applied sciences

MDPI

Article

Evaluation of Local Security Event Management System vs. Standard Antivirus Software

Antonio Pérez-Sánchez * and Rafael Palacios

Institute for Research in Technology, ICAI School of Engineering, Comillas Pontifical University, 28015 Madrid, Spain; rafael.palacios@iit.comillas.edu

* Correspondence: apsanchez@comillas.edu

Featured Application: This work can be applied to develop new anti-malware strategies based on event analysis.

Abstract: The detection and classification of threats in computer systems has been one of the main problems researched in Cybersecurity. As technology evolves, the tactics employed by adversaries have also become more sophisticated to evade detection systems. In consequence, systems that previously detected and classified those threats are now outdated. This paper proposes a detection system based on the analysis of events and matching the risk level with the MITRE ATT&CK matrix and Cyber Kill Chain. Extensive testing of attacks, using nine malware codes and applying three different obfuscation techniques, was performed. Each malicious code was analyzed using the proposed event management system and also executed in a controlled environment to examine if commercial malware detection systems (antivirus) were successful. The results show that evading techniques such as obfuscation and in-memory extraction of malicious payloads, impose unexpected difficulties to standard antivirus software.

Keywords: SIEM; antivirus; event-based threat detection; MITRE; Cyber Kill Chain

Citation: Pérez-Sánchez, A.; Palacios, R. Evaluation of Local Security Event Management System vs. Standard Antivirus Software. *Appl. Sci.* **2022**, *12*, 1076. https://doi.org/10.3390/app12031076

Academic Editors: George Drosatos, Konstantinos Rantos and Konstantinos Demertzis

Received: 24 December 2021
Accepted: 18 January 2022
Published: 20 January 2022

1. Introduction

Antivirus software is the most common tool being used to protect the user's computer from malware attacks. There are other protections methods such as local firewalls and corporate firewalls that protect the end-user from being totally exposed to attackers or malware already running on the local network. However, there are too many ways in which a piece of malicious software can get access to a personal computer. All sophisticated software, such as the operating system, programs running on a personal computer, and even software running on firewalls or other hardware equipment, contains bugs or flaws that may cause these systems to act in unexpected ways. Although most bugs are not significant on their own and do not pose a security risk, an attacker can take advantage of certain bugs to write programs called exploits to increase their arsenal of attacking tools. By combining a series of exploits into an "exploit chain", it is possible to circumvent all the defenses one-by-one until reaching the victim's computer to execute some kind of malicious software there. The fact that malware eventually reaches the end-user computer highlights the importance of malware detection software running locally.

This paper proposes a detection strategy based on events generated by the system and its corresponding mapping with Cyber Kill Chain and MITRE ATT&CK. The contributions of this paper are:

- A framework has been implemented to evaluate threats based on the analysis of events and subsequent classification in the Cyber Kill Chain and MITRE ATT&CK models.
- Using different obfuscation techniques, a set of malware samples has been built to evaluate the effectiveness of commercial antivirus systems and determine their detection and classification capabilities while dealing with obfuscated files.

For a better understanding of the contents of this paper, it was organized as follows: Section 2 describes all the elements that must be known to understand the methodology proposed in this paper; therefore, the classification of malware families, the current functioning of antivirus systems, the most common evasion techniques, and the threat classification model used to analyze events are explained. Section 3 describes the proposed methodology and its implementation. Section 4 describes the experiments performed and the results obtained and includes the comparison with several known antivirus systems. Section 5 analyzes the results obtained with the proposed methodology. Section 6 concludes this paper and presents future works.

2. Review and Background Knowledge

The detection and classification of threats [1] in networks and computer systems is an essential component for organizations. The objective is to be able to detect malicious programs (malware) that have evaded protection layers being able to reach a server or a user system. The emergence of these malicious programs originates back to 1971, when Robert Thomas created the first computer virus called "Creeper" [2,3]. Since then, the evolution in number of attacks and sophistication level has been astonishing, with different types of malwares appearing for all known computer systems.

2.1. Malware Family Classification

However, malicious code has not only evolved technically, but also in terms of threat. They not only damage files or computer systems, as they used to behave in the 1970s and 1980s, but they are also used to spy, steal information, or demand monetary ransoms.

Viruses: Usually incorporated into existing software, they are activated when the user executes the software. Their consequences are diverse, ranging from slowing down the system to corrupting or deleting information [4,5].

Worms: Do not need, at first, user intervention and have the ability, in contrast to computer viruses, to self-propagate through the network [6,7].

Trojans: Hidden inside apparently legitimate software for the purpose of going undetected. This type of malicious program can perform any action it has been programmed to perform on the system [8,9].

Spyware: Like worms, it does not require user intervention to install itself on the system and usually works in a hidden mode by collecting user or system information in an unauthorized manner [8,9].

Adware: Usually inserted in the system through the installation process of other software and its mission is to display unwanted advertising, usually in the form of pop-up windows without the user's authorization. Now, there is Adware in the form of browser extensions or plugins.

Ransomware: It is generally executed from another malware, such as a worm, virus or trojan. Its mission is to completely sequester the system, encrypting all the files and demanding a payment from the user or organization [10–13].

Rootkit: It has the function to access and hide in particularly sensitive areas of the devices it infects, including areas that are not normally accessible to users. Their purpose is to take control of the system and facilitate its remote control while remaining hidden [14,15].

2.2. Actual Antivirus Threat Detection and Classification Systems

The main detection and classification software for this type of malware is the antivirus software, that have evolved in techniques, using mechanisms based on signature, heuristics, rules, and currently the use of artificial intelligence [16,17].

Signature detection [18–20] is the traditional method used by antivirus systems and is based on a database generated by the vendor. Any file downloaded to the system is compared to the database and if there is a match, it is malware. The problem with signature-based detection is that it will only detect those samples that have been previously identified and their signature is stored in the antivirus system database.

To complement signature-based detection and provide a solution to its limitations, heuristic detection techniques were developed [21–23]. The operation of heuristic algorithms is based on different criteria, each of them with a score, which determine whether a file is malicious or not. The most common three ways of performing this analysis are [24,25]:

Generic: Compares the actions of a file with another already identified as malicious.

Passive: Analyzes the file individually and tries to determine how it works.

Active: Runs the sample in a safe environment (sandbox) and determines if its activity is malicious. This strategy is difficult to implement, imposes significant delays, and it can be avoided by implementing payload activation delays.

Heuristic analysis has two fundamental problems, the first is the large number of false negatives and the second is the workload on the system. However, it improves the ability of antivirus software to detect new samples of malware.

Recently, to increase the capabilities of antivirus systems, the use of machine learning algorithms using artificial intelligence (AI) has been introduced [26–28]. The inclusion of these techniques allows for a large-scale data analysis, the identification of patterns and trends, as well as the automatic and rapid formulation of predictions. These systems are called Endpoint Detection and Response (EDR), also known as Endpoint Thread Detection and Response (ETDR), and implement and endpoint security mechanism [29] at the clients that collects data and sends it to a centralized console for processing, as in a distributed computing environment [30]. The information collected is correlated in real time to detect and analyze suspicious activity and processed in a centralized database. However, these systems have some weaknesses, as exemplified by G. Karantzas and C. Patsakis [31], for example, the comprehension of the artificial intelligence of the model created or the huge data ingestion required for accurate decision process.

2.3. Obfuscation Techniques

Years ago, a small modification in a piece of malware was able to circumvent antivirus software, so for each virus, we had several versions as if they were different developments (only manual analysis was able to find a common origin to assign a suitable virus name). However, the evolution of antivirus software with heuristics and AI, has been very effective in detecting variants of existing malware. The remaining challenge is to detect malware that uses obfuscation techniques, which is the focus of this research.

To avoid detection, several techniques have been developed. There are diverse techniques currently available to any attacker that can be applied individually or in combination to evade the defenses of any organization.

Self-encryption: The use of encryption algorithms or functions that contain a key included in the body of the malware that allows it to perform the encryption or decryption function in an automated manner [32].

Polymorphism: It uses a polymorphic engine to mutate itself and keep its original code intact [33–35].

Metamorphism: It can transform itself according to its capabilities to translate, edit, and rewrite its own code each time it infects a computer system [36,37].

Armouring: It is programmed to prevent any attempt at analysis or disassembly, making it impossible to know the original code [38].

Stealth: It completely or partially hides its presence in the system by intercepting requests of the operating system to interact with infected objects (boot sectors, file system elements, memory, etc.) [39,40].

Covert channels: It uses unauthorized communication channels and manipulates them in an unconventional way to transmit information undetected by anyone other than the entities operating the covert channel [41–43].

These obfuscation techniques make detection process very hard because the antivirus software is not able to read and analyze the code as in the case of any other static file. Moreover, due to the internal randomness of the obfuscation process, it is not possible to create signatures of the encrypted code because each sample of the same malware is

different. In this paper, we propose to detect such kind of hidden malware by analyzing its behavior based on the chain of actions executed. Very recently evasion techniques have been applied to ransomware as well to avoid being detected while running. As an example, LockFile [44] performs file encryption by mapping the file to memory to avoid interaction with the hard drive during the process.

2.4. Threat Categorization

All adversaries will always execute a chain of actions [45–47] that are listed in the Cyber Kill Chain [48,49] (see Figure 1). This scheme allows to classify attacks into different levels of risk and to determine the perspective of the attacker at each moment. In addition, with the purpose of describing and categorizing adversary capabilities to improve system security, a structured matrix was created with the techniques and tactics used. The matrix known as MITRE ATT&CK (ATT&CK is a pseudonym of ATTACK) includes Tactics, Techniques, and Common Adversary Knowledge [50] and centralizes this information, allowing organizations to prevent possible threats to their computer systems.

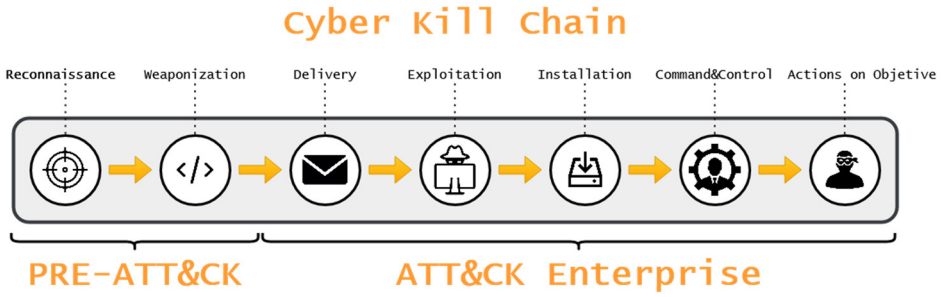

Figure 1. Cyber Kill Chain and MITRE ATT&CK.

The unification of these two schemes provides a real understanding of the type of technique and tactics used by an adversary at each stage of the attack process. By analyzing this information, we can extrapolate models that allow us to propose defensive strategies for detecting the actions of adversaries, and thus mitigate their consequences. Detection and classification systems (antivirus) on the market are very limited when performing these tasks in real time on malware that implement obfuscation techniques, because they mostly rely on static file analysis.

For these reasons, we have considered the implementation of a system based on the real-time analysis of the events [51–53] generated in the system and the classification of these events using the MITRE ATT&CK matrix and the Cyber Kill-Chain [54]. In this way, we were able to classify each event in the system, collecting all the events that have a malicious traceability.

3. Methodology and Implementation

Several Cybersecurity companies dedicate their efforts to the analysis of threats, developing and improving their detection and classification systems. These tasks resulting in the creation of a unique signature, the analysis of its operation at system and network level, or the study of its source code in depth. As a result, it is possible to obtain an updated knowledge database of known threats to defend computer systems. To evade these detection systems, adversaries can apply a multitude of obfuscation techniques, behavior modification and research of unknown techniques. Hence, antivirus companies need to create new signatures for each variant.

3.1. Related Work

This paper proposes a detection and threat classification system based on events generated by the operating system in real time, classifying them in the MITRE ATT&CK matrix of techniques and tactics together with their corresponding mapping in the Cyber Kill Chain. The criteria applied is based on the analysis of the events generated by different

attacks, checking the tactic and technique where each of them is located within the MITRE ATT&CK matrix and therefore the phase within the Cyber Kill-Chain. Although there are existing works that have carried out research on event-based threat detection and classification systems [55,56], the one proposed is very effective regardless of the obfuscation method employed.

It is important to remark that this system does not pretend to replace the current antivirus systems, since they do their job effectively in most of the known cases of malware. However, it can be considered an extension to improve antivirus in the case of certain malicious code that is obfuscated and executed in memory. In most antivirus software, the analysis of code is only triggered by read or write actions on the hard drive.

The number of events collected during the execution of suspicious programs can be very large. Therefore, the current proposal is to match those events to the widely recognized MITRE ATT&CK matrix, and then classify as malicious any vector of attack containing techniques and actions considered of high-risk (from Execution/Exploitation). In future work, it will be possible to provide intelligence to the platform to automatically detect and classify malicious codes more accurately, for example considering the progression of events. Table 1 shows the list of the 14 MITRE tactics, their correlation with the Cyber Kill Chain, the total number of techniques within each Tactic, and how many of those techniques are considered high-risk. It can be observed that for some Tactics, neither technique is considered of high-risk. For example, the Tactic Discovery has 13 different techniques, but none of them is considered malicious. On the other hand, in the case of Command and Control there 8 techniques out of 13 in this category, that are considered high-risk. Consequently, events associated with Discovery tactic will not be classified as malicious, while those associated with Command and Control will very likely classified as malicious (only 5 techniques will not). The reason for having white-listed techniques is because some techniques, such as command execution, can be used in a legitimate way. For example, system commands can be executed by any user and do not represent any threat.

Table 1. Classification method using MITRE ATT&CK matrix and Cyber Kill Chain.

MITRE ATT&CK Tactics	Cyber Kill Chain	Number of High-Risk Techniques	Classified as Malicious
Reconnaissance	Reconnaissance	0/10	✘
Resource Development	Weaponization	0/7	✘
Initial Access	Delivery	0/9	✘
Execution	Exploitation	5/10	✔
Persistence	Installation	3/9	✔
Privilege Escalation	Actions on Objectives	4/10	✔
Defense Evasion	Installation	10/27	✔
Credential Access	Actions on Objectives	3/7	✔
Discovery	Reconnaissance	0/13	✘
Lateral Movement	Actions on Objectives	5/9	✔
Collection	Actions on Objectives	0/5	✘
Command and Control	Command and Control	8/13	✔
Exfiltration	Actions on Objectives	0/8	✘
Impact	Actions on Objectives	1/1	✔

The event-based detection system was implemented by combining of a set of open-source tools to automate the analysis of malicious techniques explained above. As a central core we used Sysmon [57,58], which runs as a resident service on the system and provides logging activity through the Windows event log. The communication between systems for sending information uses Winlogbeat [59] which is an open-source tool for sending Windows event logs to Kafka, which is an open-source distributed event streaming platform. For centralized information reception and analysis, a system based on ELK (ElasticSearch, Logstah and Kibana) was also implemented [60]. The first module (Logstah) is a server-side data processing pipeline that ingests multiple sources simultaneously, the second module

(ElasticSearch) provides a data search and analytics engine and sends them to the third (Kibana) that facilitates the visualization of the data. Finally, all system information is consumed by Grafiki, which allows the creation of event graphs generated during the execution of malicious techniques in the system. We have named this detection system EDBS (Event Based Detection System). The block diagram of the system is shown on Figure 2.

Figure 2. Event Based Detection System (EBDS) Diagram.

The paper published by J. N. Praneeth and M. Sreedevi [61] similarly applied some of the proposed tools, but with very limited analysis of the results obtained. Only an unspecified piece of software is analyzed against VirusTotal using its hash code. The problem previously discussed can be observed in a similar way in other works which implement similar technologies, as the thesis published by M. Rasool [62] which uses VirusTotal as one of the central components in the classification of malware. In the research published by F. A. Bin Hamid Ali and Yee Yong Len [63], the authors describe a methodology for event-driven attack analysis based on signatures whereby their classification method only allows them to detect previously known attacks. Other publications, such as the thesis of U. Jain [64], rely on the analysis of events generated from the system of the type "Lateral Movement" but does not describe the traceability of the generated events and they are not assigned a risk level. In addition, there are other publications that implement possible architectures for threat detection and analysis [65–67]. These architectures are focused on the detection of suspicious activities in the network, and do not conceive the use of techniques that involve the execution of code directly in the memory of the victim system. Finally, there are papers that support the contextualized event-based analysis model, such as the one published by P. Giura and W. Wang [68] that describes both a methodology and a framework that can be implemented to perform the detection of different advanced attacks.

3.2. Implementation and Execution of Experimental Tests

The execution of the tests was performed based on known malicious code and techniques, which commercial antivirus systems already include in their signature databases or detect by means of heuristic algorithms that they have implemented. All of them were executed on a completely updated Microsoft Windows 10 operating system. The description of the malicious codes follows:

Code A: Powercat.ps1 is a PowerShell script equivalent to the Netcat tool that allows to open ports, to perform remote connections or do port scanning [69].

Code B: ConPtyShell.ps1 is a PowerShell script that uses the CreatePseudoConsole() function within Windows 10 version 1809 and allows the execution of a remote command console [70].

Code C: Invoke-PowershellTCP.ps1 is a PowerShell script that is part of the Nishang collection and allows the execution of a remote command console [71].

Code D: MeterpreterTCP.ps1 is a PowerShell version of one of the payloads generated by the Metasploit framework [72], that has a very wide set of functionalities and executes in memory, hence offering significant undetectability.

Code E: Mimikatz.ps1 is an open-source application that allows attackers to obtain information on the target system, such as stored credentials or Kerberos tickets [73].

Code F: Meterpreter.exe is the executable version of malicious code previously mentioned in "**Code D**".

In addition, we made use of macros 4.0 (XML) in Excel documents, which are compatible with current Microsoft Office systems [74] and represent one of the techniques currently used to reduce the detection. We also added the use of signed operating system binaries, known as LOLBins [75], to conduct code execution once the document has been opened.

Code G: The Microsoft Excel document contains a macro 4.0 that runs a local command console using the MSBuild.exe binary.

Code H: The Microsoft Excel document contains a macro 4.0 that first downloads the file with malicious code using CertUtil.exe and then executes it using MSBuild.exe, as presented in the previous case.

Similarly, we have used Visual Basic for Applications (VBA) in Microsoft Office documents. Although these are very well-known, this has allowed us to obtain an approach for the results that we have obtained with our implementation.

Code I: The Microsoft **Word** document contains a Visual Basic for Applications (VBA) macro that download and execute a malicious code using PowerShell directly into memory.

In order to study the current state of the art in obfuscated malware detection, 11 commercial threat detection and classification (antivirus) systems were used. Different virtual machines were created with a clean updated Windows 10 installation and one antivirus at a time.

After basic initial testing of the aforementioned malware code, three different obfuscation methods were implemented to perform additional tests, consisting of encryption using AES algorithms, substitution of function names and known text strings, and encryption using XOR algorithm.

AES (Advanced Encryption Standard) encryption: It is a symmetric block cipher, i.e., it operates on groups of bits of fixed length and applies invariant transformations to them. The size of the blocks it handles is 128-bits and uses 128, 192, and 256-bit encryption keys.

String substitution: A simple way to evade antivirus systems in some cases is the substitution of known function names with random ones. As a result, it is possible to avoid identification by modifying the code signature that is compared with the one stored in the antivirus system's database.

XOR encryption: It is a symmetric encryption based on the XOR logical operator and performs this operation bit by bit.

A general approach of the proposed methodology for data collection and analysis can be understood following the graph of events shown in Figure 3. This example corresponds to the execution of a Microsoft Word document that has embedded a macro (VBA) [76] with malicious code, that is Code I in the set of tests.

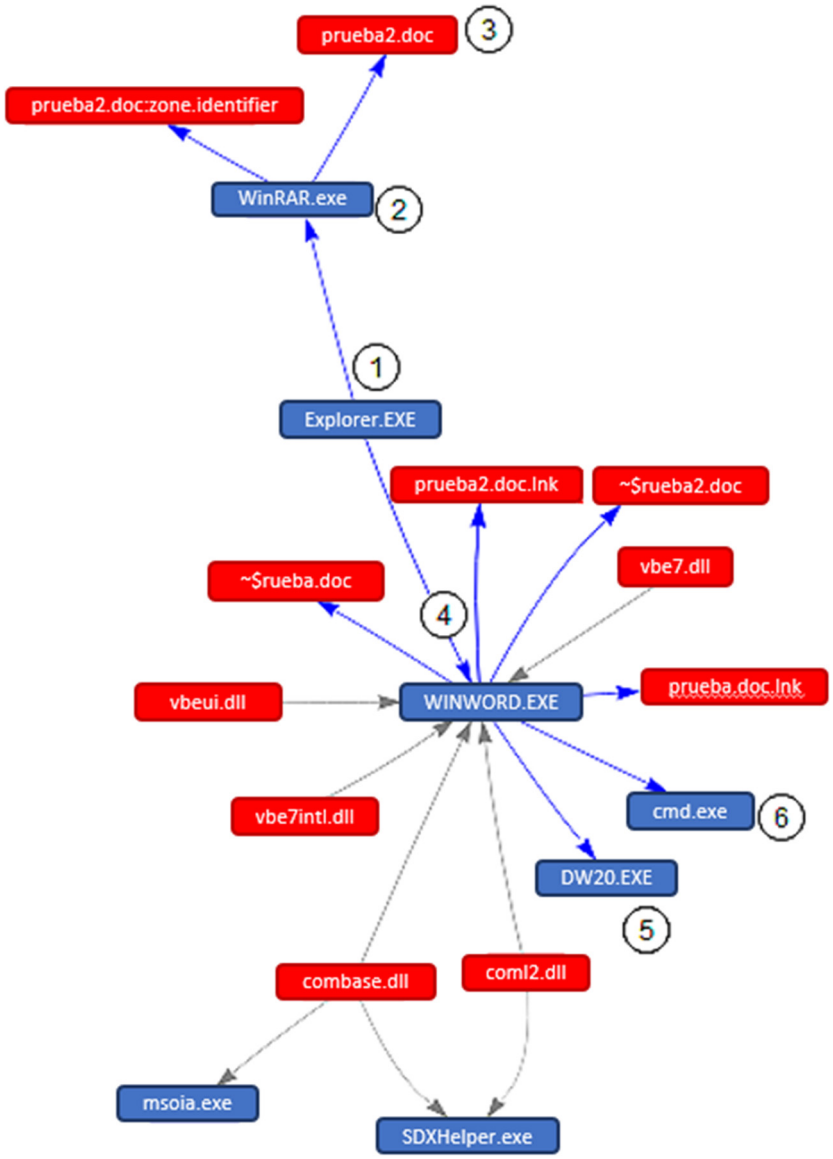

Figure 3. Result of malicious macro execution in a Microsoft Word document.

The graph of events shown in Figure 3, includes all the important events triggered by the execution of Code I. Therefore, the execution process can be described as follows:

1. The user receives compressed file and proceeds to download it to the system.
2. Launches WinRAR application to obtain the contents of the file.
3. The result of the decompression is the file "**test2.doc**" ("prueba2.doc" in the graph) which contains the macro with malicious code embedded in it.
4. It is executed by the user and the system calls the WINWORD.exe program to open the document.
5. When the user enables the execution of the macros an error occurs in the WINWORD.exe binary that calls DW20.exe, which is responsible for compiling an error report.
6. While the previous step occurs, the macro execution occurs which launches a command console on the system.

The mapping of the steps discussed above onto MITRE ATT&CK, and the Cyber Kill Chain are the following:

- The step 1 corresponds to the MITRE ATT&CK tactic "**Initial Access**" and technique "**T1566-Phishing**" and "**Delivery**" action in the Cyber Kill Chain.

- The step 2 corresponds to the MITRE ATT&CK tactic "**Execution**" and technique "**T1204-User Execution**" and "**Exploitation**" action in the Cyber Kill Chain.
- The step 3 corresponds to the MITRE ATT&CK tactic "**Persistence**" and technique "**T1137-Office Application Startup**" and "**Installation**" action in the Cyber Kill Chain.
- The step 4 corresponds to the MITRE ATT&CK tactic "**Defense Evasion**" and technique "**T1055-Process Injection**" and "**Installation**" action in the Cyber Kill Chain.
- The steps 5 and 6 corresponds to the MITRE ATT&CK tactic "**Execution**" and technique "**T1059-Command-Line Interface**" and "**Installation**" action in the Cyber Kill Chain.

In this example, the system detects several events that are matched to MITRE ATT&CK and Cyber Kill Chain, but most of them are low or medium risk. Technique "**T1204**" in steps 2 and 3 is part of the "**Execution**" tactic, but it is not considered high-risk, because it means that the user launched the execution of a program such as Word. Similarly, in step 4 the system captures a "**Defense Evasion**" tactic because of technique "**T1055**" which is neither a high-risk technique because it involves the execution of Dynamic Linked Libraries (DLLs), like almost all windows programs. Conversely, the latest event, also corresponding to the "**Execution**" tactic, is the one that actually triggers the alarm. Technique "**T1059**" is defined as type high-risk, because it means that the macro is launching a Command Tool or Power Shell for executing any kind of commands.

One of the features of Microsoft Word, is the ability to create multiple letters automatically out of a list of names and addresses stored in a database. This feature is called Mail Merge and can be used in conjunction with Open Database Connectivity (ODBC) to connect Word with a database server. It could be very useful to include a macro in that type of documents to verify, using a ping command, if there is connectivity with the database server before attempting to generate mail letters. Upon opening such document, the system will not trigger any alarm, because it contains a benign macro. The sequence of event will be very similar than before, but instead of technique "**T1059**" at the latest step, the system will be getting "**T1018**" (associated with ping command), which is a low-risk Discovery tactic that does not trigger the alarm.

Note that the procedure described with the help of the "**Code I**" can generalize to other types of attacks that execute in memory for avoiding traditional antivirus detection techniques. The methodology described, may potentially detect any attack defined by MITRE and will not reveal any risk in the case of benign events.

4. Experimental Results

The proposed methodology was evaluated for different categories of malware and obfuscation methods. The objective of the experiments is to show the effectiveness of event collection, the ability to match those events to the MITRE ATT&CK matrix and use that standard to determine if a threat is present of not. There are different attack vectors implemented in our experimental analysis that combine different tactics and techniques from the MITRE ATT&CK matrix. The evaluation is not based on individual malware samples, but on different attack vectors that ultimately implement actions that trigger system events detected by the proposed architecture. This is a more generalized assessment approach that targets non-existing malware or attack techniques and does not attempt to detect existing malware, because current antivirus can cope with such attacks.

Because of the obfuscation applied, standard antivirus programs are in disadvantage for detecting the malware tested. However, the event-based strategy can detect high-risk events, according to the MITRE specification, for the malware. Any attack that triggers high-risk system events is potentially detectable with the proposed approach regardless of the obfuscation method implemented by the attacker.

The event-bases detection system helps to perform complex analyses, such as experiments using Excel documents with macros 4.0, and signed operating system binaries. Code G and Code H include malware in the form of Excel macros. Their detail description follows:

Excel-Macro4.0-MSBuild-Cmd (Code G): The execution graph obtained after running Code G on the virtual machine is shown in Figure 4

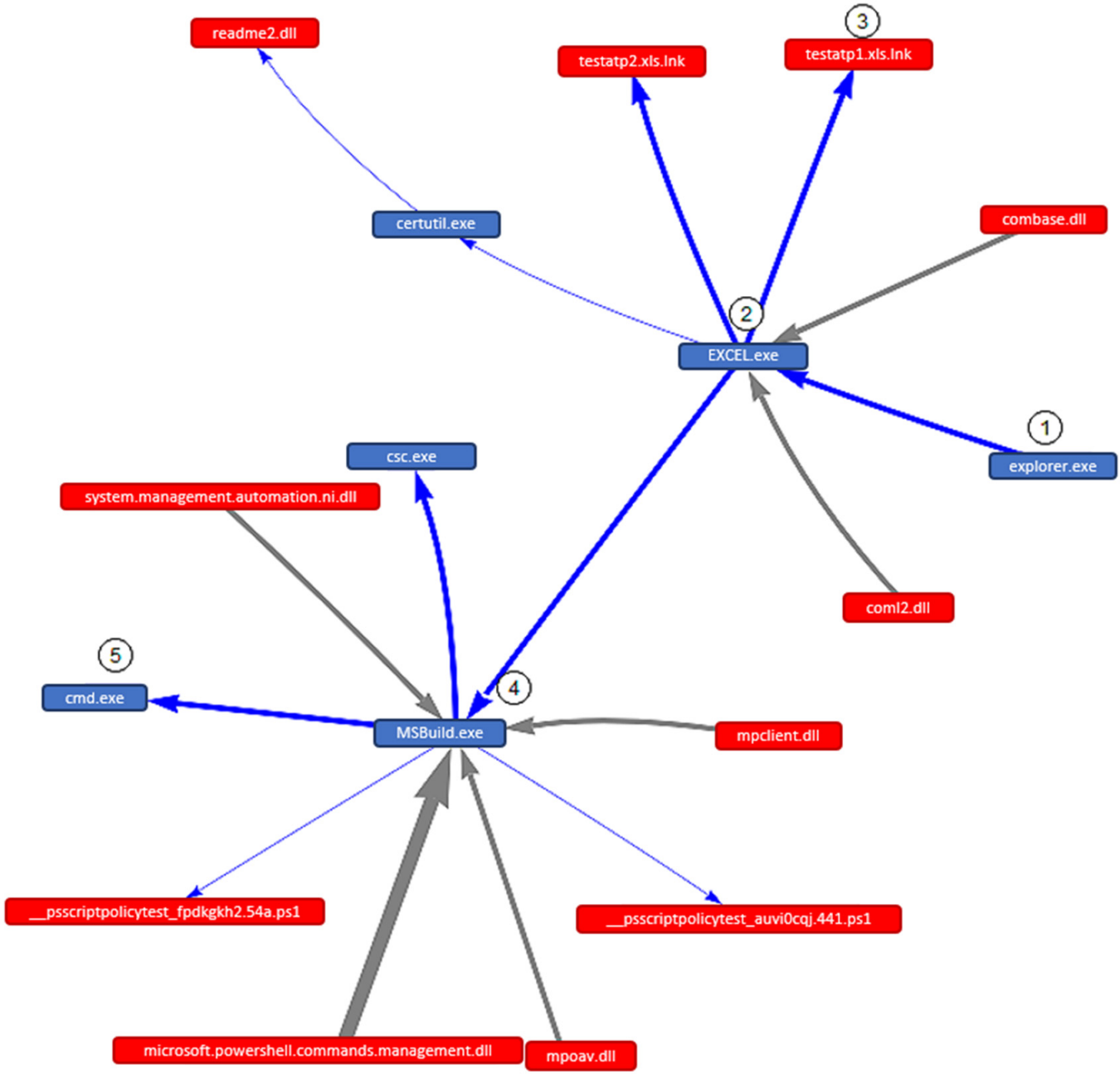

Figure 4. Result of the execution of an Excel document with MSBuild.exe.

Corresponding to the image in Figure 4, a detailed analysis of the events that have occurred in the system can be carried out, which allows for a comprehensive evaluation of the execution process:

1. The user receives an Excel document which contains the macros 4.0 that allows malicious execution on the system.
2. Opening the document, the system calls the EXCEL.exe program.
3. Document "**testapt1.xls**" opens, and the user enables the execution of macros embedded in the document.
4. Excel calls the binary MSBuild.exe that executes the commands in the system.
5. As a result, command prompt is opened on the system.

The mapping of the steps described above onto MITRE ATT&CK and the Cyber Kill Chain is the following:

- The step 1 corresponds to the MITRE ATT&CK tactic "**Initial Access**" and technique "**T1566-Phishing**" and "**Delivery**" action in the Cyber Kill Chain.
- The step 2 corresponds to the MITRE ATT&CK tactic "**Execution**" and technique "**T1204-User Execution**" and "**Exploitation**" action in the Cyber Kill Chain.
- The step 3 corresponds to the MITRE ATT&CK tactic "**Persistence**" and technique "**T1137-Office Application Startup**" and "**Installation**" action in the Cyber Kill Chain.
- The step 4 corresponds to the MITRE ATT&CK tactic "**Defense Evasion**" and technique "**T1127-Trusted Developer Utilities Proxy Execution**" and "**Installation**" action in the Cyber Kill Chain.
- The step 5 corresponds to the MITRE ATT&CK tactic "**Execution**" and technique "**T1059-Command-Line Interface**" and "**Exploitation**" action in the Cyber Kill Chain.

Excel-Macro4.0-CertUtil-MSBuild-RevShell (Code H): The execution graph obtained after running this test is detailed in Figure 5:

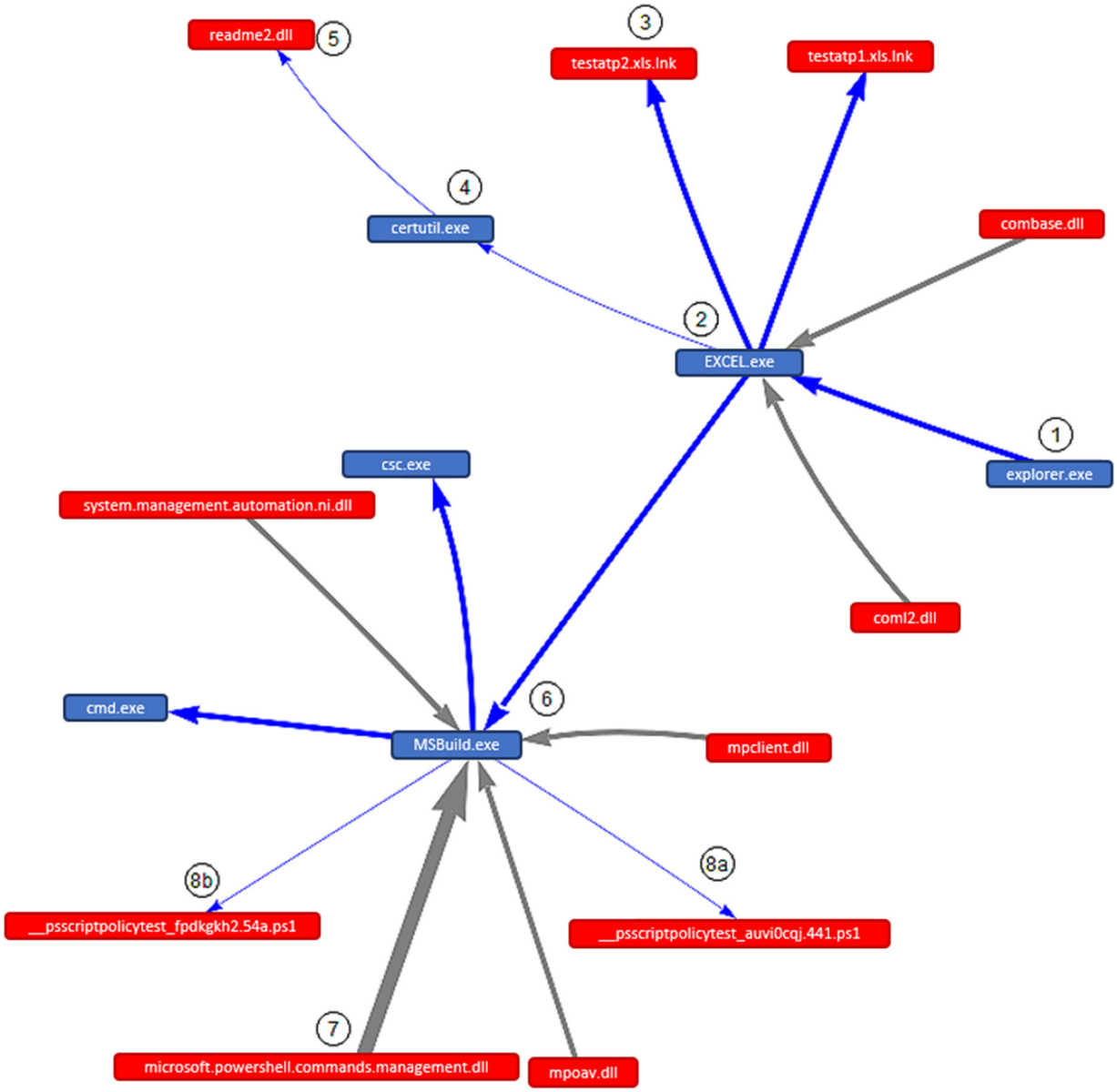

Figure 5. Result of Excel document execution with CertUtil.exe and MSBuild.exe.

The following points detail the execution process, which allows for a detailed analysis of the events that have occurred in the system.

1. The user receives an Excel document containing macros 4.0 which allow malicious execution on the system.
2. When the user proceeds to open the document, the system calls the EXCEL.exe program.
3. The document "**testapt2.xls**" is opened and the user enables the execution of macros.
4. The binary CertUtil.exe is executed at the beginning, which downloads the file with the malicious commands to the system.
5. The file "**readme2.txt**" containing the malicious code is stored in a path on the computer.
6. Afterwards, the binary MSBuild.exe is launched and it runs the contents of the file, in this case it contains C# code that executes PowerShell commands.
7. It is possible to observe how "**microsoft.powershell.commands.management**" is executed.
8. Policy compliance tests are carried out and also detected as events (8a, 8b), although they are not classified as risk events.

The mapping of the steps described above onto MITRE ATT&CK and the Cyber Kill Chain is the following:

- The step 1 corresponds to the MITRE ATT&CK tactic "**Initial Access**" and technique "**T1566-Phishing**" and "**Delivery**" action in the Cyber Kill Chain.
- The step 2 corresponds to the MITRE ATT&CK tactic "**Execution**" and technique "**T1204-User Execution**" and "**Exploitation**" action in the Cyber Kill Chain.
- The step 3 corresponds to the MITRE ATT&CK tactic "**Persistence**" and technique "**T1137-Office Application Startup**" and "**Installation**" action in the Cyber Kill Chain.
- The steps 4 and 5 correspond to the MITRE ATT&CK tactic "Defense Evasion" and technique "**T1292-Indirect Command Execution**" and "**Installation**" action in the Cyber Kill Chain.
- The step 6 corresponds to the MITRE ATT&CK tactic "**Defense Evasion**" and technique "**T1127-Trusted Developer Utilities Proxy Execution**" and "**Installation**" action in the Cyber Kill Chain.
- The step 7 corresponds to the MITRE ATT&CK tactic "**Execution**" and technique "**T1059-Command and Scripting Interpreter**" and "**Exploitation**" action in the Cyber Kill Chain.

The Event-based detection system (EBDS) proposed is able detect all malicious activity, because the malware always triggers an event classified as high-risk by MITRE. Even in the case of using obfuscation methods, EBDS can detect the attack once the original code is restored and key components (such as PowerShell) are executed. On the other hand, static analysis of the malicious file is not able to detect the key components because the payload is encrypted.

The comparison between the performance of different antivirus software, using techniques such as obfuscation and extracting malicious payloads in-memory, is shown in the following tables. Table 2, without obfuscation, shows that most malware codes are detected by several commercial antivirus packages. However, after applying different obfuscation methods, detection is much harder (Tables 3–5).

Table 2. Execution of malicious code without any encryption method.

AV/Test	A	B	C	D	E	F	G	H	I
PandaDome 20.01.00	✘	✘	✘	✘	✔	✔	✘	✘	✔
BitDefender 1.0.17.221	✘	✘	✔	✔	✔	✔	✘	✔	✔
TrendMicro 16.0	✘	✘	✘	✘	✔	✔	✔	✔	✔
Sophos 10.8.10.810	✘	✘	✘	✘	✘	✔	✘	✘	✔
Karspersky 21.2.16.590	✔	✘	✔	✔	✔	✔	✔	✔	✔
Avast 21.1.2443	✘	✘	✘	✘	✔	✔	✘	✘	✔
AVG 20.10	✘	✘	✘	✘	✔	✔	✘	✘	✔
Avira 15.0.2101.2070	✘	✘	✘	✔	✘	✘	✘	✘	✔
Eset 14.0.22.0	✘	✘	✔	✔	✔	✔	✘	✔	✔
BullGuard 21.0.387	✘	✘	✘	✔	✔	✔	✘	✘	✔
EBDS	✔	✔	✔	✔	✔	✔	✔	✔	✔

Table 3. Execution of malicious code modified using AES encryption.

AV/Test	A	B	C	D	E	F	H
PandaDome 20.01.00	✘	✘	✘	✘	✘	✔	✘
BitDefender 1.0.17.221	✔	✔	✔	✔	✔	✔	✔
TrendMicro 16.0	✘	✘	✘	✘	✔	✔	✔
Sophos 10.8.10.810	✘	✘	✘	✘	✘	✔	✘
Karspersky 21.2.16.590	✔	✔	✔	✔	✔	✔	✔
Avast 21.1.2443	✘	✘	✘	✘	✔	✔	✘
AVG 20.10	✘	✘	✘	✘	✔	✔	✘
Avira 15.0.2101.2070	✘	✘	✘	✔	✘	✘	✘
Eset 14.0.22.0	✘	✘	✘	✔	✘	✘	✘
BullGuard 21.0.387	✘	✘	✔	✔	✔	✔	✔
EBDS	✔	✔	✔	✔	✔	✔	✔

Table 4. Execution of malicious code modified by applying string substitution.

AV/Test	A	B	C	D	E	H
PandaDome 20.01.00	✘	✘	✘	✘	✘	✘
BitDefender 1.0.17.221	✘	✘	✘	✔	✔	✘
TrendMicro 16.0	✘	✘	✘	✔	✔	✘
Sophos 10.8.10.810	✘	✘	✘	✘	✘	✘
Karspersky 21.2.16.590	✔	✘	✘	✔	✔	✔
Avast 21.1.2443	✘	✘	✘	✘	✘	✘
AVG 20.10	✘	✘	✘	✘	✘	✘
Avira 15.0.2101.2070	✘	✘	✘	✔	✘	✘
Eset 14.0.22.0	✘	✘	✘	✔	✔	✔
BullGuard 21.0.387	✘	✘	✘	✔	✘	✘
EBDS	✔	✔	✔	✔	✔	✔

It is also important to mention that most antivirus are giving an alarm because they detect the obfuscation technique, giving a warning of "suspicious file", but not detecting the malware itself. As an example, code B is not detected by BitDefender without obfuscation (Table 1), but it is detected when using AES encryption (Table 2), clearly because the use of encryption tools is raising the flag, since the payload cannot be read until executed. This detection strategy typically yields false positives if the user works with legitimate encrypted or coded files. We were able to prove such behavior by creating a harmless PowerShell code to just display a "hello world" message and obfuscating the file using AES; this file was also detected as malicious by several antivirus programs. The file was uploaded to VirusTotal [77,78] on 18 May 2021, and it was labeled as malicious by 9 antivirus programs.

Table 5. Execution of malicious code modified using XOR encryption.

AV/Test	A	B	C	D	H
PandaDome 20.01.00	✘	✘	✘	✘	✘
BitDefender 1.0.17.221	✘	✘	✔	✔	✔
TrendMicro 16.0	✘	✘	✘	✘	✘
Sophos 10.8.10.810	✘	✘	✘	✘	✘
Karspersky 21.2.16.590	✔	✘	✔	✔	✔
Avast 21.1.2443	✘	✘	✘	✘	✘
AVG 20.10	✘	✘	✘	✘	✘
Avira 15.0.2101.2070	✘	✘	✘	✔	✘
Eset 14.0.22.0	✘	✘	✔	✔	✔
BullGuard 21.0.387	✘	✘	✔	✔	✔
EBDS	✔	✔	✔	✔	✔

5. Discussion

Traditional systems for detection and classification of threats in networks and computer systems are increasingly encountering difficulties for early detection of malicious code. This is because detection systems based on signature and heuristics have become outdated in the battle against novel evasion techniques. Therefore, this paper highlights the importance of implementing alternative detection methods, such as those based on the analysis and classification of events generated within the user's computer. Through this type of analysis, it is possible to detect advanced techniques and establish security checkpoints for events generated in the system that may indicate that the system is being compromised.

The results presented in this paper correspond to commercial antivirus software and the proposed Event Management System EBDS, all running locally in the end-user computer. In large companies, it is possible to use SIEM (Security Information and Event Management) systems or corporate versions of antivirus software that centralize events, therefore attaining better results than a stand-alone antivirus. However, the work presented here was carried out with the focus on home, home-office, or small businesses, which need to rely on local computer detection.

The results show that, by applying obfuscation techniques, it is possible to hide malicious code from any detection system based on code signatures. Any encrypted code (malicious or benign) uploaded to VirusTotal will yield "safe" results, or maybe a warning related to fact that the file is encrypted. In addition, most advanced attacks manage to extract the obfuscated malicious code directly in the memory, without additional hard drive access, therefore avoiding a possible signature-based detection, because antivirus softwares scan files only while reading or writing to the hard drive.

Obfuscation techniques have no limits, there are multiple techniques and combination of techniques that can be applied. Since these techniques use a randomly generated key, every sample of obfuscated malware is different, and the creation of a general file signature is not possible. In addition, completely new malware, even without obfuscation, cannot be detected through signatures.

6. Conclusions

We have presented a detection system based on open-source tools, which is able to capture security critical events of the system. The proposed event-based analysis method, linked to MITRE Cyber Kill Chain, can stop the attack at the right moment, when reaching the state of "Execution/Exploitation", preventing that PowerShell or Cmd are called.

Although standard antivirus programs are very effective and computationally efficient while detecting malware in general, they struggle with different obfuscation techniques. When obfuscation is applied to the malware, the proposed event-based strategy offers a second layer of analysis to improve detection rates compared to the set of non-detections and false positives of traditional antivirus systems alone.

For the purpose of experimentally validating the methodology proposed in this article, a comparative study has been carried out against 10 traditional detection systems (antivirus), always using the same set of samples and with the same obfuscation method in each case. Antivirus software was tested in different Windows 10 virtual machines, allowing the software to use all the potential of heuristics and real-time detection modules; nevertheless, it did not perform well for samples with obfuscation. On the other hand, the proposed methodology was able to detect the detect the sequence of events that were escalating in the severity.

As future work, Artificial Intelligence algorithms will be added to detect hazardous sequences of events, potentially being able to detect threats even in earlier stages of the attack. The current detection method was implemented by finding the state of "Execution/Exploitation", but without exploiting details about sequences of states.

Author Contributions: Conceptualization, A.P.-S.; methodology, A.P.-S. and R.P.; software, A.P.-S.; validation, A.P.-S. and R.P.; investigation, A.P.-S. and R.P.; writing—original draft preparation, A.P.-S. and R.P.; writing—review and editing, A.P.-S. and R.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research no received external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Botacin, M.; Ceschin, F.; de Geus, P.; Grégio, A. We need to talk about antiviruses: Challenges & pitfalls of AV evaluations. *Comput. Secur.* **2020**, *95*, 101859. [CrossRef]
2. Robert, J.-M.; Chen, T. The Evolution of Viruses and Worms. In *Statistical Methods in Computer Security*; CRC Press: Boca Raton, FL, USA, 2004; pp. 265–285.
3. Namanya, A.P.; Cullen, A.; Awan, I.U.; Disso, J.P. The World of Malware: An Overview. In Proceedings of the 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), Barcelona, Spain, 6–8 August 2018; pp. 420–427.
4. Zuo, Z.; Zhu, Q.; Zhou, M. Infection, imitation and a hierarchy of computer viruses. *Comput. Secur.* **2006**, *25*, 469–473. [CrossRef]
5. Schneider, W. Computer viruses: What they are, how they work, how they might get you, and how to control them in academic institutions. *Behav. Res. Methods Instrum. Comput.* **1989**, *21*, 334–340. [CrossRef]
6. Choi, Y.-H.; Li, L.; Liu, P.; Kesidis, G. Worm virulence estimation for the containment of local worm outbreak. *Comput. Secur.* **2010**, *29*, 104–123. [CrossRef]
7. Zhou, H.; Hu, Y.; Yang, X.; Pan, H.; Guo, W.; Zou, C.C. A Worm Detection System Based on Deep Learning. *IEEE Access* **2020**, *8*, 205444–205454. [CrossRef]
8. Gezer, A.; Warner, G.; Wilson, C.; Shrestha, P. A flow-based approach for Trickbot banking trojan detection. *Comput. Secur.* **2019**, *84*, 179–192. [CrossRef]
9. Dong, C.; Liu, Y.; Chen, J.; Liu, X.; Guo, W.; Chen, Y. An Unsupervised Detection Approach for Hardware Trojans. *IEEE Access* **2020**, *8*, 158169–158183. [CrossRef]
10. Meland, P.H.; Bayoumy, Y.F.F.; Sindre, G. The Ransomware-as-a-Service economy within the darknet. *Comput. Secur.* **2020**, *92*, 101762. [CrossRef]
11. Liu, W. Modeling Ransomware Spreading by a Dynamic Node-Level Method. *IEEE Access* **2019**, *7*, 142224–142232. [CrossRef]
12. Hampton, N.; Baig, Z.; Zeadally, S. Ransomware behavioural analysis on windows platforms. *J. Inf. Secur. Appl.* **2018**, *40*, 44–51. [CrossRef]
13. Lee, S.-J.; Shim, H.-Y.; Lee, Y.-R.; Park, T.-R.; Park, S.-H.; Lee, I.-G. Study on Systematic Ransomware Detection Techniques. In Proceedings of the 2021 23rd International Conference on Advanced Communication Technology (ICACT), Online, 7–10 February 2021; pp. 297–301.
14. Baliga, A.; Iftode, L.; Chen, X. Automated containment of rootkits attacks. *Comput. Secur.* **2008**, *27*, 323–334. [CrossRef]
15. Tian, D.; Ma, R.; Jia, X.; Hu, C. A Kernel Rootkit Detection Approach Based on Virtualization and Machine Learning. *IEEE Access* **2019**, *7*, 91657–91666. [CrossRef]
16. Rad, B.B.; Masrom, M.; Ibrahim, S. Evolution of Computer Virus Concealment and Anti-Virus Techniques: A Short Survey. April 2011. Available online: http://arxiv.org/abs/1104.1070 (accessed on 21 November 2021).
17. Bhaskar, V.; Patil, R.J.J. Computer Virus and Antivirus Software—A Brief Review. *Int. J. Adv. Manag. Econ.* **2014**, *4*, 4.

18. Al-Asli, M.; Ghaleb, T.A. Review of Signature-based Techniques in Antivirus Products. In Proceedings of the 2019 International Conference on Computer and Information Sciences (ICCIS), Aljouf, Saudi Arabi, 3–4 April 2019; pp. 1–6.
19. Scott, J. Signature Based Malware Detection Is Dead. February 2017. Available online: https://icitech.org/wp-content/uploads/2017/02/ICIT-Analysis-Signature-Based-Malware-Detection-is-Dead.pdf (accessed on 9 September 2021).
20. Sathyanarayan, V.S.; Kohli, P.; Bruhadeshwar, B. Signature Generation and Detection of Malware Families. In *Information Security and Privacy*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 336–349.
21. Bazrafshan, Z.; Hashemi, H.; Fard, S.M.H.; Hamzeh, A. A survey on heuristic malware detection techniques. In Proceedings of the The 5th Conference on Information and Knowledge Technology, Tehran, Iran, 28–30 May 2013; pp. 113–120.
22. Treadwell, S.; Zhou, M. A heuristic approach for detection of obfuscated malware. In Proceedings of the 2009 IEEE International Conference on Intelligence and Security Informatics, Dallas, TX, USA, 8–11 June 2009; pp. 291–299.
23. Harley, D.; Lee, A. Heuristic Analysis–Detecting Unknown Viruses 2007. Available online: https://www.welivesecurity.com/wp-content/uploads/200x/white-papers/Heuristic_Analysis.pdf (accessed on 10 October 2021).
24. Dube, T.; Raines, R.; Peterson, G.; Bauer, K.; Grimaila, M.; Rogers, S. Malware target recognition via static heuristics. *Comput. Secur.* **2012**, *31*, 137–147. [CrossRef]
25. Aslan, O.; Samet, R. A Comprehensive Review on Malware Detection Approaches. *IEEE Access* **2020**, *8*, 6249–6271. [CrossRef]
26. Wang, X.; Yang, G.; Li, Y.; Liu, D. Review on the application of artificial intelligence in antivirus detection systemi. In Proceedings of the 2008 IEEE Conference on Cybernetics and Intelligent Systems, Chengdu, China, 21–24 September 2008; pp. 506–509.
27. Singhal, P. Malware Detection Module using Machine Learning Algorithms to Assist in Centralized Security in Enterprise Networks. *Int. J. Netw. Secur. Its Appl.* **2012**, *4*, 61–67. [CrossRef]
28. De Lima, S.M.L.; de Silva, H.K.L.; da Luz, J.H.S.; do Lima, H.J.N.; de Silva, S.L.P.; de Andrade, A.B.A.; da Silva, A.M. Artificial intelligence-based antivirus in order to detect malware preventively. *Prog. Artif. Intell.* **2020**. [CrossRef]
29. Forain, I.; de Oliveira Albuquerque, R.; Sandoval Orozco, A.; García Villalba, L.; Kim, T.-H. Endpoint Security in Networks: An OpenMP Approach for Increasing Malware Detection Speed. *Symmetry* **2017**, *9*, 172. [CrossRef]
30. Latorre, J.M.; Cerisola, S.; Ramos, A.; Palacios, R. Analysis of stochastic problem decomposition algorithms in computational grids. *Ann. Oper. Res.* **2009**, *166*. [CrossRef]
31. Karantzas, G.; Patsakis, C. An Empirical Assessment of Endpoint Detection and Response Systems against Advanced Persistent Threats Attack Vectors. *J. Cybersecur. Priv.* **2021**, *1*, 387–421. [CrossRef]
32. Galteland, H.; Gjøsteen, K. Malware, Encryption, and Rerandomization–Everything Is Under Attack. In Proceedings of the International Conference on Cryptology, Kuala Lumpur, Malaysia, 1–2 December 2017; pp. 233–251.
33. Kong, D.; Jhi, Y.-C.; Gong, T.; Zhu, S.; Liu, P.; Xi, H. SAS: Semantics aware signature generation for polymorphic worm detection. *Int. J. Inf. Secur.* **2011**, *10*, 269–283. [CrossRef]
34. Wanswett, B.; Kalita, H.K. The Threat of Obfuscated Zero Day Polymorphic Malwares: An Analysis. In Proceedings of the 2015 International Conference on Computational Intelligence and Communication Networks (CICN), Jabalpur, India, 12–14 December 2015; pp. 1188–1193.
35. Tang, Y.; Xiao, B.; Lu, X. Using a bioinformatics approach to generate accurate exploit-based signatures for polymorphic worms. *Comput. Secur.* **2009**, *28*, 827–842. [CrossRef]
36. Daoud, E. Al Metamorphic Viruses Detection Using Artificial Immune System. In Proceedings of the 2009 International Conference on Communication Software and Networks, Chengdu, China, 27–28 February 2009; pp. 168–172.
37. Gibert, D.; Mateu, C.; Planes, J.; Marques-Silva, J. Auditing static machine learning anti-Malware tools against metamorphic attacks. *Comput. Secur.* **2021**, *102*, 102159. [CrossRef]
38. Filiol, E. Strong Cryptography Armoured Computer Viruses Forbidding Code Analysis: The Bradley Virus. Ph.D. Thesis, Institut National de Recherche en Informatique et en Automatique, Le Chesnay-Rocquencourt, France, 2004.
39. Rudd, E.M.; Rozsa, A.; Gunther, M.; Boult, T.E. A Survey of Stealth Malware Attacks, Mitigation Measures, and Steps Toward Autonomous Open World Solutions. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1145–1172. [CrossRef]
40. Maiorca, D.; Ariu, D.; Corona, I.; Aresu, M.; Giacinto, G. Stealth attacks: An extended insight into the obfuscation effects on Android malware. *Comput. Secur.* **2015**, *51*, 16–31. [CrossRef]
41. Patsakis, C.; Casino, F.; Katos, V. Encrypted and covert DNS queries for botnets: Challenges and countermeasures. *Comput. Secur.* **2020**, *88*, 101614. [CrossRef]
42. Nadler, A.; Aminov, A.; Shabtai, A. Detection of malicious and low throughput data exfiltration over the DNS protocol. *Comput. Secur.* **2019**, *80*, 36–53. [CrossRef]
43. Ho, J.-W. Covert Channel Establishment Through the Dynamic Adaptation of the Sequential Probability Ratio Test to Sensor Data in IoT. *IEEE Access* **2019**, *7*, 146093–146107. [CrossRef]
44. Loman, M. LockFile Ransomware's Box of Tricks: Intermittent Encryption and Evasion–Sophos News. 2021. Available online: https://news.sophos.com/en-us/2021/08/27/lockfile-ransomwares-box-of-tricks-intermittent-encryption-and-evasion/ (accessed on 31 August 2021).
45. Chen, P.; Desmet, L.; Huygens, C. A Study on Advanced Persistent Threats. In Proceedings of the IFIP International Conference on Communications and Multimedia Security, Aveiro, Portugal, 25–26 September 2014; pp. 63–72.
46. Ahmad, A.; Webb, J.; Desouza, K.C.; Boorman, J. Strategically-motivated advanced persistent threat: Definition, process, tactics and a disinformation model of counterattack. *Comput. Secur.* **2019**, *86*, 402–418. [CrossRef]

47. Virvilis, N.; Gritzalis, D.; Apostolopoulos, T. Trusted Computing vs. Advanced Persistent Threats: Can a Defender Win This Game? In Proceedings of the 2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing, Vietri sul Mare, Italy, 18–21 December 2013; pp. 396–403.
48. Yadav, T.; Rao, A.M. Technical Aspects of Cyber Kill Chain. In Proceedings of theThird International Symposium on Security in Computing and Communications (SSCC'15), Kochi, India, 10–13 August 2015; pp. 438–452.
49. Bahrami, P.N.; Dehghantanha, A.; Dargahi, T.; Parizi, R.M.; Raymond Choo, K.-K.; Javadi, H.H.S. Cyber Kill Chain-Based Taxonomy of Advanced Persistent Threat Actors: Analogy of Tactics, Techniques, and Procedures. *J. Inf. Process. Syst.* **2019**, *15*, 865–889. [CrossRef]
50. Al-Shaer, R.; Spring, J.M.; Christou, E. Learning the Associations of MITRE ATT&CK Adversarial Techniques. April 2020. Available online: http://arxiv.org/abs/2005.01654 (accessed on 21 December 2021).
51. Alexandru, M. Automation of Log Analysis Using the Hunting ELK Stack. *Rom. Cyber Secur. J.* **2021**, *3*, 59–64.
52. Al Shibani, M.; Anupriya, E. Automated Threat Hunting Using ELK Stack-A Case Study. *Indian J. Comput. Sci. Eng.* **2019**, *10*, 118–127. [CrossRef]
53. Kebande, V.R.; Karie, N.M.; Ikuesan, R.A. Real-time monitoring as a supplementary security component of vigilantism in modern network environments. *Int. J. Inf. Technol.* **2021**, *13*, 5–17. [CrossRef]
54. Kim, H.; Kwon, H.; Kim, K.K. Modified cyber kill chain model for multimedia service environments. *Multimed. Tools Appl.* **2019**, *78*, 3153–3170. [CrossRef]
55. Sapegin, A.; Jaeger, D.; Cheng, F.; Meinel, C. Towards a system for complex analysis of security events in large-scale networks. *Comput. Secur.* **2017**, *67*, 16–34. [CrossRef]
56. Tsigkritis, T.; Spanoudakis, G. Assessing the genuineness of events in runtime monitoring of cyber systems. *Comput. Secur.* **2013**, *38*, 76–96. [CrossRef]
57. Mavroeidis, V.; Jøsang, A. Data-Driven Threat Hunting Using Sysmon. In Proceedings of the 2nd International Conference on Cryptography, Security and Privacy, Guiyang, China, 16–19 March 2018; ACM: New York, NY, USA, 2018; pp. 82–88.
58. Microsoft Sysmon. 1996. Available online: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon (accessed on 18 December 2021).
59. Elastic Winlogbeat. 2018. Available online: https://www.elastic.co/es/downloads/beats/winlogbeat (accessed on 18 December 2021).
60. Elastic Elasticsearch, Logstash, Kibana (ELK). 2010. Available online: https://www.elastic.co/es/what-is/elk-stack (accessed on 18 December 2021).
61. Pranneth, J.N.; Sreedevi, M. Detecting and Analyzing the Malicious Windows Events using Winlogbeat and ELK Stack. *Int. J. Recent Technol. Eng.* **2019**, *7*, 716–720.
62. Fatemi, M.R.; Ghorbani, A.A. Threat Hunting in Windows Using Big Security Log Data. In *Security, Privacy and Forensics Issues in Big Data*; IGI Global: Hershey, PA, USA, 2020; pp. 168–188. [CrossRef]
63. Bin Hamid Ali, F.A.; Len, Y.Y. Development of host based intrusion detection system for log files. In Proceedings of the 2011 IEEE Symposium on Business, Engineering and Industrial Applications (ISBEIA), Langkawi, Malaysia, 25–28 September 2011; pp. 281–285.
64. Jain, U. Lateral Movement Detection Using ELK Stack. Master's Thesis, University of Houston, Houston, TX, USA, 2018.
65. Yang, G.; Cai, L.; Yu, A.; Meng, D. A General and Expandable Insider Threat Detection System Using Baseline Anomaly Detection and Scenario-Driven Alarm Filters. In Proceedings of the 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 763–773.
66. Torkaman, A.; Bahrololum, M.; Tadayon, M.H. A threat-aware Host Intrusion Detection System architecture model. In Proceedings of the 7'th International Symposium on Telecommunications (IST'2014), Tehran, Iran, 9–11 September 2014; IEEE: Piscataway, NJ, USA; pp. 929–933.
67. Abubakar, A.; Pranggono, B. Machine learning based intrusion detection system for software defined networks. In Proceedings of the 2017 Seventh International Conference on Emerging Security Technologies (EST), Canterbury, UK, 6–8 September 2017; pp. 138–143.
68. Giura, P.; Wang, W. A Context-Based Detection Framework for Advanced Persistent Threats. In Proceedings of the 2012 International Conference on Cyber Security, Alexandria, VA, USA, 14–16 December 2012; IEEE: Piscataway, NJ, USA; pp. 69–74.
69. Douglas, M. Powercat. 2015. Available online: https://www.sans.org/reading-room/whitepapers/testing/powercat-35807 (accessed on 23 November 2021).
70. Cocomazzi, A. ConPtyShell. Github Repository. 2019. Available online: https://github.com/antonioCoco/ConPtyShell (accessed on 23 November 2021).
71. Mittal, N. Invoke-PowerShellTcp. Github Repository. 2015. Available online: https://github.com/samratashok/nishang/tree/master/Shells (accessed on 23 November 2021).
72. More, H. Metasploit Framework. 2003. Available online: https://www.metasploit.com/ (accessed on 21 October 2021).
73. Delpy, B. Mimikatz. Github Repository. 2014. Available online: https://github.com/gentilkiwi/mimikatz. (accessed on 23 December 2021).
74. Bontchev, V. The problems of wordmacro virus upconversion. *Comput. Secur.* **1999**, *18*, 241–255. [CrossRef]

75. Oddvar, M.; Somerville, L. Living off the Land Binaries and Scripts (and also Libraries). Github Repository. 2018. Available online: https://lolbas-project.github.io/# (accessed on 23 December 2021).
76. Makris, C. Evaluation of the Detection Capabilities of the Open Source SIEM HELK. Master's Thesis, University of Piraeus, Piraeus, Greece, 2020.
77. Hsu, F.-H.; Lee, C.-H.; Luo, T.; Chang, T.-C.; Wu, M.-H. A Cloud-Based Real-Time Mechanism to Protect End Hosts against Malware. *Appl. Sci.* **2019**, *9*, 3748. [CrossRef]
78. Peng, P.; Yang, L.; Song, L.; Wang, G. Opening the Blackbox of VirusTotal. In Proceedings of the Internet Measurement Conference, Amsterdam, The Netherlands, 21–23 October 2019; ACM: New York, NY, USA, 2019; pp. 478–485.