**ABSTRACT**

Malware continues to pose substantial threats in the cybersecurity landscape, necessitating thorough analysis to uncover their attack vectors, techniques, and behavioural patterns. This project undertakes a systematic approach to analysing various types of malwares to gain comprehensive insights into their functionalities and operational methodologies.

The methodology integrates both dynamic and static analysis techniques, utilizing controlled environments and deep code inspection to dissect malware behaviour. Behavioural analysis provides understanding of how malware interacts with systems and networks, while code analysis uncovers the intricacies of its implementation and potential evasion tactics.

Key objectives include identifying patterns, command and control mechanisms, and potential payloads associated with different malware families. The project emphasizes the acquisition of detailed knowledge through rigorous examination, without focusing on mitigation strategies or countermeasures.

Case studies within the project involve the analysis of diverse malware types, including trojans, ransomware, worms, and other variants encountered in real-world scenarios. Each case study highlights the distinctive characteristics and challenges posed by different malware families, showcasing the versatility of the analysis methodology in uncovering their operational tactics.

By documenting methodologies, findings, and insights gained, this project contributes to advancing malware analysis practices within the cybersecurity community. It aims to enhance understanding, improve detection capabilities, and foster collaboration among analysts and researchers.

In conclusion, this malware analysis project represents a focused and methodological approach to exploring the complexities of modern malware. By expanding knowledge and capabilities in malware analysis, it aims to strengthen defences and support efforts to protect digital assets and infrastructures.

Keywords: malware analysis; cybersecurity; dynamic analysis; static analysis; behavioural analysis; code analysis; trojans; ransomware; worms

**INTODUCTION**

In today's interconnected digital landscape, the proliferation of malware represents a persistent and ever-evolving threat to cybersecurity. Malicious software, or malware, encompasses a broad spectrum of malicious programs designed to infiltrate, disrupt, or gain unauthorized access to computer systems and networks. From stealthy trojans and disruptive ransomware to self-propagating worms and sophisticated botnets, malware variants continue to challenge cybersecurity professionals worldwide.

The analysis of malware plays a crucial role in understanding these threats, unraveling their intricate methodologies, and developing effective countermeasures. This project focuses on the systematic and methodical analysis of various types of malware to uncover their underlying behaviors, attack vectors, and operational patterns. By dissecting malware samples through advanced analytical techniques, the project aims to deepen our understanding of their capabilities and mitigate the risks they pose to digital assets and infrastructures.

**Importance of Malware Analysis:**

Malware analysis serves as a cornerstone in cybersecurity defense strategies, offering insights into how malicious software operates, spreads, and persists within targeted environments. Through meticulous examination, analysts can identify unique characteristics and signatures that distinguish one malware variant from another. This knowledge is instrumental in developing robust detection mechanisms, enhancing threat intelligence, and fortifying defenses against evolving cyber threats.

Moreover, malware analysis provides critical insights into the tactics employed by threat actors, shedding light on their motivations and objectives. Understanding these factors enables cybersecurity professionals to anticipate and preemptively respond to potential attacks, thereby minimizing the impact on organizations and individuals.

**Objectives of the Project:**

The primary objective of this project is to conduct comprehensive and systematic malware analysis across a diverse range of malware types. By employing a combination of dynamic and static analysis techniques, including behavioral analysis and code inspection, the project seeks to achieve the following goals:

1.**Behavioral Understanding**: Gain insights into how malware interacts with its environment, including its execution flow, network communications, and system modifications.

2**. Code Examination**: Analyze the underlying code structure and functionality of malware to uncover obfuscation techniques, anti-analysis measures, and potential vulnerabilities.

3. **Pattern Identification**: Identify commonalities and distinctive features among different malware families, such as command and control mechanisms, propagation methods, and payload delivery mechanisms.

4. **Knowledge Acquisition**: Acquire detailed knowledge and documentation of the analyzed malware specimens, documenting findings and insights to contribute to the broader cybersecurity community.

**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 analyse malware is proportional to the level of sophistication of the sample. The main benefits of malware analysis are the following:

• 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 defence 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 defences 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 uses 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”, 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 or GHIDRA.

• 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 , ImmunityDbg , and WinDbg .

• Dynamic or behavioural analysis: It analyses 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.

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.

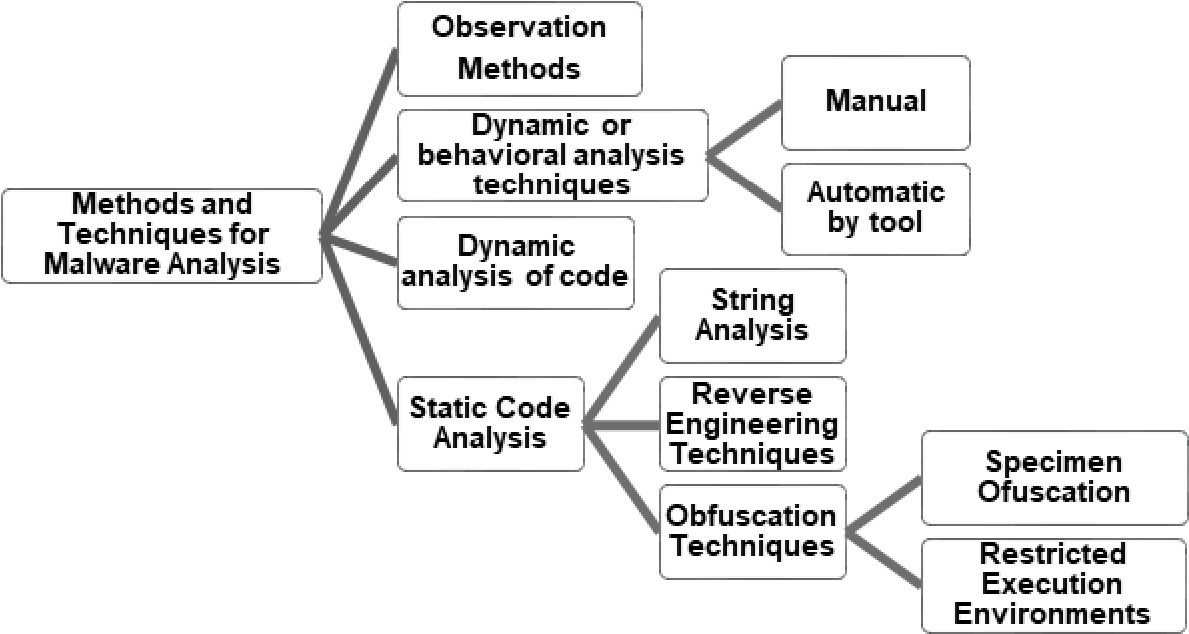


Figure. Methods and techniques for Malware analysis

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. Moser et al proposed an obfuscation approach that demonstrates that static analysis techniques alone might no longer be sufficient to analyse complex specimen of malware. Also, in 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”. 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 behavioural 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 is to support the analysts to understand the malware behaviour and to obtain data, effectively, in order to facilitate reverse engineering to design effective countermeasures.

Furthermore, 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, Norman Sandbox, Cuckoo, TTAnalyzer, Anlyz, Malwr Threat Expert, or Cobra. In 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 analysing only a single execution path, so they could ignore relevant behaviour of the malware. (i.e., malware logic behaviour under certain conditions like system date). According to Gandotra et al. “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 behaviour rather than the exact one”.

Another problem that has to be faced using this type of tools and performing dynamic analysis and behavioural 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 behaviour. It is possible to use the Pafish 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 shows a classification of tools based on its functionality.

## 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

Table. Malware analysis tools

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 analysing 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 malwares may present. 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)

**Scope and Case Studies:**

The project encompasses the analysis of various malware types encountered in real-world incidents. Case studies will focus on prominent examples such as trojans, ransomware, worms, and other sophisticated variants. Each case study will highlight unique challenges posed by different malware families and demonstrate the application of the analysis methodology in uncovering their operational tactics.

**Contributions to Cybersecurity:**

By documenting methodologies, findings, and insights gained through this project, the aim is to contribute to the advancement of malware analysis practices within the cybersecurity community. This includes:

**Enhancing Detection Capabilities**: Providing actionable intelligence and detection signatures to bolster cybersecurity defenses against current and emerging malware threats.

**Supporting Incident Response**: Equipping organizations and cybersecurity teams with the knowledge and tools needed to effectively respond to and mitigate the impact of malware incidents.

**Promoting Collaboration**: Fostering collaboration among analysts, researchers, and industry stakeholders to share best practices, methodologies, and lessons learned in combating malware.

**METHODOLOGY**

Comparing MARE and SAMA: Approaches to Malware Analysis-

At present, we have identified only one malware analysis methodology. 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 analysed malware. These phases are as follows:

1. Detection

2. Isolation and extraction

3. Behavioural 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 defence 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 specimens. 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.

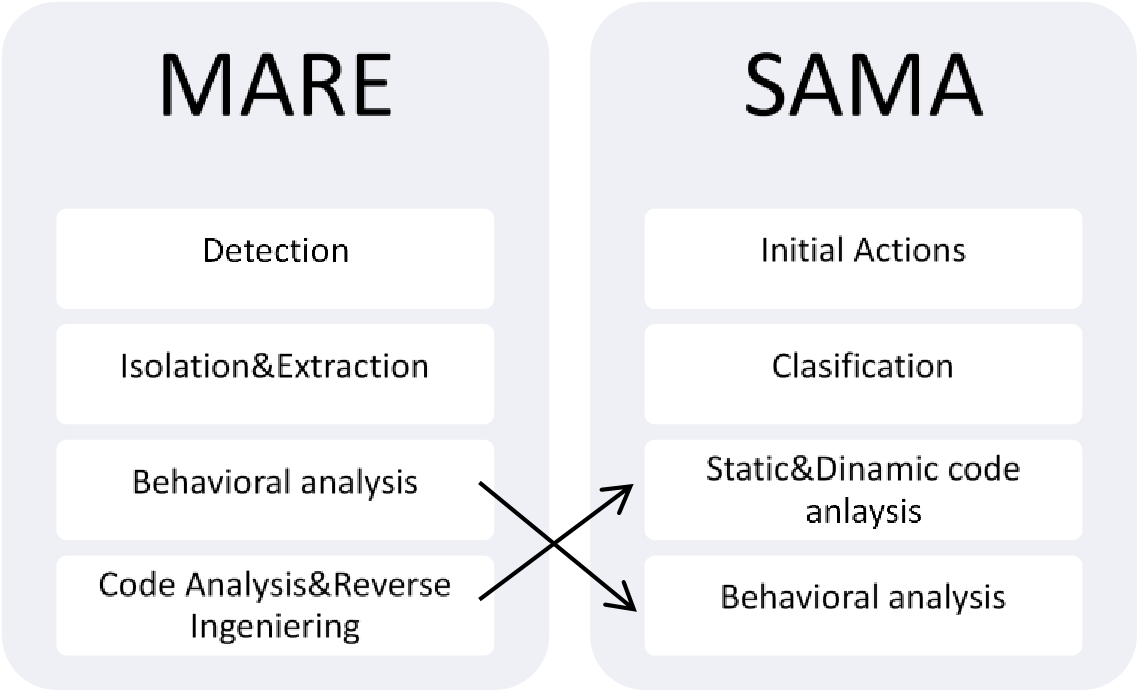
For the reasons mentioned above, it is considered not necessary to include the two previous phases in the methodology set out in this article. Apart from the aforementioned, a number of important differences are also identified:

• 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 behavioural 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 below represents MARE and SAMA comparison with the differences indicated in previous paragraphs.



 MARE**:** Malware Analysis and Reverse Engineering

 SAMA**:** Structured Analysis for Malware Analysis

**METHOD PROPOSAL**

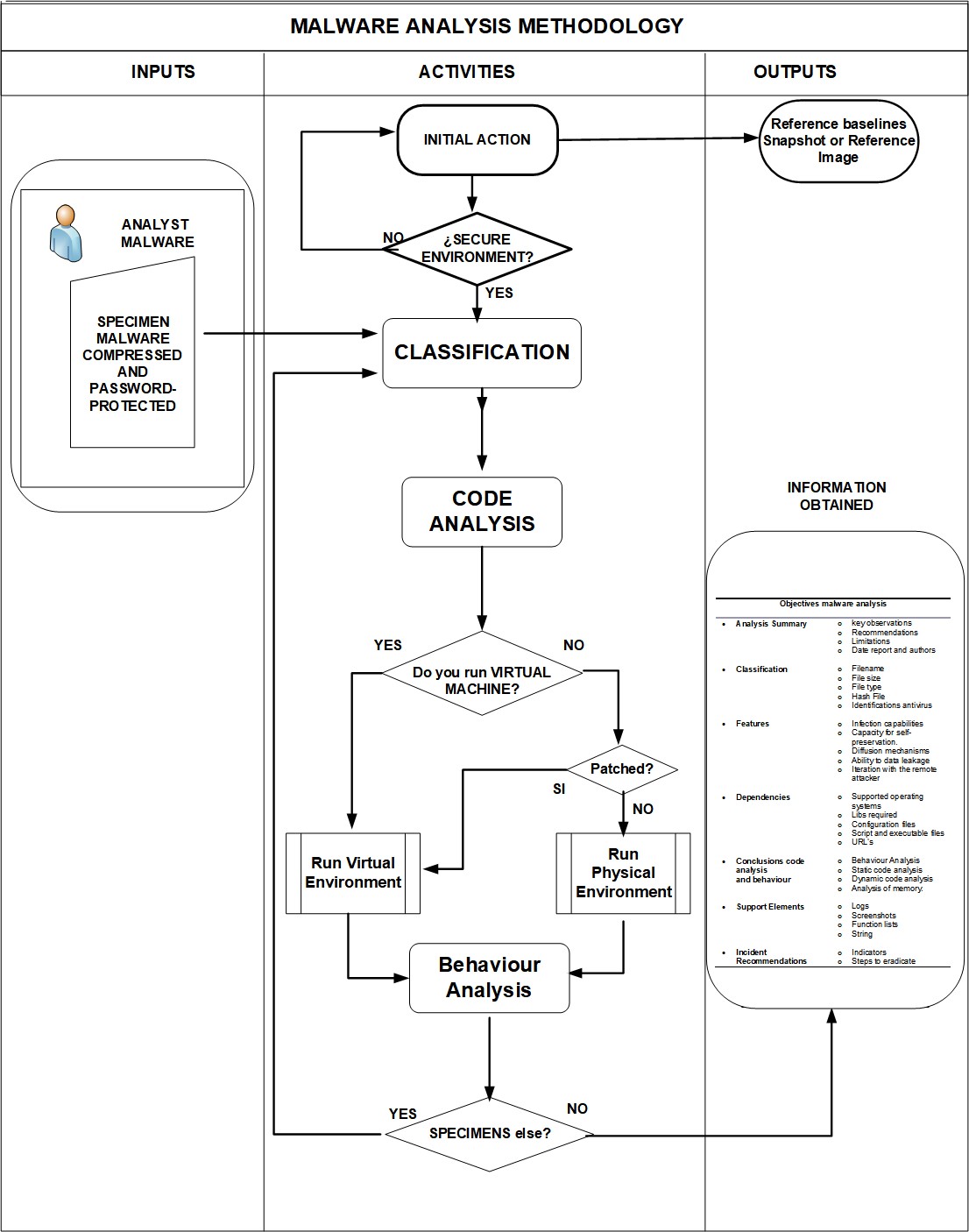
The aim of the proposed method is to provide an effective guidance during any detailed analysis of complex malware. The outcome of such a method will be valuable information that will allow the analyst to develop effective countermeasures as well as to extend our knowledge about the origin of malware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

|  |  |
| --- | --- |
| •  •  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 behaviour  • | Behaviour analysis  Static code analysis  Dynamic code analysis |
| • | Analysis of memory |
| •  •  Support Elements  • | Logs  Screenshots  Function lists |
| • | String |
| •  Incident Recommendations  • | Indicators  Steps to eradicate |

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

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 ‘behavioural 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 ‘behavioural 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



**Figure.** 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, ‘behavioural 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 behavioural 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.

• 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:

# 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”.

