown in the classification phase (00011A45)
  - Use of unconventional packaging
  - File creation date 06/05/2011 7:58:13
- Dynamic code analysis. The ‘Ollydbg’ tool is used to develop this analysis. In the first place, it is decided to navigate through the code to obtain a generalized image and the following findings are made.
  - Identification of protection measure by which it detects if the code is running in debug mode (see Figure 12)
  - The point of creation of the file ‘MSC.bat’ previously identified in the classification phase is detected (see Figure 13)
  - References to the files ‘scvhost.exe’ and ‘lhfd.gcp’, mentioned in the open sources consulted, are also identified
  - Access to the ‘RPCRT4.DLL’ library is identified that means, with high probability, that it will be used to establish the back door
  - Coded messages and text strings that mention the service ‘sshd’

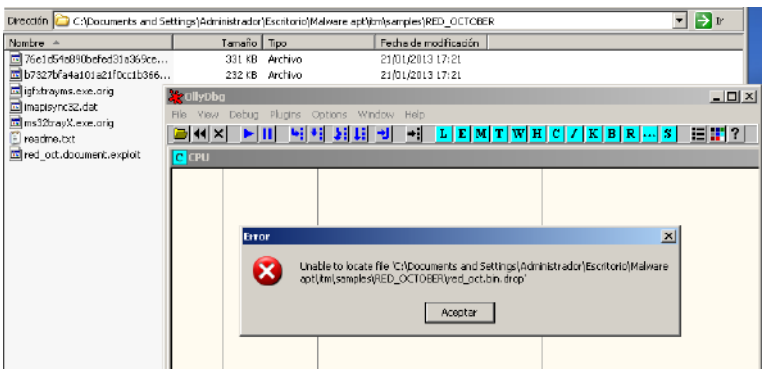

Figure 12. Identification of protection measure.

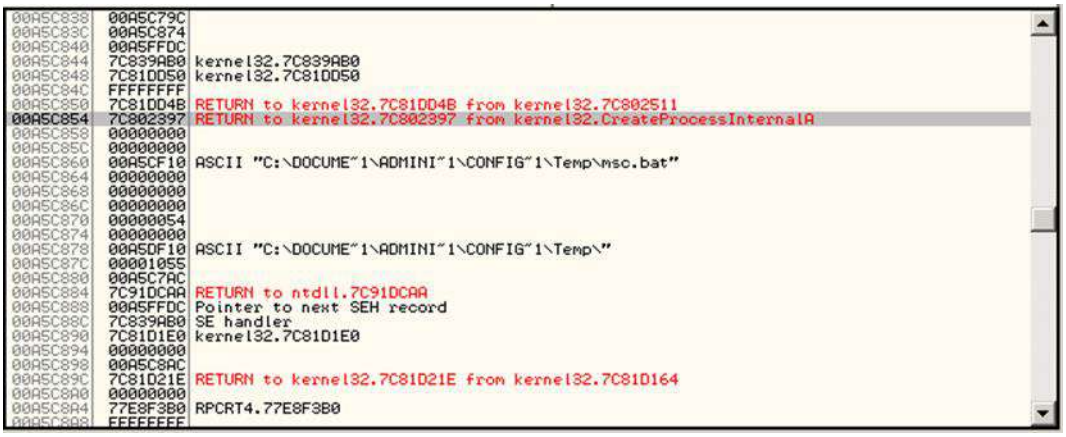

Figure 13. Creation of the file "MSC.bat".

Finally, it should be noted that during the development of the analysis it is concluded that the quality of the dynamic analysis phase is directly proportional to the analyst's expertise. This being so, the possibility to work the methodology in different phases but well-coordinated, can facilitate the management and organization of specific teams for each phase.

- Static Code Analysis. The disassembly tool 'IDA PRO' (evaluation version) is used to develop this analysis. This analysis has made it possible to obtain information of analyzed file structure at the level of memory locations, used functions and the relationship between them.

The most interesting conclusion obtained after the application of the methodology is the possibility to carry out iterations between dynamic and static analysis, allowing the analyst to check the discoveries that are obtained and thus to optimize the resources and time used. The most important results obtained during this phase are presented below:

- It is identified that within the structure of the code there is obfuscated code in each section, as well as empty memory blocks
- The use of the 'ISDebuggerPresent' function stored in the memory position '004180F0' is identified, potentially related to the protection measure detected in the previous phase (see Figure 14)

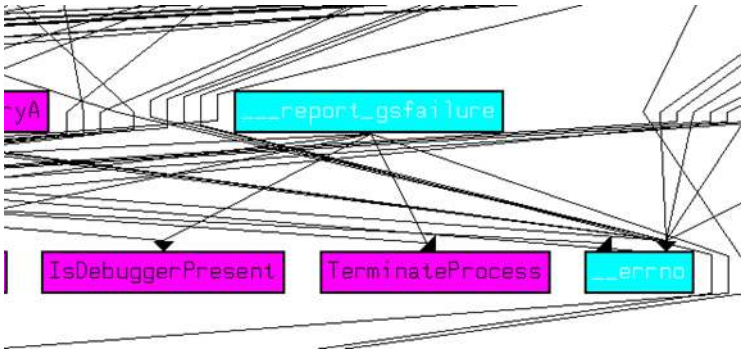

Figure 14. *Cont.*

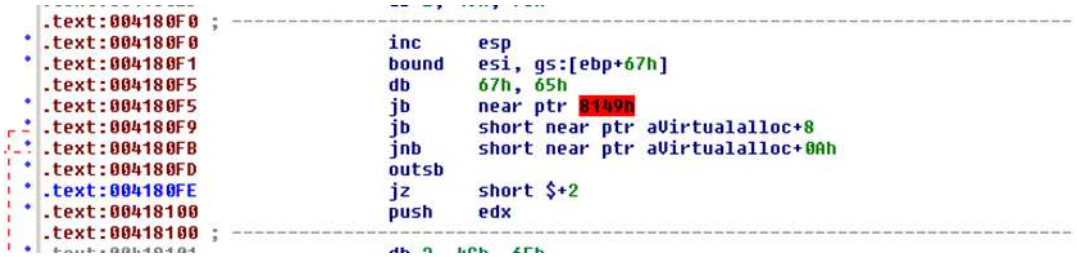

Figure 14. Ida Pro results.

5.1.3. Behavior Analysis

In this phase, we proceed to analyze the behavior of malware at runtime in order to catch and monitor the changes suffered by the victim to compare with its initial status.

The main conclusion to be drawn from executing this phase within the proposed methodological model is the large amount of information that the analyst already has at this point of analysis, reducing the number of baseline hypotheses required. This fact makes it possible to substantially optimize the analysis to be carried out.

Likewise, the application of the iteration proposal between the behavioral analysis phase and the code analysis phase demonstrates the continuous and optimized generation of information during the analysis, which translates into a continuous optimization of the analysis phases based on the findings obtained previously during the perform in the previous phases. The most relevant findings during this phase are presented below.

- Initial Tasks. Situation of the analysis starting point and establishing two work areas: the execution environment and the monitoring environment.
    - Considering that the malware has been identified requires a network communication environment, the victim's host file has been modified to allow the monitoring team resolves queries about domains
    - The monitor computer has been configured as the main DNS server being the authority over the domain.
    - The monitoring tool used was 'Wireshark'
    - The configuration of the environment of the victim computer has been prepared by capturing its initial state using the tools 'Systracer', 'Regshot. The paths in which the malware, supposedly, creates the files have been marked. Tools 'Diskpulse', 'ProcessMonitor' and VMMap' have been installed to monitor the changes in real time.
- Running malware
    - The file 'red_oct.bin.drop' is executed
    - It is detected that after 15 s of its execution, the executed file disappears from the computer
- Activation of malware services
    - Three modifications are detected in the DNS services. A first deactivation, an activation as a secondary server and an activation as a primary server.
    - Considering that, the malware being analyzed is the piece in charge of the initial infection and establishment of the Backdoor, so the detected behavior makes sense.
- Subsequent Tasks. During this stage in the behavior analysis phase, the results are analyzed after the malware execution period. It is identified that the greatest activity after the execution of the file to analyze is done during the first 20 min. The changes detected at the victim and network are presented below.

- Victim host
    - 12:04:25, the malware is run on the computer. A process named 'red_oct-bin.exe' and PID 2340 identifier is created that initiates access to registry variables related to communication processes
    - 12:04:30, the files 'svchost.exe', 'msc.bat' and 'hafd.gcp' are created
    - 12:04:36, the previously created 'svchost.exe' file is activated as a process with the PID identifier 2464 and it starts loading the malicious code into memory
    - 12:04:40, the file 'msc.bat' is accessed
    - 12:04:41, the file 'red_oct.bin.exe' is deleted. The process is created with the name 'cmd.exe'. The file 'msc.bat' is deleted
    - 12:19:44, the process 'svchost.exe' accesses registry variables related to communication processes. This phase coincides with the detection of network traffic related to the malware, concluding that it is, at this point, where it starts attempting to communicate with the C & C.
    - 12:22:30, From this moment on, the process 'svchost.exe' initiates a cyclic behavior, every 3 min approximately, by means of which it tries to establish the connection with the exterior. It is interesting to detect the attempt to access routes related to web browsing applications such as Opera or Firefox, which could mean that the sample was aimed at a victim using this type of browser.

The files responsible for the infection process are analyzed:

- The file 'msc.bat' remains temporarily in the system. This file is in charge of activating the infection inside the system (see Figure 15).
- The file 'svchost.exe' is a file that cannot be removed from the system. It is executed automatically by creating a process that contains encrypted code. Code analysis will be necessary.
- The 'hafd.gcp' file is a hidden and resident file. It is accessed only during the infection phase. It is an encrypted file.
- The use of the 'VMMAP' tool has made it possible to monitor the 'svchost.exe' process with PID2464. This monitoring has led to the following findings:
    - Text strings contained in the 'Private Data' section (00400000-00421FF) that refer to the 'Get http' method for downloading specific modules and names of external servers that can serve as proxies to communicate with the C & C.
    - The use of the tools 'Systracer' and 'Regshot' have allowed identifying the modification of user values (HKU) and the local computer (HKLM), used to allow the execution of all the tasks associated with the infection process.

Figure 15. File 'msc.bat'.

- Network traffic analysis. It is identified that the malware is programmed to initiate communication several minutes after the infection process. In this case, it has been 19 min, although this time may vary depending on the environment where the infection occurs. The analysis has been divided into three parts:

- First malware execution with the DNS service disabled, where random requests are monitored trying to contact external DNS servers that operate as proxy.
- Second execution with the active DNS server where an interaction with the malware is checked, who asks the DNS server if it is an authorized server. Once the check is established, the malware no longer interacts. It was probably waiting for confirmation from C & C.
- Thirdly, the malware was executed and the file "hafd.gcp" was deleted, the communication sequence being identical to the second scenario.

- Dump and memory analysis. This is the last step in behavioral analysis. The 'Winpmem' tool has been used together with the 'Volatility' tool and its specific 'malfind' plugin to analyze malware. The most relevant findings based on the analysis of the file 'svchost.exe' with PID 1540 are the following:
  - Identification of suspicious 'hooks' in both 'user mode' and 'kernel mode'
  - Identification of code and suspiciously injected or hidden DLLs

5.2. Flame Analysis

To test the methodology proposed in this paper, the analysis of a malware 'Flame' has been made. It belongs to APT malware type, one of the most complex and dangerous malwares discovered today. The objective of this analysis is to obtain a comprehensive understanding of its operation, composition, modules, types of encryption, injection mechanisms, communications with its command and control system, data types, etc. A more complete analysis is beyond our purpose as it would be unfeasible in a limited time.

The main objective of this malware is the theft of classified information (cyber espionage) on systems with Windows OS and using USB sticks to spread exfiltration data. Given its complexity, it has been considered to be developed by a state, which fall within the category of 'Advanced Persistent Threats'.

Their level of complexity would be very high considering its size, the number of files it handles, languages and libraries used, different encryption methods used, or the time needed for its analysis. It means that so far can be considered the most complex existing type of malware.

5.2.1. Malware Classification

- Transfer. The malware is transferred via encrypted USB and decompressed:
  - mssecmgr.ocx.orig
- Identification
  - File Type (see Table 9): type dynamic library.ocx

Table 9. Malware identification

File Name	File Size	MD5
mssecmgr.ocx.orig	6.166528 MB	bdc9e04388bda8527b398a8c34667e180

- Check for malware
  - Names of malware: Flame, Flamer, and Skywiper
  - Current detection capabilities antivirus: It is detected by 46 antiviruses
- Strings
  - Use Database SQLLite and programming language C ++ and Lua

...
CREATE VIRTUAL TABLE %T
sqlite_temp_master
sqlite_master

- UPDATE %Q.%s SET type='table', name=%Q, tbl_name=%Q, rootpage=0, sql=%Q WHERE rowid=#%d

- Obfuscation techniques
  - mssecmgr.ocx.orig: 2.4 MBit packed area, entropy 1
  - Files accessed by the malware, entropy, first detected file type and whether it was possible to decode it (See Table 10)
  - It uses three different mechanisms packaging (zlib algorithms, ppmd, libbzip) and RC4 encryption, XOR and transposition (see Table 11)
  - Diffusing Capacity: File with Microsoft valid certificate, verified as correct with an MD5 hash.
  - Exfiltrating capacity data: on the USB memory, once the malware has been executed, it creates a hidden file whose name is a single point ('.'). The content is encrypted.
  - Interact with command and control system. It uses the processes 'winlogon.exe', 'explorer.exe', and 'iexplore.exe'. The fact that it is periodically opened the 'Internet Explorer' browser, suggests that it could be used to communicate with a 'Command & Control'.

Table 10. Files created by malware

File	Entropy	Initial Content	Decoded
advnetcfg.ocx	0.46	PE32 executable	N/A
boot32drv.sys	0.32	Data encoded/encrypted	YES
ccalc32.sys	1.00	Data encoded/encrypted	NO
~dra61.tmp	1.01	Data encoded/encrypted	NO
dstrlog.dat	0.03	Data encoded/encrypted	YES
Ef_trace.log	0.56	Data encoded/encrypted	NO
File "." inside USB	0.26	Data encoded/encrypted	YES
~HLV084.tmp	1.00	Data encoded/encrypted	NO
~HLV294.tmp	1.00	Data encoded/encrypted	NO
~HLV473.tmp	0.98	Data encoded/encrypted	YES
~HLV927.tmp	0.14	Data encoded/encrypted	NO
~KWI988.tmp	0.86	Data encoded/encrypted	NO
~KWI989.tmp	1.00	Data encoded/encrypted	NO
lmcache.dat	0.97	Data encoded/encrypted	YES
mscrypt.dat	0.53	Data encoded/encrypted	YES
msglu32.ocx	0.51	PE32 executable	N/A
mssecmgr.ocx	0.70	PE32 executable	N/A
ntcache.dat	0.70	Data encoded/encrypted	YES
nteps32.ocx	0.45	PE32 executable	N/A
rccache.dat	-	Empty (0 bytes)	N/A
~rf288h.tmp	0.33	Data encoded/encrypted	YES
~rf288.tmp	0.04	Data encoded/encrypted	YES
ssitable	0.03	Data encoded/encrypted	YES

Table 11. Packaging and encryption algorithms

Algorithm	Files Detected
XOR 0xFF	boot32drv.sys dstrlog.dat lmcache.dat mscrypt.dat ntcache.dat ~rf288h.tmp ~rf288.tmp ssitable Fichero "."
Transposition by table	
Zlib	~HLV473.tmp

- **Format and structure**
  - Supported operating systems: it supports operating systems based on Windows platform.
  - Libraries required: it requires a total of 35 libraries: ADVAPI32.DLL, apphelp.dll, COMCTL32.DLL, COMDLG32.DLL, CRYPT32.DLL, Cryptui.dll, GDI32.DLL, IMAGEHLP.DLL, KERNEL32.DLL, lz32.dll, MPR.DLL, msasn1.dll, MSVCRT.DLL, NETAPI32.DLL, NTDLL.DLL, OLE32.DLL, OLEAUT32.DLL, Olecli32.dll, Olecnv32.dll, OLESVR32.DLL, OLETHK32.DLL, RPCRT4.DLL, SECUR32.DLL, SHDOCVW.DLL, SHELL32.DLL, SHLWAPI.DLL, URL.DLL, URLMON.DLL, USER32.DLL, Userenv.dll, Version.dll, WININET.DLL, Wintrust.dll, WLDAP32.DLL, WOW32.DLL.
  - URLs: https://traffic-spot.com/wp-content/rss.php. This URL, for the analysis of the sample provided, was not available; domain "traffic-spot.com" does not exist anymore.
  - DNS resolution requests, in this case, for the following domains:
    - windowsupdate.microsoft.com
    - traffic-spot.com
    - traffic-spot.biz
    - smart-access.net
    - quick-net.info
  - It works only the resolution of the 'windowsupdate.microsoft.com' domain; the remaining domains are not accessible.

5.2.2. Code Analysis

One method used to get the real text strings that malware can use is to work with a debugger such as 'ollydbg'. It allows you to execute and analyze the code step by step in a dynamic way, it is especially interesting when text strings could be encrypted or encoded (see Figure 16).

Initially a string analysis was performed on the original malware file 'mssecmgr.ocx.orig', resulting in a few interesting strings because they could be encrypted or encoded initially. However, by this method of analysis the actual text strings used by the malware were obtained.

An analysis of the text strings used by the malware processes loaded in memory, i.e., 'running', was performed. For example, the Olly DBG tool was used to decode the running process 'services.exe' in memory in real time, from which strings were extracted and used as initial input information for the next stage of 'Behavioral Analysis' in a more effective way. Strings of the following type were obtained:

- Programs that are blacklisted
- Program/malware modules and libraries
- Domain names (C & C servers): https://traffic-spot.com:443/wp-content/rss.php, https://traffic-spot.biz:443/wp-content/rss.php

- Used folder and file names, file extensions of interest...
- Registration keys accessed
- Queries to access 'sqlite' databases

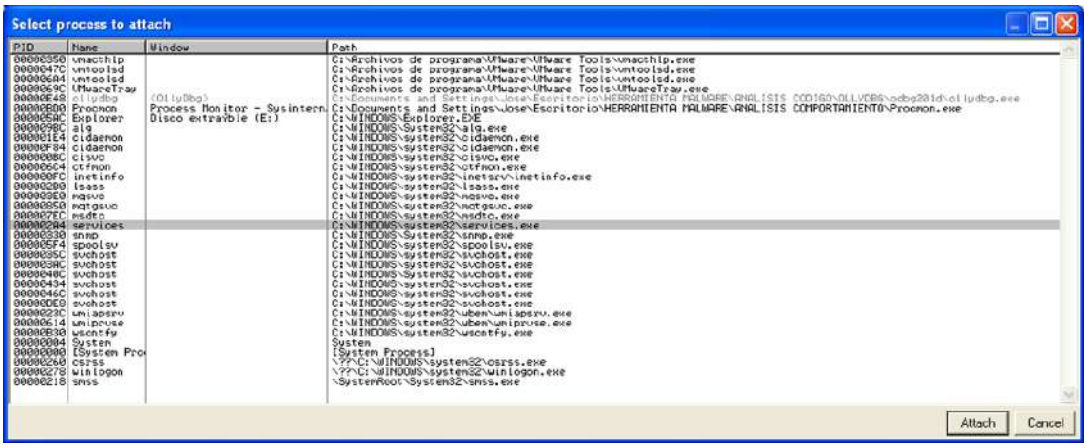

Figure 16. OllyDbg—list of processes.

The following shows what seem to be modules that are part of the design of this malware. It is extracted from the text strings, which by their name are interesting: GATOR, FROG, MICROBE, GADGET, EUPHORIA, BEETLEJUICE, Snack, Munch, AUTORUN_INFECTOR, SUICIDE, JIMMY, and FLASK.

5.2.3. Behavior Analysis

It is a very complex type of APT with a large number of components. It is also composed of large files, something customary in the malware.

- Running malware
  - Scripts and executables
    - Implementation of the specimen: start/ wait runddll32.exe mssecmgr.ocx.orig
  - Capacity of Infection
    - The malware timed their actions so that they appear intertwined with legitimate activities and migrates between processes as needed. We also see that it creates many temporary files, some are deleted shortly after being created, and others remain at least for several minutes.
    - It migrates the process 'rundll32.exe' after its first execution to the process 'services.exe'. It means that it infects using overwriting memory technique.
    - It performs a 'hook' back in the Windows API, in the 'SHGetSpecialFolderPathW' call 'Explorer.exe'. It also infects the 'iexplore.exe' process.
    - In the process 'services.exe', specifically in the memory range '0x7e6a0000-0x7ecc7fff', we can see that the 'services.exe' is another executable.
- Activation of malware services
  - Files created (see Table 12)
  - Registry modifications
    - HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\LSA\Authentication Packages added mssecmgr.ocx indicates packages that are loaded when a user logs on the victim machine.

- Memory analysis
  - In the memory of infected reference 'services.exe', it appears a process that performs an action of 'not installed programs' such as antivirus, firewalls, network monitors, etc., indicating that the malware has its own 'blacklist' within the code to avoid detection by these applications.

Table 12. Files created by malware

Name	Size (bytes)	Name	Size (bytes)
dstrlog.dat	86,016	soapr32.ocx	200,000
lmcache.dat	0	~HLV084.tmp	1604
mscrypt.dat	6,163,261	~HLV473.tmp	85
ntcache.dat	38,400	~HLV927.tmp	85
rccache.dat	0	~HLV294.tmp	200,000
ssitable	1,572,896	~HLV927.tmp	
Ef_trace.log	858	~KWI988.tmp	
advnetcfg.ocx	643,072	~dra53.tmp	
boot32drv.sys	1300	~mso2a0.tmp	
ccalc32.sys	53,534	~mso2a2.tmp	
msglu32.ocx	1,721,856	~rf288h.tmp	
mssecmgr.ocx	6,166,528	~rf288.tmp	
nteps32.ocx	827,392	~HLV084.tmp	1604

5.3. Other Analysis

The methodology has been applied to others malware like 'Stuxnet', 'Dark Comet', 'Poison Ivy', 'Locky', 'Careto', 'Wannacry', and 'Sofacy Carberp' realized by different analyst.

The Table 13 shows the result of the different phases of the methodology applied to the aforementioned malware. The degree of fulfilment will be 'C' for complete, 'P' for partial, and 'X' for none.

Table 13. Results of the methodology

	Initial Actions	Classification	Dynamic Code Analysis	Static Code Analysis	Behavior Analysis
Flame	C	C	C	P	C
Red October	C	C	C	C	C
Stuxnet	C	C	C	P	P
Dark Comet	C	C	C	C	C
Poison Ivy	C	C	C	C	C
Locky	C	C	C	C	C
Careto	C	C	C	C	P
Wannacry	C	C	C	C	P
Sofacy Carberp	C	C	P	P	C

Through these analyses, it is demonstrated that the application of this methodology, its procedures and activities, in the study and analysis of a malware, allows obtaining effective results on the action of a malware on the infected systems. It provides techniques and knowledge that will generate an effective application base. The results of the 'Code Analysis' depends greatly on the analyst's reverse engineering skills and the complexity of the malware.

The complexity of the malware requires not only organizational resources such as proposed methodology, tools and a good laboratory, but also it is essential to have a very high knowledge in reverse engineering, by the analyst who will perform the analysis.

6. Discussion

The main lessons learned from the completion of the two study cases are related to the proposed methodology of malware analysis and to its suggested modifications of the analysis process:

- Considering the current situation in the development of malware and its industrialization, within the analysis of malware, the technical analysis of the solution is a fundamental issue. It has been verified that the phases of classification and search in open sources are key to establish an optimal strategy of analysis in the following phases. Currently, there is a lot of information published by expert analysts that can help greatly to economize the use of resources in a specific analysis, either by reuse of code, by reuse of vulnerabilities to exploit or reuse of entire malware packages if the attack is characterized by the use of specific modules.
- During the application of the methodology to the piece of malware, two highly complex defense mechanisms have been identified: one is the packaging and obfuscation and the second is the detection of operation in debugging mode. To be able to deal with these types of measures, it requires a high level of specialization of the analyst team that is not available to everyone.
- However, thanks to the structure of the methodology, it is possible to identify this situation and, in case of incapacity, to derive that part of the analysis to qualified teams. This situation implies an optimization of resources at the level of analysis, guaranteeing a good use of work teams, starting with the detection of highly complex situations, as well as the possibility of deriving parts of the analysis to different teams.
- Code analysis capabilities are essential to develop a good analysis. This implies that the methodology depends, of course, on the capacity of the analyst, especially in the static analysis phase. However, the structure of the methodology makes it possible to derive this phase to a team or several teams in parallel, providing the possibility of crossing results and improving the analysis in two ways:
    - By iteration between code analysis and behavior analysis
    - By iteration between static and dynamic code analysis
    - Through the joint work of different teams in these phases in order to consolidate the results

7. Conclusions

It has been reported the application of the method to a case study with two of the most complex and sophisticated malwares known as a first evaluation of the method. These two APT are 'Flame' and 'Red October' malware. The main conclusions of its applications are the following:

- The methodology is applicable regardless of the complexity of the malware. This makes it possible to analyze different types of malware models and obtain a homogeneous analysis result.
- The identification of the phases allows its execution by human teams with its specific tools. It allows to systematize the analysis optimizing each phase based on the existing capacities and to share the results of each phase with the rest of the involved analysts to improve the effectiveness of the results.
- The methodology establishes the type of analysis to be performed but it is independent of the tools. This guarantees the versatility and adaptability of the method to different analysis environments.
- The order established by the methodology guarantees an effective obtaining information process, starting from the classification based on searching in open sources. Later, this will make it possible to define an operative strategy in the analysis of the selected malware. Likewise, it allows, in the face of the same malware, to develop specific analysis work on specific aspects, a highly interesting issue for the development of intelligence on complex malware such as APTs.

- The inclusion of an iteration process in different phases of the methodology has made it possible to resolve and complete the different phases of the analysis by generating a greater global knowledge than the sum of each of the phases independently.
- The validity of the applicability of the methodology taken as a reference is strongly confirmed. The execution of the complete process according to the proposed order has guaranteed an orderly and coherent advance in the analysis of the 'Red October' and 'Flame' malware. The results of which, in accordance with those obtained by expert sources, proves its success.
- The classification phase has been identified as a critical one for the realization of this pilot. The information gathered has allowed the tests carried out to be effectively oriented, optimizing the analysis. This fact shows that this phase is vital in the analysis of this type malware that have been previously detected by third parties. In the case of unidentified malware, the relevance of this phase would diminish, drastically making the analysis more complex.
- During the code analysis phase, it is confirmed that the proposed innovation of exchanging the order, placing the dynamic code analysis first to the static one, facilitates the development of the malware study if it is highly complex and uses advanced obfuscation techniques. However, the need to include a cyclical and iterative process between both analyses has been detected—possibly conditioned by the level of the analyst—with the aim of focusing and performing deeper investigation as the knowledge about malware increases. This could be an innovation derived from the one described above.
- The need for the malware behavior analysis phase is confirmed, as it greatly increases the knowledge about the malware behavior being analyzed. However, the effectiveness of this phase is conditioned by the quality of the information obtained in the previous phases which drastically reduces any possible interest in executing this phase before the previous ones. The methodology includes an iterative mechanism consistent with the possibility of having to analyze new samples due to the complexity of malware. It can be materialized during execution, a situation that would have occurred if phase 2 of the APT 'Red October' had been initiated and specific modules were being downloaded from the C & C.
- During the execution of the tests, taking into account their scope, it has become clear that the tests in the phases of analysis of code analysis (static as well as behavior) can focus on the victim system, simplifying the number of tools to install in the rest of systems involved in the lab. This allows the analyst to focus on the target of the monitor system and the system services needed as key parts in the constitution of a realistic scenario in order to force the malware to show all its behavior.

As a continuation of this paper, it has been identified in the FIGURE lines of future work:

- Management and classification of tools. The application of the methodology is based on the use of multiple tools, many of them of free use and others previous acquisition of license. Bearing in mind that the second type of tool already has legal conditions, the first one is more difficult to control (support, continuity, etc.). A line of development focused on the effective management and maintenance of these tools would improve the management of technical and human resources and could lead to possible standardizations in specific sectors.
- Apply the proposed methodology to other operating systems. While here it has been developed and validated for Windows based systems, as this type of operating system used mainly in the world, is can also be valid for other operating systems like Linux, Android, etc. Since, nowadays, malware is present in any of them. New lab design should consider other victim machines' operating systems such as Linux and Android.
- Determine whether different analysis tools are compatible and, if not, look for the equivalent versions for Linux and Android operating systems.

Author Contributions: Conceptualization, J.B.H., C.A.A., and J.-R.B.H.; Methodology, J.B.H. and C.A.A.; Validation, J.-R.B.H., J.A.S.M., and M.A.S.U.; Investigation, J.B.H., C.A.A., J.-R.B.H., J.A.S.M., and M.A.S.U.; Resources, J.B.H., C.A.A., J.-R.B.H., J.A.S.M., and M.A.S.U.; Writing—Original Draft Preparation, J.B.H. and C.A.A.; Writing—Review and Editing, J.B.H., C.A.A., J.-R.B.H., J.A.S.M., and M.A.S.U.; Visualization, J.B.H., C.A.A., J.-R.B.H., J.A.S.M., and M.A.S.U.; Supervision, J.B.H., C.A.A., J.-R.B.H., J.A.S.M., and M.A.S.U.; Project Administration, J.-R.B.H., J.A.S.M., and M.A.S.U. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bencsáth, B.; Pék, G.; Buttyán, L.; Felegyhazi, M. The cousins of Stuxnet: Duqu, Flame, and Gauss. *Future Internet* **2012**, *4*, 971–1003. [CrossRef]
2. Gandotra, E.; Bansal, D.; Sofat, S. Malware Analysis and Classification: A Survey. *J. Inf. Secur.* **2014**, *5*, 56–64. [CrossRef]
3. Burrows, A.; Lockwood, M.; Borowczak, M.; Janak, E.; Barber, B. Integrated STEM: Focus on Informal Education and Community Collaboration through Engineering. *Educ. Sci.* **2018**, *8*, 4. [CrossRef]
4. Orcos, L.; Jordán, C.; Magreñán, A. 3D visualization through the hologram for the learning of area and volume concepts. *Mathematics* **2019**, *7*, 247. [CrossRef]
5. Grout, I. Remote Laboratories as a Means to Widen Participation in STEM Education. *Educ. Sci.* **2018**, *7*, 85. [CrossRef]
6. Jordán, C.; Magreñán, Á.A.; Orcos, L. Considerations about flip education in the teaching of advanced mathematics. *Educ. Sci.* **2019**, *9*, 227. [CrossRef]
7. Prieto, M.C.; Palma, L.O.; Tobías, P.J.B.; León, F.J.M. Student assessment of the use of kahoot in the learning process of science and mathematics. *Educ. Sci.* **2019**, *9*, 55. [CrossRef]
8. Orcos, L.; Hernández-Carrera, R.M.; Espigares, M.J.; Magreñán, Á.A. The Kumon method: Its importance in the improvement on the teaching and learning of mathematics from the first levels of Early Childhood and Primary Education. *Mathematics* **2019**, *7*, 109. [CrossRef]
9. Sikorski, M.; Honig, A. Practical Malware Analysis. In *The Hands-On Guide to Dissecting Malicious Sofware*; No Starch Press: San Francisco, CA, USA, 2012.
10. Monnappa, K.A. *Learning Malware Analysis*; Packt Publishing Ltd.: Birmingham, UK, 2018; ISBN 978-1-78839-250-1.
11. Theerthagiri, D. Reversing Malware: A Detection Intelligence with In-Depth Security Analysis. Ph.D. Thesis, Linköpings University, Linköping, Sweden, 2009.
12. IDA Pro Disassembler. 2019. Available online: https://www.hex-rays.com/idapro/ (accessed on 2 November 2019).
13. National Security Agency. Central Security Service. GHIDRA. 2019. Available online: https://www.nsa.gov/resources/everyone/ghidra/ (accessed on 2 November 2019).
14. Olly Debugger. 2019. Available online: https://www.ollydbg.de/ (accessed on 2 November 2019).
15. Immunity Debugger. 2019. Available online: https://www.immunityinc.com/products/debugger/ (accessed on 2 November 2019).
16. Windows Debugging Tools. WinDBG. Available online: https://docs.microsoft.com/es-es/windows-hardware/drivers/debugger/debugger-download-tools (accessed on 2 November 2019).
17. Ori Meir, N.N.; Yuval, E.; Lior, R. Dynamic Malware Analysis in the Modern Era—A State of the Art Survey. *ACM Comput. Surv.* **2019**, *52*, 1–48. [CrossRef]
18. Moser, A.; Kruegel, C.; Kirda, E. Limits of Static Analysis for Malware Detection. In Proceedings of the Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), Miami Beach, FL, USA, 10–14 December 2007; pp. 421–430.
19. Manuel, E.; Theodoor, S.; Engin, K.; Christopher, K. A survey on automated dynamic malware-analysis techniques and tools. *ACM Comput.* **2012**, *44*, 42.
20. Mulukutla, V. Wolfsting: Extending Online Dynamic Malware Analysis Systems by Engaging Malware. Ph.D. Thesis, Faculty of North Carolina State University, Raleigh, NC, USA, 2010.
21. Rieck, K.; Trinius, P.; Willems, C.; Holz, T. Automatic analysis of malware behavior using machine learning. *J. Comput. Secur.* **2011**, *19*, 639–668. [CrossRef]

22. Yoshioka, K.; Matsumoto, T. Multi-Pass Malware Sandbox Analysis with Controlled Internet Connection. IEICE TRANSACTIONS on Fundamentals of Electronics. *Commun. Comput. Sci.* **2010**, *93*, 210–218.
23. Wang, L.; Wang, B.; Liu, J.; Miao, Q.; Zhang, J. Cuckoo-based Malware Dynamic Analysis. *Int. J. Perform. Eng.* **2019**, *15*, 772–781. [CrossRef]
24. Liu, X.; Zhang, J.; Lin, Y.; Li, H. Atmpa: Attacking machine learning-based malware visualization detection methods via adversarial examples. In Proceedings of the IEEE/ACM International Symposium on Quality of Service, Phoenix, AZ, USA, 24–25 June 2019.
25. Analiz. Malware Dashboard. 2019. Available online: https://sandbox.anlyz.io/dashboard (accessed on 2 November 2019).
26. Malwr. 2019. Available online: http://malwr.com/ (accessed on 2 November 2019).
27. Threat Analyzer. Automated Threat Analysis. 2019. Available online: https://www.threattrack.com/malware-analysis.aspx (accessed on 2 November 2019).
28. Pafish. 2019. Available online: https://github.com/a0rtega/pafish (accessed on 2 November 2019).
29. Gómez Ramos, F.; Bermejo, J. We must simulate to improve our Cyber Defense. *J. Atenea* **2012**, *35*, 56–60.
30. Cory, Q.N.; James, E.G. Malware analysis reverse engineering (MARE) methodology & malware defense (M.D.) timeline. In Proceedings of the Information Security Curriculum Development Conference (InfoSecCD '10), Kennesaw, GA, USA, October 2010; pp. 8–14.
31. Rahman, S.; Khan, M.N.A. Review of Live Forensic Analysis Techniques. *Int. J. Hybrid Inf. Technol.* **2015**, *8*, 379–388. [CrossRef]
32. Rafique, M.; Naeem, M.; Khan, A. Exploring Static and Live Digital Forensics: Methods, Practices and Tools. Computer Science: USA. 2013. Available online: https://www.semanticscholar.org/paper/Exploring-Static-and-Live-Digital-Forensics%3A-and-Rafique-Khan/45e51f18c4e8157836bb936997293adeced32a24 (accessed on 2 November 2019).
33. Coryn, C.L. The fundamental characteristics of research. *J. Multidiscip. Eval.* **2007**, *3*, 124–133.
34. Demeyer, S. Research Methods in Computer Science. In Proceedings of the IEEE 27th International Conference on Software Maintenance, ICSM 2011, Williamsburg, VA, USA, 25–30 September 2011.
35. Runeson, P.; Höst, M. Guidelines for conducting and reporting case study research in software engineering. *Empir. Softw. Eng. J.* **2009**, *14*. [CrossRef]
36. sKyWIper Analysis Team. A Complex Malware for Targeted Attacks. In *Laboratory of Cryptography and System Security (CrySyS Lab)*; Budapest University of Technology and Economics: Budapest, Hungary, 2012.
37. Kaspersky Labs. "Global Research & Analysis Team. Red October" Diplomatic Cyber Attacks Investigation". 2013. Available online: https://securelist.com/analysis/36740/red-october-diplomatic-cyber-attacks-investigation/ (accessed on 2 November 2019).
38. Christopher, C.E. *Advanced Malware Analysis*; McGraw-Hill Education: Pennsylvania Plaza, NY, USA, 2015; ISBN 978-0-07-181975-6.