The Impact of Malware Attacks on the Performance of Various Operating Systems

Maria-Mădălina Andronache[1], Alexandru Vulpe[2], Corneliu Burileanu[3]
Research Institute "CAMPUS", National University of Science and Technology Politehnica Bucharest, Bucharest, Romania[1]
Telecommunication Department, National University of Science and Technology Politehnica Bucharest, Bucharest, Romania[2]
Speech and Dialogue Research Lab, National University of Science and Technology Politehnica Bucharest, Bucharest, Romania[3]

***Abstract*—Latest research in the field of cyber security concludes that a permanent monitoring of the network and its protection, based on various tools or solutions, are key aspects for protecting it against vulnerabilities. So, it is imperative that solutions such as firewall, antivirus, Intrusion Detection System, Intrusion Prevention System, Security Information and Event Management to be implemented for all networks used. However, if the attack has reached the network, it is necessary to identify and analyze it in order to be able to assess the damage, to prevent similar events from happening and to build an incident response adapted to the network used. This work analyzes the impact of malicious and benign files that have reached a network. Thus, during the work, various analysis methods (both static and dynamic) of real malicious software will be developed, in two different operating systems (Windows 10 and Ubuntu 22.04). Thereby, both the malware and benign files and their impact on various operating systems will be analyzed.**

***Keywords*—Cybersecurity; network security; network monitoring; incident analysis; incident response**

I. Introduction

Network security includes a large number of technologies and devices that must work together based on a predefined set of rules. These rules are primarily intended to protect sensitive information in a system. However, it must be considered that security aspects cannot work with the same set of rules indefinitely because the threat environment is constantly changing, attackers are always trying to find new vulnerabilities, and network architectures are increasingly complex and different. This is the main reason why network security management tools or applications are also constantly updated.

In this security context, malicious files are an extremely difficult aspect to ignore. So, they are represented by malicious software that is created to produce, according to study [1], various exfiltration of information or to cause various interruptions. Their general purpose is to obtain ideas, to damage the reputation of a company or a system, or material gains. Malicious files can achieve these aspects precisely because of the possibility of exploiting some vulnerabilities or due to the negligence of certain people who perform various actions unfavorable to the security context (disabling the antivirus, deactivating the firewall). From study [2], malicious files and cyber-attacks have expanded their action exponentially in recent years, being encountered more and more often both within companies and for ordinary users. Their greatest impact is given by affecting critical systems such as the health area, the financial-banking area, the area of government attacks or the industrial area. Although malicious files and new types of attacks appear daily, a main part of them are based on the skeletons of older malware to which various code improvements are added. Thus, it is imperative to analyze the existing malware and understand their characteristics because they can generate patterns of future attacks. To perform this type of analysis, it is important to distinguish between static and dynamic analysis of files. This differentiation is made considering various sources such as [3], [4] and [5]. Static analysis of a malicious file involves testing it without executing it. So, this involves the analysis of the source code and other aspects such as the magic number or the hash of the file in order to identify whether it is malicious or not. Dynamic analysis is how the file is analyzed after it is executed to observe how it affects various files or various system registries. Given the fact that this method also involves the execution of the file, it is necessary for this to be done in a closed and controlled environment. Following these two types of analysis, it will be determined whether the file is malicious. The most frequently encountered types of malware present in a system are those known as zero-day. However, the static and dynamic analysis methods cannot detect this type of attack if it does not have a known pattern. For all the other types of malicious files: ransomware, trojan, virus, worm, backdoor, this analysis can be performed for the purpose of documentation and for the purpose of identifying the main characteristics necessary to prevent a possible subsequent attack. These characteristics are also considered based on the literature in [6], [7], [8] and [9].

The purpose of this work is to perform a comparative analysis of how different types of files are executed within two different types of operating systems. For this purpose, a test environment is created, which includes both static and dynamic analysis methods. The contributions of this work consist in creating a test environment (which is similar to a regular user, within a company and does not rely on existing sandboxes), choosing the most recent files from a public malware database (that are not predefined and used in another labeled database), analyzing various events and logs (after the execution of the files) and making a comparison on the key characteristics of these files.

This paper is organized as follows. Section II provides an evaluation of the specialized literature and research related to the analysis of malicious files. Section III provides the area of background work. Experiments and results are presented in Table IV. Discussion is given in Section V and finally, Section VI concludes the paper.

II. Related Work

To perform a complete analysis of malware files, a thorough review of the research activity and tools required in the process is required. Through this section, a deeper understanding of the existing research in the field will be achieved, an aspect that will also lead to the framing of this work in the current security context.

Taking into consideration the paper [10], it is found that it introduces a study related to the dynamic analysis of malicious files, evaluating an open source SIEM (Security Information and Event Management) system called Elastic Stack. Thus, by capturing the event logging mode in Windows, a complete description of the events within the system could be achieved. Malware analysis included in the paper contains a Dynamic Analysis. This type of analysis is also used in the current paper, but compared to the work [10], this paper includes experiments within both Linux and Windows operating system.

The vulnerabilities of a system are an extremely sensitive subject for traditional detection methods because they contain extremely complex functions and algorithms, which cannot be interpreted by them. In the paper [11], detection of these vulnerabilities is carried out in a binary code, by means of neural network algorithms and by means of the NDSS18 database. This database contains CVEs (Common Vulnerabilities and Exposures) for both Windows and Linux operating systems. The results indicate a good performance of the model given by machine-learning algorithms. In this paper, both operating systems will be used to see the interaction of some malware files with the default processes of these operating systems.

In paper [12], a method for detecting the behavior of malicious Android activities is proposed through a hybrid, static and dynamic approach with automatic learning. The results of the work indicate an accuracy of 97% for the detection of anomalies. The advantages of the work would be the reduced use of some permission functions and the consumption of resources, which improve the efficiency of the system. In the present work, the absence of experiments in the area of the Android operating system is identified, as it is not a desktop environment. However, the work in [12] indicates various similarities between the operating modes of malicious files in the desktop area and in the mobile area, the static analysis being carried out identically.

According to study [13], IoT devices occupy a special place in the area of malware detection because they have different characteristics depending on the environment and the platform studied, making it extremely difficult to identify. The proposed analysis method includes both static analysis, against software shells, and dynamic analysis, through nine different sandboxes. The analysis method required the creation of a database and, for this, samples from the Padawan sandbox, VirusTotal [33] and other open-source areas were taken into account. The malware detection accuracy presented in the solution, evaluated using XGBoost, SHAP and Scikit-Learn exceeds 98%. In the current work, compared to the work [13], the sandbox used for dynamic analysis is not a commercial one, but is given by common virtual machines of common users of a company. Thus, the tools used for the experiments are also different, but the key aspects pursued are similar.

In the paper [14], a solution for detecting malicious files using the YARA tool is presented. Thus, during the work, five rules were developed for malware detection in the static analysis area. From the expressed results, it is stated that the presented solution reduces the identification time, improving the detection efficiency. In this work, Yara rules will be also used for static analysis.

Considering [15], the paper presents a way of detection and prevention of malicious files before they corrupt the test system. Thereby, a Virtual Box type environment is used in which a static analysis of malicious files is carried out. The experiments were carried out by means of IDAPro, a tool that will be used in this work as well, and the malicious files were of the type of Trojan horses. The results indicated various functions, strings, imports and exports made by the malware program, and their detection was done through Reverse Engineering.

Taking into account the experiments made in study [16], it is concluded that a static analysis is carried out on a malicious file, with the aim of detecting its behavior. The way to detect malicious files is by extracting the APIs and checking them, and the authors have developed a program that analyzes PE files. The files on which this test is performed are benign, Ransomware, Backdoor and Keylogger and this approach is also found in the present work.

From study [17], it can be concluded how ransomware works. Thus, through the specified paper, certain experiments are carried out through the Kaspersky ransomware signature database and a virtual environment. So, attacks are detected, based on rules based on the signatures of these files, and an analysis is made on their functionality and prevention. Therefore, all analyzed files are either restricted or sent to the detection area for further processing. The present work also addresses ransomware files within the experiments, but the files are different within the two papers.

In study [18], an analysis of the vulnerabilities of a software system is presented. During the work, both the main types of system vulnerabilities (from Buffer overflow to DOS or Memory Corruption) are highlighted, as well as the detection methods, which include, as in this work, static analysis, dynamic analysis and hybrid analysis. The experimental data used in the paper are the authors' own data and include historical vulnerability data. These are evaluated for vulnerability detection by means of machine-learning techniques. The essential difference between the work [18] and the current paper includes their motivation: one trains Machine Learning algorithms and the other only extracts and analyzes key features to form a complex database of malware files.

The previously cited literature sources highlight various aspects of the cyber security area and various approaches similar to the one in the present paper. A good part of them is based on machine-learning algorithms and the development of robust models through this technology. However, in the real security environment, within various companies, there is still quite a lot of skepticism regarding the area of artificial intelligence and the methods used by it. Although the advantages of these solutions are immense and solve a large part of repetitive tasks, knowledge of traditional methods is still imperative to ensure a complete and deep understanding of the field.

III. BACKGROUND WORK

In order to carry out a complete analysis of the behavior of a malicious file, both a static analysis and a dynamic analysis of it are necessary. Thus, during the experiments carried out in this work, both types of analysis will be performed on legitimate or malicious files from the Windows 10 and Ubuntu 22.04 operating systems. The goal is to identify the essential characteristics of various types of malware in order to create a complete database of characteristics. This will be able to serve, later, to create an automatic intrusion detection system within a computer network of various sizes. These types of analysis, both static and dynamic, will be implemented using various open-source tools, and the results of the study will be analyzed comparatively.

The analyzed files were downloaded from the MalwareBazaar Database [25] and include various types of files related to Windows and Linux operating systems. The method of choosing the malware samples used did not follow any specific algorithm but included the identification of similar types of malware for the two types of operating systems used. This need appeared after consulting the literature regarding similar, relevant articles in the field, finding a predilection for known or even outdated databases, which are permanently tested. Therefore, out of the desire to perform the experiments with real malware samples and not with pre-tagged databases, the MalwareBazaar Database [25] source was chosen.

Within this database there are many different malware samples, but the categories are represented by the main types of malware. Thereby, similar samples were chosen (from the same malware family, from the same category, appeared on the same day, etc.), both for Windows and for Linux. By means of this approach, the research can evaluate various ways of functioning of malicious files in its own way and can lead to unpredictable conclusions, which can contribute both in the literature and in the area of commercial applications.

A. Static Analysis

The static analysis of this work will be carried out using various open-source tools. These tools were chosen due to their popularity and effectiveness in detecting key characteristics of malicious files for both Windows and Linux operating systems.

- IDA Pro [19] is a tool that is able to disassemble the files and identify the way in which the instructions are executed in the assembly language. This tool is suitable for both Windows and Linux OS.
- PeStudio [20] is an integrated tool for static analysis of malicious software that indicates various information about files (file headers, file entropy, character strings or imported or exported functions). This tool is characteristic of the Windows operating system, but for Linux it will be an used an alternative named Malcat [26] and Detect-It-Easy [27].
- YARA rules [21] – methods of identifying malicious programs by means of commands written in a .txt program that include instructions for identifying similar sequences used or that have similar patterns.

B. Dynamic Analysis

Dynamic analysis of malicious files means executing them to be able to observe the actual behavior. So, it is found that for this aspect, it is necessary to create a safe and isolated sandbox environment. Except for this aspect, it is also necessary to introduce various tools that can be used to identify the described behavior at runtime. Also, tests will be carried out through which it will be possible to observe whether or not these types of files raise various alerts through the antiviruses specific to each OS.

The essential characteristics that must be monitored in a dynamic analysis are:

- Network traffic monitoring: To be able to track IPs and DNSs contacted for file downloads or data exfiltration.
- System file monitoring: Monitoring how certain registries are created, modified or deleted.
- CPU monitoring: To be able to observe if it becomes overloaded by certain unknown requests.
- Memory area monitoring: To be able to identify activities that are not visible.
- Code monitoring: If it can be decrypted, it provides important information about how the malicious file works.

This information can determine several aspects of the actual attack and whether the system in which it was identified is the target or only an intermediate step towards the final target.

It is mentioned that some of the tools used in the created sandboxes are characteristic of the operating system, and another part of them is common to both Windows and Linux. The chosen tools have the advantages of being open-source and, according to the literature, have increased efficiency in monitoring malicious activity.

- Regshot [22] is a dynamic analysis tool that performs a comparison of a created file with the status of registers and system keys before and after the execution of the malicious file. This tool identifies the changes made by the malicious file and is characteristic of the Windows operating system, but it can also be adapted for Linux, through a series of commands.
- Wireshark [23] is a tool that can be used to capture and analyze data packets from a network. In this work, this analyzer will be used to identify the exchange of messages between the malware file executed within the network and any IPs to which the request is made. Although this tool is normally used in the Windows operating system, adaptations can also be found for the Linux area.
- FakeNet-NG [24] is a tool that simulates Internet traffic, specially created for the malware analysis area. Thus, the malicious programs consider that the workstation is connected to the Internet and try to access various resources. These are later captured to be analyzed in the file behavior characteristics area. This tool can be implemented both in the Windows operating system area, as well as in the Linux OS area.

IV. Experiments and Results

As can be seen from the Fig. 1, the steps necessary to perform the two types of analysis are: choosing the file type, choosing the analysis method with the related implications (analyzing the file, without executing it, and analyzing it by executing it, in a sandbox environment) and choosing the appropriate tools for each analysis, separately. Considering the fact that, in the case of dynamic analysis, the execution of malicious files will involve affecting some registers or even the entire test environment, the experiments will be performed in a virtual work environment. This offers the possibility of returning to the previous settings through the Snapshot function.

This work approach will be preserved for both operating systems. In addition to the basic tools, which will be used to detect malware files, additional resources will be used, such as various functions from Microsoft or the detection of differences in the state of the CPU and memory during the attack. Also, in order to be able to analyze the involvement of anti-virus software in the experiments, this resource will also be used, and its involvement will be analyzed comparatively, depending on the operating system.

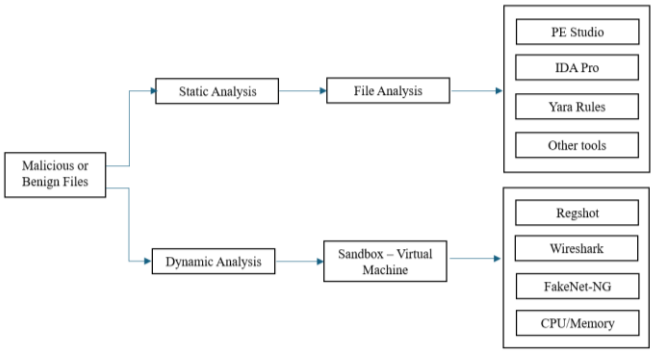

Fig. 1. Workflow diagram of the system.

A. Experiments in the Windows Operating System

a) Static analysis: In the framework of the static analysis present in the Windows operating system, all the tools that were previously presented, in Section III, will be used.

The files used for the experiments in this scenario are of different types, containing both non-malicious and malicious files downloaded from an online database or from public resources (for benign files). For the experiments carried out, the legitimate files are of various types – from executable files to document-type files, archived or photo-type files. The benign file that has .exe extension is the strings64.exe file from the Strings library [28]. Apart from benign files, several different types of malicious files were also chosen from MalwareBazaar Database [25]. Each of these was analyzed using PEStudio, strings64.exe, YARA Rules and IDA Pro. By means of these tools, it was possible to analyze the hash values of the files (MD5-Message Digest Method 5, SHA1-Secure Hash Algorithm 1, SHA256-Secure Hash Algorithm 256), their imphashes, entropy values, imported libraries or APIs. In Table I, you can find the values of file types, their imphash or MD5 hash and the entropy values for each one of them.

TABLE I. Static Analysis of Malicious files in the Windows Operating System

File Type	Characteristics	
	MD5/ Imphash	Entropy
Benign .exe	4d936b630620ff7c59da22b1206636e	6.42
Benign .doc	6698b2a4a15f86ddd4fc90ad65521cf7	7.76
Benign .txt	42631b1af161defcf4844fb1e26cfc70	4.40
Benign .jpg	2be5d32efb9c3f4b6acf94a1d1e707b4	7.96
Benign .mp4	37d7f751daa745beba4cf44b6373f2be	8.00
Benign .pdf	f9067cb2369fa0ec4e3753f67638fbca	7.34
Benign .zip	8273b4301f7a6d678c0523bb07fedd80	7.99
Benign .html	40c15f040bc8f4aeb81909fb36aa9905	4.29
Benign .iso	089a3a344f301a34dc40cc3702f2b873	7.93
Botnet .exe	5e146bf6c1ef160162ed271c0ddde908	3.54
Backdoor .dll	f34d5f2d4577ed6d9ceec516c1f5a744	4.42
Keylogger .exe	008a6a7f7e2610edadf3e2f26c73b646	7.63
Malware .exe	11ea24073ee65343ee563e3160c77fde	7.81
Ransomware .exe	914685b69f2ac2ff61b6b0f1883a054d	7.18
RAT .exe	8d5087ff5de35c3fbb9f212b47d63cad	6.59
Trojan .exe	d6d4965d7fe2d90a52736f0db331f81a	6.59
Worm .exe	2dfc2c74864b84f5530ab40a343c56d8	5.36

Imphash is, from study [31], the method by which a hash is calculated based on the libraries and APIs imported by the file. This is useful to determine if two apparently different files come from the same source or belong to the same family. Within the values presented in Table I, the imphash values are not similar and, therefore, the analyzed files are different and do not belong to the same malware family. Given the fact that the imphash value can only be calculated for executable files, for benign files, which also contain other types of files (except for .exe), the values related to the MD5 type string were added.

The entropy value gives, according to [30] the level of randomness of a file. Thus, the higher the value of the file, apart from the interval [0, 8], it can be concluded that the file is encrypted or packed and can be identified as malware. This aspect is not respected within the values in Table I because the benign files, which are also executable, have, sometimes, a higher value than malicious files such as backdoor or worm. In the case of benign files, the highest entropy values are recorded in the case of .doc, .jpg, .mp4, .pdf, .zip and .iso files.

The explanation for this phenomenon is that, in the case of .jpg or .zip files, the compression algorithms used increase the randomization of the data in order to compact them in a safe way. In the case of .mp4 type files, they encode various waveforms, which leads to a random appearance of the file. For files of type .doc or .pdf, the entropy can have a high value due to various images or text with different fonts. In the case of .iso files, they contain several types of smaller files (which can have various extensions), so its randomization index will be high.

Since the static analysis of a file, especially in the case of those considered malicious, includes aspects such as access and manipulation of memory resources, reading the source code or imported functions or libraries, a deeper look at these resources is necessary.

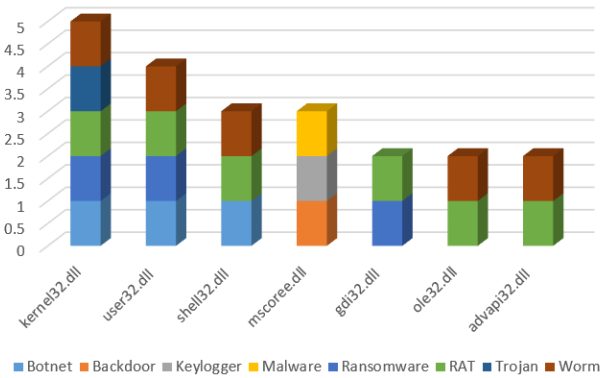

Fig. 2. DLLs imported by malicious files in Windows operating system.

Thus, bearing in mind that the files are of different types and, therefore, the imported functions are also diverse, a comparative analysis at their level would be extremely difficult to achieve. However, what can be analyzed is the domain of libraries imported by each individual file. For the current work scenario, this data can be found in Fig. 2. From this figure, you can see what type of libraries can be imported by the file, if it will be executed. All these libraries are considered legitimate libraries and compatible with the Windows operating system, having various specific functions. As an example, the kernel32.dll library deals with memory management and the control of various processes, while user32.dll considers the user interface and various inputs given by them. However, in the case of malicious files, these libraries have different functionalities. Basically, an attacker names a part of the malicious file identical to a known library but attaches it to a different directory within the file suite. This is also a direct way to detect if a library is legitimate or not. Moreover, even the fact that a file wants to create or overwrite a system library is a fairly solid indicator of the presence of malicious software.

Taking into account the fact that, through static analysis, the considered parameters can create confusion regarding the malware characteristics of a file, a dynamic analysis is also needed, which can clarify certain aspects. Therefore, the analysis of these files will have to be carried out in the next paragraph, in order to be able to observe how legitimate and malicious software functions in a real environment.

b) Dynamic analysis: Dynamic analysis is important because it allows evaluating the behavior of a software program during its execution. In the case of this scenario, the motivation for its realization is the more defined perspective on the key aspects of some legitimate files, compared to some malicious ones.

The aspects identified in the dynamic analysis will be related to the presence or absence of system changes. Thereby, by means of values in Table II, it will be identified if there are modified key values or registers (identified by the Regshot tool), if the connection to various Internet sources (Fake Net and Wireshark) is started, if there are changes to the CPU resources and memory and if the presence of the malware is identified by the antivirus.

TABLE II. DYNAMIC ANALYSIS OF MALICIOUS FILES IN THE WINDOWS OPERATING SYSTEM

File Type	Characteristics			
	Values/ Keys Modified	*Network Activity*	*CPU/ Memory*	*Detected by AV*
Benign .exe	√	X	X	X
Benign .doc	√	X	X	X
Benign .txt	√	X	X	X
Benign .jpg	√	X	X	X
Benign .mp4	√	X	√	X
Benign .pdf	√	X	√	X
Benign .zip	√	X	X	X
Benign .html	√	√	X	X
Benign .iso	√	X	X	X
Botnet .exe	√	√	√	√
Backdoor .dll	√	√	√	√
Keylogger .exe	√	√	√	√
Malware .exe	√	√	√	√
Ransomware .exe	√	√	√	√
RAT .exe	√	√	√	√
Trojan .exe	√	√	√	√
Worm .exe	√	√	√	√

During the experiments carried out in the Windows sandbox, all files were executed sequentially. After running each type of malware, the test environment needed to be replaced via the Snapshot function. This aspect was also valid for benign files, even if their execution does not endanger other files. However, it was desired that the execution of one of the files to not influence the execution of subsequent files in any way. The files that strongly affected the test environment were Worm and Ransomware. In the case of these two types of experiments, the results, given by running the malicious files, led to temporary interruptions of the virtual machines or even to the irretrievability of some data.

A slightly more atypical aspect, resulting from experiments, was in the case of the experiments carried out with the Worm type file, because the Fake Net area could not stop making recordings, having this behavior for a few minutes. During all this time, many HTTP (POST) or DNS related events were recorded, through which the malicious file tried to access ihcnogskt.biz or kkqypycm.biz multiple times.

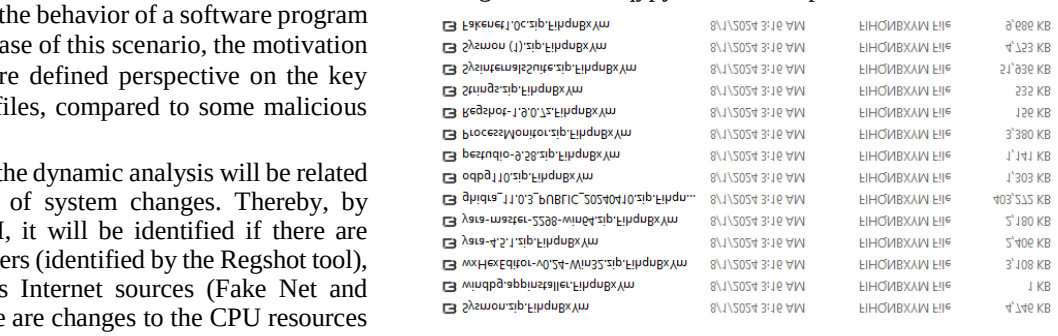

Fig. 3. Changing the file extension after conducting experiments with the malicious ransomware type file.

In the case of the experiments carried out with the Ransomware type file, after execution, it encrypted all the files of the working system, giving them a new type of file extension that can be seen in Fig. 3. As a result of this aspect, the virtual machine became inaccessible, and the files could no longer return to the previous extension type. Even after trying to return to the default settings of the virtual machine, the initial files could not be recovered. Thus, it was necessary to create another working environment.

It is mentioned that the duration of all the experiments done in the test environment did not last more than 5-10 minutes/experiment out of concern not to irreparably affecting the test environment.

B. *Experiments in the Ubuntu 22.04 Operating System*

a) Static analysis: In the case of the experiments carried out in this scenario, benign files, that are completely different from those used in Windows operating system, contain different file types, from a python executable called impelf.py [29], to some other file types corresponding to the Linux operating system. The malicious files are also different, but the way in which they were chosen was by comparing them with the previous chosen ones (be from the same malware family, be announced around the same time, etc.). Each of these files were analyzed using Malcat, Detect-it-Easy, readelf, md5sum, sha1sum, sha256sum. Since .elf and benign files have no information about their executable mode, the imphash hash could only be generated for Keylogger and Worm files, which were .exe. For the rest of the files, in Table III, their md5 hash was added.

Taking into consideration that a complete static analysis needs to include aspects related to various properties of the files (the imports made or file structures), it is also necessary to understand the malicious files in this scenario. Thus, an important amount of the files analyzed within the experiments have the extension .elf. From [32], these files are characteristic of the Linux systems, being executable files or libraries. This type of file is structurally divided into two parts: the header area and the segment area. The header contains various metadata, and the segments describe various memory operations, which are performed during execution. These segments are of various types, but the most common ones are those found in Fig. 4.

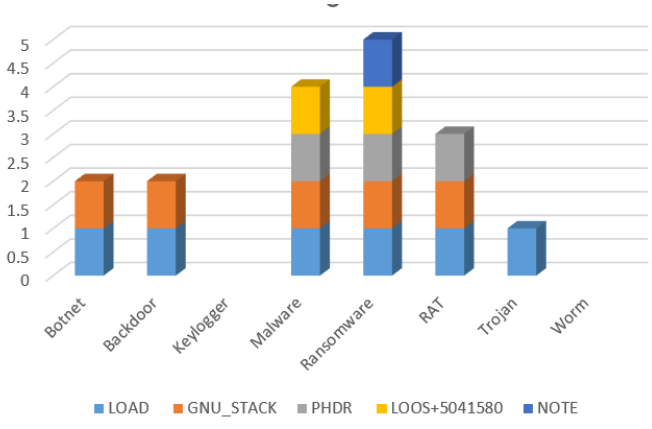

Fig. 4. Segments imported by the malicious files in the ubuntu 22.04 operating system.

Thus, as can be seen from the figure, the first type of segment is LOAD segment and it indicates the memory location where the file will be loaded and its various permissions, while the NOTE type segment includes information that can be used by the system kernel such as version, debug data, etc.

In the context of the analysis of some malicious files, although certain segments appear to be legitimate, being usually found in executable files, they can be modified to have the character of malware. Thereby, the malicious behavior can be hidden in "stuffing" segments in order to be able to pass undetected by various types of antiviruses. If these segments also have file modification or execution privileges, this behavior may indicate the presence of malware. Another detection index of these types of files is the atypically large size of some segments that are apparently legitimate.

In addition to the general segments, such as LOAD or GNU_STACK, which are present in almost all files of the .elf type, there are also segments that are a little more atypical for an ordinary file. Among these types of segments is LOOS+5041580. The fact that, even at the level of Internet resources, extremely little data about this type of segment can be identified, may indicate the presence of malware. It is emphasized that, even the searches based on the LOOS segment, without the numerical suffix, did not indicate details about a usual work segment.

Similar to the experiments carried out in the Windows operating system, the presence of malicious activity cannot be clearly defined to be able to conclude that one of the chosen files is truly legitimate or has malicious characteristics (entropy has different values, segment type files also have a legitimate character, etc.). Thus, the dynamic analysis of the files is also needed to be able to have a more explicit conclusion on the mode of operation and the influence they have on the Ubuntu 22.04 operating system.

TABLE III. STATIC ANALYSIS OF MALICIOUS FILES IN THE UBUNTU OPERATING SYSTEM

File Type	Characteristics	
	MD5/ Imphash	***Entropy***
Benign .py	b6014a53db0e1797301ec118f2625c45	4.53
Benign .txt	6045aa2bdbbfa5839a382fbc383307ac	4.75
Benign .mp4	a6b8790aeeffaa6b08b1b7dfa2b0a1f7	7.99
Benign .sh	fc331af161311d2000fb18d02764a062	5.49
Benign .tar	38544f88237f2b1184c8822289a1899d	7.99
Benign .iso	05fde34ce38913489a1a988175240f27	7.99
Benign .pdf	e9ef095f7dec56b483d2c31f915e177c	7.98
Benign .jpg	0fe826c9fad792732c9081b59bbcb613	7.85
Benign .conf	e95d5425c026ab1142a025d49bf23dc9	4.81
Backdoor .elf	9a85bf5e1b4ca4db7b5654aa48df5f2e	5.65
Botnet .elf	4d58d0cae526ee6364f7c738b83f2961	6.01
Keylogger .gz	888988a74b67d0e75f5293688ab07b71	4.12
Malware .elf	171d2a50c6d7e69281d1c3ef98d510f2	6.00
Ransomware .elf	56cabcf95add39a6feb09391ccc40dcd	6.18
RAT .elf	9f539613aae69eec04ed66550f814f6b	7.98
Trojan .elf	1655222d44cfc33dcc3d10f8a4f2e2db	1.51
Worm .tar	5a46892a133f6e380a5a2acb389c5af6	6.93

b) Dynamic analysis: Following the experiments carried out, in the Ubuntu 22.04 operating system, the same structure found in the dynamic analysis in the Windows 10 sandbox was kept. So, both the benign files and the malicious files were run one by one.

The results of the experiments can be found in Table IV, where the key elements of the detection are reproduced. Thereby, it can be observed that for all types of files, there are modified values or registry keys, and different CPU and memory values recorded during the attack. However, the situation is different for the network activity when the file is executed and the files that are detected by the ClamAV antivirus indicators. Thus, although some files are declared as malware in public databases, in our experiments, it can easily pass the security system of the virtual machine. The less favorable aspects happened in the case of running two different types of files, the one with malware and the one with ransomware. So, after running the malware file, the message "Exporting key" is recorded, after which the workstation becomes unusable, performing a restart. After the execution of the ransomware file, not even this message is displayed, but after restarting the test environment, some files are unusable. For these experiments, recording the malicious behavior was quite difficult due to the continuous restart of the virtual machine. Thereby, in order to succeed in capturing the parameters for Table IV, it was necessary to repeat the tests several times.

It is mentioned that the recording duration of the malicious events was carried out in an interval of approximately 5-10 minutes after the execution of the malware file. The values presented in Table IV reveal only the aspects recorded within this interval. It is also mentioned that, even for the benign files, after each execution, the test environment was returned to the default settings, so that the experiments are not inter-influenced.

TABLE IV. DYNAMIC ANALYSIS OF MALICIOUS FILES IN THE UBUNTU OPERATING SYSTEM

File Type	Characteristics			
	Values/ Keys Modified	***Network Activity***	***CPU/ Memory***	***Detected by AV***
Benign .py	√	**X**	√	**X**
Benign .txt	√	**X**	√	**X**
Benign .mp4	√	**X**	√	**X**
Benign .sh	√	**X**	**X**	**X**
Benign .tar	√	**X**	√	**X**
Benign .iso	√	**X**	**X**	**X**
Benign .pdf	√	**X**	**X**	**X**
Benign .jpg	√	**X**	**X**	**X**
Benign .conf	√	**X**	**X**	**X**
Backdoor .elf	√	**X**	√	√
Botnet .elf	√	**X**	√	√
Keylogger .gz	√	√	√	**X**
Malware .elf	√	**X**	√	**X**
Ransomware .elf	√	**X**	√	**X**
RAT .elf	√	**X**	√	√
Trojan .elf	√	**X**	√	√
Worm .tar	√	**X**	√	**X**

V. DISCUSSION

Taking into account the aspects analyzed in the previous section, a comparative analysis of how the benign files and the malicious software affected the test systems is necessary. Although, in Fig. 1, these analyses are treated independently, in fact, an analysis of malware files is complete only by encompassing all aspects analyzed through all experiments performed and tools used. Thus, in the static analysis, carried out in the case of both analyzed operating systems, it is found that the test files have different entropies, included in the range [3.54-8.00] for Windows and [1.51-7.99] for Ubuntu. Although the initial expectations were that malicious files have a higher entropy value (corresponding to the way in which the malicious file is encrypted and packaged to avoid detection), it is observed that, in the case of both operating systems, some of the files that do not contain malware has a higher entropy value than some files containing malware. This aspect is caused by the randomization achieved by compressing certain files (archive type), by including certain metadata such as photos or different fonts (in the case of document type files) or by processing certain different waveforms (in the case of .mp4 files).

Regarding the hash part, for both operating systems it was necessary to discover various types of hashes. Although, initially, for the tests, the representation of the data by imphash was chosen, this aspect proved to be inconsistent for files that are not Portable Executable. This aspect had an impact in both operating systems because, given the fact that the construct part of some files is not executable or is a binary one, imphash cannot be represented. Therefore, it was chosen to complete the table with the MD5 value for these types of files. A good aspect that needs to be mentioned is the purpose for which it was necessary to find out the hash of a file. Basically, according to study [3], the hash of files within the tables had two directions of development. The first direction is given by the fact that a good part of the malicious files is recognized in some tools by hashes. The second direction, which derives from the first, is the fact that, depending on various hash values (MD5, SHA, SHA256, SHA512, imphash), the characteristics of malicious files can be found in various public databases. The most used example, in this case, is through the VirusTotal tool through which, after searching for a hash, it shows extremely relevant information about the respective file. This tool can also be included in the static analysis area, being an extremely useful aspect in discovering similar files or the characteristics of a certain type of malware.

However, this approach was not necessary in the case of benign files because these files could be represented by the actual name. Their values, that are, in Table I and III, also in the form of hash have the purpose of a unitary way of data representation.

Considering that a complete static analysis must also include aspects related to the actual content of the file, it is necessary for it to include various information such as the strings used, APIs or DLLs imported or the source code, as in study [3], [4] and [10]. Therefore, in the case of both scenarios, these aspects were also analyzed. Unfortunately, due to the difference in the structure of the analyzed files, they cannot be the basis of a comparative analysis, being also extremely different. Thus, the

approach consisted in delimiting the two scenarios and performing a comparative analysis only according to the malicious files within the respective scenario. Therefore, within the Windows operating system, the most frequently encountered imported DLLs were chosen, to be able to observe if there are common characteristics between the analyzed malware files and the intentions of each malicious file, separately. It was concluded that these files import both common libraries and independent libraries. Within Fig. 2, only the common ones were included, the most frequently encountered being kernel32.dll, user32.dll, shell32.dll and mscore.dll. These have as their main characteristics file operations, privilege escalation, avoid analysis, user interface or memory management operations. The importance of these DLLs imported by malicious files can lead to conclusions about the behavior of the malicious file (as in [16]), its inclusion in a certain family, the creation of similarities with other files known to be malicious or the creation of action patterns of certain types of files.

In the Ubuntu 22.04 operating system, the analyzed files, from a static point of view, imposed the analysis of some files with .elf extensions. These are, in essence, binary files and therefore it is quite difficult to extract some essential features, in a similar way to the previous scenario. The initial analysis of these files consisted of examining the file structure (the size of the headers, their number, etc.) or the strings used, but these characteristics could not define a comparative analysis that would indicate an atypical character of the file. Therefore, the chosen feature consisted of presenting the segments loaded in memory during execution. Considering Fig. 4, it is indicated that the most frequently used segments for malicious files were LOAD, GNU_STACK and PHDR. Their essential characteristics include indications on an executable code, on working with the memory stack or on various memory locations that need to be accessed or written. In the case of malicious files, these segments include the possibility of attacks that include self-modifying, returning addresses from the memory stack or escalation of privileges. Therefore, it is extremely important for these characteristics to be known, in order to get an overview of the behavior of the malicious files. Apart from the mentioned aspects, in Fig. 4 you can see the absence of segments for two types of files - Keylogger and Worm. This aspect is due to the fact that the two types of files are not of the .elf type, but of the .gz type. Thus, they must be characterized in a different way.

Considering the aspects mentioned for static analysis and papers [3], [4], [5] and [15], it is necessary to emphasize that they are quite insufficient for a complete characterization of a file type. Therefore, a dynamic analysis would complete the unknown aspects or for which the static analysis is insufficient and would offer the possibility of an overview of the malicious file's mode of action.

Dynamic event analysis involved conducting experiments on how file execution works in a sandbox environment. However, in order to distinguish between a malicious file and a benign one, it was necessary to establish some indicators of the presence of malicious software, similar to [4], [5], [10] and [17]. Therefore, detection by OS-specific antivirus was one of the main parameters that should be considered. Also, the steep increase in machine performance, given by the CPU and memory values, is one of the necessary indicators of the presence of non-compliant activity. Other important aspects were given by the monitoring of the traffic within the network and the values or keys changed at the level of the operating system registries.

Although analyzed independently, the performance parameters could be considered insufficient (CPU and memory values increase even when running a legitimate application), in the analysis, they work as a whole. The discrepancy in results between the two operating systems is due to both the way the malicious files work and the actual file type. Not having the same applicable file available for both operating systems leads to different experimental values.

Taking into consideration all this, it was found that, if at the level of the Windows operating system, aspects are more uniform, the malware files leaving positive traces for all the mentioned indicators, in Ubuntu operating system, things are less favorable because some malicious files are not even recognized by the antivirus. Thus, they can go unnoticed and can attack systems without being detected. The worst aspect is that ransomware and malware files that, after execution, lead to the temporary inaccessibility of the virtual machine, are, initially, undetected by the antivirus. This does not mean that keylogger or worm files cannot affect systems or perform various lateral movements. Therefore, in the case of the Linux-type operating system, the fact that the antivirus used fails to effectively identify certain malicious files is an important problem.

The importance of a preliminary analysis of a file is recalled here to avoid, as much as possible, the unfavorable aspects of executing a malicious file. Thus, considering the results presented in Fig. 3 and [17], attention is drawn to the fact that, for the execution of ransomware, the working environment or even the network may become unusable. And, if the file backup area is located within the same network, it will most likely be affected as well. In all experiments performed, the performance of the virtual machine was affected by running the malicious software, with related CPU values increasing instantly, certain processes stopping or becoming temporarily inaccessible.

For benign files, there was also an increase in CPU and memory values, especially for larger files. So, in the case of the experiments carried out through the Windows operating system, the CPU and memory values increased for the .pdf and .mp4 files, while, in the case of the Ubuntu 22.04 operating system, the same values increased for the .py type files, .txt, .mp4 and tar. The only common aspect is the fact that, in both cases, the CPU and memory values increased when executing the .mp4 file because they are quite resource consuming. No conclusions can be drawn for the rest of the files, as the CPU and memory values may also increase depending on the characteristics of the file itself (its size, the information contained, etc.).

As for the area of activity of the network resources, it is different within the analyzed operating systems. Thus, if in the first scenario all malicious files register the need for access outside the network (trying to access various resources from the Internet), in the case of the second scenario, this event was registered only for the keylogger type file.

For the benign files, they did not register the need to access additional resources from the Internet, with the exception of the

html file, in Windows operating system. Although, initially, its execution can be done in an offline mode, as additional aspects are opened within it, internet resources are needed. It is very likely that, in this case too, there will be changes to the system after the working time allocated to the realization of the experiments.

Regarding the Values/Keys Modified parameter, whose value is based on the Regshot tool, but also on various processing tools such as Process Monitor, it is observed that, for all types of files analyzed (both benign and the malicious ones), there are changes at the level of various registries or keys of the operating system, regardless of whether we consider the Windows or Ubuntu 22.04 platform.

Considering all these aspects, it is necessary to open a discussion about how the execution of these files affects the performance of the analyzed test systems. Thus, taking into account both the aspects mentioned in the static analysis and those in the dynamic analysis, it is appreciated that, if a file is unknown to a user, a simple check of its hash or entropy can be a sufficient index good to see if the file is, in fact, malicious software. These aspects can be done both individually and using various tools or resources from the Internet. It is particularly important that the unknown file is not executed, if no data is known about it, in order to protect both the system through which the file was received, and the network of which it is a part, if applicable.

If the file has been executed, the way in which the performance of a system is affected includes some effective evaluation indicators. Among them, the high CPU and memory values are counted because the malicious file requires various resources to be able to increase its coverage area or to access various types of sensitive information. An immediate effect would be longer response times for applications or even the impossibility of accessing them, various crashes or even restarts of the operating systems. Another performance parameter, taken into account in the case of this work, is the access area to unknown internet resources. As a result of this fact, Internet resources can become difficult to access, and latency can increase.

Other ways in which operating systems are affected include a slower boot, changes to registries or their keys, deletion of various information or files and even their full encryption. It needs to be mentioned that since a system is compromised, it can open various backdoors for further infections.

If, when running, the file is detected by the antivirus, it will block its effective execution and delete it. But, as was observed from the experiments, sometimes antiviruses can also let malicious files pass. Therefore, increased attention is required in the case of unknown files.

Even benign files can affect the performance of operating systems. Thus, although they do not cause the same damage as malicious files, the fact that they are large files or archives with many files of various types affects the operating system. Another way in which they can impact the performance of the operating system is if they contain outdated software. These not only affect the total performance, not having any updates, but they can also create many vulnerabilities, which facilitate various types of attacks. Also, if there are files that contain errors or have modified extensions, they can also corrupt other files, even from the installer area of the operating system.

Considering all this, it is imperative that the files be verified, especially when they come from unknown sources. The verification methods can be both simple (checking the hash and comparing it with the effective extension of the file), as well as complex (which can include reverse engineering). It is also necessary to keep in mind that even benign files can affect system performance. Therefore, increased attention is needed regarding the public sources for downloading them and the way of implementation, both within the respective system and in the case of various communication networks.

In this paper, certain limitations related to the field of cybersecurity are also admitted. Among these, the method of selecting malware samples is listed, since they focused only on executables from the Windows and Linux operating systems. Also, the selection method did not have a predefined set of rules, the files being chosen according to the malware family they come from, their category and the publication date (the aim being to perform experiments with the most recent files). Thus, it is possible that files with relevant characteristics were omitted from the study. Another important limitation is given by the isolated environment in which the experiments were performed, because it was not possible to record the working mode of the malicious files at the network level and the impact that it may have on other network resources. Mentioning these limitations is important for identifying how to perform future experiments.

VI. Conclusions

In this work, a comparative analysis was presented on the way of working of various operating systems with malicious or legitimate files. Therefore, for this aspect, it was necessary to download samples of malicious files from public resources and to choose various types of legitimate files, to be able to make a comparison between them. Later, they were analyzed at the level of the Windows and Ubuntu operating systems, both through static and dynamic analysis.

The static analysis was carried out by means of open-source tools and made an identification of the malicious and benign files, without them being executed. This type of analysis was based on the identification of file hashes, their entropy, the strings used within the system, the imported libraries and the APIs called. However, given the fact that they were quite diverse and ambiguously related, they could not be the basis of a comparative analysis between the two types of operating systems used. Thus, the aspects taken into account at the level of this work were the imphash or the MD5 hash, the entropy of the files and, depending on the operating system, the imported libraries and the structural segments of the file.

In the dynamic analysis area, two sandbox virtual machines were built, with two different operating systems (Windows 10 and Ubuntu 22.04). Within these virtual machines, an isolated file execution environment was created, and the network area was simulated by means of a public tool. The analysis required the execution of the files, and this aspect led, in some cases, to the temporary unavailability of the virtual machines or even to the corruption of all the files within it.

Taking into consideration all this analysis, a discussion was also carried out on the impact that various types of files have on operating systems, either from the perspective of legitimate files or from the perspective of malicious files. The conclusion of this discussion led to underlining the importance of preliminary verification of files received from unknown authors or downloaded from various less obscure public sources, in order to prevent damage to the actual operating system or the network of which it is a part.

The most surprising conclusions that can be drawn from the experiments are given by the way in which the ransomware file, run in the Windows operating system, led to the impossibility of accessing them and by the fact that some malicious files went unnoticed by the Linux operating system antivirus. These aspects may impact the academic and work environment to carry out similar experiments to try to make antiviruses more efficient or to analyze new methods for restoring files affected by ransomware attacks.

For future work, other types of malicious files will be taken into account (adware, rootkits, bots), and the analysis will also be performed at the level of other types of operating systems (MAC OS, for desktop area, and Android and iOS, for mobile area). The final goal is to create a robust database, which contains more recent malware samples and their key features, which can be adapted to current security systems and can be considered can be considered a working basis for other experiments that will be carried out in the literature.

REFERENCES

[1] F. Ullah et al., “Data Exfiltration: A Review of External Attack Vectors and Countermeasures,” Journal of Network and Computer Applications, 2017, doi: 101. 10.1016/j.jnca.2017.10.016.

[2] A. Ghosh, “An overview article on 600% increase in Cyber Attack in 2021,” 2021, doi: 10.13140/RG.2.2.18205.52968.

[3] R. Sihwail, K. Omar, A. Zainol, A. Khairul Akram, “A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis,” 2018, doi: 8. 1662. 10.18517/ijaseit.8.4-2.6827.

[4] A. Belea, “Methods for Detecting Malware Using Static, Dynamic and Hybrid Analysis,” International Conference on Cybersecurity and Cybercrime, 10, 258–265, 2023, doi:10.19107/CYBERCON.2023.34

[5] R. Baker del Aguila, Carlos Daniel Contreras Pérez, Alejandra Guadalupe Silva-Trujillo, Juan C. Cuevas-Tello, Jose Nunez-Varela. “Static Malware Analysis Using Low-Parameter Machine Learning Models,” 2024, Computers 13, no. 3: 59, doi: 10.3390/computers13030059

[6] F. Almeida, M. Imran, J. Raik, S. Pagliarini, “Ransomware Attack as Hardware Trojan: A Feasibility and Demonstration Study,” IEEE Access, 2022, doi: 10. 44827 - 44839. 10.1109/ACCESS.2022.3168991.

[7] N. Ravichandran, T. Tewaraja, V. Rajasegaran, S. Kumar, S. Gunasekar, S. Sindiramutty, “Comprehensive Review Analysis and Countermeasures for Cybersecurity Threats: DDoS, Ransomware, and Trojan Horse Attacks,” 2024, doi:10.20944/preprints202409.1369.v1.

[8] A. Sheikh, “Trojans, Backdoors, Viruses, and Worms,” 2021, doi: 10.1007/978-1-4842-7258-9_5.

[9] B. Rajesh, P. Praveen Yadav, C. V. Chakradhar, “Malicious Computer Worms and Viruses: A Survey,” International Journal of Trend in Research and Development (IJTRD), , ISSN: 2394-9333, Special Issue | RIET-17 , December 2017, URL: http://www.ijtrd.com/papers/IJTRD13399.pdf

[10] R. Mahmoud, M. Anagnostopoulos, S. Pastrana and J. M. Pedersen, “Redefining Malware Sandboxing: Enhancing Analysis Through Sysmon and ELK Integration,” in IEEE Access, vol. 12, pp. 68624-68636, 2024, doi: 10.1109/ACCESS.2024.3400167

[11] S. Yuan et al., "Research on Vulnerability Detection Techniques Based on Static Analysis and Program Slice," 2024 6th International Conference on Electronic Engineering and Informatics (EEI), Chongqing, China, 2024, pp. 965-969, doi: 10.1109/EEI63073.2024.10696068.

[12] Y. Tian et al., “Research on Personal Privacy Security Detection Techniques for Android Applications,” 2024 9th International Conference on Electronic Technology and Information Science (ICETIS), Hangzhou, China, 2024, pp. 375-379, doi: 10.1109/ICETIS61828.2024.10593754.

[13] S. Wang et al., “A Novel Detection System for Multi-Architecture IoT Malware,” 2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Tianjin, China, 2024, pp. 1758-1763, doi: 10.1109/CSCWD61410.2024.10580682.

[14] R. H. Mahdi and H. Trabelsi, “Detection of Malware by Using YARA Rules,” 2024 21st International Multi-Conference on Systems, Signals & Devices (SSD), Erbil, Iraq, 2024, pp. 1-8, doi: 10.1109/SSD61670.2024.10549308.

[15] M. F. Ismael and K. H. Thanoon, “Investigation Malware Analysis Depend on Reverse Engineering Using IDAPro,” 2022 8th International Conference on Contemporary Information Technology and Mathematics (ICCITM), Mosul, Iraq, 2022, pp. 227-231, doi: 10.1109/ICCITM56309.2022.10031698

[16] H. A. Noman, Q. Al-Maatouk and S. A. Noman, “A Static Analysis Tool for Malware Detection,” 2021 International Conference on Data Analytics for Business and Industry (ICDABI), Sakheer, Bahrain, 2021, pp. 661-665, doi: 10.1109/ICDABI53623.2021.9655866.

[17] K. Khaliq et al., “Ransomware Attacks: Tools and Techniques for Detection,” 2024 2nd International Conference on Cyber Resilience (ICCR), Dubai, United Arab Emirates, 2024, pp. 1-5, doi: 10.1109/ICCR61006.2024.10532926.

[18] H. Durgapal and D. Kumar, "Software Vulnerabilities Using Artificial Intelligence," 2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT), Greater Noida, India, 2024, pp. 1-6, doi: 10.1109/ICEECT61758.2024.10739067.

[19] Hex-rays, “IDA Pro,” Retrieved June 18, 2024 from https://hex-rays.com/ida-pro/

[20] Winitor, “pestudio” Retrieved June 19, 2024 from https://www.winitor.com/download

[21] yara, “yara v3.4.0,” Retrieved June 19, 2024 from https://yara.readthedocs.io/en/v3.4.0/gettingstarted.html

[22] GitHub, “Regshot,” Retrieved June 19, 2024 from https://github.com/Seabreg/Regshot

[23] Wirehark, “The world's most popular network protocol analyzer,” Retrieved June 19, 2024 from https://www.wireshark.org/

[24] GitHub, “flare-fakenet-ng” Retrieved June 19, 2024 from https://github.com/mandiant/flare-fakenet-ng

[25] MalwareBazaar by ABUSE, “MalwareBazaar Database” Retrieved June 20, 2024 from https://bazaar.abuse.ch/browse/

[26] MALCAT - the binary file dissector, “Malcat,” Retrieved June 20, 2024 from https://malcat.fr/index.html

[27] GitHub, “Detect-It-Easy” Retrieved June 20, 2024 from https://github.com/horsicq/Detect-It-Easy

[28] Microsoft, “Strings v2.54,” Retrieved June 21, 2024 from https://learn.microsoft.com/en-us/sysinternals/downloads/strings

[29] GitHub, “impelf,” Retrieved June 21, 2024 from https://github.com/signalblur/impelf

[30] IBM, “Analyzing files for embedded content and malicious activity”, Retrieved Nov 4, 2024 from https://www.ibm.com/docs/en/qsip/7.5?topic=content-analyzing-files-embedded-malicious-activity

[31] Chris Balles and Ateeq Sharfuddin, “Breaking Imphash”, 2019, 10.48550/arXiv.1909.07630

[32] I. Seung-Soon, “Tool interface standard (TIS) executable and linking format (ELF) specification.”, 1995.

[33] VirusTotal, “VirusTotal”, Retrieved Nov 4, 2024 from https://www.virustotal.com/gui/home/upload