Evaluating Anti-Virus Effectiveness in Linux

Giuseppe Raffa
Royal Holloway, University of London
Egham, UK
giuseppe.raffa.2018@live.rhul.ac.uk

Daniele Sgandurra
Royal Holloway, University of London
Egham, UK
daniele.sgandurra@rhul.ac.uk

Daniel O'Keeffe
Royal Holloway, University of London
Egham, UK
daniel.okeeffe@rhul.ac.uk

***Abstract*—Anti-virus (AV) software is widely recognized as one of the most important defensive tools against malware. Although historically many Linux users considered this operating system to be malware-free, recent research suggests that Linux malware is on the rise. However, to date there has not been a comprehensive observational study on the effectiveness of modern Linux AVs.**

In this work, we evaluate a range of Linux AVs using a dataset of 43,553 Linux malware samples, conducting our analysis over a period of ten months to identify possible regression effects. We measure the detection rates of Linux AVs available in our local test environment and on an online malware scanning service. Furthermore, we perform a Linux malware capability analysis using the open-source tool CAPA. Overall, the results of this work show that Linux AVs' signature databases are not well maintained by AV vendors, and that several Linux AVs are affected by regression. In addition, our capability analysis suggests that malware authors are trying to further specialize existing approaches for evading AV software rather than developing new capabilities.

***Index Terms*—Linux malware, Anti-virus regression, CAPA tool, Signature-based detection**

I. Introduction

Anti-virus (AV) software plays an important part in protecting end-users and networks from several types of malware. Consequently, installing and keeping up-to-date an AV program is widely considered an essential step in securing a vast range of computing devices regardless of the particular operating system. As the figures published by a major AV vendor show [1], Linux users are a lot less affected by malicious software in comparison with Windows users. However, Linux cannot be considered completely malware-free, as confirmed by documented examples of Linux-specific malware, e.g., the one discovered in the 17.3 version of Linux Mint [2]. Linux systems are also exposed to cross-platform threats coming from HTML, PDF and JavaScript [1], [3].

Currently, Linux is the most widely used OS for server computers. Statistics published by Trend Micro in 2021 [4] estimate that 96.3% of the top 1 million Internet web servers run Linux. Furthermore, its market share in cloud computing has been around 90% for the last three years [5]. These statistics, along with the fact that Linux servers can act as a storage or command-and-control server for Windows malware [4], explain why major AV companies have developed Linux-compatible products targeting this kind of system.

978-1-6654-8045-1/22/$31.00 ©2022 IEEE

On desktops, Black Lotus Labs reported in September 2021 [6] that malicious Linux ELF files can also be used to compromise computing platforms relying on the Windows Subsystem for Linux (WSL) [7], which enables the execution of native Linux applications in Windows hosts [8]. In the academic community, the growing popularity of IoT devices running variants of Linux has produced a body of recent research [9], [10] that aims to characterize malware targeting them. Such malware, in fact, is on the rise [11] and its complexity is growing as well, with cybercriminals, for instance, increasingly exploiting IoT platforms to run crypto-mining malware [9]. Despite this threat landscape, except for an online report published in 2015 [12], very little information about the effectiveness of Linux AVs can be found in the literature.

With this work, we perform a systematic measurement study on the status of Linux AVs. Due to the complexity of modern AV software and the lack of detailed documentation [13], a comprehensive analysis is non-trivial, as it requires using a large number of malware samples and measuring potential regression effects over time [14], [15]. To this end, we carry out an extensive AVs evaluation by measuring the detection rate of locally-installed AVs as well as of those available through the online malware scanning service VirusTotal. We then assess the AVs' effectiveness over a period of ten months to find out whether they are affected by regression such that an AV is no longer capable of identifying malware samples that were successfully detected in the past. To better understand why some malicious files are misidentified by the locally-installed software, we also perform a malware capability analysis with the CAPA tool [16] focused on a subset of false negative and true positive samples.

Our results show that three out of four locally-installed AVs exhibit detection rates well above 90%. However, we observe that the detection rates over the same malware dataset increase very marginally over time, and that two of the tested products show file-level regression. Additionally, the capability analysis highlights that there are no significant differences between detected and undetected malicious samples in terms of their abilities to carry out specific malicious activities. This suggests that malware authors are focusing their attention on efficiently employing consolidated techniques instead of identifying novel ones. For the tested VirusTotal AVs, unexpectedly, one third have a detection rate of at most 30%, and 24 out of 58 are affected by regression. Our tests, therefore, confirm that AV vendors should thoroughly review how Linux-

specific signature databases are maintained and distributed.

In summary, we make the following contributions:

- We provide up-to-date figures regarding the detection rate of four prominent locally-installed Linux AVs, and quantify their overall and file-level regression effects.
- We conduct a malware capability analysis to gain insight into the results obtained with the locally-installed AVs. To this end, we open source a new framework[1] that was used to process multiple sets of files with CAPA and to efficiently generate test reports.
- We assess the effectiveness of the AV engines available on an online malware scanning service against Linux-specific malicious samples.
- We compare the tested locally-installed AVs to their online versions and quantify the differences in terms of detection rates and regression effects.

II. AV Software Overview

AV software is specifically designed to detect malicious software (*malware*) and is one of the most widely adopted preventative security measures [3]. It can also be deployed as a reactive control, being in most cases capable of removing the malicious code once the detection mechanism has successfully been triggered. As further explained in this section, given the level of sophistication of modern malware, developing an AV program is a very complex task.

Modern AVs rely on three broad types of detection mechanism: *signature-based*, *heuristic detection* and *behaviour-based* [17], [18]. A signature can be extracted by a malware sample in many different ways. Pattern matching (e.g., identification of a specific string), cyclic redundancy checks (CRCs) applied to the whole file or only chunks of data, and cryptographic hash functions are all widely used [3]. Furthermore, there exist more sophisticated methods, which are generally based on complex heuristic patterns, for example, aiming at analysing specific features included in Windows Portable Executable (PE) files [3]. Heuristics of this type, which do not require the execution of the sample to be inspected, are also referred to as static heuristics [18].

By contrast, when behaviour-based detection is adopted, the AV monitors the execution of the suspicious file in an attempt to identify actions that could have a malicious intent, for example access to an operating system configuration file. Behaviour-based approaches, which include dynamic heuristic detection [18], have been introduced because the number of signatures that have to be generated and distributed to the end-user has become unsustainably high [5], despite the frequent adoption of cloud models for AV updates [18]. However, signature-based detection techniques are still widely used, because they are normally very specific, which implies they are less likely to cause undesirable false positives. On the contrary, behaviour-based approaches normally cause an increase in false positives, as they rely on monitoring actions that could well be performed by legitimate code [18].

[1] https://github.com/giusepperaffa/capa-tool-launcher

To overcome the aforementioned limitations, machine learning (ML)-based methodologies aiming at detecting malware have been studied and tested. Polymorphic and metamorphic variants [19], in fact, enable the malware author to reuse existing malicious code after changing its appearance. In many circumstances, this is sufficient to evade the AV, especially when signature-based detection is employed. As a result of all this, several ML-based techniques have been developed [20], [21] to improve the effectiveness of AV products, irrespective of the particular operating system used. As pointed out by Anderson *et al.* [22], AV vendors are currently using ML-based techniques both for primary detection engines and supplementary detection heuristics. However, ML models for AVs have been shown to be easily evaded [23] and they cannot therefore completely replace signature-based detection systems.

III. Research Objectives and Approach

The overall objective of this work is to provide a comprehensive evaluation of a range of Linux-compatible AV solutions. To this end, we completed two sets of tests, which were performed with local installations and an online malware scanning service. Furthermore, to gain insight into the results obtained in our local test environment, we carried out an additional capability analysis focused on a subset of the tested malware samples. More specifically, we formulated four research goals for our evaluation.

RG1: How effective are Linux AVs in terms of detection rate? We want to perform a thorough evaluation by measuring the AVs' detection rates in a test environment under our control, i.e., where both the AV engines' versions and the used signature databases are known. To achieve this goal, we evaluate actual AV installations (§ IV-A). We note that this approach cannot be adopted to measure the effectiveness of all anti-malware solutions. In fact, in addition to compatibility-related issues [24], installing, configuring, updating and submitting malware samples to locally-installed AVs does not scale well. To address this, we perform as second set of tests (§ IV-C) to assess the effectiveness of a larger number of AVs by employing an online malware scanning service and, as in [14], we chose VirusTotal [25].

RG2: Do Linux AVs show any regression effects over an extended period of time? We want to perform a comparison of detection rates over a period of ten months to assess the AVs' performance against the same set of malware samples, and verify whether their signature databases are promptly updated. Recent research [14], in fact, has highlighted the importance of measuring the detection rate of AVs multiple times during an observation period. This has allowed us to quantify the effectiveness and timeliness of the AV update mechanisms, and to identify regression effects, which are caused by signature database updates and changes to the adopted heuristics.

RG3: What are the capabilities of the undetected malicious samples? We want to identify the capabilities of the undetected malicious samples to provide Linux AV vendors with a reference for future improvements. To reach this goal, we use CAPA [16], an open-source capability analysis tool that also processes Linux ELF files [26], [27]. Moreover, to facilitate the analysis of multiple file repositories and to efficiently generate test reports, a new framework has also been designed as part of this research (§ IV-B).

RG4: Is there any difference in detection results and regression effects among local and online versions of Linux AVs? We want to compare detection rates and regression results of local and online versions of Linux AVs over the same dataset. Previous AV evaluation studies [14], [18] have reported discrepancies between results obtained with locally-installed AVs and those provided by web-based services. While the inevitable differences in AV engine and signature database versions are the cause of these discrepancies [28], there is not much information on this about Linux-compatible AVs. Even though there are no guarantees on the OS version of the VirusTotal AV scanners[2], analysing these potential inconsistencies allows evaluating the AV solutions comprehensively, as it shows to what extent the online services-originated results are reliable when Linux malware is considered. As regards the regression effects, differently from [14], we want to provide Linux-specific figures both for the locally-installed and the online AVs.

IV. Evaluation Methodology

In this section, we define the methodology used in our study, which is based on two sets of tests. The first were conducted using virtual machines (§ IV-A). Their results were further analysed by extracting the capabilities of the scanned malicious files with an open-source tool (§ IV-B). The other set of tests were performed with an online malware scanning service (§ IV-C). Finally, we conclude this section by detailing all our test environments (§ IV-D).

A. Local Tests with Virtual Machines

The first set of tests aimed at assessing the effectiveness of four locally-installed AV programs, which we chose based on them being among the most widely used. To preserve their anonymity, we refer to them as AV1, AV2, AV3 and AV4 in the rest of the paper. In order to avoid potential issues caused by the installation of more than one AV on the same computer [24], multiple virtual machines (VMs), each including one selected AV, were installed and configured. The targeted AVs were then tested by executing five scans of the same set of malicious samples during a period of ten months. The first four were run over the course of three weeks between June and July 2020, to measure changes in AVs' detection rates during a short period of time, whilst the last one was performed in April 2021, to quantify AVs' regression effects over a longer period of time. Each scan was run after updating the AV signature database, and the first four tests were separated by approximately one week to avoid the short-lived result changes highlighted in previous research [15].

As far as the used malware repository is concerned, we carried out our evaluation with the archive of malicious Linux Executable and Linkable Format (ELF) files made available on VirusShare [30] on 5^{th} April 2020, which contained 43,553 samples. Note that such samples constitute our ground truth dataset, as they were all considered malicious and executable. In addition, our malware capability analysis (§ IV-B) shows that the VirusShare dataset includes a variety of architectures. Having performed our local tests with desktop installations though, we could have limited our evaluation to 32 and 64-bit x86 samples, which directly target desktop computers. However, modern AVs are modular systems, which implies that their engines are implemented in such a way that they can be deployed on multiple platforms, e.g., cloud-based, with minimal changes. Consequently, we reckon that considering architectures other than x86, e.g. ARM and MIPS, would assist us in carrying out a more comprehensive evaluation.

B. Malware Capability Analysis

To gain insight into the results obtained with the first set of tests (§ IV-A), we conduct a malware capability analysis with the CAPA tool. In this context, a *capability* can be described as the ability to carry out a certain (malicious) task, e.g., establishing a network connection. CAPA extracts capabilities by relying on *rules*, which detail capability-specific signatures. In our work, we leverage only the default ELF rules available in the version of the tool we used (§ IV-D). We note also that CAPA currently has two limitations for our purposes: ❶ it includes a low number of ELF rules[3] and ❷ it supports only 32 and 64-bit x86 architectures[4].

To achieve **RG3** (§ III), we identified two subsets for each evaluated AV: ❶ the first contained the samples that were never detected during our testing campaign, whereas ❷ the second included the malicious files that were flagged at least once. Both these subsets were analysed with CAPA in November 2021. Note that, while our **RG3** is focused on the undetected specimens' capabilities, analysing the other subsets enabled us to better understand similarities and differences between false negatives and true positives.

Since the CAPA standalone binary returns the results in JSON format and cannot be launched to automatically analyse files stored in multiple repositories, we also implemented a new Python command-line tool to generate detailed reports

[2]The authors contacted VirusTotal in May 2021 to obtain more information about the version (i.e., Linux or Windows-specific) of the available AV engines. The Support Team clarified that VirusTotal AV engines are able to determine the best way of scanning each malware sample, regardless of the targeted OS – see also [29].

[3]There are currently 36 rules available for ELF files according to [26], i.e., one order of magnitude lower than the number of Windows PE rules.

[4]We were unable to find evidence of this limitation in the CAPA documentation [16], [26], [27]. However, it is explicitly mentioned in the CAPA log files we analysed.

for each tested repository along with summaries that facilitate the comparison of results obtained with different sets of files.

C. Online Tests with VirusTotal

To measure the effectiveness of a larger number of AVs against Linux-based malware, our second set of tests use VirusTotal, an online malware scanning service, which included 62 AV products at the beginning of our study. VirusTotal offers an Application Programming Interface (API), which we used to scan 4,000 randomly selected malware specimens from the same dataset used in the local tests. In addition, to ensure consistency with the evaluation of the locally-installed AV software, during the first part of our study (June - July 2020) the malicious files were submitted twice over a period of two weeks to identify short-term regression effects. To evaluate possible degradation effects after a longer period of time, we also conducted a third test in April 2021 when the number of AVs available in VirusTotal was 59.

D. Test Environments

We now provide further details on the test environments.

Local Tests with Virtual Machines. Table I specifies the main features of the host and of the guest systems used in our testing environments. In both cases we relied on a recent version of the Ubuntu Linux distribution.

TABLE I: Summary of the host and guest system features

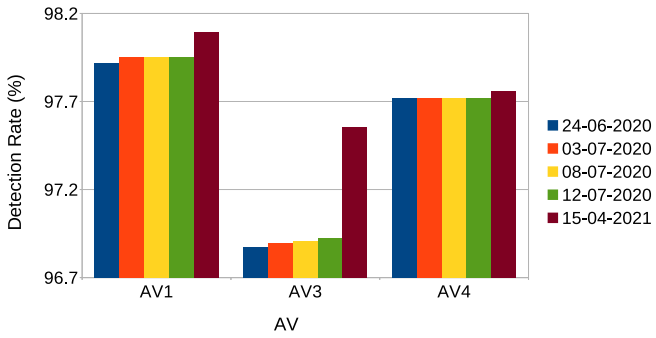
Feature	Host System	Guest System
Memory	16 GB DDR4	12 GB
OS	Ubuntu 18.04 LTS	Ubuntu 18.04 LTS
Hypervisor	VirtualBox 6.1	✗
Processor	Core i7 8750H	✗
Virtual HD Size	✗	32 GB

Malware Capability Analysis. We conducted our malware capability analysis with the version v3.0.2 of CAPA, which we executed in a Ubuntu Linux VM. After preliminary experiments with our reference guest system we set the RAM to 12 GB (see Table I), and enabled 4 CPU cores to minimize the number of cases where our analysis timeout (5 minutes) was reached.

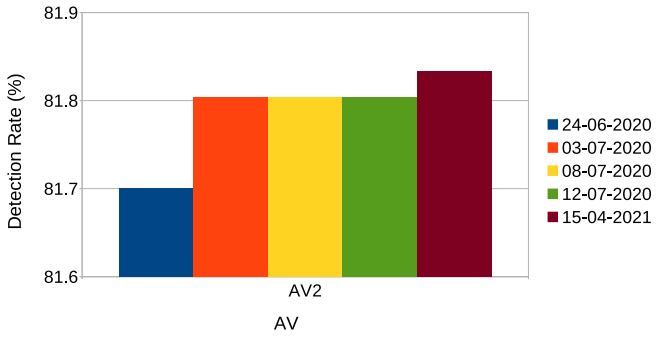
Online Tests with VirusTotal. For the tests conducted with VirusTotal, the first and the second were carried out between 30^{th} June 2020 and 15^{th} July 2020, whilst the third was performed between 17^{th} and 21^{st} April 2021. An API-based Python scanner was used to submit the selected malware samples to VirusTotal.

V. Evaluation Results

We now analyse and summarize the results of all the performed tests. Note that additional insights along with specific recommendations are provided in § VI.



(a)



(b)

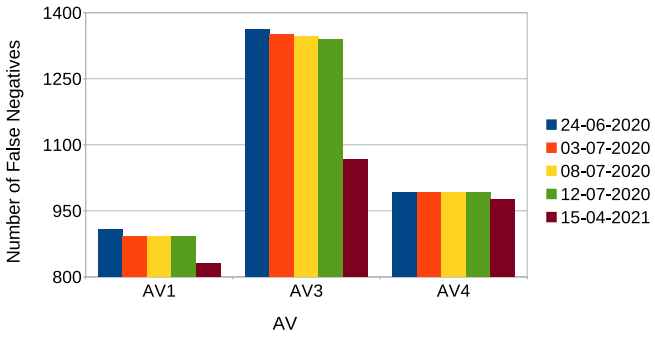
Fig. 1: Detection rate vs AV. Fig. 1a reports the detection rates of AV1, AV3, and AV4, whilst Fig. 1b reports those recorded with AV2. We show AV2 in a separate figure with a different scale to better visualize the differences

A. Local Tests with Virtual Machines

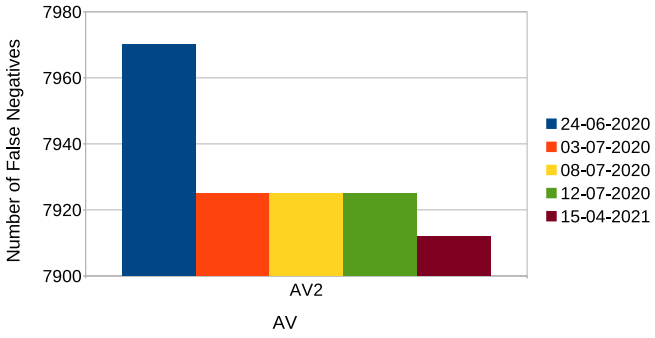
To present the results obtained within our local test environment, which are instrumental in achieving **RG1**, **RG2**, and **RG4** (§ III), we start by analysing the AVs' detection rates. We then measure the update timeliness, the regression effects, and the execution times.

Detection Rate. Congruently with previous AV evaluation studies [18] and online reports [12], [31], we focus on measuring the detection rate (DR) of the selected AVs, which is defined as $DR = \frac{TP}{TP+FN}$, where TP is the number of true positives (meaning that the AV is correctly flagging a malware sample as such), and FN is the number of false negatives (meaning that the AV is not flagging a malware sample as such). As reported in Figure 1, we can see that tested AV's average detection rates (i.e., averaged over the total number of scans) range from 81.8% for AV2 (the lowest) to 98% for AV1 (the highest). It is noteworthy that the results of the last scan show only marginal improvements, with the largest detection rate increase in comparison to the penultimate test being 0.6% (AV3).

Update Timeliness. As regards the timeliness of the AVs' update mechanisms, although our tests started more than ten



(a)



(b)

Fig. 2: False negatives vs AV. Fig. 2a reports the number of false negatives for AV1, AV3, and AV4, whilst Fig. 2b reports those recorded with AV2. We show AV2 in a separate figure with a different scale to better visualize the differences

weeks after the ELF file dataset was made available, the highest detection rate ever recorded in our study is 98.1%. In addition, only the detection rate of AV3 improves in each scan, with the last one being 0.7% higher than the first. By contrast, AV1 and AV2 increase their detection rates in only two scans (with an average of 0.1%), while AV4 shows a detection rate increase in the last test only (less than 0.1% in comparison to the previous scan).

Therefore, as illustrated in Fig. 2, although the false negatives rate (i.e., $100 - DT$) for AV1 is always lower than 2.1%, we record, on average, 883 false negatives over 43,553 samples. This is a surprisingly high number of undetected malicious files, given that the dataset contained malware known for more than two months when our evaluation began. The amount of false negatives is even higher for the other AV products. In particular, the AV with the worst detection rate (AV2) shows 7,970 false negatives during the first test, which very marginally decrease to 7,912 in the last one.

Regression Effects. To measure detection regression [14], where previously detected malware samples are no longer flagged as malicious, we post-process our results on a per-file basis to compare the last scan with all the others. This analysis shows that AV1 and AV4 are affected by file-level

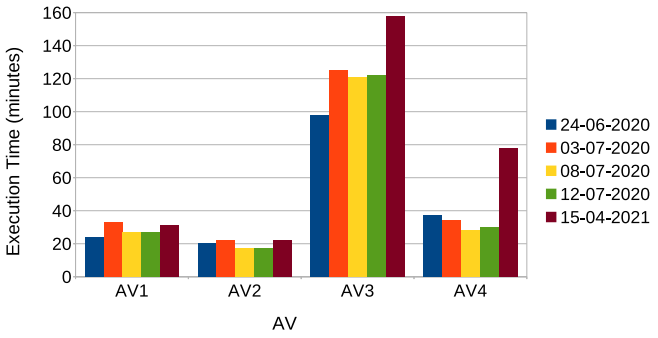


Fig. 3: Test execution time vs AV

regression, albeit limited to 6 and 3 specimens, respectively. Interestingly, both AVs are able to detect these malware samples in all tests apart from the last one.

Execution Times. We also measure the time taken by each AV to scan of the entire VirusTotal dataset of 43,553 samples. As shown in Fig. 3, except for AV3, which has an average scan execution time of 125 minutes, all the other AVs show comparable values, with average scanning times ranging from 20 (AV2) to 41 minutes (AV4). We note that all AVs exhibit an increase in scan execution time during the last test.

Summary. The locally-installed AVs, which are assessed against a repository containing 43,553 malicious ELF files, exhibit an overall average detection rate of 94%. The performed tests also indicate that the detection rates increase very marginally over time, the maximum recorded being 0.7% during the entire observation period. Furthermore, two of the tested products show file-level regression, albeit limited to at most six files.

B. Malware Capability Analysis

This section is focused on our malware capability analysis, which we perform with CAPA to better interpret the results obtained with the locally-installed AVs and, more specifically, to achieve our **RG3** (§ III). As previously explained (§ III), we first identify, for each evaluated AV, two subsets. The first contains the malicious files that have never been flagged as such in our local tests (*false negatives*), whereas the second includes the malware samples that have been detected at least once (*true positives*): both sets are detailed in Table II. Note that, unsurprisingly, some malicious files belong to more than one subset. Consequently, as further illustrated in the remainder of this section, we analyse in total 8,337 different false negatives and 42,962 different true positives.

False Negatives Analysis. The results obtained with CAPA are presented in Fig. 4. Note that capabilities were identified for 313 malicious files, given that, in most cases, the tool was unable to complete its analysis because of either an

TABLE II: Number of samples in the AV-specific subsets identified for our malware capability analysis (FN=False Negatives, TP=True Positives)

AV	FN Subset	TP Subset
AV1	824	42,729
AV2	7,912	35,641
AV3	1,135	42,418
AV4	971	42,582

unsupported architecture or an unsupported OS[5].

CAPA returns the detected capabilities in a hierarchical structure, i.e., a *namespace* such as `anti-analysis/packer/upx`, which helps with their categorization. To facilitate the interpretation of our results, we first extract the top level of the returned namespaces, which we refer to as *high-level capabilities*. In total, ten of them are identified, and, as illustrated in Fig. 5, the two with the highest overall number of occurrences are `host-interaction` (641) and `anti-analysis` (477). To gain additional insight, we then analyse the namespaces as a whole[6], and, as detailed in Table III, we find that the most frequently detected are `anti-analysis/packer/upx` and `anti-analysis/anti-vm/vm-detection` with 199 and 181 occurrences, respectively.

As regards the samples classified by CAPA as unsupported architecture, they are, as expected, all recognized as ELF by the standard Linux `file` command. However, the latter also shows that the architecture of these specimens is always different from 32 and 64-bit x86, which is the only one supported by the used CAPA version. Therefore, the results of our experiments are affected by the composition of our dataset (§ IV-A) and by the limitations of CAPA (§ IV-B).

The unsupported OS files were also further checked with the Linux `file` command. Similarly to the previous case, they are all classified as ELF, but somewhat surprisingly their architecture is always either 32-bit or 64-bit x86. Our analysis of the CAPA source code indicates that the tool features a mechanism for identifying multiple ELF-compatible OSes. However, among other things, the mechanism relies on processing the file header, which, as pointed out by Cozzi *et al.* [10], is routinely manipulated by malware authors to evade detection tools. We therefore hypothesise that our inability to identify any capabilities for these malicious files is a consequence of a malware implementation technique already documented in the literature.

[5]CAPA is not always able to specify why a capability analysis has failed. We classify these cases as *Other* in Fig. 4.

[6]We note that CAPA namespaces do not uniquely identify capabilities. For instance, as illustrated in some of the examples included in [16], the namespace `communication/socket/tcp` matches both the `connect TCP socket` and the `create TCP socket` capability. This implies that there is scope for an additional level of analysis, which, for simplicity, we did not perform. The information provided by the namespaces, in fact, is in most cases sufficient to complete the malware characterization within the scope of this research.

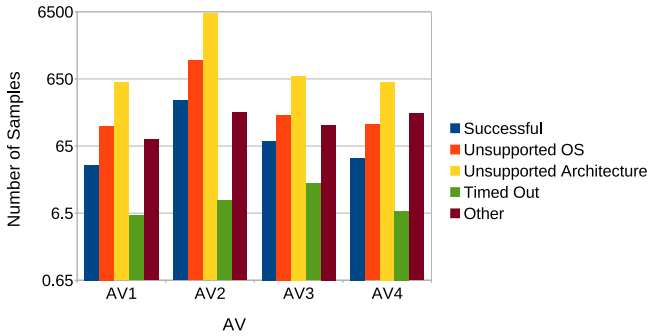


Fig. 4: Summary of the false negatives analysis performed with CAPA. The bar chart shows the number of both successful and unsuccessful analyses for each evaluated AV. Note the logarithmic scale on the vertical axis

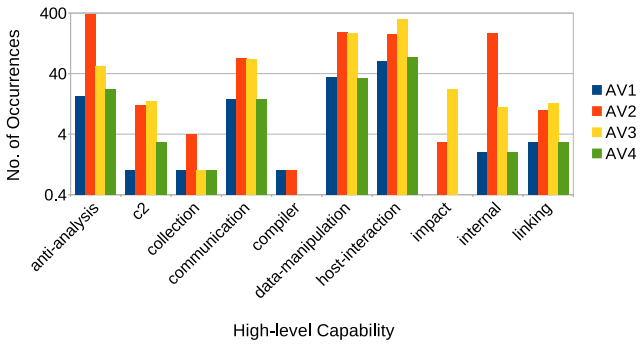


Fig. 5: High-level capabilities of the false negatives samples. Note the logarithmic scale on the vertical axis

TABLE III: CAPA namespaces (false negatives analysis)

Namespaces	Occurrences
anti-analysis/packer/upx	199
anti-analysis/anti-vm/vm-detection	181
host-interaction/file-system/write	108
data-manipulation/encoding/xor	103
host-interaction/file-system/read	96

True Positives Analysis. As illustrated in Fig. 6, the overall pattern of our results is very similar to the false negatives case. CAPA, in fact, is able to identify capabilities for 1,273 out of 42,962 samples, with the vast majority of the samples unprocessed due to either an unsupported architecture or an unsupported OS.

We record no change to the set of high-level capabilities in comparison with the previous analysis. However, as shown in Fig. 7, we observe that the two most detected are `anti-analysis` and `data-manipulation`, with a total number of occurrences equal to 7,509 and 4,102, respectively. It is therefore worth noting that the most detected high-level capability among the true positives is the second most detected in the false negatives case. By contrast, the analysis of the extracted namespaces, summarized in Table IV, reveals that the two most frequently detected capabilities are the same as

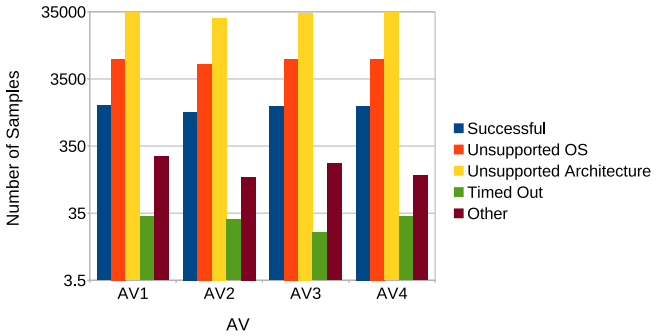


Fig. 6: Summary of the true positives analysis performed with CAPA. The bar chart shows the number of both successful and unsuccessful analyses for each evaluated AV. Note the logarithmic scale on the vertical axis

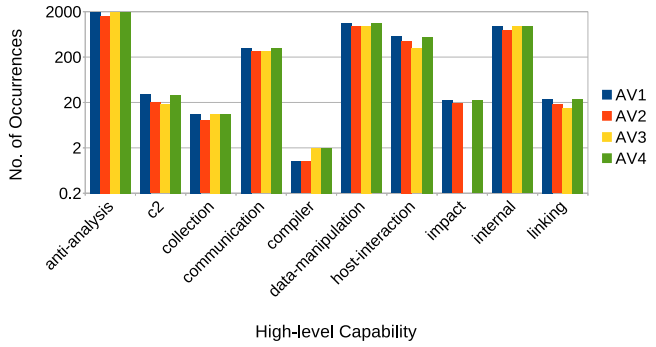


Fig. 7: High-level capabilities of the true positives samples. Note the logarithmic scale on the vertical axis

those identified for the false negatives samples (Table III).

To better interpret our results given the outcomes of the false negatives analysis, we also check all the true positives specimens with the Linux `file` command. The latter allows us to verify that: ❶ all the samples are recognized as ELF, and ❷ the tested AVs are capable of detecting malicious files targeting architectures different from 32 and 64-bit x86, e.g., ARM, MIPS and Renesas SH. This result is noteworthy, as it shows that, even though we evaluate desktop-oriented products, their engines are not platform-specific.

TABLE IV: CAPA namespaces (true positives analysis)

Namespaces	Occurrences
anti-analysis/packer/upx	3,601
anti-analysis/anti-vm/vm-detection	2,593
data-manipulation/encoding/xor	965
data-manipulation/encryption/rc4	891
anti-analysis/obfuscation/string/stackstring	793

Summary. To gain insight into the results obtained with our local tests (§ V-A), we conduct a malware capability analysis focused both on the false negatives, i.e., samples that have never been detected by the evaluated AVs, and on the true positives, i.e., samples that have been detected at least once. CAPA was able to identify capabilities only for 313 specimens out of 8,337 in the first case, and 1,273 out of 42,962 in the second. In addition to the aforementioned limitations of CAPA (§ IV-B), we hypothesise that the widespread practice of manipulating ELF headers to evade detection has further decreased the number of files successfully processed by CAPA.

Our experiments show that the most detected high-level capability among the false negatives is `host-interaction`, whilst it is `anti-analysis` for the true positives. Nevertheless, the analysis of the obtained CAPA namespaces, which can be described as fully-categorized capabilities, shows that there is no major difference between the two groups of malicious files, as the most detected namespaces are always `anti-analysis/packer/upx` and `anti-analysis/anti-vm/vm-detection`. As further discussed in § VI, malware authors therefore seem to be focused on effectively reusing established approaches rather than developing new ones.

C. Online Tests with VirusTotal

To measure the effectiveness of a larger number of AVs, which is essential to **RG1** and **RG2** (§ III), we perform a similar set of experiments using an online malware scanning service, VirusTotal. We present the results of their detection rates, and then we discuss the regression effects and the file-level analysis. Next, to reach **RG4**, we compare these results with those for the locally-installed AVs.

Detection Rates. During the first two tests (June - July 2020), 62 AVs were available in VirusTotal, whilst 59 AVs were available during the third test (April 2021). Therefore, in the following, when comparing the results of the last test with the others, we only consider the AVs available in all tests (58 AVs). By contrast, figures regarding exclusively the two tests conducted in 2020 take into account all the AVs available at that time.

As illustrated in Table V, the average detection rates of the AVs queried via VirusTotal API are significantly lower than those of the locally-installed ones, as the number of detected malware samples is on average 2,453 out of 4,000, which corresponds to a detection rate of 61.3%.

TABLE V: Average values of VirusTotal AVs performance indicators

Indicator	June 20	July 20	April 21	Avg
# of Detections	2,433.3	2,469.1	2,457.7	2,453.3
Detection Rate	60.8%	61.7%	61.4%	61.3%

A further breakdown of the detection rate figures for the first and the second test, which is summarized in Fig. 8, shows that nearly 50% of the VirusTotal AVs have a detection rate above 90%, whilst it is at most 30% for one third of them. The distribution of the detection rates was also recomputed to take into account the final test, but no major change was identified. This implies that the overall average detection

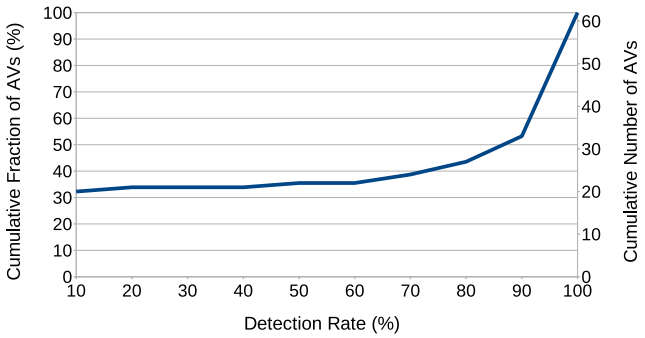


Fig. 8: VirusTotal AVs average detection rate distribution (tests performed in June and July 2020)

rate is significantly affected by a rather high number of AV engines that perform very poorly.

Regression Effects. Our analysis of the results from the first two tests identified 13 AVs affected by regression, as the total amount of files flagged as malicious during the second test is lower in comparison to the first. Interestingly, when we compare the data from the second and the third test, we find 24 regression-affected AVs, which include three of the AVs tested locally (namely, AV1, AV2 and AV4), as the number of specimens detected by these anti-malware solutions decreases by 36, 18 and 4, respectively.

File-level Analysis. We also evaluate the performance of VirusTotal AVs on a per-file basis. As illustrated in Table VI, even though our study began more than two months after the release of the used malware repository, all tests report some malicious files not detected by any AV. It is also worth noting that their number increases by two in the second test with respect to the first.

Furthermore, less than 50% of the 4,000 submitted samples are detected by a higher number of AVs during the second test in comparison to the first. As detailed in Table VII, in fact, 1,485 malicious files are flagged by the same total amount of AVs, whilst 774 samples are detected by a lower number of anti-malware solutions. We also observe that the amount of specimens that are flagged by a higher number of AVs during the third test in comparison to the second was even lower. Specifically, the number of samples with unchanged and fewer detections increased to 1,714 and 1,317, respectively (Table VII).

TABLE VI: Summary of file-level evaluation parameters

Parameter	June 20	July 20	April 21
Min # of Detections	0	0	0
Average # of Detections	37.3	37.9	35.6
Max # of Detections	44	44	41
# of Samples Not Detected	18	20	20

TABLE VII: Summary of recorded samples with unchanged and decreasing number of detections

Parameter	June 20 → July 20	July 20 → April 21
Unchanged # of Detections	1,485	1,714
Decreasing # of Detections	774	1,317

Validation of VirusTotal AVs with Locally-installed AVs. To compare the online with the local version of the AVs, we recomputed the detection rates for the four locally-installed AVs considering only the 4,000 malware samples tested with VirusTotal. As detailed in Table VIII, the two sets of results show only small discrepancies, with the largest being 2.2% (AV1). Our figures are in line with those obtained by Botacin *et al.* [14], who made a similar comparison, albeit limited to three AVs. Differently from those previously published results though, ours are Linux-specific.

Summary. To evaluate the effectiveness of a larger number of AVs, we conduct three tests by submitting 4,000 specimens of Linux-based malware to the online scanning service VirusTotal. Although there is no guarantee that the latter uses Linux-specific AV engines (§ III), we note that one third have a detection rate in the range between 0 and 30%. Consequently, the online AVs exhibit an overall average detection rate of approximately 60%, which is significantly lower than expected. Furthermore, the tests carried out with VirusTotal show additional evidence of regression, which affects 24 out of 58 online AV products.

TABLE VIII: Average detection rate of locally-installed AVs compared to VirusTotal results (DR=Detection Rate, VT=VirusTotal)

AV	Avg DR (%) (Local)	Avg DR (%) (VirusTotal)	Delta (%) (VT Vs Local)
AV1	98.5	96.3	-2.2
AV2	82.8	81.6	-1.2
AV3	97.1	96.5	-0.6
AV4	98.1	98.2	0.1

VI. Discussion and Recommendations

We now provide additional insights into the results as well as recommendations to AV vendors and for future research.

Linux AVs Evolution. In our evaluation, the average of all the detection rates recorded with the locally-installed AVs when using the entire VirusShare repository is 94% (§ V-A). To understand how Linux AV software is evolving, it is interesting to compare this with the 2015 online report by AV-Test [12], where the tested security solutions exhibited an average detection rate against Linux-based malware equal to 80.1%. Even though this result was obtained with a much lower number of malware specimens (i.e., 900) in comparison to ours, and with a mixture of desktop and server-oriented

AVs, it suggests that, in general, Linux-compatible anti-malware solutions have improved over the years. However, our study additionally reveals that the detection rates increase very marginally over time. Consequently, we believe that our results identify an area of improvement for Linux AV vendors, as signature databases do not appear to be well maintained and distributed, thus creating attack opportunities.

Linux vs Windows AVs Comparison. We also considered statistics recently published for Windows AVs. The report released in February 2021 by AV-Test [32] indicates that the average detection rate is 98.8%. However, the details of the test methodology [33] suggest that the AV-Test evaluation relies on malware samples collected over a period of four weeks prior to the test and does not include multiple scans. Similarly, the March 2021 report published by AV-Comparatives [31] shows that the average detection rate is 93.7%. Note that AV-Comparatives's evaluation was done with a larger number of specimens (i.e., 10,013), but was not based on a series of tests.

Considering the different methodologies, and that Linux-based malware, despite its growing complexity, is not as sophisticated as its Windows counterpart [9], [10], we believe that comparing these Windows AVs results with ours is not scientifically sound. Our evaluation though highlights that there is a significant proportion of underperforming VirusTotal AVs. This implies that their engines should be improved to increase the detection rates of Linux-based malware.

Regression Effects. Differently from previous research [14], [15], we provide Linux-specific figures that illustrate that regression affects several Linux AVs, albeit limited to a generally low number of files. It should also be observed that this phenomenon is much more pronounced over periods of time lasting several months, rather than weeks. Therefore, Linux AV developers should consider improving their testing practices and methodologies to ensure that new signatures and updated heuristics do not have a detrimental impact.

Malware Capability Analysis. While the most detected high-level capabilities differ for false negatives and true positives (§ V-B), three out of the five namespaces with the highest number of occurrences are the same for both groups of specimens, and their ranking orders differ in one case only[7]. Even though our capability analysis was successfully completed only for a minority of the samples that we set out to inspect, this lack of distinct patterns suggests that Linux malware authors are fine-tuning well-known techniques instead of identifying novel ones. However, our results might have been affected by the limited number of ELF rules used by CAPA when this research was conducted. Therefore, future Linux malware characterization analyses based on this tool could reveal more complex patterns.

[7] `data-manipulation/encoding/xor`

Scan Execution Times. In our tests, three out of four locally-installed AVs exhibited reasonable scan execution times. However, these substantially increased during the tests conducted in April 2021, which we hypothesise is due to the larger sizes of the virus signature databases. Scan execution times affect the usability of AV software and should therefore be regularly monitored by AV vendors.

VII. Related Work

Linux Security and Malware. Yaswinski *et al.* [34] suggest methods for protecting a Linux system from different threats. Installing an AV is considered an essential step along with downloading software from official repositories and scanning programs before execution via a compatibility layer, e.g., Wine. The latter case is relevant as malware that can only be successfully run in Linux via Wine has been observed [35]. However, unlike our work, this work only mentions one widely used anti-malware solution, i.e., ClamAV [36], and do not analyse its actual effectiveness. Cozzi *et al.* [10] identify the challenges and the most significant behaviours of Linux-based malware targeting IoT devices. Their analysis considers multiple CPU architectures and provides important information on evasion techniques. However, they do not evaluate the effectiveness of Linux AV solutions, the main objective of our work.

AV Comparative Studies. Although entirely focused on Windows, the work by Zarghoon *et al.* [18] proves that figures summarizing the detection rates of AV software, e.g. provided by vendors or specialized websites, can be misleading as they frequently refer to old malware samples. The authors tested their malware samples with the online scanning service NoDistribute [37] as well as manually in virtualized environments. The results showed a substantial gap between the detection rates reported on AV-Test [38] and those from NoDistribute.

As illustrated by Botacin *et al.* [14], evaluating AVs comprehensively is a complex task because these are frequently affected by regression. To take this into account along with delays in updating the signature databases, the authors propose six AV evaluation metrics. However, differently from this work, their study is not focused on Linux-compatible AVs. Zhu *et al.* [15] analyse the behaviour over time of VirusTotal AVs, concentrating their attention on PE and APK files. After collecting daily snapshots of malware labels for one year, the authors show that these are characterised by short-term and long-term fluctuations. In addition, the performance metrics of high-reputation engines are evaluated with a set of obfuscated files. The obtained results reveal a higher than expected percentage of misidentified samples, but the work does not include any Linux-specific analysis.

VIII. Conclusion

In this paper, we present a comprehensive evaluation of the effectiveness of AVs currently available for Linux. This was achieved by performing multiple tests over a period of ten

months with local installations and an online malware scanning service. Furthermore, we conduct a capability analysis of a subset of the used malicious samples with a rule-based tool.

The results of our study suggest that AV vendors should thoroughly review the mechanisms adopted to update and distribute signature databases. In addition, AV testing methodologies should take into account that new signatures and updated heuristics can cause regression effects, especially over periods of time lasting several months. Finally, our capability analysis shows the absence of major differences between false negatives and true positives in terms of their abilities to perform specific tasks. This suggests that malware authors are attempting to further specialize well-known techniques rather than developing new ones.

References

[1] A. Goretsky. (2015, Jan.) Do you really need antivirus software for linux desktops? [Online]. Available: https://www.welivesecurity.com/2015/01/13/really-need-antivirus-software-linux-desktops/

[2] M. Casserly. (2018, Jun.) Does linux need antivirus? [Online]. Available: https://www.techadvisor.co.uk/feature/software/does-linux-need-antivirus-3678945/

[3] J. Koret and E. Bachaalany, *The Antivirus Hacker's Handbook*. Indianapolis, IN, USA: John Wiley & Sons, 2015.

[4] T. Micro. (2021, Aug.) Linux threat report 2021 1h: Linux threats in the cloud and security recommendations. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/linux-threat-report-2021-1h-linux-threats-in-the-cloud-and-security-recommendations

[5] T. Panker and N. Nissim, "Leveraging malicious behavior traces from volatile memory using machine learning methods for trusted unknown malware detection in linux cloud environments," *Knowledge-Based Systems*, vol. 226, 2021.

[6] B. L. Labs. (2021, Sep.) No longer just theory: Black lotus labs uncovers linux executables deployed as stealth windows loaders. [Online]. Available: https://blog.lumen.com/no-longer-just-theory-black-lotus-labs-uncovers-linux-executables-deployed-as-stealth-windows-loaders/

[7] V. M. Ionescu, M. Patel, and D. Hindocha, "Alternatives for running linux applications in windows," in *2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2019, pp. 1–4.

[8] P. Kochberger, A. Tauber, and S. Schrittwieser, "Assessment of the transparency of the windows subsystem for linux (wsl)," in *2019 International Conference on Software Security and Assurance (ICSSA)*, 2019, pp. 60–69.

[9] J. Carrillo-Mondéjar, J. Martínez, and G. Suarez-Tangil, "Characterizing linux-based malware: Findings and recent trends," *Future Generation Computer Systems*, vol. 110, pp. 267–281, 2020.

[10] E. Cozzi, M. Graziano, Y. Fratantonio, and D. Balzarotti, "Understanding linux malware," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 161–175.

[11] M. Maganu. (2022, Jan.) Linux-targeted malware increases by 35% in 2021: Xorddos, mirai and mozi most prevalent. [Online]. Available: https://www.crowdstrike.com/blog/linux-targeted-malware-increased-by-35-percent-in-2021/

[12] AV-Test. (2015, Oct.) Linux: 16 security packages against windows and linux malware put to the test. [Online]. Available: https://www.av-test.org/en/news/linux-16-security-packages-against-windows-and-linux-malware-put-to-the-test/

[13] M. Botacin, F. Ceschin, R. Sun, D. Oliveira, and A. Grégio, "Challenges and pitfalls in malware research," *Computers & Security*, vol. 106, p. 102287, 2021.

[14] M. Botacin, F. Ceschin, P. Geus, and A. Grégio, "We need to talk about antiviruses: challenges & pitfalls of av evaluations," *Computers & Security*, vol. 95, 2020.

[15] S. Zhu, J. Shi, L. Yang, B. Qin, Z. Zhang, L. Song, and G. Wang, "Measuring and modeling the label dynamics of online Anti-Malware engines," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 2361–2378.

[16] W. Ballenthin and M. Raabe. (2020, Jul.) capa: Automatically identify malware capabilities. [Online]. Available: https://www.mandiant.com/resources/capa-automatically-identify-malware-capabilities

[17] F. A. Garba, K. I. Kunya, S. A. Ibrahim, A. B. Isa, K. M. Muhammad, and N. N. Wali, "Evaluating the state of the art antivirus evasion tools on windows and android platform," in *2019 2nd International Conference of the IEEE Nigeria Computer Chapter (NigeriaComputConf)*, 2019, pp. 1–4.

[18] A. Zarghoon, I. Awan, J. P. Disso, and R. Dennis, "Evaluation of av systems against modern malware," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2017, pp. 269–273.

[19] K. A. Asmitha and P. Vinod, "A machine learning approach for linux malware detection," in *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, 2014, pp. 825–830.

[20] S. Hou, A. Saas, L. Chen, and Y. Ye, "Deep4maldroid: A deep learning framework for android malware detection based on linux kernel system call graphs," in *2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*, 2016, pp. 104–111.

[21] S. Y. Yerima, S. Sezer, and I. Muttik, "High accuracy android malware detection using ensemble learning," *IET Information Security*, vol. 9, no. 6, pp. 313–320(7), Nov. 2015.

[22] H. S. Anderson, A. Kharkar, B. Filar, D. Evans, and P. Roth, "Learning to evade static pe machine learning malware models via reinforcement learning," 2018. [Online]. Available: https://arxiv.org/abs/1801.08917

[23] L. Demetrio, S. E. Coull, B. Biggio, G. Lagorio, A. Armando, and F. Roli, "Adversarial exemples: A survey and experimental evaluation of practical attacks on machine learning for windows malware detection," *ACM Trans. Priv. Secur.*, vol. 24, no. 4, Sep. 2021.

[24] Dr.Web. (2020, May) Dr.web for linux version 11.1 user manual. [Online]. Available: https://www.drweb.com/

[25] VirusTotal. (2020, Jun.) Analyze suspicious files and urls to detect types of malware, automatically share them with the security community. [Online]. Available: https://www.virustotal.com/gui/home/upload

[26] W. Ballenthin, M. Raabe, M. Hunhoff, and A. M. M. Gomez. (2021, Sep.) Elfant in the room – capa v3. [Online]. Available: https://www.fireeye.com/blog/threat-research/2021/09/elfant-in-the-room-capa-v3.html

[27] J. Kennedy. (2021, Sep.) Teaching capa new tricks: Analyzing capabilities in pe and elf files. [Online]. Available: https://www.intezer.com/blog/malware-analysis/analyzing-capabilities-in-PE-and-ELF-files/

[28] VirusTotal. (2020, Jun.) Av product on virustotal detects a file and its equivalent commercial version does not. [Online]. Available: https://support.virustotal.com/hc/en-us/articles/115002122285-AV-product-on-VirusTotal-detects-a-file-and-its-equivalent-commercial-version-does-not

[29] ——. (2020, Jun.) How it works. [Online]. Available: https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works

[30] VirusShare. (2020, Jun.) Repository of malware samples. [Online]. Available: https://virusshare.com/

[31] AV-Comparatives. (2021, Mar.) Malware protection test. [Online]. Available: https://www.av-comparatives.org/wp-content/uploads/2021/04/avc_mpt_202103.pdf

[32] AV-Test. (2021, Feb.) The best windows antivirus software for home users. [Online]. Available: https://www.av-test.org/en/antivirus/home-windows/

[33] ——. (2021, Feb.) Test modules under windows. [Online]. Available: https://www.av-test.org/en/about-the-institute/test-procedures/test-modules-under-windows-protection/

[34] M. R. Yaswinski, M. M. Chowdhury, and M. Jochen, "Linux security: A survey," in *2019 IEEE International Conference on Electro Information Technology (EIT)*, 2019, pp. 357–362.

[35] R. Duncan and Z. C. Schreuders, "Security implications of running windows software on a linux system using wine: a malware analysis study," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 39–60, Apr. 2019.

[36] ClamAV. (2020, Jun.) Open source antivirus engine. [Online]. Available: https://www.clamav.net/

[37] NoDistribute. (2020, Jun.) Online virus scanner without result distribution. [Online]. Available: https://nodistribute.com/

[38] AV-Test. (2020, Jun.) The independent it-security institute. [Online]. Available: https://www.av-test.org/en/