



On the antivirus detection delay and malware defense evasion techniques: a VirusTotal-based evaluation

Alessandro Marchetto¹

Received: 6 July 2026 / Accepted: 8 August 2026
© The Author(s) 2026

Abstract

Malware stands for **MAL**icious soft**WARE** that aims to compromise digital systems and devices, interrupt services, and leak sensitive information. The antivirus detection capability is largely experimented by companies and organizations. However, such experiments: (i) focus only on the antivirus detection rate, without performing any in-depth analysis, and (ii) often report an almost perfect antivirus detection capability achieved under not clearly documented experiments' setup. This paper reports on two experiments conducted to evaluate the detection capability of antivirus, explore the role of defense evasion techniques, and the use of crowd-sourced detection rules. While the first experiment has been conducted installing the antivirus locally, the second and larger one has been conducted via VirusTotal, an online multi-antivirus platform. The results show that: (i) a non-trivial percentage of malware samples (up to 26.8%) remained undetected by antivirus in VirusTotal, sometimes for a long period (e.g., 5% more than 23 days), and most of them are related to potentially critical threats; (ii) groups of AVs demonstrated similar behaviors in terms of malware detection trends, agreement patterns, and overall effectiveness; (iii) few defense evasion techniques result to be frequently incorporated into malware samples for which we observed a high detection delay; (iv) specific defense evasion techniques support malicious tactics; and (v) more than half of malware samples (up to 54%) do not trigger crowd-sourced detection rules in the online platform. Furthermore, we identified a set of lessons learned that can represent future research directions to improve the antivirus detection capability. The experiments show that antivirus on VirusTotal can miss or delay malware detection for a long period. Furthermore, we observe that malware samples frequently use specific defense evasion techniques to elude these antivirus engines, and often do not trigger crowd-sourced detection rules.

Keywords Cybersecurity · Experimentation · Malware · Antivirus

1 Introduction

Several definitions exist in the literature to describe what a *malware* is, e.g., [1, 2] (NIST). However, it is common sense that the term malware derives from the combination of **MAL**icious and soft**WARE**, thus it is a piece of software with malicious intents. Typically, an attacker uses a malware to gain access to a target device (victim's machine) affected by weaknesses and vulnerabilities [3] that could be exploited by the attacker, or by victim induced to act in a way that can compromise her security (e.g., by phishing and social engineering). Nowadays, *malware* is a generic term used to identify families of malicious software, e.g., Tro-

jans, Spywares, and Ransomwares, depending on the way in which the malware evades the security mechanisms, and on the attack target.

Antivirus (AVs) are one of the first barriers against malware and their detection capability is largely and continuously experimented by companies (e.g., Gartner [4], Security.org [5] Forrester [6]). They provide market and technical reports in which existing AVs are ranked based on several criteria, including information from the vendors and expert feedback. Laboratories and organizational bodies (e.g., AV-TEST [7], AV-Comparatives [8], SE-Labs [9], and MITRE ATT&CK [10]) also continuously provide reports about their experiments. However, it is widely recognized in the scientific literature that the real capability of AVs to detect malware is not well-studied and documented by such experiments. In the scientific literature, in fact, we can observe that: (*limit 1*) these experiments focus only on the AVs' detection

✉ Alessandro Marchetto
alessandro.marchetto@unitn.it

¹ University of Trento, Trento, Italy

rate, without conducting any in-depth analysis on malware strategies and AVs' defense mechanisms [11, 12]; (*limit 2*) the experimental setup and the used datasets are generally not well-documented; (*limit 3*) experiments report almost perfect detection rate, usually, higher than 95%; and (*limit 4*) similar experiments conducted by researchers report lower results, e.g., [13] reports 80% as the maximum malware detection rate, [11, 12] report a maximum rate up to 45% and 85%, depending on the used malware samples and AVs. These results can be partially explained by the fact that everyday the AV producers has to choice which detection signature add to their AVs' database, by considering the large number of malwares samples discovered every days [14]. To avoid performance degradation of their products, in fact, they often prioritize the development of detection rules, thus focusing on those ones related to the most frequent and aggressive malware samples, i.e., the samples that pose a substantial threat to users' software and devices. This choice is clearly critical since can overall decrease the performance of the AVs.

Therefore, there is a need of experimentation that deeper investigates effectiveness and strategies of malware attacks and detection [12]. This paper first documents two experiments conducted to evaluate the capability of AVs to detect malware samples. Then, the paper deeply analyzes the collected data to study the relationship between the measured malware detection capability and the defense evasion techniques adopted by malware samples, as well as the support of crowd-sourced detection rules in the malware detection.

1.1 Contribution

To summarize, in this paper we report on two experiments conducted to evaluate the antiviruses' detection capability, and two further analysis focused on defense evasion techniques and crowd-sourced detection rules. The main collected results are the following:

- a non-trivial percentage of malware samples (up to 26.8%) remained undetected by the VirusTotal AVs for an extended period (e.g., exceeding 23 days for 5% of them); and most of them are associated with highly critical security threats;
- groups of AVs show similar behaviors in terms of malware detection trends, agreement patterns, and overall effectiveness;
- few specific defense evasion techniques (e.g., *Obfuscated files or Information*, *Scripting*, and *PortableExecutableInjection*) are frequently incorporated into malware samples for which we observed a high detection delay;
- a large set of defense evasion techniques (up to 41%) is used only by a specific malware family;

- specific defense evasion techniques support specific malicious tactics;
- more than half of malware samples (up to 54%) did not trigger the crowd-sourced detection rules considered by the online platform.

Finally, in our experiments, we identified lessons learned for both users and AV vendors, that can also represent future research directions in the field.

2 Background

Anti-malware software aims at detecting and analyzing existing malware and new variants.

The analysis of a malware [15] is mainly conducted by means of: (1) **static analysis** – executable and binary malware files (code) are analyzed statically to identify representative instruction patterns that can distinguish malicious code (i.e., the “signature” of the malware); (2) **dynamic analysis** – executable and binary malware files are executed, in a protected and closed environment (e.g., sandbox), for checking predefined behavioral properties of the malware file that could lead to its characterization and detection; (3) **Machine learning** (ML) based analysis – learning algorithms are trained to identify a malware or predict malicious intentions of a software, mainly via log-based analysis; and (4) **hybrid analysis** – a combination of previous methods.

Based on the type of analysis carried out, state-of-the-art malware detection approaches can be classified into three main groups [15, 16]. **Signature-based** approaches identify malware by using software signatures derived by applying static analysis. Each known malware is characterized by a signature stored in a database (e.g., via Hash function), so that it can be quickly matched with the signature of the code to be analyzed. These approaches are easy to apply for detecting already known malware, but often they are only partially adequate for detecting new malware and variants.

Behavioral-based approaches identify malware by applying dynamic analysis. The code to be analyzed is exercised in a protected environment (e.g., sandbox) and several parameters (e.g., executed processes, files used, network activities, scheduled tasks, resources used) are computed to characterize the software and compare it against the ones of known malicious code. These approaches are typically complex to be applied, time-consuming, and require the execution of the code to be analyzed. **Heuristic-based** approaches use both static and dynamic methods to derive heuristics for the malware detection. Based on previous observations on software signatures and parameters, heuristics can be derived to classify a software as malicious. For instance, the identification of specific sets of API calls and their execution order. These approaches are less precise in detecting known malware but

they are flexible enough to identify new malware and variants.

Malware detection and analysis are challenging mainly due to: (i) a large variety of software technology and environments; (ii) an increased complexity and continuous connectivity of software and systems; and (iii) the development of sophisticated defense evasion techniques. Defense evasion techniques are adopted by malware to bypass the AVs, i.e., to remain stealth, avoid detection, and increase the effort required to the anti-malware. Several works present and review existing malware defense evasion techniques (e.g., [17–19]). They rely on the ways of: **Spreading** – spreading the malware (e.g., malicious links, emails, social engineering, phishing campaigns); **Packing** – hiding the attacker’s signature, code, and data (e.g., packing the malicious code inside compressed files like zip and apk or inside other innocuous executable files such as JS and docx); **Obfuscating** – making an executable or a file difficult to discover or analyze (e.g., adopting polymorphic software to automatically change, compress, and encrypt the malicious code, so that it cannot be recognized easily); **Concealmenting** – analyzing, characterizing, and changing the host environment and the anti-malware software to deceive it (e.g., identifying and changing the security mechanisms setup as well as the host environment setup and characteristics in order to hide themselves); **Attacking** – identifying and neutralizing defense mechanisms (e.g., acquiring and neutralizing the security software and defensive tools); **Injecting** – injecting and executing the malicious code into the host environment as a “normal” software (e.g., executing the malicious payloads by hijacking the way operating systems run programs). In this work, we also use these categories for classifying malware samples.

3 Methodology

This section presents the methodology adopted to execute two experiments with the **goal** of evaluating the malware detection capability of AVs (**context** of our goal).

3.1 Research questions

According to goal and context, we identified the following research questions.

– **RQ1 (AVs effectiveness).** *How effective are AVs in detecting malware?* RQ1 gives an idea about how secure our systems and devices are. To answer RQ1, we tested different AVs against a set of real malware samples.

– **RQ2 (Non-detected malware).** *Is there any malware family that is frequently not detected?* RQ2 provides insights into the malware characteristics that help it evade AVs. To answer RQ2, we evaluated the malware detection capability of AVs

by grouping malware samples in families, and identifying different technologies adopted by each one.

– **RQ3 (Detection delay).** *How many days malware survive without being detected?* RQ3 gives an idea about how much our devices are exposed to non-detected malware. To answer RQ3, we evaluated the number of days needed to detect each malware sample.

3.2 Used services

We used two online services: MalwareBazaar [20] and VirusTotal [21]. This latter service has been used only in the second experiment, conversely in the first experiment the considered AVs have been locally installed.

MalwareBazaar is a community project that collects and makes malware samples available for further study and analysis. VirusTotal is a service that inspects suspected malware samples submitted via file/ URL/ SHA with several AVs (up to 70 at the time of the experiment). VirusTotal uses several AV engines to analyze each provided sample, and produces a static report (also named “verdict”) with several types of information about the conducted analysis, among which number and details about the used AVs, and the list of the engines that recognized the sample as malicious (named VT_{score}).

3.3 Experiments

We conducted two experiments.

1. **In-lab experiment:** we locally installed 4 AVs into an isolated virtual laboratory, and we measured their detection capability with respect to 360 malware samples downloaded from MalwareBazaar. Then we conducted a manual analysis on a subset of samples for identifying non-malware incorrectly detected as malware, and real but not detected malware samples.
2. **In-large experiment:** we conducted a large experiment for testing the capability of 12 AV engines in VirusTotal against a set of 8.9k malware samples provided by MalwareBazaar.

3.4 In-lab experiment

To execute this first experiment, we built a virtual laboratory composed of 5 virtual machines (VMs) to create an isolated and controlled environment, thus avoiding the risk of malware propagation. One of the VMs accesses the Internet to collect malware samples and act as malware dispatcher towards the other 4 by means of a 1-to-1 connection, i.e., the other 4 VMs act as victims. In this experiment, we focus on desktop Windows-based environments, as it is the most widespread OS at the time of the experiment, 75%

of desktop market [22]. Specifically, in the 4 victim VMs we installed Windows 10 OS (the most widespread Windows OS at the time of the experiment, even if Windows 11 was already available, i.e., 73% against 23% respectively [22]). In the remaining VM, we installed a Linux-based OS (i.e., Ubuntu 20.04). This latter VM acts as a malware dispatcher server, it can: (i) reach the internet to download the malware samples from MalwareBazaar, while the other VMs cannot reach the internet, for security reasons; (ii) act also as web server from which the victims' VMs download the files required for the experiment by means of specific GET requests towards an internal web server. The samples are hence downloaded in the Ubuntu VM and added to a compressed file protected by a password for a further protection before being distributed to the victims' VMs during the experiment. In the victims' VMs, we installed four popular AVs (one per VM) highly scored in terms of detection performance by independent laboratories [14]: Avast (version 22.12), MalwareBytes (version 4.5), BitDefender (version 26.0), and WindowsDefender (version 4.18). Each AV has been used with its default configuration. The first three AVs use both signature and behavioral-based approaches, with the additional possibility of using a cloud-based heuristic analysis (not part of our experiment). The last one uses only the signature-based approach, and since it was no longer supported, we included it only as a baseline. During the experiment (the process is detailed below), we downloaded our malware samples into each victim VM, and started the AV scan, either automatically or manually. This process has been replicated in multiple paths and folders of the victim OS. Typically, "landing" could be important since, often, an AV scans every new file that appears on the computer by focusing, in particular, on those folders that are more likely to be the landing place for unwanted software, e.g., since they contain temporary or OS files. Furthermore, in the experiment, an expert manually analyzed some randomly selected malware samples to identify the presence of some non-malicious samples (false positives and negatives).

3.4.1 Process

We executed the In-lab experiment as follows.

- e1_{1a}** We analyzed 160 malware samples (out of 360) against our 4 AVs (we updated the AVs just before starting the experiment), and computed the *undetected score* ("uscore") for each AV (i.e., the percentage of non-detected malware samples).
- e1_{1b}** After 10 days, we updated the AVs' database and re-tested the 160 samples.

Table 1 Malware samples' families

Id	Description
Js	JavaScript code mainly used to pack malicious code, often a command and control malware
Heodo	Heodo is based on an infrastructure-as-a-service for content delivery, i.e., used to install Ransomware and Remote Access Trojan (RAT)
Docx	Word documents can contain VBS executable code under the form of MACRO; such code can be user triggered or even on-open triggered
Mirai	Mirai targets IoT devices that are later added to a botnet used for large-scale attacks
DLL	A Dynamic Link Library is an executable file that allows applications to share code; a malicious code inside a dll can execute the attacker's code
RemCosRat	Remcos was born as a Remote Administration Tool, then modified to act as backdoors for attackers
Stealer	(RedLine)Stealer steals information from networking applications, e.g., web browsers
Trojan	Trojan is a malicious code that disguises itself as legitimate code, aiming at mislead the victim's that could executed it

- e1_{2a}** After 20 days, we conducted a second round of analysis by testing other 200 malware samples against our AVs (configured as in e1_{1b}).
- e1_{2b}** After 10 days, we updated the AVs and repeated the analysis (step e1_{2a}).
- e1₃** By involving a security expert, malware samples have been deeply analyzed for identifying false positive and false negative cases. Specifically, 14 samples selected randomly among the ones not detected by at least two AVs, have been analyzed.

3.4.2 Malware samples

In this first experiment, we used 360 malware samples [23] related to 8 malware families (listed in Table 1).

We selected these samples by considering both the type of malware and the time of the first upload in the platform, thus selecting recent samples. This selection approach opens up to the possibility of some wrongly submitted malware, i.e., files that are supposed to be malicious but they are not. Hence, for each malware sample, we also collected its VT_{score} . We observed that only 4 of our samples (3 *Docx* and 1 *DLL*) received a VT_{score} equal to 0, meaning that only these 4 (out of 360) could be non-malware samples.

3.5 In-large experiment

In the In-large experiment, for one month we downloaded malware samples from MalwareBazaar and then provided them to VirusTotal for the analysis on 12 selected AV

engines provided by the platform: AVG, AVAST, BitDefender, Kaspersky, McAfee, Microsoft, Symantec, TrendMicro, Malwarebytes, Panda, Tencent, and SentinelOne. In this second experiment, hence, we provided the malware samples to VirusTotal and we analyzed the produced verdicts. Table 2 details both versions and number of updates of the VirusTotal AVs used in the experiment for each of the two experiment steps (named e_{21} and e_{22} , see Sect. 3.5.1 for the process adopted in this In-large experiment). Specifically, the column “Most used AV version” of the table reports the AV version that has been mainly used by VirusTotal (on average, to test at least 60% of samples); the last two columns detail the number of AV versions used during the experiment, and the average number of AV database updates per sample, respectively. We can see that overall, we considered 130 versions of VirusTotal AVs, i.e., 92 and 38 in e_{21} and e_{22} , respectively. In addition, we observe a large variability of the AVs’ engines updates declared in the period of the experiment by the platform. For instance, in e_{22} , SentinelOne has been updated 7 times in the VirusTotal in the period of the experiment, while BitDefender and McAfee only one time. While in e_{21} the database update is 1 by construction (see the experiment process). On average, in e_{22} the AV databases have been updated 2.3 times per tested sample in VirusTotal, excluding SentinelOne, for which the number of updates is less, 1.2 per sample. This can be probably explained by considering the high frequency of updates for such specific AV. Although by using VirusTotal platform the AVs setup is not under our control, we resort to the use of this service since it gives us the possibility to: (i) involve a large number of AVs and malware samples; (ii) collect detailed reports about the conducted analysis (the so called “VirusTotal verdict”); (iii) adopt a standardized experimental environment, thus making the experiment highly repeatable; and (iv) use a platform that provides several AVs’ engines used in similar works documented in the literature, such as [11, 12] (see Sect. 9 for a detailed analysis).

3.5.1 Process

The In-large experiment has been conducted in two steps.

- e_{21}** Every two days for one month, 500 new malware samples have been identified in MalwareBazaar and provided to VirusTotal, for the analysis on 12 AVs. The output has been stored locally for the post-analysis used to answer RQ1 and RQ2.
- e_{22}** Starting from an initial set of 500 malware samples tested daily against our AVs for one month, we evolved the dataset by keeping only the undiscovered samples and replacing the detected ones. After one month, we stopped replacing the discovered samples with new ones and simply continued checking the remaining

Table 2 In-large experiment: AV setup

AV	Most used version	# AV versions	(avg) # DB updates
Experiment step: e_{21}			
AVG	22.11.7701.0	9	1
Avast	22.11.7701.0	7	1
BitDefender	7.2	1	1
Kaspersky	22.0.1.28	5	1
McAfee	6.0.6.653	4	1
Microsoft	1.1.23050.3	21	1
Symantec	1.20.0.0	17	1
TrendMicro	11.0.0.1006	5	1
Malwarebytes	4.5.5.54	8	1
Panda	4.6.4.2	2	1
Tencent	1.0.0.1	1	1
SentinelOne	23.2.0.1	12	1
Experiment step: e_{22}			
AVG	22.11.7701.0	3	2.3
Avast	22.11.7701.0	3	2.3
BitDefender	7.2	1	2.4
Kaspersky	22.0.1.28	3	2.3
McAfee	6.0.6.653	1	2.3
Microsoft	1.1.23060.1005	8	2.3
Symantec	1.20.0.0	4	2.3
TrendMicro	11.0.0.1006	1	2.3
Malwarebytes	4.5.5.54	5	2.3
Panda	4.6.4.2	1	2.4
Tencent	1.0.0.1	1	2.3
SentinelOne	23.2.0.1	7	1.2

samples, ensuring that each sample was detected or tested for a total of 30 days.

To answer RQ3, we computed the number of days for which each malware sample remained non-detected. Furthermore, we identified the malware family and the used technology that characterize the most resistant (undetected) samples. To identify the possible presence of false positive cases, after a long period, we additionally conducted a post-experiment analysis by (re-)checking the samples and by collecting their VT_{score} . We assume that if, after this long period, a sample has VT_{score} equal or close to 0, it can be reasonably considered as a non-malware.

3.5.2 Malware samples

According to the sample selection process applied in the In-lab experiment, we started the selection of the malware samples to be used in the In-large experiment from Malware-

Table 3 In-large Experiment: Malware families

Id	Freq. (%)		Description
	e2 ₁	e2 ₂	
Msil	28.4	21.4	Trojan based on the Microsoft intermediate language
Linux	9.5	11.2	Trojan that steals information (variants of Mirai)
Unknown	7.8	7.9	Not identified
Mirai	5.9	8	Trojan that targets connected devices
Stealer	2.8	1.6	Trojan steals information
Cryxos	2.6	<1%	Trojan that steals information by using JS
Zusy	2.2	4.4	Trojan that steals (bank) information by using man-in-the-middle attacks
Amadey	1.7	<1%	Trojan that steals system information via botnet agents
Disabler	1.6	<1%	Trojan that uses Microsoft Intermediate Language
Lazy	1.6	3.9	Ransomware that encrypts user data by using Microsoft Intermediate Language
Barys	1.3	<1%	Malware that opens the system to other infections
Stop	1	2.2	Ransomware that encrypts victim's data
Pswx	<1%	1.9	Spyware
Genericfca	<1%	1.2	Trojan that steals information
Injuke	<1%	1.2	Trojan that steals information
Botx	<1%	1.2	Trojan that steals information via botnet agents
Gcleaner	<1%	1.2	Trojan that steals information
Others	32.8	32.7	

bazar by considering: (i) the time of the first upload in the platform, aiming to select modern and recent samples; and (ii) the type of malware, aiming to select different types of malware samples. We did not applied any additional selection filters. Overall, we analyzed 8.9k malware samples (i.e., 7623 in **e2₁** and 1298 in **e2₂**), related to 10 threat categories related to malware families/types (e.g., Trojan, Ransomware, Virus, Spyware), delivery mechanisms (e.g., Dropper, Downloader, Phishing) e risk categories (e.g., Pua, Hacktool, Adware). We randomly selected such samples among the ones added in MalwareBazaar. Table 3 reports the distribution of samples into the main malware families. We grouped in the row “Others” the malware families that are represented by less than 1% of samples. In this paper, we will use the term “malware families” as a broader classification to encompass specific malware strains (e.g., Mirai, Amadey, Stop, Cryxos) as well as their behavioral types and capabilities (e.g., Stealer, Pswx),

Table 4 In-lab - **e1₁**: undetected malware (#) and undetection score (uscore)

Malware family	Avast		BitDefender		Malwarebytes	
	#	uscore	#	uscore	#	uscore
e1_{1a}						
Js	0	0%	0	0%	20	100%
Heodo	1	5%	1	5%	2	10%
Docx	12	60%	10	50%	0	0%
Mirai	0	0%	0	0%	0	0%
DLL	3	15%	6	30%	2	10%
RemC.	0	0%	1	5%	4	20%
Stealer	0	0%	0	0%	0	0%
Trojan	4	20%	1	5%	4	20%
Total	20		19		32	
Average		12.5%		11.9%		20%
e1_{1b} (after 10 days and AVs update)						
Js	0	0%	0	0%	20	100%
Heodo	0	0%	0	0%	0	0%
Docx	11	55%	10	50%	20	100%
Mirai	0	0%	0	0%	20	100%
DLL	2	10%	4	20%	2	10%
RemC.	0	0%	0	0%	0	0%
Stealer	0	0%	0	0%	0	0%
Trojan	3	15%	1	5%	3	15%
Total	16		15		65	
Average		10%		9.4%		40.7%

thus remaining consistent with the labels used by VirusTotal. The table shows that the most-frequent malware families were consistent in **e2₁** and **e2₂**: Msil, Mirai, and Stealer, i.e., Trojan malware that mainly aim at accessing the victim's device and stealing information. The full details such as the complete list of the considered malware samples (SHA 256) are available online [23].

4 Results

4.1 In-lab experiment

Tables 4 and 5 report the number of undetected malwares (#) and the undetection score (*uscore*) for each AV, collected in **e1₁** and **e1₂**. The tables do not report results of Windows Defender since it detected only one malware, as expected since the AV uses only the signature-based detection approach and it is no longer supported.

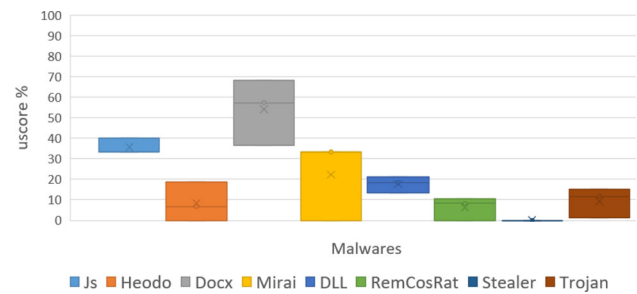
The results show that Avast and BitDefender achieved comparable results outperforming Malwarebytes: they both adopt signature-based and behavioral-based approaches to detect malware. In **e1_{1a}** (160 malware samples), Avast and

Table 5 In-lab - $e1_2$: undetected malwares (“#”) and undetection score (“uscore”)

Malware family	Avast #	Avast uscore	BitDefender #	BitDefender uscore	Malwarebytes #	Malwarebytes uscore
$e1_{2a}$: (after 20 days and with outdated AVs)						
Js	19	76%	18	72%	25	100%
Heodo	2	8%	2	8%	15	60%
Docx	17	68%	17	68%	25	100%
Mirai	0	0%	7	28%	25	100%
DLL	18	72%	17	68%	21	84%
RemC.	21	84%	15	60%	22	88%
Stealer	11	44%	10	40%	15	60%
Trojan	22	88%	13	52%	20	80%
Total	110		99		168	
Average		55%		49.5%		84%
$e1_{2b}$: (after 20 days and with updated AVs)						
Js	4	16%	1	4%	25	100%
Heodo	1	4%	0	0%	13	52%
Docx	5	20%	13	52%	25	100%
Mirai	0	0%	0	0%	25	100%
DLL	4	16%	11	44%	1	4%
RemC.	1	4%	1	4%	6	24%
Stealer	0	0%	0	0%	0	0%
Trojan	0	0%	1	4%	0	0%
Total	15		27		95	
Average		7.5%		13.5%		47.5%

BitDefender were not able to detect on average 12% of malware samples. In step $e1_2$ (after 20 days), Avast undetected on average 55% and BitDefender 49.5% of the malware samples, with their outdated databases, 7.5% and 13.5% of the samples, respectively, with their updated version. Figure 1 shows boxplots about the undetected malware families in $e1_1$ and $e1_2$. The families that are less detected are: *Trojan*, *Docx*, and *Js*. They have also shown a rapid evolution, as suggested by the high undetection score in $e1_{1a}$ and $e1_{1b}$, in contrast to $e1_{2a}$ and $e1_{2b}$.

Malwarebytes overall achieved less competitive results than other AVs. Like other AVs, it combines signature-based and behavioral-based approaches, but differently from others that are declared to be AntiVirus, Malwarebytes is an AntiMalware. Even if nowadays these two terms are used as synonyms (as we did in this paper) they are not exactly the same. Essentially, Malwarebytes will not go beyond the potentially malicious file under analysis, such as a JS code or a Docx macro. This difference with the other two AVs is empirically confirmed in our experiment and, in fact, explains the low result achieved in cases such as *Js* and *Docx*. Focusing on *Mirai*, we recall that it mainly targets Linux-based IoT devices, so it is unlikely to be detected by a Windows-only

**Fig. 1** In-lab - Boxplots for undetected malware families ($e1_1$ and $e1_2$)

AV. Finally, concerning *Heodo*, we noticed that most of the *Heodo* samples not detected in $e1_2$ were composed of files of type: *.docx* and *.xlsx*.

Table 6 shows results of the qualitative analysis performed in $e1_3$ in terms of technology (“Tech.”), number of analyzed samples (“#”) and false negatives. While the expert did not detect any false positive samples, she identified some false negative cases (missing detections of real malware). Overall, the considered AVs failed to detect from 57% to 71% of the samples analyzed by the expert. For instance, only one of the two *dll* samples considered is recognized as a non-real malware (confirmed also by the VT_{score} of 0/68). Instead the other *dll* sample (detected only by Malwarebytes) is a real malware that can read different registry keys of the hosting OS without however doing anything that looks harmful. This explains the low VT_{score} of the malware (13/69) and the non-detection of Avast and BitDefender. For all other samples, the expert was able to recognize code with malicious and fraudulent impact, so they are false negative cases (missing detection).

4.2 In-large experiment

To assess the detection effectiveness of various AVs and capture statistically significant differences, we analyzed the malware samples in the AVs provided in the VirusTotal platform, and then analyzed the platform verdicts. Specifically, we computed the malware undetection score (*uscore*) for the samples in $e2_1$, as the percentage of undetected malware samples per AV, and subsequently also per malware family. We then applied the Wilson Score Interval for binomial proportions at a 95% confidence interval (CI). For each computed *uscore*, the Wilson Score Interval provides an asymmetric range within which the true, theoretical error rate resides. To evaluate the detection effectiveness of the AVs, we analyzed the *uscore* values and applied the “Overlap Rule” to examine size and overlap of their respective confidence intervals. For instance, if the upper bound of AV A’s interval is strictly lower than the lower bound of AV B’s interval, the intervals do not overlap. This lack of overlap confirms that such AVs perform significantly different with a confidence level of

Table 6 In-lab qualitative analysis (e1₃)

Tech.	#	Non-detected real malwares		
		Avast	BitDefender	Malwarebytes
Malware Family: Js				
<i>js</i>	2	1	2	2
Malware Family: Heodo				
<i>docx</i>	1	0	0	0
<i>xlsx</i>				
Malware Family: Docx				
<i>doc</i>	4	2	4	4
Malware Family: DLL				
<i>dll</i>	2	1	1	0
Malware Family: RemCosRat				
<i>doc, exe</i>	2	1	2	1
Malware Family: Trojan				
<i>apk</i>	3	3	1	3
<i>zip</i>				
total	14	8 (57%)	10 (71%)	10 (71%)

95%, i.e., the observed difference is statistically significant ($p < 0.05$) and not due to random chance. We also performed a *Cochran's Q* test to globally compare the performance of the AVs. If the computed test p -value < 0.05 , we can reject the null hypothesis, which states that there is no statistically significant difference in the detection effectiveness across AVs. In such cases, we then applied the post-hoc analysis to compare the performance of each AV with the average of the other ones using two-tailed McNemar tests. Since for some malware families we collected few samples (e.g., less than 15), in such cases, exact binomial procedures were employed to compute reliable p -value. Bonferroni adjustment was also incorporated to identify specific strengths and weaknesses of each AV. This approach helps to pinpoint, for instance, which malware families are significantly more difficult to detect for a given AV.

Table 7 reports the malware undetection score achieved in e2₁ per each considered VirusTotal AV, and the undetection score per threat category (the last column “Others” contains samples of several families for which we have a limited number of instances). On average, we see that 26.77% of the samples remains undetected by almost all VirusTotal AVs. However, we observe that the average uscore ranges from 9.33% up to 56.85%, we argue that this is mainly due to the different policies and detection paradigms adopted by the considered antivirus engines. Most of the undetected samples resort to the *Trojans*: *Pua*, *Adware*, and *Dropper*. For instance, *Adware* and *Dropper* are recognized as malicious samples since they are “unwanted software” (e.g., software

for advertising, and software installers), but AVs can classify them differently or give them lower detection priority. The boxplots in Figure 2 report the uscore per threat category. The percentage of undetected malware samples is almost similar for all considered AVs in case of *Trojan*, *Virus* and *Ransomware*. Conversely, a large variability characterizes *Downloader*, *Phising*, and *Adware*, thus showing that AVs have different detection capability, or they use different criteria and priority to categorize these threats as malicious. In terms of confidence uscore Wilson Score Intervals overlap, we can identify three groups of AVs that exhibit comparable results: (g1) AVG, Avast, BitDefender, Kaspersky, Microsoft, and Symantec; (g2) BitDefender, McAfee, and SentinelOne; and (g3) the other AVs. In terms of *Cochran's Q* test, the obtained p -value is less than 0.001, thus confirming a significant difference in the behavior of the AVs. In detail, the post-hoc analysis (in Table 7 the outcome is represented by $\uparrow$ and $\downarrow$ to highlight a statistically significant positive and negative trend, respectively) indicates interesting performance of some AVs (e.g., AVG, Avast, Kaspersky, Microsoft, and Symantec), in detecting *Trojan*, while other ones seem to suffer the detection of specific malware families, such as *Trojan* for TrentMicro, Panda, and *Downloader* and *Adware* for Microsoft.

Table 8 reports the percentage of undetected malware samples for the most frequent families of Trojans (e2₁), *Cryxos*, *Msil*, and *Linux* (Mirai/Linux) are among the most undetected Trojans, e.g., 13.7% of the analyzed *Msil* samples remain undetected. The table also shows the results of the *Cochran's Q* test computation and subsequent residual analysis conducted to compare the performance of the AVs. For instance, it can be observed that AVG and Avast exhibit specific limitation in detecting *Cryxos*, a JS-based Trojan to steal information, whereas they successfully detected the majority of the samples related to *Msil*, *Linux*, and *Mirai*. Table 9 details the main technology adopted for the malware propagation (e.g., JS, MS-Office documents, Linux executable). The technology that is frequently used is related to the MS-Windows environment (i.e., on average, 46% of executable files, 4.1% of DLL files, and 5.3% of MS-Office documents such as *Docx*), even if a non-trivial percentage of samples (on average, 10%) uses JS, Linux executable (Elf) files, and compressed files' format (e.g., ZIP, TAR, and APK).

Survival Analysis

To analyze the collected detection delay systematically, we performed a Kaplan-Meier survival analysis on the malware samples. Figure 3 depicts the overall survival function $S(t)$, representing the probability that a malware sample remains undetected by an AV over time. The curve shows that most of the detected malware samples are identified within 10 days. However, the curve plateaus significantly after day

Table 7 In-large: undetected malware samples and threat categories (e21)

Absolute number of analyzed Malware by threat category									
Malware	Trojan	Dropper	Hacktool	Downloader	Adware	Virus	Pua	Ransomware	Others
7623	7266	9	8	31	9	14	7	10	266
Undetection score ("uscore" - percentage % of undetected Malware)									
Frequently undetected Malware by family									
uscore	Trojan	Dropper	Hacktool	Downloader	Adware	Virus	Pua	Ransomware	Others
(95% CI)	(95% CI)	(95% CI)	(95% CI)	(95% CI)	(95% CI)	(95% CI)	(95% CI)	(95% CI)	(95% CI)
AVG	14.39	11.49 ↑	55.56	6.45	33.33	57.14	85.71	10	87.22 ↓
	13.62–15.2	10.78–12.25	26.67–81.12	1.79–20.72	12.06–64.58	32.59–78.62	48.69–97.43	1.79–40.42	82.67–90.71
Avast	14.39	11.49 ↑	55.56	6.45	33.33	57.14	85.71	20	87.22 ↓
	13.62–15.2	10.78–12.25	26.67–81.12	1.79–20.72	12.06–64.58	32.59–78.62	48.69–97.43	1.79–40.42	82.67–90.71
BitDefender	32.76	30.32 ↓	55.56	80.65	66.67	71.43	85.71	10	87.97 ↑
	31.71–33.82	29.27–31.39	26.67–81.12	63.72–90.81	35.42–87.94	45.35–88.28	48.69–97.43	1.79–40.42	83.51–91.35
Kaspersky	9.33	6.84 ↑	44.44	0	11.11	14.29	14.29	10	75.56 ↓
	8.69–10	6.28–7.44	18.88–73.33	0–11.03	1.99–43.5	4.01–39.94	2.57–51.31	1.79–40.42	70.06–80.34
McAfee	20.6	17.96 ↑	33.33	58.06	77.78	64.29	71.43	10	81.71 ↓
	19.7–21.52	17.09–18.86	12.06–64.58	40.77–73.58	45.26–93.68	38.76–83.66	35.89–91.78	1.79–40.42	77.7–86.78
Microsoft	11.47	8.35 ↑	55.56	61.29 ↓	88.89 ↓	50	71.43	10	82.33 ↓
	10.77–12.2	7.74–9.01	26.67–81.12	43.82–76.27	56.5–98.01	26.8–73.2	35.89–91.78	1.79–40.42	77.3–86.44
Symantec	14.13	10.98 ↑	77.78	51.61	44.44	71.43	85.71	10	86.09 ↓
	13.36–14.93	10.28–11.72	45.26–93.68	34.84–68.03	18.88–73.33	45.35–88.28	48.69–97.43	1.79–40.42	81.42–89.74
TrendMicro	44.08	42.07 ↓	88.89	64.52	77.78	64.29	85.71	20	91.35 ↑
	42.97–45.19	40.94–43.21	56.5–98.01	46.95–78.88	45.26–93.68	38.76–83.66	48.69–97.43	5.67–50.98	87.36–94.17
Malwarebytes	39.05	36.58 ↓	88.89	96.77	88.89	78.57	57.14	10	93.61 ↑
	37.96–40.15	35.48–37.7	56.5–98.01	83.81–99.43	56.5–98.01	52.41–92.43	25.05–84.18	1.79–40.42	90–95.97
Panda	56.85	55.16 ↓	88.89	96.77	88.89	71.43	100	20	94.36 ↑
	55.74–57.96	54.02–56.3	56.5–98.01	83.81–99.43	56.5–98.01	45.35–88.28	64.57–100	5.67–50.98	90.91–96.55
Tencent	40.57	38.49 ↓	66.67	83.87	55.56	64.29	57.14	10	89.85 ↑
	39.48–41.68	37.38–39.62	35.42–87.94	67.37–92.91	26.67–81.12	38.76–83.66	25.05–84.18	1.79–40.42	85.63–92.93
SentinelOne	23.68	22.9 ↓	44.44	12.9	88.89	57.14	57.14	20	38.72 ↑
	22.74–24.65	21.95–23.88	18.88–73.33	5.13–28.85	56.5–98.01	32.59–78.62	25.05–84.18	5.67–50.98	33.07–44.7
average	26.77	24.39	62.96	51.61	62.96	60.12	71.43	12.5	83.08
	25.79–27.78	23.41–25.39	32.39–85.78	34.84–68.03	32.39–85.78	35.12–80.77	35.89–91.78	2.61–43.2	78.11–87.11

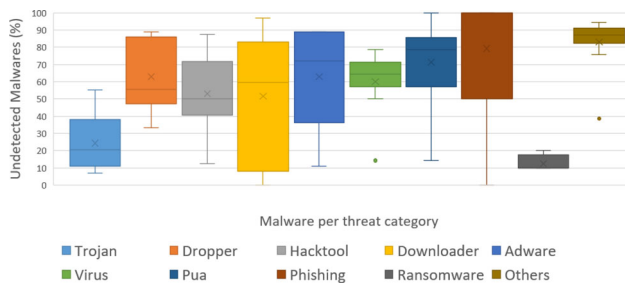


Fig. 2 In-large: Boxplots for undetected malwares per threat category (e2₁)

Table 8 In-large: Undetected Trojan families (e2₁). ↑ and ↓ indicate a statistically significant (positive or negative) trend observed with the Cochran's *Q* test

	Undetected malware samples (%) for the most frequent Trojan families				
	Cryxos	Msil	Linux	Mirai	Stealer
AVG	18.9 ↓	4.3 ↑	1.7 ↑	0.0 ↑	0.0
Avast	18.9 ↓	4.3 ↑	1.7 ↑	0.0 ↑	0.0
BitDefender	0.0 ↑	29.6 ↓	0.0 ↑	6.0 ↑	5.8 ↓
Kaspersky	7.0 ↓	2.4 ↑	2.6	0.0	0.0
McAfee	13.3 ↓	5.7 ↑	14.3 ↓	13.4 ↓	0.0 ↑
Microsoft	3.5	4.8	0.0 ↑	0.0	0.0
Symantec	14.8 ↓	6.5 ↑	7.4	3.3	0.0
TrendMicro	5.9	27.3 ↓	2.6 ↑	2.9 ↑	2.7 ↓
Malwarebytes	7.1 ↑	5.7 ↑	26.1 ↓	16.1 ↓	0.0 ↑
Panda	4.8 ↑	17.7 ↑	17.2 ↓	10.7 ↓	0.0 ↑
Tencent	2.0 ↑	28.0 ↓	0.0 ↑	0.0 ↑	0.0 ↑
SentinelOne	0.0 ↑	27.6 ↓	0.0 ↑	0.0 ↑	0.0 ↑
average %	8.0	13.7	6.1	4.4	0.7

Table 9 In-large: Undetected Trojan technology (e2₁)

	Most used technology mean (%)					
	JS	Ms-Office documents	MS-Windows executable	Compressed (e.g., ZIP)	MS-Windows DLL	Linux executable/Elf
AVG	19.9	6.8	31.8	15.8	4.6	2.2
Avast	19.9	6.8	31.8	15.8	4.6	2.2
BitDefender	1.9	1.4	72.8	4.8	2.7	8.9
Kaspersky	9.3	4.2	54.1	9.0	7.6	2.8
McAfee	17.0	6.9	24.6	9.8	2.0	23.4
Microsoft	8.1	11.7	31.9	14.5	4.6	3.5
Symantec	16.5	3.2	30.4	18.8	4.3	8.5
TrendMicro	7.2	1.9	65.4	8.1	3.1	5.2
Malwarebytes	9.2	7.6	22.5	8.6	2.4	38.5
Panda	6.4	5.2	43.9	8.7	1.9	26.3
Tencent	4.2	1.9	69.3	11.1	3.8	2.3
SentinelOne	0.0	6.4	73.9	7.4	7.3	0.7
average %	10.0	5.3	46.0	11.0	4.1	10.4

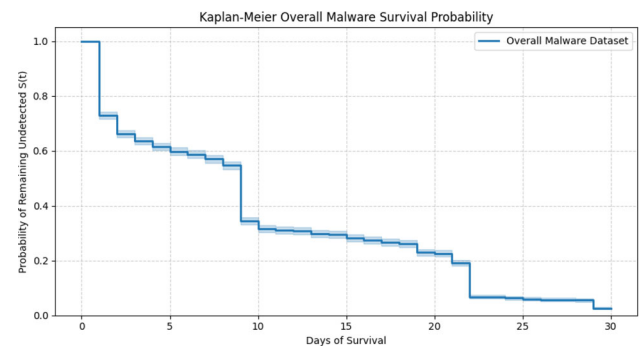


Fig. 3 In-large: Trojan detected by day, overall survival analysis (e2₂)

10, exhibiting a long tail. In particular, towards the end of the observation period (i.e., after 23 days), the curve does not drop to zero. Instead, approximately 5% of the malware samples remain undetected for more than 23 days. This detection delay shows that some Trojan malware can survive for a long period, thus potentially creating relevant problems for the victims' devices. Figure 4 shows the survival curve detailed for each AV. The curves reveal noticeable disparities in detection responsiveness across different AVs. To test whether these observed differences are statistically significant, we performed a multivariate Log-Rank test across AVs. The test yielded a statistically significant difference (p -value < 0.001), rejecting the null hypothesis of equal survival distributions. This confirms that malware detection delay depends on the AV, its setup and configuration (e.g., updates). As a further analysis, Table 13 reports the number of the samples that remain undetected for at least 23 days and for at least 30% of the considered AVs. Trojan families such as *Linux* (variant of Mirai also known as Mirai/Linux), *Mirai*, and *Zusy* have not been detected at least by half of the

AVs for a long period. In terms of adopted technology, these resistant Trojans mainly use Linux (Elf) and MS-Windows executable, thus suggesting that they adopt evasion strategies effective with respect to the AVs. We can note that most of such malware samples are often classified as “critical” in terms of severity. For instance, *Zusy* and *Alien* are banking Trojan designed to steal sensitive financial data and login credentials in Windows and Android systems. Again, *Mirai* and *Linux* target smart devices, enlisting them into a massive botnet used to launch large-scale DDoS attacks. This seems to clearly confirms the relevance of this type of analysis.

4.2.1 False Positives and Metrics

In the presented analysis, we did not consider the presence of false positives (samples wrongly identified as malicious), as we lack an oracle capable of distinguishing between malicious and non-malicious samples within our large dataset. Instead, we assumed as “correct” the labels provided by VirusTotal. To limit the impact of this possible issue, we focused on the 12 selected AVs known to be effective. However, recent studies (e.g., [24–26]) have shown that to completely “nullify the presence of false positives” a VirusTotal VT_{score} threshold of at least 20 AVs should be considered. In other words, if fewer than 20 AVs in VirusTotal flag a sample as malicious, we cannot be entirely certain of its true malicious nature. Therefore, aligned with such previous experiences, we adopted this threshold to estimate the presence of false positive cases within our sample dataset, and measure *consensus-based metrics*. To prevent circular reference bias, which occurs if the AVs’ detection outcome is also considered in the VirusTotal AVs set, we adopted a Leave-One-Out (LOO) consensus approach. Specifically, for each evaluated engine AV_i , its detection outcome was excluded from the VirusTotal consensus score ($VT_{score} - AV_i$) before applying the detection threshold (e.g., $TH = 20$). We refer to these metrics as *consensus-based metrics* (e.g., consensus-based F1-score), as they measure an individual AV’s detection alignment with the qualified majority consensus of the AVs in the VirusTotal platform.

Table 10 reports on the percentage of false positive cases per AV (FP %) along with the computed detection consensus-based metrics for $e2_1$ and $e2_2$. Specifically, for each AV, we measure consensus-based accuracy (Acc), precision (Prec), recall (Rec), and F1-score (F1) as follows.

- **Accuracy** – Measures the proportion of correctly identified samples among all considered samples:

$$Acc = \frac{TP + TN}{TP + FP + FN + TN} \quad (1)$$

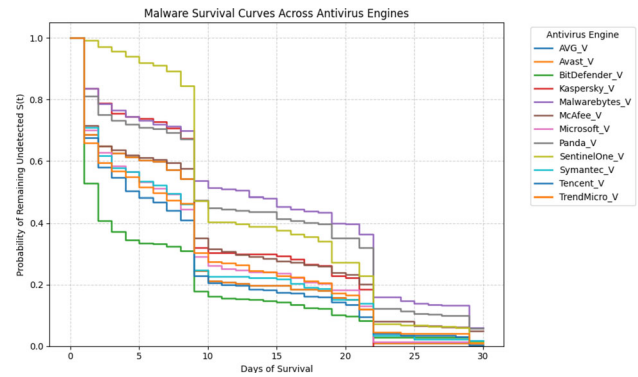


Fig. 4 In-large: Trojan detected by day, survival analysis by AV ($e2_2$)

- **Precision** – Measures the proportion of correctly identified malware samples among all detected ones:

$$Prec = \frac{TP}{TP + FP} \quad (2)$$

- **Recall** – Measures the proportion of actual malware samples correctly identified:

$$Rec = \frac{TP}{TP + FN} \quad (3)$$

- **F1-score** – The harmonic mean of precision and recall:

$$F1 = \frac{2TP}{2TP + FP + FN} \quad (4)$$

where, **TP** (True Positives) refers to correctly identified malicious samples, **FP** (False Positives) are non-malicious samples misidentified as malicious, **FN** (False Negatives) are malicious samples incorrectly identified as non-malicious, and **TN** (True Negatives) are correctly identified non-malicious samples.

In $e2_1$, we obtain, on average, a low FP rate of 2.92% (i.e., 211 samples out of 7623 analyzed), so with a limited potential impact on the conducted analysis. While Kaspersky and Microsoft achieved the highest percentage: 6.71% and 5.27%, respectively; Panda and TrendMicro achieved the lowest percentage: 0.42% and 1.1%, respectively. The detection consensus-based metrics show high variability for the considered AVs. For instance, F1-score ranges from 0.65 of Panda to 0.95 of Symantec. We observe that some AVs are quite prudent and selective. For instance, Panda reaches a precision of 0.99, with very few false positives, but with a recall of 0.48. Conversely, other AVs such as Kaspersky achieved a better balance in terms of precision and recall (F1-score of 0.94), but paying much in terms of false positives. Results of $e2_2$ are consistent with the ones in $e2_1$. This further confirms the different policies and detection paradigms adopted by the AVs.

To quantify the dispersion of the consensus-based F1-score calculated with the LOO method at different thresholds, for a given AV j across different threshold settings $T = \{t_1, t_2, \dots, t_K\}$, we computed the Coefficient of Variation (CV_j). The CV_j is defined as the ratio of the sample standard deviation (σ_j) to the sample mean (μ_j) of the metric under evaluation:

$$CV_j = \frac{\sigma_j}{\mu_j} = \frac{\sqrt{\frac{1}{K-1} \sum_{k=1}^K (x_{j,k} - \mu_j)^2}}{\frac{1}{K} \sum_{k=1}^K x_{j,k}}$$

where $x_{j,k}$ denotes the consensus-based F1-score achieved by AV j at threshold t_k , μ_j represents the consensus-based F1-score average value across all considered thresholds, and K is the total number of evaluated threshold configurations. Low values of CV_j represent almost stable performance of the AV j . Conversely, high values of CV_j represent threshold-related performance. Table 11 reports the obtained values for the calculated consensus-based F1-score at different thresholds (i.e., from 25% to 40% of the VirusTotal AVs). This analysis let's evaluate the sensitivity of the consensus-based F1-metrics at different thresholds. As shown in Table 11, the average CV across all AVs is less than 0.30 in **e2₁** and 0.46 in **e2₂**, with no single engine exceeding 2.91. This low relative difference demonstrates that the consensus-based metrics remain stable to the specific threshold considered.

4.2.2 False positives in GE23

Experiment **e2₂** shows that several malware samples remained undetected for at least 23 days and by more than 30% of the evaluated AVs in VirusTotal. To empirically confirm the malicious nature of these samples (Table 13), we re-analyzed them on the VirusTotal platform after one year to collect their updated verdicts and VT_{score} . The assumption is

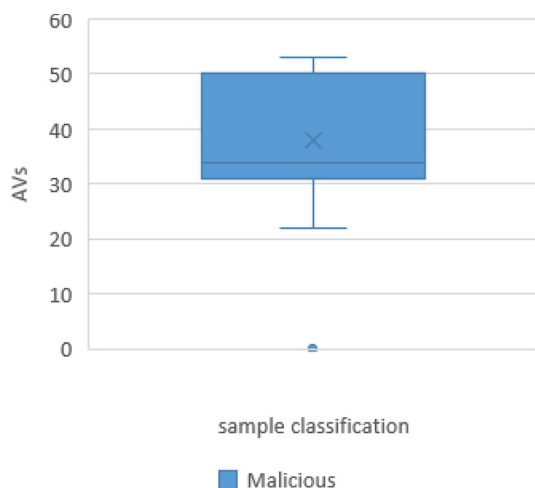


Fig. 5 In-large experiment: boxplot for GE23 samples' VT_{score}

Table 10 In-large: Consensus-based metrics for Undetected malware samples

Undetected malware samples: metrics					
	FP %	Acc	Prec	Rec	F1
e2₁					
AVG	3.41	0.89	0.96	0.91	0.93
Avast	3.42	0.89	0.96	0.91	0.93
BitDefender	3.19	0.73	0.95	0.72	0.82
Kaspersky	6.71	0.89	0.92	0.95	0.94
McAfee	3.59	0.83	0.95	0.84	0.90
Microsoft	5.27	0.86	0.94	0.90	0.92
Symantec	3.43	0.91	0.96	0.94	0.95
TrendMicro	1.1	0.65	0.98	0.61	0.75
Malwarebytes	1.43	0.69	0.98	0.66	0.79
Panda	0.42	0.54	0.99	0.48	0.65
Tencent	1.58	0.66	0.97	0.62	0.76
SentinelOne	1.47	0.62	0.97	0.58	0.73
average	2.92	0.76	0.96	0.76	0.84
e2₂					
AVG	0.82	0.92	0.96	0.95	0.95
Avast	0.82	0.92	0.96	0.95	0.95
BitDefender	0.93	0.86	0.95	0.89	0.92
Kaspersky	1.10	0.89	0.94	0.94	0.94
McAfee	0.70	0.89	0.96	0.91	0.93
Microsoft	1.24	0.91	0.94	0.97	0.95
Symantec	0.81	0.93	0.96	0.97	0.96
TrendMicro	0.24	0.72	0.98	0.70	0.82
Malwarebytes	0.64	0.68	0.95	0.68	0.79
Panda	0.17	0.70	0.99	0.68	0.80
Tencent	0.47	0.79	0.97	0.79	0.87
SentinelOne	0.69	0.67	0.95	0.66	0.78
average	0.72	0.82	0.96	0.84	0.89

that if a sample still exhibits zero or negligible VT_{score} , it can be considered as a false positive sample (i.e., non-malicious). To avoid circular references, we adopted a Leave-Twelve-Out strategy, in which the detection outcome of the 12 considered AVs was excluded from the VirusTotal consensus score before applying the detection threshold. We hence considered on average 54 AVs of the VirusTotal platform. We observed that all malware samples (with a single exception) have been recognized as malware by at least 20 AVs. The boxplot in Fig. 5 shows that on average each sample reaches a VT_{score} of about 37.9, i.e., on average 37.9 AVs recognized the sample as malicious. Table 14 reports on a sensitivity analysis conducted to check the variation of the number of false positive samples when increasing threshold values are considered. We can observe the relative stability of the results. Specifically, up to a threshold of 35%, the value of FP remains 1,

Table 11 In-large: Consensus-based metrics - sensitivity analysis

	20%	25%	30%	35%	40%	CV
e2₁						
AVG	0.93	0.94	0.93	0.93	0.92	0.79
Avast	0.93	0.94	0.93	0.93	0.92	0.79
BitDefender	0.82	0.82	0.82	0.82	0.82	0.21
Kaspersky	0.95	0.95	0.94	0.92	0.90	2.41
McAfee	0.90	0.90	0.90	0.90	0.89	0.23
Microsoft	0.93	0.92	0.92	0.90	0.88	2.14
Symantec	0.95	0.95	0.95	0.94	0.92	1.25
TrendMicro	0.73	0.74	0.75	0.77	0.78	2.39
Malwarebytes	0.78	0.78	0.79	0.80	0.81	1.55
Panda	0.63	0.64	0.65	0.66	0.67	2.91
Tencent	0.75	0.75	0.76	0.76	0.76	0.61
SentinelOne	0.72	0.72	0.73	0.73	0.73	1.14
average	0.83	0.84	0.84	0.84	0.83	0.30
e2₂						
AVG	0.96	0.95	0.95	0.95	0.94	0.84
Avast	0.96	0.95	0.95	0.95	0.94	0.84
BitDefender	0.91	0.92	0.92	0.92	0.91	0.39
Kaspersky	0.95	0.95	0.94	0.93	0.92	1.26
McAfee	0.93	0.94	0.93	0.93	0.92	0.84
Microsoft	0.96	0.96	0.95	0.94	0.92	1.60
Symantec	0.97	0.96	0.96	0.96	0.94	0.97
TrendMicro	0.80	0.81	0.82	0.83	0.83	1.41
Malwarebytes	0.80	0.79	0.79	0.78	0.78	0.91
Panda	0.79	0.80	0.80	0.81	0.82	1.30
Tencent	0.86	0.87	0.87	0.87	0.87	0.51
SentinelOne	0.78	0.78	0.78	0.78	0.78	0.29
average	0.89	0.89	0.89	0.89	0.88	0.46

around 40% and 45% is 2. Overall, we can confirm that only a very few samples not discovered after 23 days were actually a non-malicious sample. These observations are consistent with the expert feedback collected during **e1**, confirming the validity of the results discussed results in **e2₂**.

4.2.3 Labeling changes

Several studies (e.g., [11, 25, 27]) highlight the relevance of online services such as VirusTotal, but they also notice that their results can change over time and be inconsistent. To evaluate the consistency of the collected results, we randomly selected more than 4k samples on the full set used in **e2₁** to be re-analyzed in the VirusTotal platform almost more than one year after the original analysis. For such samples, we (re-)collected the VirusTotal verdicts and the classification labels for the 12 considered AVs and applied the McNemar's test to determine the sample classification

Table 12 In-large experiment: labeling changes between first and second period

	Odds ratio	p-value	McNemar's test
Malware sample classification			
malicious vs undetected	1.37	< 0.01	Moderate improve
Malware sample identification per VirusTotal AV			
AVG	0.9	0.25	Stable (no change)
Avast	0.89	0.231	Stable (no change)
BitDefender	11.83	< 0.0001	Substantial improve
Kaspersky	0.77	0.063	Stable (no change)
McAfee	11	< 0.0001	Substantial improve
Microsoft	1.31	0.006	Marginal improve
Symantec	1.16	0.107	Stable (no change)
TrendMicro	0.34	< 0.0001	Substantial decrease
Malwarebytes	3.42	< 0.0001	Moderate improve
Panda	27.15	< 0.0001	Substantial improve
Tencent	390.33	< 0.0001	Substantial improve
SentinelOne	45	< 0.0001	Substantial improve

consistency between the two distinct periods. Specifically, the test evaluates the marginal homogeneity of a 2×2 contingency table by focusing exclusively on the discordant pairs, i.e., samples not undetected initially but detected later (Gained), and those detected initially but missed in the second period (Lost). The Edwards' correction for continuity was also considered to ensure statistical robustness. For each sample and AV, an obtained p -value < 0.05 allows us to reject the null hypothesis of stability in the detection performance, thus confirming a statistically different performance in the two period rather than a random difference. Additionally, to quantify the magnitude of such a difference (i.e., effect size), we computed the paired Odds Ratio ($OR = \text{Gained/Lost}$). An $OR > 1$ indicates a trend toward a better detection capability ($OR > 10$ represents a substantial upgrade), whereas an $OR < 1$ denotes a significant degradation in the detection capability. Conversely, cases with $p \geq 0.05$ characterize stable performance, confirming that any nominal variance was statistically negligible.

Table 12 reports on the results collected with the McNemar's test applied to the threshold-based sample classification and the VirusTotal AV detection in the two periods. Results confirm the expected changes in both labels and detection capability. Specifically, we observed an overall moderate improvement, this was expected since the VirusTotal engines have been updated and their set of malware signatures largely extended. In detail, we observe the relative stability or the marginal improvement of results collected by half of the considered VirusTotal AVs (e.g., AVG, Kaspersky, Symantec). Instead, the engines large performance improve-

Table 13 In-large: Trojan families and detection delay (e_2)

	Trojan samples undetected by at least 30% of AVs for at least 23 days										
	Alien	Anmon	Botx	Deyma	Jaik	Linux	Mirai	Stealer	Zusy	Filerepmalware	
AVG	0	1	0	0	1	0	0	0	0	0	0
Avast	0	1	0	0	1	0	0	0	0	0	0
BitDefender	0	1	1	1	0	4	1	2	0	1	1
Kaspersky	0	0	0	0	0	0	0	0	0	0	0
McAfee	1	1	1	0	0	17	5	0	1	0	0
Microsoft	1	0	0	0	0	0	0	0	2	0	0
Symantec	1	0	0	0	0	6	1	0	0	0	0
TrendMicro	0	1	1	1	1	6	1	2	13	1	1
Malwarebytes	1	1	0	1	1	50	24	2	18	1	1
Panda	1	1	1	0	1	50	23	1	0	1	0
Tencent	0	0	1	1	1	0	0	0	14	0	1
SentinelOne	1	0	0	0	0	1	0	0	0	0	0
(%) VirusTotal AVs not detecting at least one trojan sample for 23 days											
	Alien	Anmon	Botx	Deyma	Jaik	Linux	Mirai	Stealer	Zusy	Filerepmalware	Pwsx
	50	58.33	41.6	33.3	50	58.3	50	33.3	41.6	33.3	33.3

ment (e.g., BitDefender, Panda, Tencent) can be explained by a strong update of the engine database or the adoption of improved detection algorithms and rules. Only for one AV we observed a decrease of the performance. Since we empirically exclude that this was due to more aggressive tuning of heuristics aimed at mitigating false positives, alternative motivations can be related changes in the signature lifecycle management (e.g., optimization of the signature storage and use) or AV architecture (e.g., moving towards the integration of a cloud-based database). One alternative option can concern the AV setup used in the VirusTotal platform. Overall, these observations confirm that VirusTotal labels change as expected, and that the adoption of a selected VirusTotal AVs and a threshold-based policy can effectively mitigate the impact of such changes.

Table 14 In-large: GE23 samples' VT_{score} TH-based sensitivity

TH	# AVs (out of on avg 54)	# FP (out of 79)	Malicious samples (%)
e_2			
15%	8	1	98.73
20%	10	1	98.73
25%	13	1	98.73
30%	16	1	98.73
35%	19	1	98.73
40%	21	2	97.46
45%	24	2	97.46

4.2.4 Dataset saturation and robustness verification

To empirically check the robustness of our evaluation and verify whether our malware dataset size is adequately to yield stable VirusTotal AV consensus-based performance metrics, we performed a dataset saturation analysis applying a bootstrap resampling procedure to e_2 . Specifically, we evaluated the (Leave-one-out) consensus F_1 -score across five (sub)datasets of increased size (20%, 40%, 60%, 80%, and 100% of the total dataset size). At each increment $p \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, we performed $B = 100$ bootstrap iterations. In each iteration, a subset of size $N_p = \lfloor p \cdot N \rfloor$ was drawn with replacement from the full dataset (N), and the (Leave-one-out) F_1 -score was computed across the considered AVs.

Figure 6 shows that the mean F_1 -score remains almost stable across all sample sizes, displaying a plateau around 0.841 with very limited fluctuations ($\Delta F_1 < 0.0015$). Furthermore, the figure also shows that the standard deviation across iterations continuously decreases as the dataset size increases, reflecting a narrowing confidence band. The figure confirms empirical stochastic convergence, thus proving that expanding the dataset size further yields diminishing returns and that the current dataset size is saturated enough. Hence, we can overall state that the observed results, including degradation and effective trends, reflect systemic capabilities of the respective detection AVs rather than random fluctuation dependent on the selected malware samples.

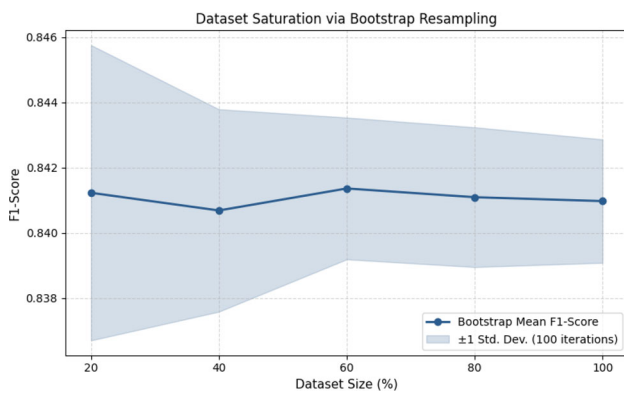


Fig. 6 In-large experiment: bootstrapping resampling dataset saturation (e2₁)

4.2.5 Dataset cross-validation

To cross-validate our dataset, we leveraged HybridAnalysis [28], an enterprise platform built upon the CrowdStrike's Falcon Sandbox technology [29]. Given a file to be analyzed, HybridAnalysis mainly relays on a mix of three dynamic analysis, rather than relying exclusively on static signatures and traditional AVs like VirusTotal. Specifically, by using platforms such as CrowdStrike Falcon and MetaDefender [30] it conducts signature and heuristics-based analysis, runtime behavioral analysis in a sandbox, MITRE ATT&CK indicator mappings, and threat intelligence reputation analysis (i.e., cloud-based searches for suspicious activities related to the file). The results is a four-label classification of the file: “malicious”, if the file is identified as malicious; “suspicious”, if it shows limited unexpected behaviors; “no specific threat”, if it is identified as non-malicious; and “unknown” if the platform is not able to collect enough evidence for classifying the file.

We analyzed the same subset of our sample dataset used in Sect. 4.2.3 with HybridAnalysis, and only three samples are not present in the platform, for the other ones we collected the classification outcome. We obtained that 96.5% of samples have been classified equally by VirusTotal and HybridAnalysis. We hence computed the F1-score among the two classifications and this reveals a near-total overlap in predictions (98.2%). Therefore, we can conclude that the two platforms provided a classification output with a strong overlap. Additionally, to determine whether the remaining discrepancies (about 2% of samples) were due to chance or systematic differences, we applied the McNemar's test. The test revealed a statistically significant difference ($p < 0.05$), thus we can reject the null hypothesis of equal classification and observe that the disagreement between the two classification, even if quantitatively limited, follows a non-random distribution. We argue that this result is mainly due to the used unbalanced datasets (strongly in favor of

malicious samples), and by the different types of analysis applied but the two platforms. VirusTotal tends to be more conservative in the classification than HybridAnalysis coherently with the different detection paradigm adopted by the platforms. VirusTotal aggregates more traditional signature-based static analysis, which are sensitive to historical known patterns. Conversely, HybridAnalysis relies on runtime indicators within a sandbox, thus it is more sensitive with respect to malware behaviors, leading to a high number of “unknown” classification (2.9%). These statistical findings cross-validated our dataset and demonstrate that static signature harvesting and dynamic behavioral profiling form a complementary, non-redundant pipeline necessary for robust threat verification.

5 Further analysis on defense evasion techniques

To better understand the collected results, we conducted an in-depth investigation focused on the identification of the used defense evasion techniques, i.e., techniques adopted by malware to avoid the detection from the victim's security software mechanisms. MITRE [31] characterizes the known defense evasion techniques, e.g., uninstalling and disabling security software (e.g., *Disable or Modify Tools*), obfuscating/ encrypting data and scripts (e.g., *Obfuscated Files or Information*). Additionally, MITRE [32] also lists the common objectives that drive attacker while conducting an attacks (named *tactics*).

In our analysis, we were interested in answering the following research questions:

– **RQ4 (Used defense evasion techniques).** *What defense evasion techniques are used by malware?* To answer RQ4, we identified the defense evasion techniques that are frequently adopted by malware samples.

– **RQ5 (Defense evasion techniques and detection delay).** *Are there any defense evasion techniques that, when used by malware or incorporated into some malware families, reduce detection more than others?* To answer RQ5, we correlated the defense evasion techniques adopted by malware samples and detection delay.

– **RQ6 (Defense evasion techniques and malicious tactics).** *Are there any other malicious tactics applied by malware samples in conjunction with the defense evasion techniques?* To answer RQ6, we correlated the defense evasion techniques adopted by each malware sample with other malicious tactics incorporated into the sample for achieving their malicious objectives (e.g., data leak, software execution, privilege escalation, system's corruption).

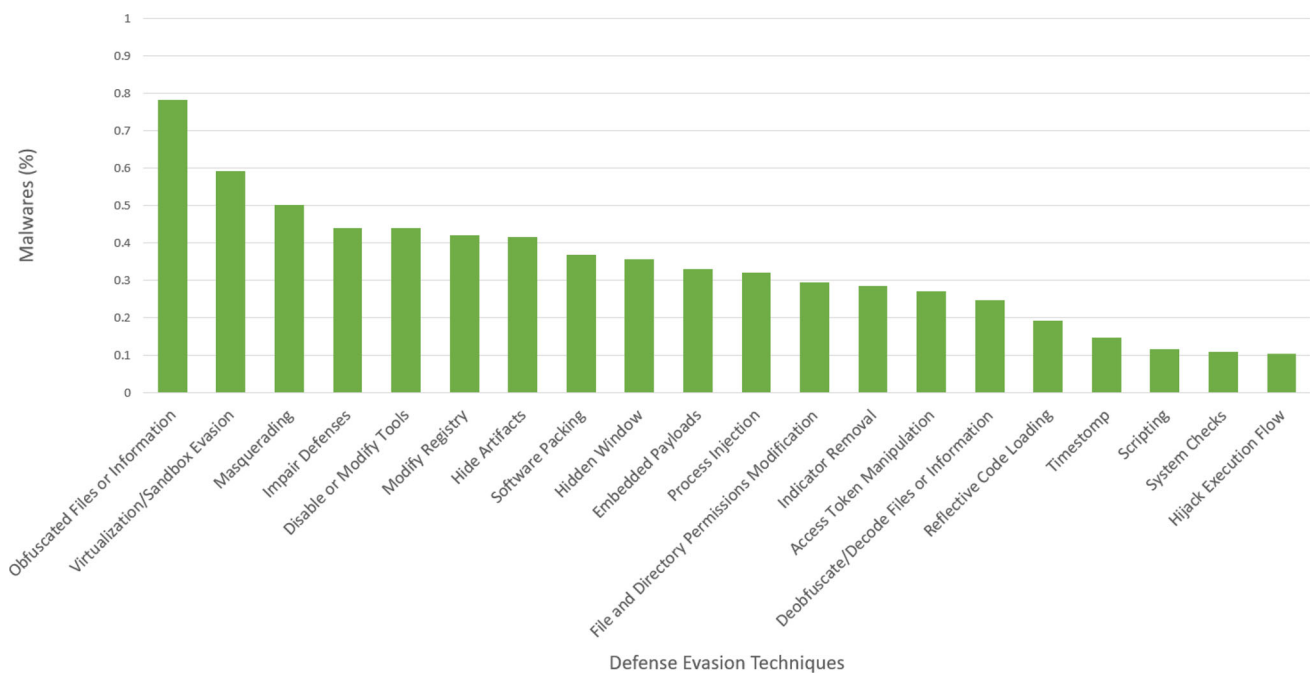


Fig. 7 Defense evasion techniques on malware samples (frequency higher or equal to 10%)

Table 15 In-large: defense evasion techniques on malware samples (frequency higher or equal to 10%)

Defense evasion technique	Malware %	Proportion %	CI 95%	Cumulative pareto %
Obfuscated files or information	0.78	59.98	58.87–61.07	10.42
Virtualization/sandbox evasion	0.59	45.28	44.17–46.4	18.29
Masquerading	0.50	38.38	37.3–39.48	24.96
Impair defenses	0.44	33.77	32.71–34.84	30.83
Disable or modify tools	0.44	33.73	32.67–34.8	36.69
Modify registry	0.42	32.26	31.22–33.32	42.3
Hide artifacts	0.42	31.9	30.87–32.96	47.84
Software packing	0.37	28.18	27.18–29.2	52.74
Hidden window	0.36	27.25	26.26–28.26	57.48
Embedded payloads	0.33	25.28	24.32–26.27	61.87
Process injection	0.32	24.6	23.64–25.58	66.14
File and directory permissions modification	0.30	22.59	21.66–23.54	70.07
Indicator removal	0.29	21.87	20.95–22.81	73.87
Access token manipulation	0.27	20.78	19.88–21.7	77.48
Deobfuscate/decode files or information	0.25	18.9	18.04–19.8	80.77
Reflective code loading	0.19	14.65	13.88–15.46	83.31
Timestomp	0.15	11.2	10.51–11.93	85.26
Scripting	0.12	8.84	8.22–9.5	86.8
System checks	0.11	8.3	7.71–8.94	88.24
Hijack execution flow	0.10	7.96	7.38–8.59	89.62

5.1 RQ4: Used defense evasion techniques

By analyzing the malware samples from the In-large experiment **e2₁**, we identified 72 different defense evasion techniques used by such samples. To assess the distribution of such techniques across the malware samples, we conducted a three-phase analysis. First, we performed a χ^2 (Chi-Square) Goodness-of-Fit test to evaluate how these techniques are distributed across the samples; a p -value less than 0.05 allows us to reject the null hypothesis, which states that the defense evasion techniques are used uniformly by the malware samples. Second, for each evasion technique, we quantified its prevalence within the dataset in terms of proportion alongside its 95% confidence interval (CI). Third, we conducted a Pareto analysis with the rule “80/20” to identify the most critical defense evasion techniques. The result of the χ^2 test ($p < 0.05$) confirms a statistically significant difference in the distribution of defense evasion techniques across the malware samples. Figure 7 plots the distribution of the most frequent evasion techniques, and Table 15 reports the computed numbers for them. *Obfuscated Files or Information* and *Masquerading* (both under the *Obfuscating* category), and *Virtualization/Sandbox Evasion*, *Impair Defenses*, *Disable or Modify Tools* (category *Attacking*) are very frequent. The Pareto analysis reveals that these techniques are integrated into more than 35% of considered malware samples. Overall, we observe a high variability among the defense evasion techniques incorporated by the samples, as only 20 out of 72 exhibit a prevalence higher than 10%. The Pareto analysis reports also that a set of 14 defense evasion techniques (the top ones listed in Table 15, i.e., approximately 20% of the identified techniques) is responsible for over 80% of the total cumulative occurrences across the malware dataset.

The identified defense evasion techniques can be grouped into five categories (see Sect. 2): *Obfuscating*, *Concealmenting*, *Attacking*, *Injecting*, and *Packing*. Figure 8 shows the distribution of the identified defense evasion techniques per each category. Additionally, we conducted an association rules analysis by applying the Apriori algorithm [33]. The Apriori algorithm can mine frequent patterns of associated items in a dataset, such as sets of multiple co-occurrent items. Each set inferred has a *support* that indicates how frequently the item set appears in the dataset. We used the Apriori algorithm to identify the co-occurrent sets of defense evasion techniques. Table 16 reports the identified co-occurrent defense evasion techniques (with minimum support to 30%), together with their related category. We can note that *Obfuscated Files or Information* - category *Obfuscating*, is almost part of all inferred rules, and that it is frequently used in combination with techniques of the same category (e.g., *Masquerading* and *Hide Artifact*) and of the categories *Attacking* (e.g., *Impair Defense*), *Concealmenting* (e.g., *Modify Registry*) and *Packing* (i.e., *Software Packing*). Overall, most

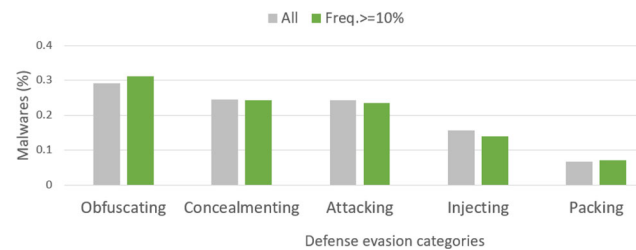


Fig. 8 Distribution of the defense evasion techniques on the malware samples

of the co-occurrent techniques are related to *Obfuscating*, *Attacking*, and *Packing*, indicating that such categories are often used in combination. This means that several malware samples adopt, for instance, polymorphic software or encryption mechanisms, packing strategies, as well as strategies to identify and neutralize the AVs’ defense mechanisms.

5.2 RQ5: Defense evasion techniques and detection delay

To answer RQ5, we analyzed the samples from the In-large experiment **e2₂**. To assess the distribution of defense evasion techniques across the malware samples while differentiating between those discovered within and beyond 23 days (“L23” and “GE23”, respectively), we performed a χ^2 (Chi-Square) test of independence. A p -value < 0.05 allows us to reject the null hypothesis, which states that the use of defense evasion techniques is independent of the detection time (i.e., evasion techniques are distributed identically between L23 and GE23 samples). Then, a post-hoc analysis based on adjusted standardized residuals with a Bonferroni correction was performed to identify which specific evasion techniques are significantly associated with malware samples detected beyond 23 days. By applying the χ^2 test of independence, we obtained a p -value less than 0.05, thereby confirming that defense evasion techniques are not distributed equally between malware samples discovered within and beyond 23 days. Figure 9 shows the distribution of defense evasion techniques on malware samples, by differentiating between samples discovered within and beyond 23 days. Figure 10 reports the malware undetection days per defense evasion technique. While there are techniques (e.g., *Bootkit* and *Compile After Delivery*) that are only incorporated by samples detected in less than 23 days, there are no techniques that are only incorporated into samples detected late (i.e., after 23 days). However, Fig. 11 identifies defense evasion techniques that are more frequently incorporated into samples within GE23, than in L23 (e.g., *Bypass User Account Control*, *Obfuscated Files or Information*, *Scripting*, *Install Utils*). For instance, *Bypass User Account Control* is incorporated into 7.5% and 4.7% of samples of GE23 and L23 respectively, and

Table 16 Examples of inferred association rules related to of defense evasion techniques (min. support of 30%)

Id	Defense Evasion Set of techniques	Corresponding categories
r1	Process injection	Injecting
r2	obfuscated files or information, embedded payloads	Obfuscating, injecting
r3	Obfuscated files or information, modify registry	Obfuscating, concealmenting
r4	Obfuscated files or information, hide artifacts, hidden window	Obfuscating, obfuscating, concealmenting
r5	Obfuscated files or information, masquerading, software packing	Obfuscating, obfuscating, packing
r6	Obfuscated files or information, virtualization/sandbox evasion, software packing	Obfuscating, attacking, packing
r7	Obfuscated files or information, impair defenses, disable or modify tools, software packing	Obfuscating, attacking, attacking, packing
r8	Obfuscated files or information, masquerading, virtualization/sandbox evasion, impair defenses, disable or modify tools	Obfuscating, obfuscating, attacking, attacking, attacking

Obfuscated Files or Information into 50.6% and 44%. The conducted post-hoc analysis confirms that *Obfuscated Files or Information*, *Scripting*, and *PortableExecutableInjection* are the unique defense evasion techniques statistically more frequently integrated into malware samples within GE23.

Figure 12 shows a 2D clustered heatmap built from the matrix that identifies the defense evasion techniques (rows) related to the *ten* most frequent malware families (columns) whose samples have been mainly detected after 23 days. Each row of the initial matrix was composed of a sequence of 1 or 0, where 1 indicates that the samples in the malware family (column) incorporate the defense evasion technique (row), 0 otherwise. Data pre-processing, PCA, and heatmap clustering (based on correlation distance and average linkage) were carried out in ClustVis [34] by centering rows and applying unit variance scaling (Z-scores); relative variations were mapped onto a color gradient ranging from blue for below-average values to red for above-average values. The heatmap captures and visualizes the correlation degree between defense evasion techniques and those malware families often detected after 23 days, by using such a color scale in each cell of the map. While red indicates the existence of correlation, blue indicates the absence. The color intensity of each cell represents the correlation degree. The heatmap documents a sparse set of strong defense evasion techniques' incorporation. We see that *Pswx* and *Zusy* are malware families that incorporated the largest number of defense evasion techniques. Conversely, *Gcleaner*, *Msil*, and *Jaik* incorporated only one or two defense evasion techniques each. A limited number of defense evasion techniques (32%) is used into several malware families. For instance, *Software Packing*, *Masquerading*, and *System Checks* are techniques incorporated by at least half of the malware families. However, most of the defense evasion techniques are incorporated

into specific malware families. Indeed, 41% of the defense evasion techniques characterize only one malware family. For instance, *File and Directory Permissions Modification*, *File Deletion*, and *Disable or Modify System Firewall* are incorporated only into Mirai.

5.3 RQ6: Defense evasion techniques and malicious tactics

To answer RQ6, we analyzed the malware samples from the In-large experiment **e2₁** to identify the tactics for which the defense evasion techniques are used.

Table 17 reports a brief description of the considered tactics, as provided by MITRE. Additionally, the table reports the average number of tactics' approaches and methods that have been identified in our samples. Figure 13 presents the percentage of malware samples per tactic. For instance, Collection [35] groups approaches (e.g., Adversary-in-the-Middle, DHCP Spoofing, Browser Session Hijacking) typically used to gather data of interest for the attacker's goal. Collection's approaches are used by more than 72% of the malware samples. Differently, Impact [36] groups approaches (e.g., Data Encrypted for Impact, Disk Wipe, Network Denial of Service) typically used to manipulate, interrupt, or destroy the target systems and data. Impact approaches are used by (about) 30% of the malware samples. Overall, we can see that our malware samples are mainly used for Defense Evasion [31], Discovery [37], and Execution [38].

Figure 14 shows a 2D clustered heatmap built from the matrix that captures the association between Top-10 defense evasion techniques and tactics. Data pre-processing, PCA, and heatmap clustering (based on correlation distance and average linkage) were carried out in ClustVis [34] by cen-

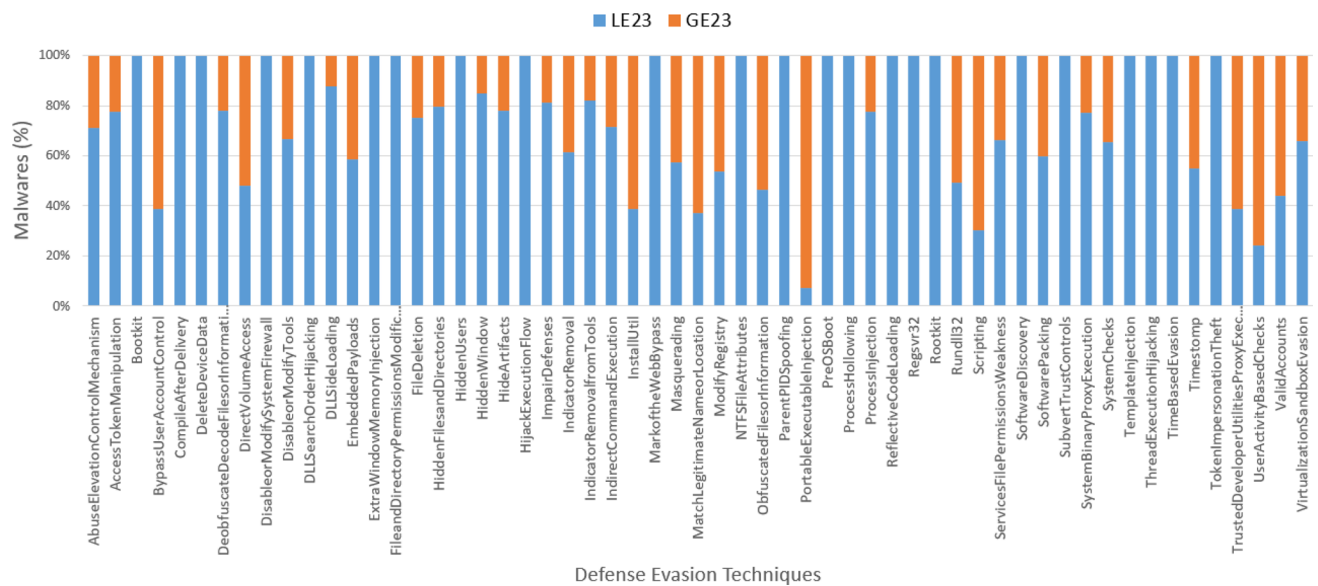


Fig. 9 Defense evasion techniques and detection delay (undetected days)

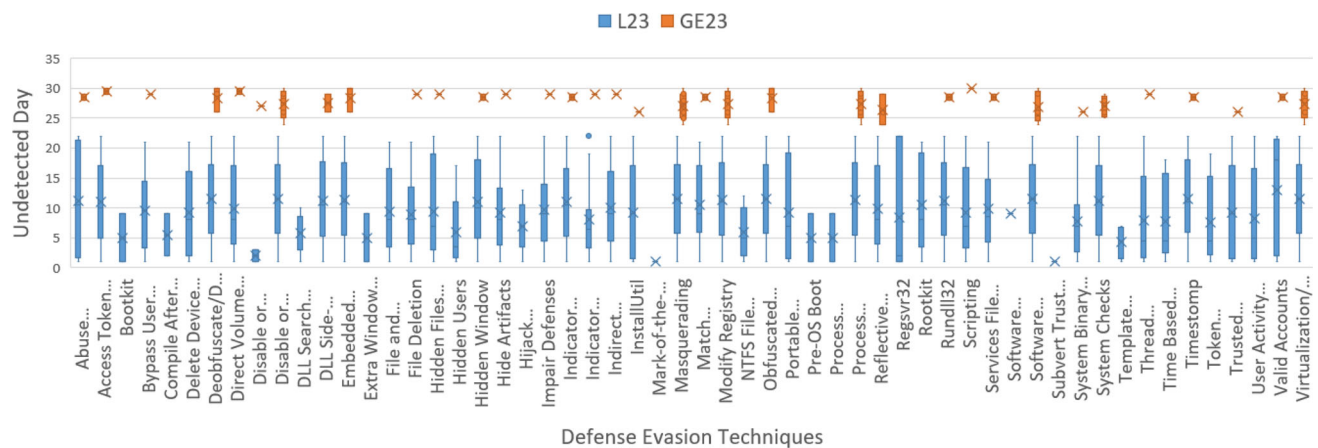


Fig. 10 Defense evasion techniques more frequent on malware in GE23

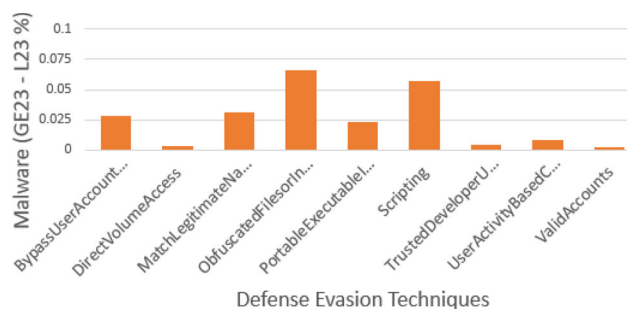


Fig. 11 Defense evasion techniques (Y-axis) frequent in malware families of the set GE23 (X-axis)

tering rows and applying unit variance scaling (Z-scores); relative variations were mapped onto a color gradient ranging from blue for below-average values to red for above-average values. Dark-red indicates that the technique is very fre-

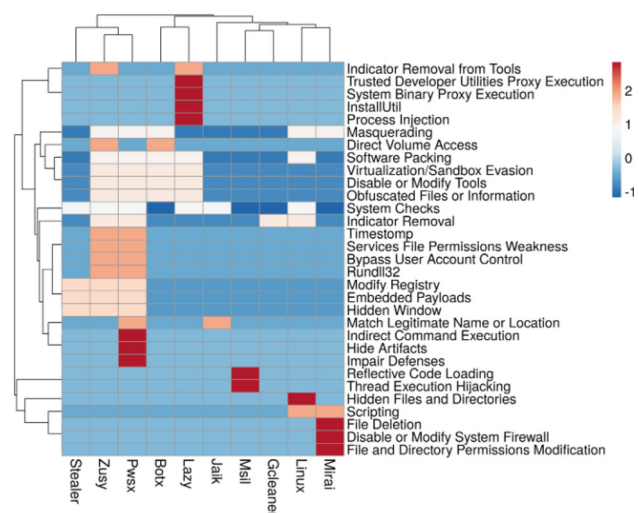


Fig. 12 Malware samples per tactic

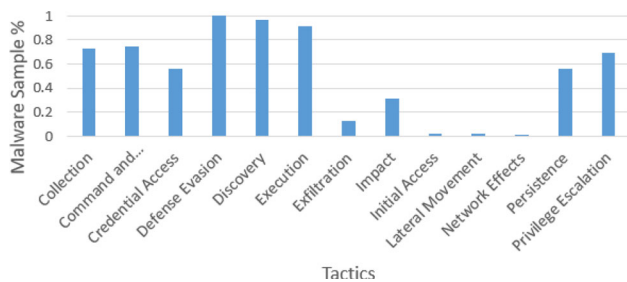


Fig. 13 Top-10 Defense evasion techniques (Y-axis) and tactics (X-axis)

quently used for the specific tactic, while dark-blue cell indicates that the technique is not used for the specific tactic.

Overall, the heatmap shows that a limited number of defense evasion techniques used for a large set of tactics. Generally, we can identify three clusters of malicious tactics: (1) the first one (i.e., Collection, Command and Control, Credential Access, and Network Effects) focuses on the collection of data related to the attack target environment, and uses defense evasion techniques of the category Obfuscating (strong correlation) and Attacking (less strong correlation), such as *Masquerading* and *Impair Defenses*; (2) the second one (i.e., Exfiltration, Execution, Discovery, and Impact) focuses on the manipulation of the target environment and uses defense evasion techniques of the category Injecting (strong correlation) and Concealmenting (less strong correlation), such as *Embedded Payloads* and *Access Token Manipulation*; (3) the third one (i.e., Initial Access Lateral Movement, Persistence, and Privilege Escalation) focuses on the movement inside the target environment and uses defense evasion techniques of the category Concealmenting (strong correlation) and Injecting (less strong correlation), such as *Indicator Removal* and *Process Injection*.

6 Further analysis on crowd-sourced detection rules

Malware can generate anomalous network traffic and activity on the attacked environment. Hence, we conducted an in-depth investigation focused on the malware violation of intrusion detection rules raised in the VirusTotal's sandbox executions. Specifically, we in the results obtained from the VirusTotal platform for each sample, we checked whether our malware samples violated crowd-sourced detection (cIDs) rules, such as the ones provided by: Snort [39], Proofpoint [40], and Suricata [41]. These rules are defined by the community and among the most diffused to detect attacks and malicious network traffic and activity. In particular, we were interested in answering the following research questions:

– **RQ7 (Violated cIDs rules).** *What cIDs rules are violated by malware samples?* To answer RQ7, we identified the cIDs rules that are frequently triggered in the VirusTotal's sandbox by malware samples.

– **RQ8 (Defense evasion techniques, cIDs rules, and detection delay).** *Are there any defense evasion techniques that, when used by malware or incorporated into malware families that violated cIDs rules, contribute to increase the detection delay?* To answer RQ8, we compared the defense evasion techniques incorporated into malware samples, by considering whether the malware violated cIDs rules, as well as the AVs' detection delay.

6.1 RQ7: Violated cIDs rules

By checking the reports of the malware samples of the In-large experiment $e2_1$, we identified 3507 samples (46%) that triggered at least one cIDs rule in the VirusTotal sandbox, i.e., 54% of the executions in the VirusTotal sandbox did not trigger cIDs rules in the considered platform. This latter number can be due to several reasons, for instance, it can be due to: (i) the detection of the VirusTotal sandbox by the malicious samples by means of specific evasion techniques; (ii) samples that do not generate network traffic (either by design or because the associated Command & Control (C2) infrastructure was inactive/offline during the analysis); (iii) execution failures due to unsupported environments (e.g., missing dependencies or platform mismatches); and (iv) insufficient execution time adopted for the analysis. Since raw execution traces (e.g., network PCAPs) are unavailable for a retrospective analysis on each sample, we cannot unequivocally isolate the exact cause for the non-triggering samples. We therefore leave a detailed trace-level investigation for future work. In our analysis, we observed that overall, 745 rules from 8 different cIDs have been violated. Table 18 lists the used rules, the number of rules violated, and the number of malware samples that violated such rules. Most malware samples violated cIDs rules from Snort and Proofpoint, with a large overlap (67%) between these two systems.

The violated rules are related to 37 rule categories. To assess how the malware samples violated the cIDs rules across different categories, we conducted a three-phase analysis. First, we computed a χ^2 (Chi-Square) Goodness-of-Fit test to evaluate whether these rule categories are uniformly violated by the samples; a p -value less than 0.05 allows us to reject this hypothesis. Second, for each cIDs rule category, we quantified the sample violations in terms of proportion alongside the 95% Wilson confidence intervals. Third, we conducted a Pareto analysis with the rule “80/20” to identify the most effective rule categories (i.e., those that were most frequently violated, thus that frequently detected malware samples). Table 19 reports on a short description of the rule categories that have been violated by at least 10%

Table 17 Considered tactics

Tactic	Nr. approaches Average	Std.Dev.
Command and Control: communicate with compromised systems to control them	2.6	2.1
Collection: gather data for the attacker's goal	3	2.8
Credential Access: steal account names and passwords on the target system	1.4	1.8
Defense evasion: avoid being detected	9.2	6
Discovery: figure out the target environment	9.4	5.6
Execution: run malicious code in the target system	3.2	2.2
Exfiltration: steal data	0.1	0.3
Impact: manipulate, interrupt, or destroy the target systems and data	0.6	1
Initial Access: get into the target network	0.03	0.2
Lateral Movement: move in the target environment	0.02	0.1
Network Effects: intercept or manipulate network traffic to or from a device	0.004	0.1
Persistence: maintain the attacker's foothold	1.9	2.3
Privilege Escalation: gain higher-level permissions in the target system	2.8	3.1

of the malware samples, and Table 20 reports on the computed metrics for such rule categories. The result of the χ^2 test (p -value < 0.05) confirms a statistically significant difference in how the cIDs rule categories are violated by the malware samples. The relatively high violation rates obtained by few specific cIDs rule categories (including: “A Network Trojan was detected”, “Misc-activity”, “Potentially Bad Traffic”, and “Trojan-activity”), with their narrow 95% confidence intervals, confirm that these detection rates are stable across the dataset. Furthermore, the Pareto analysis shows that these four categories account for more than half of the total cumulative violations (i.e., they are violated by 53% of malware samples).

The 37 rule categories violated can be clustered into 5 groups, depending on their detection target: *Malware attacks*, *Reconnaissance*, *Policy Abuse* and *malicious behaviors*, *Attack against web apps*, and *Attacks to protocols*. Table 21 reports a short description of such groups and the distribution of categories, rules, and malware samples per group. Almost all violated rules (90%) resort to *Malware attacks* and *Policy Abuse* and *malicious behaviors*.

Figure 15 plots the average number of cIDs rules violated per malware family, and the average percentage of violated rules for each malware family. About half of the malware families violated 2 or 3 rules, on average, with a non-trivial 9% of families whose samples violated up to 6 rules. Among the malware families with a high number of violated rules we can find Botx, Linux, and Lazy which violated 5 rules each.

Table 18 cIDs source, rules, and percentage of violated rules

Source	Rules	Violated rules (%)
Snort registered user ruleset	194	0.83
Proofpoint Emerging Threats Open	476	0.78
Suricata	44	0.16
Travis Green: Threat hunting rules	7	0.033
Abuse.ch Suricata-JA3-Fingerprint Ruleset	16	0.030
Abuse.ch SSL Blocklist	4	0.006
Positive Technologies: PT Open Ruleset	1	0.003
Abuse.ch Botnet C2 IP Ruleset	3	0.0008

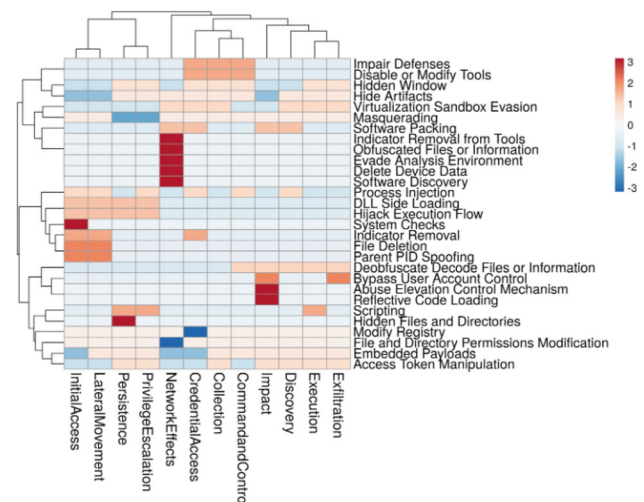
**Fig. 14** cIDs rules violated on average for malware families (top), percentage of malware families for number of violated rules (down)

Table 19 Most frequently violated cIDs rule categories

cIDs rule category: description
A Network Trojan was detected: rules to reveal activities related to Trojan Horse at network level (e.g., communication with C2 servers)
Misc-activity: rules to reveal anomalous activities such as: unusual network traffic, malicious activity relates to known vulnerabilities
Potentially Bad Traffic: rules to reveal potentially malicious network traffic, not related to specific malware types
Trojan-activity: rules to reveal activities related to Trojan Horses
Policy-violation: rules to reveal violations of company security policies and configurations (e.g., use of not authorized protocols)
Successful-recon-limited: rules to reveal successful tentative of reconnaissance (e.g., port scanning)
Malware Command and Control Activity Detected: rules to reveal C2 communication associated with Trojans, Botnet, and Ransomware
Generic Protocol Command Decode: rules to reveal malicious activity at protocol-implementation level, e.g. structure and syntax of protocol's commands
Attempted-recon: rules to reveal tentative of reconnaissance (e.g., port scanning)

Table 20 Most frequently violated cIDs rules - metrics

cIDs rule category	Malware %	Proportion %	CI 95%	Cumulative Pareto %
A Network Trojan was detected	0.48	22.26	21.34–23.21	15.05
Misc-activity	0.45	20.62	19.73–21.54	28.99
Potentially Bad Traffic	0.40	18.19	17.34–19.08	41.29
Trojan-activity	0.38	17.57	16.73–18.44	53.16
Policy-violation	0.25	11.68	10.97–12.42	61.05
Successful-recon-limited	0.21	9.55	8.91–10.23	74.01
Malware Command and Control Activity Detected	0.18	8.07	7.48–8.7	79.46
Generic Protocol Command Decode	0.15	6.78	6.24–7.37	84.05
Attempted-recon	0.11	5.14	4.67–5.66	87.52

Table 21 Groups of cIDs rule categories

Group	cIDs rule categories (%)	cIDs rules (%)	Malware samples (%)
Malware attacks: detecting malware, such as: trojans, botnets, ransoms	0.31	0.42	0.34
Reconnaissance: collecting information, e.g., port scanning	0.09	0.03	0.1
Policy Abuse and malicious behaviors: detecting policy violations and anomalous traffic	0.34	0.47	0.53
Attack against web apps: detecting attacks against web application (e.g., SQL injections)	0.09	0.01	0.003
Attacks to protocols: detecting malicious activity towards abuse of protocols	0.15	0.05	0.009

6.2 RQ8: defense evasion techniques, cIDs rules, and detection delay

To answer RQ8, we analyzed the malware samples from the In-large experiment **e2₂**. We aim to identify the defense evasion techniques incorporated into samples, and distinguish between those samples that violated or complied with cIDs rules, while categorizing them based on whether they were detected within (**L23**) or beyond 23 days (**GE23**). Although

the 54% subset of samples that did not trigger cIDs rules in the VirusTotal platform (see the previous section) represents an aggregate of both stealthy network evasion and sandbox execution artifacts (e.g., offline C2s or execution timeouts), evaluating AV detection on this set provides potentially insights into defense-in-depth efficacy. Existing AVs rely mainly on signatures, file heuristics, and local behavior analysis, rather than active network traffic analysis, so their performance on these network-silent samples reflects their

capability to compensate for unobserved or absent network-level alerts.

We hence conducted three types of analysis to identify the existing relationships between:

- *analysis-1*: violation of cIDS rules and integration of specific defense evasion techniques into malware samples;
- *analysis-2*: integration of defense evasion techniques into malware samples and their detection delay, when cIDS rules are violated;
- *analysis-3*: integration of defense evasion techniques into malware samples and their detection delay, when cIDS rules are not violated, for any reasons.

In all the three analysis, we first performed a χ^2 (Chi-Square) test of independence. A p -value < 0.05 allows us to reject the null hypothesis, which states that the elements of the relationship under investigation are independent. For instance, in *analysis-1*, the null hypothesis states that the integration of defense evasion techniques is independent from the cIDS rule violation. Then, if a p -value less than 0.05 is obtained, we applied a post-hoc analysis using adjusted standardized residuals with the Bonferroni correction to identify specific trends. For instance, in *analysis-1*, we aim to distinguish which defense evasion techniques are significantly integrated into malware samples that either violated or bypass cIDS rules.

(*analysis-1*) Figure 16 shows the distribution of malware samples with respect to the violation of cIDS rules, depending on the defense evasion techniques they incorporate. Although there are few techniques (i.e., *Bootkit*, *Disable or Modify System Firewall*, *Software Discovery*, and *Pre-OS Boot*) incorporated only into malware samples that did not violate any cIDS rules, there is a prevalence of techniques for which most malware samples that incorporated them violated cIDS rules.

By applying the χ^2 test of independence, we obtained a p -value less than 0.05, confirming a statistically significant association between the integration of defense evasion techniques and cIDS rule violations. Table 22 (top) reports on the results of the post-hoc analysis. There are techniques that exhibit a significant correlation with the cIDS rule violation. For instance, we observe that while *Obfuscated Files or Information*, *Indicator Removal from Tools*, and *Valid Accounts* are significantly associated with samples that have frequently evaded cIDS rules. Conversely, only *Bypass User Account Control* demonstrate a strong positive correlation with cIDS violation, indicating that it is relatively easy to detect for cIDS rules.

(*analysis-2*) Overall, in *textbfe22*, we have a prevalence of defense evasion techniques used by samples that violated cIDS rules. Furthermore, 90.8% of malware samples that violate cIDS rules are part of L23 (detected in less than 23

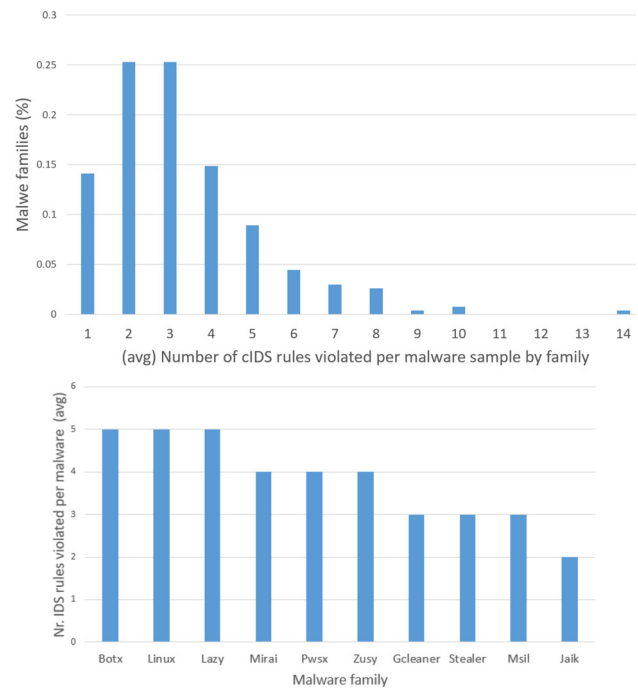


Fig. 15 Distribution of the defense evasion techniques on the malware samples, triggering and not triggering cIDS rules

days), while only the remaining 9.2% of GE23. In terms of malware families, only a limited percentage (5.9%) of them contain samples that are mainly detected in more than 23 days, in particular: *Atraps*, *Dinvoke*, *Gen4*, *Malformed*, *Lotok*, and *Obfsstrm*. Table 23 reports a brief description for each one. Most of the other families contain malware samples that violated cIDS rules and are mainly part of L23. The unique two exceptions, are *Mirai* and *Obfsobjdat* (obfuscated element OLE data in Microsoft Office documents), in which the distribution of samples among L23 and GE23 is well-balanced. Concerning the χ^2 test of independence, we obtained a p -value less than 0.05, confirming a statistically significant association between the integration of defense evasion techniques and detection delay, when cID rules are violated. Table 22 (middle) reports on the results of the post-hoc analysis. There exist defense techniques that exhibit a significant correlation with high detection delay (meaning that samples that integrate such techniques are often part of GE23), even when cIDS rules are violated. For instance, samples that incorporate *Bypass User Account Control* and *Rndll32* tend to violate cIDS rules and be part of GE23 (e.g., detected after 23 days).

(*analysis-3*) By focusing on the set of malware samples that did not violated any cIDS rules (for any reasons), the p -value of the χ^2 test of independence is less than 0.05 and confirms a statistically significant association between the integration of defense evasion techniques and such a non-violation of cIDS rules. Table 22 (bottom) reports on the

results of the post-hoc analysis by listing the defense evasion techniques that exhibited a significant correlation with the non-violation of the cIDs rules, i.e., samples that integrate such techniques tend to survive more than 23 days. For instance, we observe that while *Scripting* and *Valid Accounts* are significantly associated with samples that did not violate any cIDS rules.

7 Discussion and lessons learned

Based on the collected results, we answer the research questions and identify future research directions.

- **RQ1 (AVs effectiveness).** We observed that the AVs in VirusTotal show highly effectiveness but they can leave undetected a non-trivial percentage of malware samples, on average up to 26.8%, most of which are considered critical. A statistically relevant variability of the effectiveness has been observed. However, we identified groups of AVs that exhibit comparable behavior, in terms of sample classification outcome and alarm triggering. This can indicate that such groups can potentially adopt comparable strategies, e.g., alert triggering strategies, and false negative sensitivity (*Lesson 1*). Hence, when combination of multiple AVs are used in a device, the selection of the engines to be adopted has to be carefully conducted.
- **RQ2 (Non-detected malwares).** We observed that a statistically relevant variability on the malware detection effectiveness, depending on AV and malware target family. We confirm that groups of VirusTotal AVs exhibit comparable relevant performance and difficulties with respect to the detection of specific malware families. There exist Trojan families that are frequently not detected by the VirusTotal AVs, e.g., Cryxos, Msil, and Linux (Mirai/Linux). Often, they use technologies related to the MS-Windows environment (i.e., executable/DLL files and MS-Office documents such as Docx) and Linux executable/Elf files. For instance, technologies such as **Docx** files are not malicious as they act maliciously, but they are malicious because of the information that they carry, i.e., the file is not actually going to directly harm the victim, but it is used to trigger something malicious. Hence, the detection capability can be potentially improved if preliminary analysis against packing are conducted (*Lesson 2*).
- **RQ3 (Days of non-detection).** Most malware samples have been detected by several VirusTotal AVs in less than 10 days. However, some samples persisted up to more than 23 days, thus being potentially problematic. For instance, several samples of Linux (Mirai/Linux), Mirai, and Zusy, remained undetected for almost the full period of the experiment. Hence, even if we cannot confirm if this was due to technical issues or detection prioritization strategies. However, we need to be conscious that potentially critical malware samples can persist for several days, i.e., from 10 up to 23 days (*Lesson 3*).
- **RQ4 (Used defense evasion techniques).** We observed a statistically significant difference in the distribution of defense evasion techniques across the malware samples analyzed by the VirusTotal AVs (*Lesson 4*). A large variability of defense evasion techniques is incorporated into malware samples. However, three of such techniques, i.e., *Obfuscated Files or Information*, *Masquerading*, and *Virtualization/Sandbox Evasion*, are incorporated into half of the samples. By mining co-occurrent sets of defense evasion techniques, we see that Obfuscating techniques are often used in conjunction with techniques of category Attacking and Concealment. Therefore, we noticed that malware hide itself also by observing and altering the target environment.
- **RQ5 (Defense evasion techniques and detection delay).** We statistically confirm that defense evasion techniques are not distributed equally across malware samples discovered within and beyond 23 days in the VirusTotal AVs. There are defense evasion techniques (e.g., *Obfuscated Files or Information*, *Scripting*, and *Portable-ExecutableInjection*) that are mainly incorporated into samples that survived for several days, thus deserving more attention by the community (*Lesson 5*). Overall, we did not identify a clear winner strategy in how the malware incorporate defense evasion techniques. In fact, most malware families (e.g., *Gcleaner*, *Msil*, and *Jaik*) incorporate only few and specific defense evasion techniques, depending on their target attack. However, a limited number of families (e.g., *Pswx* and *Zusy*) incorporate a large set of defense evasion techniques. A more specific investigation can discover underlying incorporation strategies (*Lesson 6*).
- **RQ6 (Defense evasion techniques and malicious tactics).** We identified three groups of malicious tactics according to how they use of defense evasion techniques: (i) collection of information about the attacked environment, (ii) manipulation of the environment, and (iii) lateral movement on the environment for avoiding the detection. Even if the number of defense evasion techniques used for other malicious tactics is rather limited, we observed that specific techniques are used for specific tactics. Hence, by knowing the type of defense evasion techniques adopted by malware, we can potentially identify the tactics, i.e., the reason of the attack (*Lesson 7*).
- **RQ7 (Violated cIDs rules).** We statistically confirm that there is a significant difference in how cIDs rule categories are triggered (violated) by malware samples. The high violation rates observed for few specific rule categories (including “A Network Trojan was detected”,

Table 22 Violation of cIDS rules by malware samples that integrate defense evasion techniques

Defense evasion techniques and cIDS violation/evasion
HIGH cIDS rules VIOLATION Malware samples are effectively detected by cIDS rules • Bypass User Account Control HIGH cIDS rules EVASION Malware samples are not effectively detected by cIDS rules • Indicator Removal from Tools, Obfuscated Files or Information, Valid Accounts Defense evasion techniques, cIDS violation, and detection delay HIGH cIDS rules EVASION Malware samples that trigger cIDS, but are often in GE23 • Bypass User Account Control, Rundll32 Defense evasion techniques, cIDS evasion, and detection delay HIGH cIDS rules EVASION Malware samples do not trigger cIDS, but are often in GE23 • Abuse Elevation Control Mechanism, Disable or Modify System Firewall, Scripting, Thread Execution Hijacking, Valid Accounts

Table 23 Malware families that violated cIDS rules, and detected in more than 23 days

Malware family: description
Atraps: it is a Trojan diffused by phishing and specialized in steal bank account information and credentials. It mainly affects Android Dinvoke: it is a framework based on PowerShell to execute code in Windows systems without calling APIs Gen4: it is a malware of the generation 4, typically they use artificial intelligence or machine learning algorithms to bypass the security measures Malformed: it is a malware that manipulates files or URLs to corrupt them, e.g., it can malformed URL prefixes, or embedding malformed hyper-links in PDF files, with the aim of bypassing security measures Lotok: it is a Trojan of type Remote Access Trojan (RAT) that often affects Windows-based systems. If installed in a system, it allows the attacker to control the system remotely Obfsstrm: it is type of malware that obfuscates the communication streams to hide the malicious code

“Misc-activity”, “Potentially Bad Traffic”, and “Trojan-activity”) is mainly motivated by the high number of Trojans in the used dataset.

However, up to 54% of the analyzed malware samples did not trigger cIDS rules considered in the VirusTotal platform. This can indicate that such samples do not generate anomalous network traffic and activity, or alternatively that existing rules have to be extended and refined to capture the traffic they generate. Hence, more effort is required to analyze the effectiveness of crowd-sourced rules in depth and identify the reasons behind their failure to trigger (*Lesson 8*).

- **RQ8 (Defense evasion techniques, cIDS rules, and detection delay)** Few defense evasion techniques (e.g., *Bootkit*, *Disable or Modify System Firewall*, and *Software Discovery*) are incorporated only into malware samples that did not trigger the considered cIDS rules. Overall, we observed that a statistically significant association between the integration of defense evasion techniques

and the cIDS rule triggered. For instance, while *Obfuscated Files or Information* and *Valid Accounts* are significantly associated with samples that did not violate any cIDS rules, *Bypass User Account Control* demonstrated a strong positive correlation with cIDS violation (i.e., almost all samples that incorporated such a technique violated cIDS rules).

Moreover, among the malware samples that triggered cIDS rules, only a small set (9.2%) is not detected before 23 days, and resorts to a limited number of families (i.e., *Atraps*, *Dinvoke*, *Gen4*, *Malformed*, *Lotok*, and *Obfsstrm*). The defense evasion techniques are differently incorporated into malware families which samples have been detected before and after 23 days. In particular, samples that incorporated *Bypass User Account Control* and *Rundll32* are frequently detected after 23 days. Furthermore, we identified only few defense evasion techniques, such as *Scripting* and *Valid Accounts*, that are significantly associated with samples that did not trigger any

cIDS rules, and tend to survive more than 23 days. Therefore, defense evasion techniques differently correlate with cIDS rule violations and detection time (*Lesson 9*). Specific evasion techniques seem to be strongly associated with samples detected later than 23 days, even if they finally triggered cIDS rules. However, we have to observe that samples of a given malware family cannot be considered equally. In fact, we noticed that samples of the same family use different sets of defense evasion techniques, thus exhibiting different relationship with the cIDS rule violation and detection delay (*Lesson 10*).

Given the set of malware of our datasets, to summarize the comparison carried out about the identified defense evasion techniques, we computed *WOS*, a unique multi-dimensional index. Specifically, for each defense evasion technique d_i in the set of identified techniques (named *DET*), $d_i \in DET = \{d_1, \dots, d_K\}$, we compute the a set of metrics $M = \{m_1, \dots, m_P\}$, where P is the maximum number of metrics, m_1 is **Popularity**, m_2 is *DEAssociationism*, m_3 is *RevDetectionRate*, m_4 is **DetectionDelay**, m_5 is *RevIDSViolation*, and m_6 is *TacticAssociationism*, as defined in Table 24. The metrics are computed, square root transformation is applied as following to reduce the data skewness, without over-compressing high-frequency outliers:

$$m'_p(d_i) = \sqrt{m_p(d_i)}$$

Then, *Min-Max Scaling* data normalization is applied as following to translate each metrics' value in the range $[0, 1]$, thus limiting the impact of the different malware samples' frequency:

$$N_{m'_p}(d_i) = \frac{m'_p(d_i) - \min_{1 \leq j \leq K} (m'_p(d_j))}{\max_{1 \leq j \leq K} (m'_p(d_j)) - \min_{1 \leq j \leq K} (m'_p(d_j))}$$

Finally, the *WOS* metric is computed as the weighted average of the metrics' value as:

$$WOS(d_i) = \sum_{m_p=1}^{m_p=P} w_{m_p} * N_{m'_p}(d_i)$$

where

$$\sum_{m_p=1}^{m_p=P} w_{m_p} = 1$$

Finally, the *WOS* index is calculated as a value in the range between 0 and 1, where 1 indicates a defense evasion technique frequently used and is expected to be effective, while 0 indicates a possible marginal techniques. To

evaluate the *WOS* stability against subjective metric weighting, we conducted a Monte Carlo sensitivity analysis (with 1000 iterations). Weight vectors for *WOS* were randomly sampled from a Dirichlet distribution, simulating alternative evaluation scenarios, with the aim to confirm that top-ranked defense evasion techniques remain robustly prioritized regardless of weight configuration. To quantify the stability of *WOS*, we applied the Weighted Spearman Rank Correlation (ρ_w) across the Monte Carlo simulations. This metric specifically penalizes rank shifts among high-priority techniques more heavily than among low-ranked ones. In our case, we observed that the computed arithmetic mean of ρ_w ($\mu = 0.646$) is pulled down by rare, degenerate boundary samples generated by the used data distribution. However, the median weighted correlation reaches a significant value $\rho_w = 0.983$ (IQR = $[0.805, 0.993]$). This confirms that across over 75% of the sampled scenarios, the top-tier techniques under *WOS* remains stable, nevertheless the metric weighting choices.

The measured *WOS* for the Top-10 defense evasion techniques is reported in Fig. 17 (top), while the contribution of each metric in Fig. 17 (down). We see that *Obfuscated Files or Information* [42], *Virtualization/Sandbox Evasion* [43], and *Masquerading* [44], are three defense evasion techniques that present the greatest threat to the security of our devices and software. They, in fact, obtained a median *WOS* around 80%, meaning that they are popular, frequently used for different tactics, and in conjunction with other defense evasion techniques, as well as they resulted to be difficult to detect. Specifically, *Obfuscated Files or Information* achieved the best result in terms of Popularity, *DEAssociationism*, and *TacticAssociationism*, meaning that it is very frequently used by malware samples also in conjunction with other defense evasion techniques, and for different tactics (different malicious actions). However, the low result in terms of *RevDetectionRate* and *RevIDSViolation* show that most malware samples that adopt such a defense evasion technique are discovered by most AVs. Conversely, in terms of *DetectionDelay*, *Virtualization/Sandbox Evasion* obtained the best result, meaning that malware samples that adopt such a defense evasion technique can require some days to be discovered. Finally, for both *RevDetectionRate* and *RevIDSViolation* the best result is achieved by *Process Injection* [45], meaning that such a defense evasion technique is particularly effective in hiding malicious software to security mechanisms. Hence, detection strategies can increase their attention toward defense evasion techniques such as *Virtualization/Sandbox Evasion* and *Process Injection*, since they seem to be particularly effective in hiding malicious software (*Lesson 11*).

Several approaches and methods exist to realize each of these defense evasion techniques. For instance, an attacker can obfuscate files of information by compressing, archiv-

Table 24 Overall score: metrics (high value is preferable)

Metric name: description	Rationale
Popularity (Met_1): average number of malware samples that incorporated $d_i \in DET$	It is a proxy for estimating both complexity of use and effectiveness of a defense evasion technique, we assume that a popular technique is almost easy to use but also effective
DEAssociationism (Met_2): average number of different defense evasion techniques incorporated into the malware samples together with $d_i \in DET$	A defense evasion technique used together other techniques is flexible and sophisticated enough to be jointly used with other techniques
RevDetectionRate (Met_3): based on the inverse of the average number of AVs that detected the malware samples that incorporated $d_i \in DET$	It is a proxy of the effectiveness of the evasion technique in contributing to hide the malware to the security mechanisms
DetectionDelay (Met_4): average number of days required to the AVs to detect the malware samples that incorporated $d_i \in DET$	It is an index of the strength of the effectiveness of the defense evasion technique in contributing to hide the malware to the security mechanisms
RevcIDSViolation (Met_5): based on the inverse of the the average number of cIDs rules violated by the malware samples that incorporated $d_i \in DET$	It is a proxy of the capability of the defense evasion technique in hiding the operations and the flow of data generated by the malware's activity to the security mechanisms
TacticAssociationism (Met_6): average number of different malicious tactics used by the malware samples together with $d_i \in DET$	It can give an idea about the possible impact of a successful adoption of the defense evasion technique since it concerns the adoption of other possible malicious capabilities (e.g., collecting information, executing software)

ing or encrypting malicious payloads, she can split a payload into separate files that only reveal malicious functionality when reassembled, or she can encode portions of files to hide the plain-text strings that would otherwise help defenders with discovery. Similarly, an attacker can employ various means to identify artifacts typically indicative of the presence of a virtual machine environment (VME) or sandbox in the target system, and hence be able to change its behavior (e.g., dropping additional and specific payloads, use of sleep timers to trick a temporary sandbox). We can expect that these approaches and methods can have slightly different complexity and sophistication and require moderate to high effort and knowledge to be implemented by malware writers. Moreover, they are often used in conjunction with other defense evasion techniques and tactics, making them effective and increasing their detection delay.

8 Threats to validity

One threat that could hamper the generalizability of results and observations reported in this paper concerns the selected sets of malware samples and AVs considered. Different sets can lead to different results, if they are not enough representative. To limit this threat, we used a non-trivial number of real and recent malware samples, and popular AVs. Moreover, we provided the repetition package, hoping that other researchers can repeat and also extend our experiment.

Another threat concerns the use of only one source for the selection of our malware samples. We are aware that other sources exist (e.g., VirusShare [46] and Malshare [47]), we

selected MalwareBazaar since it is built by the community. The use of a single source can introduce sampling bias, as the content of such a malware samples platform may reflect highly active or commercially prevalent threat categories, while under-representing stealthy, targeted attacks. Consequently, the AVs' detection effectiveness observed might vary with different malware sets. To mitigate this threat and evaluate the statistical robustness of our findings, we conducted an internal validation by extracting multiple random subsets of our dataset, re-computing the detection performance metrics, and performing specific statistical test to evaluate the metrics' stability. While this sub-sampling process does not inherently alter the baseline characteristics of the source, it statistically confirms that our findings are stable with respect to sample variants. Future replications of the experiment can use additional malware sources to further generalize the observations.

Another threat concerns the use of VirusTotal, an online service to run the AVs. We resort to this service, rather than deploying and manually configuring AVs locally, to safely evaluate a large set of AVs within a secure environment. While we are aware that this choice limits our control over the AVs' configurations, update cycles, and runtime detection mechanisms, this approach guarantees a highly standardized, vendor-agnostic, and widely reproducible benchmarking environment, thus eliminating any potential configuration bias introduced by the researchers. We moreover cross-checked the validity of our dataset by analyzing the samples in a second platform, named HybridAnalysis. This platform mainly conduct dynamic analysis, thus representing a complementarity approach with respect

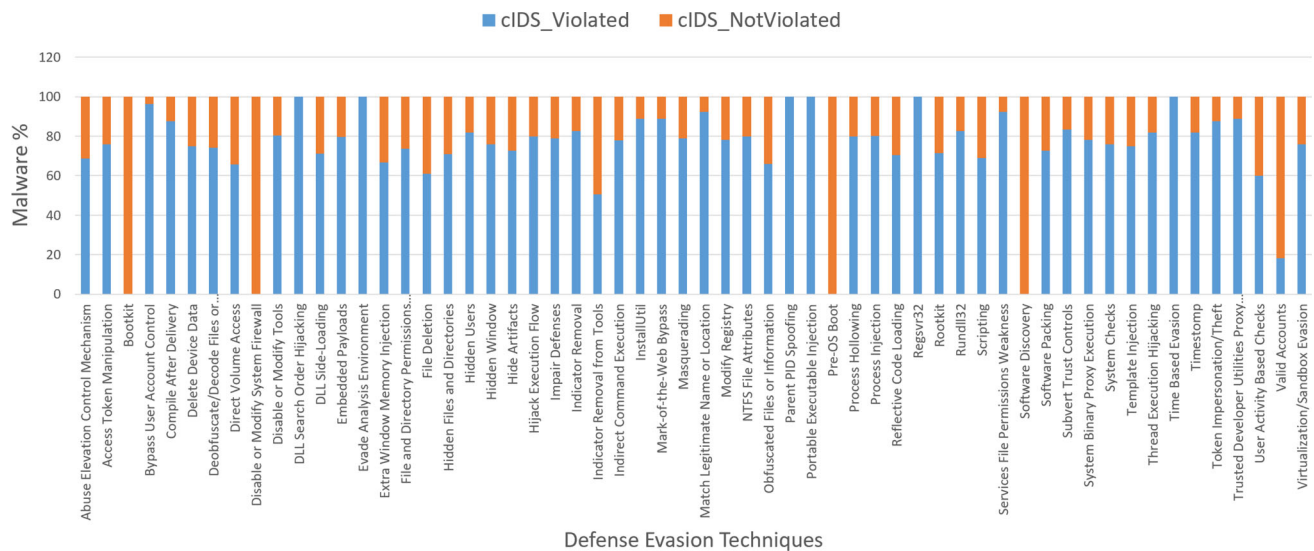


Fig. 16 Comparison of Top-10 defense evasion techniques: median WOS (top), and metrics' contribution (down)

to the traditional one applied by VirusTotal. We believe that the interesting insights of this work can encourage future repetitions and new experiments across different environments, configurations, and settings. A specific threat to validity affects the analysis conducted to check how malware samples of e_2 triggered the cIDS in the VirusTotal sandbox. The absence of a matched cIDS rule does not strictly guarantee that a malware sample evaded the detection through stealthy network behavior, thus that existing cIDS rules are not enough. It may also stem from unobserved network activity caused by inactive Command and Control (C2) servers, anti-analysis/sandbox evasion techniques, unsupported execution operations and environments, or execution analysis timeouts. In our view, only an ad-hoc analysis of the sample execution traces produced in the sandbox can allow us to isolate active traffic events, thus being able to classify the malware samples depending on the reason of their cIDS rules non-detection. The quality of the samples' analysis, as well as of the samples' labels (output of the AV analysis) are key aspects for this work. We are conscious that wrong information and labels can lead to misleading results and that different AVs can label the same sample in a different manner. We selected VirusTotal since it is well-known and used in similar works such as [11, 12]. We conducted the preliminary In-lab experiment also with the purpose of evaluating the quality of the collected samples by involving the expert. Moreover, in the In-large experiment we adopted a conservative approach by considering threshold of more AVs and all possible labels. According with the literature (e.g., [26]), we however observed a substantial reliability of such a service for evaluation purposes. However, we hope that future repetition of the experiment with other platforms or by involving more experts can corroborate the results.

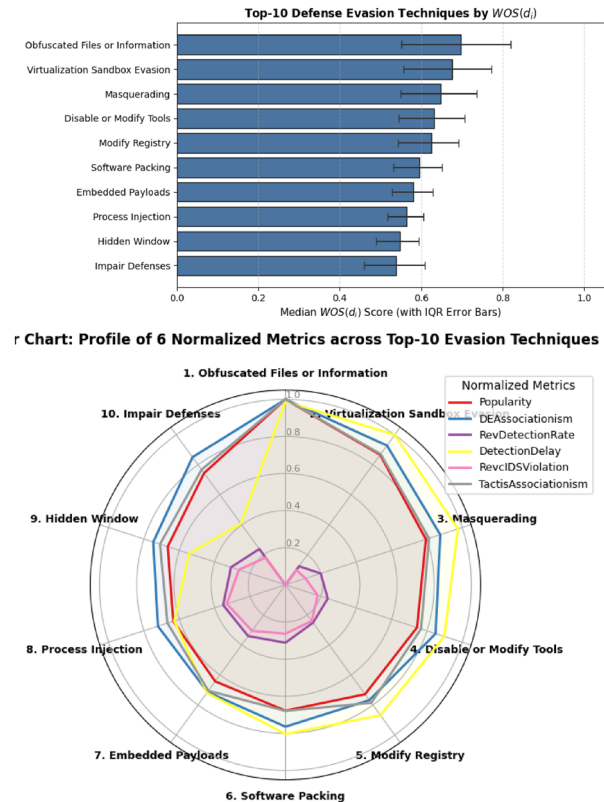


Fig. 17 Comparison of Top-10 defense evasion techniques: median WOS (top), and metrics' contribution (down)

9 Related work

The community is providing a huge effort to develop new techniques and tools to deal with malware, and increase the AVs detection capability with respect to new malware

Table 25 Related work: analysis summary

Rif.	Investigated Aspect	Exp.	AV	Artifact Source	Nr. (Malicious)	Type	AV's detection rate % Max	Avg	Min
[8]	AV's detection rate	In-lab	20 COTS	own	428 (100%)	real malware	100	99	94.2
[7]	AV's detection rate	In-lab	17 COTS	own	358 (100%)	real malware	100	99.3	95.1
[48]	DL/ML detection rate	In-lab	3 own	public	262 (20%)	execution traces	97	81.6	n.a
[11]	AV's detection rate and delay	In-lab+ Online	3 COTS + n.a. COTS	public+ own	28,875 (100%)	real malware	85	61.5	42.5
[12]	AV's-DL/ML detection rate, static and net-work detection rate	In-lab	4 COTS	public	3536 (72%)	real malware	87.3	40.1	0
[18]	AV's detection rate	In-lab	6 COTS	own	21 (100%)	generated pay-loads	100	90.8	71.4
[49]	AV' detection rate	In-lab	4 own	own	10,854 (40%)	execution traces	95.6	56.3	29
[50]	AV's detection rate	In-lab+ Online	4 COTS+ COTS	own	2 (100%)	generated pay-loads	100	62.5	0

Table 25 continued

Rif.	Investigated Aspect	Exp.	AV	Artifact Source	Nr. (Malicious)	Type	AV's detection rate %	Max	Avg	Min
[13]	Detection rate: AVs-DL/ML, static and dynamic detection rate	In-lab+ Online	5 COTS+ COTS	own	15 (40%)	generated loads	pay-	80	34.3	0
[51]	AV' detection rate	Online	20	public	635,000(100%)	real malware		80	67	33
[51]	AV' detection rate by days	Online	20	public	1500(100%)	real malware		-	69	-
[52]	DL/ML detection rate wrt malware hiding	In-lab	2 COTS	public +own	8084(50%)	real malware + generated pay-load		97	90.4	74.1
[52]	AV's detection rate wrt malware hiding	Online	n.a. COTS	public +own	12538(67%)	real malware + generated pay-load		-	-	-
[53]	AV's detection rate wrt malware hiding	Online	79 COTS	public +own	140(100%)	real malware + generated pay-load		40.3	13.9	2
[54]	AV's detection rate wrt malware hiding	Online	72 COTS	public +own	370(81%)	real malware + generated pay-load		98	4	1
Our	AV's detection rate and delay, role of evasion techniques and crowd-sourced detection rules	In-lab+ Online	4 public+ 12 public	public	9281 (100%)	real malware		90.7	73.2	43.1

variants. However, within the “lifecycle” of malicious and anti-malicious software, also malware writers are developing new code by adopting sophisticated and innovative defense evasion strategies. For instance, [55] investigates the use of LLMs to generate malware variants. [16] shows that traditional malware analysis techniques could fail in detecting unexpected patterns and complex processes. They study a specific evasion technique that can resist to traditional analysis methods by using a split-personality software. It uses a technique known as shadow attack to analyze the target environment and collect information about security mechanisms, thus adapting its behavior by partitioning itself into multiple smaller “shadow processes”. [18] examines existing methods and tools for bypassing AVs; their results show that the tools’ evasion effectiveness is positively correlated with their age and popularity. [56] investigates the relationship of malicious and anti-malicious software by means of a co-evolutionary computation technique, based on genetic programming, for developing new variants of a malware, and improving security approaches. Some papers review the existing state-of-the-art in the field of malware and anti-malware. For instance, [15, 57] discuss issues and challenges of the malware analysis. [58] surveys anti-malware dynamic techniques and defense evasion techniques for evaluating their effectiveness.

Pattern-based rules are collected to identify and characterize malware families. For instance, YARA [59] is a set of pattern-based rules often used to identify and classify malware samples. [60] adopts such YARA rules to detect specific malware families and functionality. Other crowd-sourced detection rules have been also applied (e.g., [61, 62]). Such rules collect customer experiences on several vulnerabilities of IoT networks, thus allowing to build intelligent and adaptive detection systems. Moreover, an interesting trend consists in corroborating malware detection by using network-based scanners and dynamic techniques [63, 64].

Recently, machine-learning (ML) and artificial intelligence techniques have been adopted for malware detection and analysis. For instance, [65–68] investigate the adoption of difference AI-based architectures (e.g., convolutional neural networks and recurrent neural networks) to create an intelligent malware detection systems. [12] notices the lack of available ML-based malware detectors, especially the ones that can detect zero-day malware (i.e., never-before-seen). They conducted an experiment and observed that ML-based malware detectors can achieve high precision, but recall is often low, leading to an F1-score of about 70%. [69, 70] survey the literature to provide a systematic overview about ML and deep learning (DL) techniques for malware detection, discussing advances and trends. Recent literature shows that also such modern AI-based AVs have limitations. For instance, [68, 71] study attacks based on adversarial examples and possible counter-measures.

Malware detection and analysis is not limited only to computers and traditional executable software. In fact, modern programming languages and technologies, devices (e.g., smartphones), domains, and embedded systems (including Internet of Things and cyber-physical systems), are a new target of cyber-attacks. For instance, [72] provides an learning-based AV for a modern and dynamic programming language such as JavaScript. Moreover, [19] analyzes Android-specific malware evasion techniques and compares them with anti-malware tools. [48] documents an experiment conducted to assess the performance of data-driven strategies in detecting and classifying Android-specific malware. In the experiment, they used a public dataset of mobile network traffic traces [73] and a set of 400 malware samples. [1] presents a literature review that focuses on ML techniques to automatically detect malware in CPSs, thus identifying the most common attacks against these systems.

Differently from these works, we focused on traditional devices (e.g., computers) that could be attacked by malware, and we did not introduce new detection techniques or survey the literature. Instead, we were interested in studying the capability of AVs to detect malware. Several commercial companies (e.g., Gartner, Security.org, Forrester) rank existing AVs based on information from vendors, experts, and end-user feedback. Similarly, laboratories including AV-TEST, AV-Comparatives, SE-Labs, and MITRE ATT&CK, conduct experiments and publish reports. However, researchers [11, 12] observe that these experiments: (1) focus only on the AVs’ detection rate, (2) do not conduct any in-depth analysis, (3) use not documented datasets, design and tools’ setup, and (4) achieve almost perfect detection rate, typically higher than 95%. These experiments hence do not capture peculiarities and differences among AVs and malware samples. Furthermore, researchers also observe that similar experiments reported in the scientific literature achieved lower results. For instance, [18, 50, 74] present experiments in which payloads have been created with different tools, such as Metasploit [75], and analyzed by existing AVs, e.g., Bitdefender. Results demonstrate that such tools are able to generate malicious files that can bypass AVs’ checks in 50% of cases. [49] presents an experiment conducted to evaluate the capability of ML techniques, such as multi-layer perceptron and random forest, to detect malware. The dataset used in the experiment was composed of traces related to 42 malware families, such as Ransomware, Adware, Scareware, and SMS malware. ML techniques can obtain results that strongly depend on the data used for the model training and on the model’s parameters. [11, 13, 51] reports on experiments similar to ours, but that focus only on the AV detection rate. Specifically, [13] reports on two experiments that conclude that AVs achieve a maximum detection rate of 80%. [11] conducted an experiment in which they measure the AVs’ detection rate of malware sam-

ples from different sources. Results show that the detection rate ranges between 45% and 85%, depending on the used set of malware samples, and that the AVs can need up to 28 days before detecting some malware samples. [51] concludes that over 60% of AVs detect 67% of samples, and that AVs exhibit lower detection capabilities for certain malware families. [51] reports on a study in which 1500 malware samples are tested for 90 days with 20 AVs; the study concludes that on average 69% of the samples are detected. Recent experiments mainly focus on the impact of different malware obfuscation and repackaging mechanisms (e.g., code obfuscation, repackaging, encryption, compression, and steganography) into the AVs' detection rate. For instance, [52–54] consider malware samples and ad-hoc obfuscated variants tested on AVs provided by VirusTotal. They observe that while obfuscation reduces detection rates, several AV engines remain resilient to state-of-the-art obfuscation tools. Specifically, [54] shows that the detection rate can decrease till less than 10% for more than two thirds of the considered AVs. [53] observes that malware samples hidden mechanisms such as steganography, encryption, and compression are highly effective. [52] shows that code obfuscation often confuses AVs, i.e., obfuscated malicious and benign samples tend to be frequently marked by AVs as benign and malicious, respectively (e.g., 95% of obfuscated benign samples were marked as malware by the AVs). Furthermore, [52] report on a second experiment conducted by locally installing two learning-based AVs and providing them training sets mixed with different proportions of obfuscated samples. They observe a detection rate reduction ranging from 5.14% to 35.62%. Overall, there is a need of scientific and rigorous experimentation that deeper investigates the effectiveness of AVs [12]. However, all these works still focus only on the AV detection capability.

Other works focus on defense evasion techniques, e.g., to classify and characterize them. For instance, [76] surveys the techniques used to evade the security checks of intrusion detection and AV tools. [17] identifies three main classes of evasion techniques: transformation (malware that alter their structure and behavior), concealment (malware that conceal the behavior of the malware), and attack-based (malware that engage in an attack on the detector to render it inoperable). [19] characterizes and compares various Android-specific evasion techniques, including Obfuscation, Packing, Code reordering, and System monitoring, among the others.

Table 25 summarizes relevant characteristics of experiments similar to our (e.g., [11, 13]), as well as it reports results of two representative experiments conducted by laboratories ([7, 8]). Even if every experiment is clearly conducted under different setup and scenarios, some common trends exist, in fact, most experiments: (i) focus only on the AVs' detection rate; (ii) consider three types of artifacts: malware samples, malware execution traces, and ad-hoc generated malicious payloads; and (iii) use proprietary sources of malicious arti-

facts. Moreover, (iv) the AVs' detection rate of scientific work tends to be lower than the one collected in the laboratories' experiments. Similar to [13, 50, 54], our experiment setup is organized in an in-lab and an online experiment that involve two different sets of Commercial off-the-shelf (COTS) AVs. Differently from such works, we use real malware instead of generated payloads. Similar to [11, 12], we use real malware samples, compare COTS AVs, and achieve comparable AVs' detection rate, even if both the average and the minimum rate measured by [12] are lower. Similar to all these works, we experimented on the capability of AVs to detect different types of malware. In contrast to such works, we move beyond detection rate and delay. We conduct two in-depth studies exploring the relationship between these measurements and: (i) defense evasion techniques; and (ii) crowd-sourced detection rules. This allows us to increase our knowledge about how to improve the malware detection capability.

10 Conclusions

This paper reports on two experiments conducted to evaluate the detection capability of antiviruses, explore the role of defense evasion techniques, and the use of crowd-sourced detection rules. Specifically, the first experiment has been conducted installing the antiviruses locally and by involving an expert for a qualitative analysis, the second and larger experiment has been conducted by analyzing verdicts produced by VirusTotal, an online multi-antivirus platform. We further conducted two in-depth studies on the VirusTotal verdicts to evaluate the use of defense evasion techniques and of crowd-sourced detection rules. The collected results evidence that the malware detection engines are effective, and that their effectiveness can be improved in several directions; we hence discussed possible further research directions.

Possible extensions of our work are: (i) repeating the experiment with a different set of malware samples and antiviruses; (ii) conducting the experiment in a controlled environment with locally deployed AV products, so hence to avoid the use of online multi-AV services; (iii) extending the expert-based qualitative analysis by involving additional experts; and (iv) extending the in-depth analysis by considering other techniques applied for different tactics, rather than focusing on defense evasion techniques.

Acknowledgements The author thanks the expert T. Andreatta involved in the In-lab experiment, who inspired this work. The work was partially supported by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

Author Contributions A.M. idea, data collection, data analysis, paper write, paper review

Funding Open access funding provided by Università degli Studi di Trento within the CRUI-CARE Agreement. No funding was received for conducting this study.

Data Availability For replication purpose, the complete list of malware samples for reproducibility: <https://tinyurl.com/4ab7kwm3> (2026).

Materials Availability Not applicable.

Code Availability Not applicable.

Declarations

Conflict of interest The authors have no conflict of interest to declare that are relevant to the content of this article.

Ethics approval and consent to participate Not applicable.

Consent for publication Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Montes, F., Bermejo, J., Sanchez, L.E., Bermejo, J.R., Sicilia, J.A.: Detecting malware in cyberphysical systems using machine learning: a survey. *KSII Trans. Internet Inform. Syst. (TIIS)* **15**(3), 1119–1139 (2021). <https://doi.org/10.3837/tiis.2021.03.016>
- Barker, E.B., Smid, M., Branstad, D.: A profile for u. s. federal cryptographic key management systems. National Institute of Standards and Technology (NIST SP 800-152) (2015)
- NIST: NVD (2026). <https://nvd.nist.gov>
- Gartner: Gartner (2025). <https://www.gartner.com/reviews/market/endpoint-protection-platforms>
- Security.org: the best antivirus software in 2026, tested by security researchers (2026). <https://www.security.org/antivirus/best>
- Forrester: the forrester wave (2025). <https://www.forrester.com/research/wave>
- AVTEST: AV-TEST: test results (2026). <https://av-test.org>
- AV-comparatives: independent tests of anti-virus software (2026). <https://www.av-comparatives.org>
- Lab, S.: SE-Lab: reports (2026). <https://selabs.uk>
- MITRE: Adversary Emulation (2026). <https://ctid.mitre.org/resources/adversary-emulation-library>
- Botacin, M., Ceschin, F., de Geus, P., Grégio, A.: We need to talk about antiviruses: challenges & pitfalls of av evaluations. *Comput. Secur.* **95**, 101859 (2020). <https://doi.org/10.1016/j.cose.2020.101859>
- Bridges, R.A., Oesch, S., Iannacone, M.D., Huffer, K.M.T., Jewell, B., Nichols, J.A., Weber, B., Verma, M.E., Scofield, D., Miles, C., Plummer, T., Daniell, M., Tall, A.M., Beaver, J.M., Smith, J.M.: Beyond the hype: an evaluation of commercially available machine learning-based malware detectors. *Digit. Threats* (2023). <https://doi.org/10.1145/3567432>
- Zarghoon, A., Awan, I., Disso, J.P., Dennis, R.: Evaluation of av systems against modern malware. In: International Conference for Internet Technology and Secured Transactions, pp. 269–273 (2017). <https://doi.org/10.23919/ICITST.2017.8356397>
- AVTest: Malware statistis (2026). <https://www.av-test.org/en/statistics/malware>
- Mira, F.: A systematic literature review on malware analysis. In: IEEE International IOT, Electronics and Mechatronics Conference (2021). <https://doi.org/10.1109/iemtronics52119.2021.9422537>
- Murali, R., Ravi, A., Agarwal, H.: A malware variant resistant to traditional analysis techniques. In: International Conference on Emerging Trends in Information Technology and Engineering (2020). <https://doi.org/10.1109/ic-etite47903.2020.264>
- Geng, J., Wang, J., Fang, Z., Zhou, Y., Wu, D., Ge, W.: A survey of strategy-driven evasion methods for pe malware: transformation, concealment, and attack. *Comput. Secur.* **137**, 103595 (2024). <https://doi.org/10.1016/j.cose.2023.103595>
- Samociuk, D.: Antivirus evasion methods in modern operating systems. *Appl. Sci.* (2023). <https://doi.org/10.3390/app13085083>
- Mirza, S., Abbas, H., Shahid, W.B., Shafqat, N., Fugini, M., Iqbal, Z., Muhammad, Z.: A malware evasion technique for auditing android anti-malware solutions. In: IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 125–130 (2021). <https://doi.org/10.1109/WETICE53228.2021.00034>
- Abuse.ch: MalwareBazaar (2026). <https://bazaar.abuse.ch/browse>
- VirusTotal: Analyzers of suspicious files, domains, and IPs (2026). <http://www.virustotal.com>
- GlobalStats: OS market statistis (2026). <https://gs.statcounter.com/os-market-share/desktop>
- Experiment Data Samples: The complete list of malware samples for reproducibility (additional details will be provided upon acceptance) (2026). <https://tinyurl.com/4ab7kwm3>
- Lan, T., Demetrio, L., Nait-Abdesselam, F., Han, Y., Aonzo, S.: Trust under siege: label spoofing attacks against machine learning for android malware detection. *IEEE Trans. Inf. Forens. Secur.* (2026). <https://doi.org/10.1109/TIFS.2026.3671128>
- Wang, J., Wang, L., Dong, F., Wang, H.: Re-measuring the label dynamics of online anti-malware engines from millions of samples. In: ACM on Internet Measurement Conference (2023). <https://doi.org/10.1145/3618257.3624800>
- Zhu, S., Shi, J., Yang, L., Qin, B., Zhang, Z., Song, L., Wang, G.: Measuring and modeling the label dynamics of online anti-malware engines. In: USENIX Conference on Security Symposium. USENIX Association, USA (2020)
- Wang, L., Wang, H., Zhang, T., Xu, H., Meng, G., Gao, P., Wei, C., Wang, Y.: Android malware family labeling: Perspectives from the industry. In: International Conference on Automated Software Engineering (2024). <https://doi.org/10.1145/3691620.3695280>
- CrowdStrike: Hybrid Analysis (2026). <https://hybrid-analysis.com>
- CrowdStrike: Defeat malware (2026). <https://www.crowdstrike.com/en-us/platform/threat-intelligence/malware-analysis>
- OPSWAT (2026). <https://www.opswat.com>
- MITRE: Tactic:TA0005 (2025). <https://attack.mitre.org/tactics/TA0005>
- MITRE: Tactics (2026). <https://attack.mitre.org/tactics>
- Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann Publishers Inc, USA (1994)
- BIIT: ClustVis (2025). <https://biit.cs.ut.ee/clustvis>
- MITRE: MTactic:TA0009 (2025). <https://attack.mitre.org/tactics/TA0009>

36. MITRE: Tactic:TA0040 (2025). <https://attack.mitre.org/tactics/TA0040>
37. MITRE: Tactic:TA0007 (2025). <https://attack.mitre.org/tactics/TA0007>
38. MITRE: Tactic:TA0002 (2025). <https://attack.mitre.org/tactics/TA0002>
39. Cisco: Snort (2025). <https://www.snort.org>
40. Proofpoint (2025). <https://www.proofpoint.com>
41. OISF: Suricata (2025). <https://suricata.io>
42. MITRE: Technique:T1027 (2025). <https://attack.mitre.org/techniques/T1027>
43. MITRE: Technique:T1497 (2025). <https://attack.mitre.org/techniques/T1497>
44. MITRE: Technique:T1036 (2025). <https://attack.mitre.org/techniques/T1036>
45. MITRE: Technique:T1055 (2025). <https://attack.mitre.org/techniques/T1055>
46. VirusShare: Malware repository (2025). <https://virusshare.com>
47. Malshare: Malware repository (2025). <https://github.com/malshare>
48. Bovenzi, G., Cerasuolo, F., Montieri, A., Nascita, A., Persico, V., Pescape, A.: A comparison of machine and deep learning models for detection and classification of android malware traffic. In: IEEE Symposium on Computers and Communications. IEEE, Greece (2022). <https://doi.org/10.1109/iscc55528.2022.9912986>
49. Chandrakala, D., Sait, A., Kiruthika, J., Nivetha, R.: Detection and classification of malware. In: International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation, pp. 1–3 (2021)
50. Arunanshu, G.S., Srinivasan, K.: Evaluating the efficacy of antivirus software against malware and rats using metasploit and asynchr. In: Innovations in Power and Advanced Computing Technologies (2023)
51. Muzayan Haq, M.Y., Abhishta, A., Zeijlemaker, S., Chau, A., Siegel, M., Nieuwenhuis, L.J.M.: Measuring malware detection capability for security decision making. In: European Symposium on Security and Privacy Workshops (EuroS&PW) (2024). <https://doi.org/10.1109/EuroSPW61312.2024.00044>
52. Cuiying, G., Wu, Y., Li, H., Yuan, W., Jiang, H., He, Q., Liu, Y.: Uncovering and mitigating the impact of code obfuscation on dataset annotation with antivirus engines. In: International Symposium on Software Testing and Analysis (2024). <https://doi.org/10.1145/3650212.3680302>
53. Dell'Orco, D., Valeriani, L., Bianchi, G., A, P., Merlo, A.: Challenging malware against elusive android malware over time. In: ITASEC/SERICS (2025)
54. Aswal, K., Pathak, H., Singh, N., Gupta, N.: Strength and limitations of publicly available anti-malware tools against obfuscated malware. In: International Conference on Inventive Research in Computing Applications (ICIRCA) (2023). <https://doi.org/10.1109/ICIRCA57980.2023.10220600>
55. Akil, M.A., Li, A.S., Karim, I., Iyengar, A., Kundu, A., Parla, V., Bertino, E.: Llmalmorph: on the feasibility of generating variant malware using large-language-models. In: International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (2025). <https://doi.org/10.1109/TPS-ISA67132.2025.00026>
56. Sen, S., Aydogan, E., Aysan, A.I.: Coevolution of mobile malware and anti-malware. IEEE Trans. Inf. Foren. Secur. **13**(10), 2563–2574 (2018). <https://doi.org/10.1109/TIFS.2018.2824250>
57. Joshi, Y.K., Tiwari, N.: A comprehensive survey on malware detection techniques. In: International Conference on Information Management & Machine Intelligence. Association for Computing Machinery, India (2024). <https://doi.org/10.1145/3647444.3647830>
58. Afianian, A., Niksefat, S., Sadeghiyan, B., Baptiste, D.: Malware dynamic analysis evasion techniques: a survey. ACM Comput. Surv. (2019). <https://doi.org/10.1145/3365001>
59. VirusTotal: Yara documentation. <https://yara.readthedocs.io/en/stable/index.html> (last acc. 2026)
60. Huang, G.-C., Chang, K.-C., Lai, T.-H.: Chaotic-based shellcode encryption: a new strategy for bypassing antivirus mechanisms. Symmetry (2024). <https://doi.org/10.3390/sym16111526>
61. Venkatraman, S., Surendiran, B.: Adaptive hybrid intrusion detection system for crowd sourced multimedia internet of things systems. Multimed. Tools Appl. (2020). <https://doi.org/10.1007/s11042-019-7495-6>
62. Kantipudi, M.P., Aluvalu, R., Velamuri, S.: An intelligent approach of intrusion detection in mobile crowd sourcing systems in the context of iot based smart city. Smart Sci. **11**(1), 234–240 (2023). <https://doi.org/10.1080/23080477.2022.2117889>
63. Mangayarkarasi, R., Vanmathi, C., Ravi, V.: A robust malware traffic classifier to combat security breaches in industry 4.0 applications. Concurr. Comput. Pract. Exp. (2023). <https://doi.org/10.1002/cpe.7772>
64. Lu, T., Wang, J.: F2dc: android malware classification based on raw traffic and neural networks. Comput. Netw. (2022). <https://doi.org/10.1016/j.comnet.2022.109320>
65. Kandasamy, C.A., Chandru, P., Kanimozhi, J.K., Loganathan, E., Prakash, N., Yoganathan, A.: Intelligent malware detection: enhancing accuracy with recurrent neural network. In: International Conference on Trends in Electronics and Informatics (ICOEI) (2025). <https://doi.org/10.1109/ICOEI65986.2025.11013413>
66. Farfoura, M.E., Mashal, I., Alkhatib, A., Batyha, R.M.: A lightweight machine learning methods for malware classification. Clust. Comput. (2024). <https://doi.org/10.1007/s10586-024-04755-2>
67. Dustova, S., Abuzalova, M., Adizova, N., Ruzieva, D., Ruziev, Y.: Ai-powered malware detection: a hybrid cnn-rnn model for real-time threat analysis. In: International Conference on Optical and Computational Technologies for Measurements and Industrial Applications (2025). <https://doi.org/10.1117/12.3078041>
68. Ban, Y., Kim, M., Cho, H.: An empirical study on the effectiveness of adversarial examples in malware detection. Comput. Model. Eng. Sci. (2024). <https://doi.org/10.32604/cmes.2023.046658>
69. Gibert, D., Mateu, C., Planes, J.: The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. J. Netw. Comput. Appl. **153**, 102526 (2020). <https://doi.org/10.1016/j.jnca.2019.102526>
70. Pachhala, N., Jothilakshmi, S., Battula, B.P.: A comprehensive survey on identification of malware types and malware classification using machine learning techniques. In: International Conference on Smart Electronics and Communication (ICOSEC). IEEE, India (2021). <https://doi.org/10.1109/icosec51865.2021.9591763>
71. Tsingenopoulos, I., Cortellazzi, J., Bošanský, B., Aonzo, S., Preuveneers, D., Joosen, W., Pierazzi, F., Cavallaro, L.: How to train your antivirus: RL-based hardening through the problem space. In: International Symposium on Research in Attacks, Intrusions and Defenses (2024). <https://doi.org/10.1145/3678890.3678912>
72. Lima, S.M.L., Souza, D.M., Pinheiro, R.P., Silva, S.H.M.T., Lopes, P.G., Lima, R.D.T., Oliveira, J.R., Monteiro, T.D.A., Fernandes, S.M.M., Albuquerque, E.D.Q., Silva, W.W.A., Santos, W.P.: Next-generation antivirus for javascript malware detection based on dynamic features. Knowl. Inf. Syst. (2024). <https://doi.org/10.1007/s10115-023-01978-4>
73. Lashkari, A.H., Kadir, A.F.A., Gonzalez, H., Mbah, K.F., Ghorbani, A.A.: Towards a network-based framework for android malware detection and characterization. In: Annual Conference on Privacy, Security and Trust. IEEE, Canada (2022). <https://doi.org/10.1109/PST.2017.00035>

74. Ahmed, A., Garba, F., Abba, A.: Evaluating antivirus evasion tools against bitdefender antivirus. In: International Conference on FIN-TECH Opportunities and Challenges (2021)
75. Metasploit: Metasploit Framework (2025). <https://www.metasploit.com>
76. Marpaung, J.A.P., Sain, M., Lee, H.-J.: Survey on malware evasion techniques: state of the art and challenges, pp. 744–749. (2012). **(International Conference on Advanced Communication Technology)**

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.