A Comparative Analysis of VirusTotal and Desktop Antivirus Detection Capabilities

Conference Paper · July 2022

DOI: 10.1109/IISA56318.2022.9904382

5 authors, including:

Christoforos Ntantogian
Ionian University
61 PUBLICATIONS **673** CITATIONS
SEE PROFILE

Stylianos Karagiannis
Ionian University
26 PUBLICATIONS **128** CITATIONS
SEE PROFILE

Emmanouil Magkos
Ionian University
85 PUBLICATIONS **967** CITATIONS
SEE PROFILE

Vassilios S. Verykios
Hellenic Open University
287 PUBLICATIONS **8,260** CITATIONS
SEE PROFILE

A Comparative Analysis of VirusTotal and Desktop Antivirus Detection Capabilities

Christian Leka
Department of Informatics
Ionian University
Corfu, Greece
p17leka@ionio.gr

Christoforos Ntantogian
Department of Informatics
Ionian University
Corfu, Greece
dadoyan@ionio.gr

Stylianos Karagiannis
Department of Informatics
Ionian University
Corfu, Greece
skaragiannis@ionio.gr

Emmanouil Magkos
Department of Informatics
Ionian University
Corfu, Greece
emagos@ionio.gr

Vassilios S. Verykios
School of Science and Technology
Hellenic Open University
Patras, Greece
verykios@eap.gr

2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA) | 978-1-6654-6390-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/IISA56318.2022.9904382

***Abstract*— VirusTotal has been widely used and being adopted by researchers mainly for the classification of files as malicious or not. Unfortunately, it is not well understood how reliable the results from the antivirus engines on VirusTotal are, especially compared to their desktop counterparts. In this paper, we shed light on the blackbox testing functionality of VirusTotal by evaluating the detection results of VirusTotal antivirus engines and their equivalent desktop versions. Based on our results, we arrive to the conclusion that there are discrepancies between the engines on VirusTotal and the desktop engines. In general, the malware detection rate of the engines on VirusTotal is lower compared to desktop products. This is mainly attributed to the fact that VirusTotal engines do not take advantage of cloud-based detection deteriorating their performance.**

Keywords—VirusTotal, antivirus evasion, malware detection

I. INTRODUCTION

Malware, a portmanteau for malicious software, is designed to cause harmful consequences to computer systems varying from minor inconveniences such as slowing computers and popup ads to more severe repercussions such as unauthorized access and data theft and loss. Because of the alarming ramifications that malware can lead to, it is imperative to detect malware as fast as possible. Nowadays, several online scanning platforms exist that are able to scan files to detect malware or URLs for detecting phishing and malware hosts [1][2][3][4][5]. These platforms scan files and URLs using multiple vendors and present the combined output of different products. VirusTotal is perhaps the most popular one out of these scanning platforms. In VirusTotal, users can upload files to be checked by over 70 antivirus (AV) vendors, in order to determine if the files are malicious or not [6]. After scanning the file, the platform provides a total detection result from all the employed detection engines. As such, the platform does not label applications as malicious and benign; instead, it is up to the user to decide upon strategies to interpret the provided information.

According to [27], VirusTotal has been extensively used by researchers in more than 115 published papers[1] published between 2009-2018, while this list is constantly growing with new papers [29], [32], [33], [34]. Most of these works utilize VirusTotal to label datasets as malicious using either a threshold (i.e., if *t* or more engines return a "malicious" label, then the file is labeled as malicious) or by considering the results of high-reputation engines. A small number of previous works use VirusTotal's results to build their own system or as their comparison baselines. VirusTotal also includes an API for integration with external services [8].

Despite the heavy usage of VirusTotal by previous works, there is no clear understanding of how this platform delivers its results. Since the platform operates basically as a blackbox, the produced results may be questionable. Even some previous works acknowledge this fact such as [31], which mentions that "AV vendors only deployed light-weighted engines on VirusTotal, with most of them being signature-based in order to achieve instant detection". Moreover, as stated by [28], "VirusTotal runs stripped-down engine versions that do not always reflect the best detection capability of an AV vendor". After all, VirusTotal itself reports, "AV solutions on VirusTotal may differ from their public commercial versions" [7]. That being said, it is crucial to determine how big this difference is, why it exists, and how reliable VirusTotal is in general, since it affects published papers and, more critically will affect future research.

In this paper, we evaluate the reliability of VirusTotal by examining discrepancies in the detection results between VirusTotal and desktop engines. We focus exclusively on malware analysis since URL scanning has been extensively examined in the past [28]. More specifically, our main objectives are:

1. Pinpoint discrepancies between VirusTotal and desktop engines.
2. Identify how large those discrepancies are and if they could majorly affect the reliability of the VirusTotal platform.
3. Analyze the reasons behind these discrepancies.

In order to answer the research questions posed above, first we created our malware dataset using a set of 16 open source AV evasion tools that obfuscated 2 Metasploit payloads resulting in 50 malicious files in total. We also selected the 12 most popular AV products to measure their detection capabilities. To perform the experiments, each malicious file was downloaded into a virtual machine. If the malware was not detected at download, then it was scanned. If it was not detected by scanning either, then it was executed for runtime detection. For each malicious file, it was recorded whether it was detected or not by the AV products and the detection phase (download, scanning or execution). We have considered two different scenarios. In the first scenario the deployed

[1] https://sfzhu93.github.io/projects/vt/paper-list.html

978-1-6654-6390-4/22/$31.00 ©2022 IEEE

virtual machines did not have access to the Internet to avoid AVs use their cloud-based detection capabilities, while in the second scenario, the virtual machines had Internet access. At the same time, we uploaded the 50 malicious files of our dataset into VirusTotal to obtain the related results from the platform. The comparison of the detection results between VirusTotal and desktop AVs revealed that in general VirusTotal exhibited lower malware detection rates for most AV engines compared to their desktop counterparts in both scenarios. This may be attributed to the fact that AV engines on VirusTotal do not include cloud-based detection, or they are shipped with different settings compared to their desktop counterparts. To the best of knowledge, this is the first work that comprehensively studies the inconsistencies of VirusTotal and desktop AV products.

The rest of the paper unfolds as follows. Section II presents the related work and the contradictory results obtained. Section III describes the methodology adopted for the carried-out experiments. Section IV presents the results while Section V lists a set of observations. Finally, Section VI concludes the article.

II. Related work

As we mentioned previously, VirusTotal has been extensively used by researchers to annotate sample datasets as malicious (executable .exe or Android APK files) or for system evaluation. However, very few papers have examined the reliability of VirusTotal in terms of its malware detection results. For example, the work in [30] mentions but does not verify that AV vendors change the settings of their products specifically for VirusTotal compared to their commercial versions.

The closest works with ours are [27] and [28] that compare the detection capabilities of desktop and VirusTotal engines. More specifically, the authors in [28] set up malicious phishing websites in order to test the reliability of the engines on VirusTotal in regard to detection of phishing websites. The results obtained were that VirusTotal had lower detection compared to the AV vendors' own scan APIs. This may be attributed to the fact that third-party vendors do not always give VirusTotal the scan permission or the most updated blacklists. The work in [28] did not analyze malicious files and focused only on phishing websites.

Moreover, the main idea of [27] was to evaluate the characteristics and usage of VirusTotal service with regards to malware classification. It arrived at several conclusions including: i) Threshold-based classification of malware is beneficial and should be utilized by researchers as virus engines exhibit large discrepancies; ii) The impact of a poorly chosen threshold can have significant impact on the malware classification; iii) Certain groups of engines are strongly correlated and should not be treated independently; iv) Certain engines fail to perform in-depth analysis on submitted files and can easily produce false positives. Apart from the above insightful results, the work in [27] presented also a comparison of VirusTotal engines with desktop engines. Their conclusion was that: "Inconsistency exists between the desktop version and the online version (at VirusTotal) for all engines. Surprisingly, for most of the vendors, their VirusTotal engines are able to detect more malware than their desktop versions." This outcome was contradictory to the results of [28]. The authors in [27] justify their results by arguing that "desktop versions are more conservative to keep a small number of false alarms". This argument seems valid, but the same argument holds also for the detection of phishing websites (i.e., URL scanning should trigger only the minimum number of false alarms). However, as we mentioned previously, the work by [28] showed that desktop AVs were able to identify more phishing websites from VirusTotal. We also noticed that [27] utilized only 2 commercial obfuscators named Code Virtualizer and Themida developed by the same company to generate their dataset. Apart from the limited obfuscators utilized, it also seems that the former is a small subset of the latter with regards to its evasion capabilities. Thus, the results from these two obfuscators may not be independent.

Another recent research work [29] examines VirusTotal from a different point of view. In particular, over six months of experimentation, the authors investigated the VirusTotal's learning behavior. To this end, a custom packer was developed and constantly updated with new features. Then, the response time in terms of detecting the new features was recorded. The authors concluded that some specific AVs were very quick to detect modifications in the packer. At the end of the experimental run, only 4 out of 56 VirusTotal AV engines were not able to detect the packer.

In general, due to the very limited number of published papers regarding the reliability of VirusTotal for malware detection and the contradicting results of existing works, there is a need for further investigation into this topic. To this end, this paper will explore and compare in a comprehensive manner the VirusTotal and desktop engines by utilizing several evasion techniques and tools.

III. Methodology

The main focus of this paper is to understand how effective the AV engines on VirusTotal are compared to their counterpart desktop versions. The methodology to achieve the above was to generate malware samples that will first be tested on desktop AV products and later uploaded to VirusTotal and compare the results. The first step to accomplish this, was to create our malware dataset. To this end, we used two popular shellcodes from the open-source Metasploit framework: i) Reverse TCP Meterpreter and ii) Reverse TCP shell. Next, we gathered 16 cutting edge open-source AV evasion tools from GitHub. Using these tools, we were able to generate 50 obfuscated malware samples to bypass AV products. This in turn helps simulate a more realistic environment since attackers often use AV evasion techniques on their malware in order to bypass AV products. Please note that some tools are able to generate more than 1 sample (because they employ several evasion techniques). Also, since all of our files are malicious, we will not examine false positives between desktop engines and VirusTotal engines.

Table 1 presents the AV evasion tools along with a description of the generated malicious files, which were 50 in total. Note here that all tools generated an executable file in the form of a Portable Executable (PE), except for the tool named Unicorn that generated a .bat file and the tool named Chimera that generated PowerShell (.ps1) files. All generated files run under the Windows operating system. Most of the samples are generated for 64-bit architecture, which in theory should provide a better evasion rate. When an evasion tool did not support creating x64 bit executables, x86 bit executables were created instead. The evasion tools use a variety of AV techniques to bypass signature based detection, such as code

obfuscation, encoding, encryption, signing the executable and many more. Additionally, the chosen evasion tools include techniques to bypass heuristic or behavioral based detection such as NTDLL API unhooking, sandbox evasion techniques (e.g., sleep, heuristics to detect whether the executable is executed in sandbox environment) and many others.

According to [27], out of all the VirusTotal vendors, 36 vendors also offer desktop versions of their products. Out of those 36, we have selected the 12 most popular, in order to conduct our experiments and carry out our comparisons. Table 2 presents a list of the vendors that were selected with their respective product and its corresponding version. To conduct our experiments, we set up 12 virtual machines (VM), each with a fully patched Windows 10 64-bit operating system. On each VM, we installed the latest version of each desktop AV at that time. We fully updated the signatures of each product and left everything else unchanged (meaning that each AV software was used with its default settings).

Each malicious file was first downloaded into the VM from a remote server in our lab. If the malware was not detected at download, then it was scanned. If it was not detected by scanning either, then it was executed for runtime detection. For each malicious file, it was recorded whether it was detected or not by the AV products and the detection phase (download, scanning or execution). A file was considered that bypassed detection only if a remote shell connection was established (based on our Metasploit shellcodes) and simple commands could be executed. After our experiments with the desktop versions were completed, the malicious files were uploaded to VirusTotal. As we mentioned above, only the results from our 12 chosen engines were recorded in VirusTotal. Since we utilized 50 malicious files on 12 AVs, the total number of samples utilized for detection by all AVs were 600 in total.

We have considered two different scenarios. In the first scenario, we derived detection results with the VMs being disconnected from the internet. This was required because almost every AV vendor nowadays uses the cloud to some extent as part of its malware detection process. There is no public information about how each vendor uses the cloud, but we could not guarantee that some vendors do not upload files to VirusTotal and then use the results as part of their malware labeling process. This would possibly taint the results of our research, since some of our PE files could have been uploaded to VirusTotal if internet was enabled. So all of our installed AV products operated without internet.

In the second scenario, detection results were calculated with internet connectivity but only for 3 specific AV vendors namely Microsoft Defender, Bitdefender, and Avira. The reason that we selected these 3 specific AV vendors is that they utilize heavily the cloud to improve their detection performance even for their free versions. This information was provided by the documentation of these 3 AV products.

Table 1: AV Evasion Tools and Description of each Generated File

AV Tools	Malicious File
Avet [11]	Meterpreter with XOR encoding and sleep x64
	Shell with XOR encoding and Fibonacci computations x64
	Meterpreter with Avet encoding and Fibonacci computations x64
	Shell with Avet encoding and sleep x64
Phantom-Evasion [12]	Stager (signed, fake process path, unhook NTDLL, junkcode) x64
	Meterpreter with double key XOR encryption (signed, fake process path, unhook NTDLL, junkcode) x64
	Shell with double key XOR Vigenère encryption (signed, fake process path, unhook NTDLL, junkcode) x64
Shellter [13]	Meterpreter shellcode injection with mIRC version 7.66 x86
	Meterpreter shellcode injection with PuTTY version 0.76 x86
	Meterpreter manual mode with WinRAR version 6.02 x86
	Shell shellcode injection with mIRC version 7.66 x86
	Shell shellcode injection with PuTTY version 0.76 x86
	Shell shellcode injection with WinRAR version 6.02 x86
	Shell shellcode injection with UniKey version 4.3 x86
Veil [14]	C Meterpreter x86
	Python Meterpreter with sandbox evasion x86
	Ruby Meterpreter x86
	C# shell with Arya crypter x86
	Go shell x86
	Ruby shell x86
TheFatRat [15]	Meterpreter with Fudwin 1.0 x86
	Meterpreter with UPX and Fudwin 1.0 x86
	Meterpreter Powerstager with Fudwin 1.0 x86
	Meterpreter C# and PowerShell with PwnWinds 1.5 x86
	Meterpreter Apache PowerShell with PwnWinds 1.5 x86
	Meterpreter with MSFvenom x86
	Shell with MSFvenom x86
WinPayloads [16]	Meterpreter with sandbox evasion x86
	Shell stageless with sandbox evasion x86
Xeexe [17]	Meterpreter stageless x64
	Shell stageless x64
Unicorn [18]	Meterpreter with PowerShell x86
	Shell with PowerShell x86
Chimera [19]	Invoke-PowerShellTcp.ps1 using custom options
	PowerShell TCP one line shell using custom options
	Generic PowerShell shell using custom options
MsfMania [20]	Meterpreter with sleep (signed, junkcode, XOR encoding) x64
	Shell with sleep (signed, junkcode, XOR encoding) x64
PEzor [21]	Meterpreter with SGN encoding, syscalls, unhooking x64
	Shell with SGN encoding, syscalls, unhooking x64
RapidPayload [22]	Meterpreter with Hyperion (signed) x64
	Shell with Hyperion (signed) x64
FourEye [23]	Meterpreter with DarkArmour x64
	Shell with DarkArmour x64
	Meterpreter with XOR encryption using fiber x64
	Shell with ROT13 encryption using Queueuserapc x64
Crypter [24]	Python Meterpreter (compiled using PyInstaller)
Onelinepy [25]	Python Meterpreter with base64 encoding (compiled using PyInstaller)
Hercules [26]	Meterpreter x86
	Shell stageless x86

Table 2: Selected AV Vendors, Products and Desktop Versions

AV Vendors	Products	Desktop Versions
Microsoft	Windows Defender	4.18.2107.4
Kaspersky	Kaspersky AV	21.3.10.391f
Bitdefender	Bitdefender AV Plus	25.0.26.89
ESET	ESET NOD 32 AV	14.2.24.0
McAfee	McAfee Total Protection	16.0
Avast	Avast Free AV	21.7.2481
AVG	AVG AV Free	21.7.3196
Avira	Avira AV Pro	15.0.2107.2107
Vipre	Vipre AV Plus	11.0.6.22
Malwarebytes	Malwarebytes Premium	4.4.5.130
K7	K7 AV Premium	16.0.0672
Panda	Panda Free AV	21.00.00

IV. Results

In this section, we present the results of our research. Regarding the first scenario of the experiments (i.e., VMs without Internet), the first important finding is that the overall detection of malware on VirusTotal is lower compared to the overall detection of malware on desktop products. Please note that the overall detection is calculated by adding the number of detected malware samples from each AV.

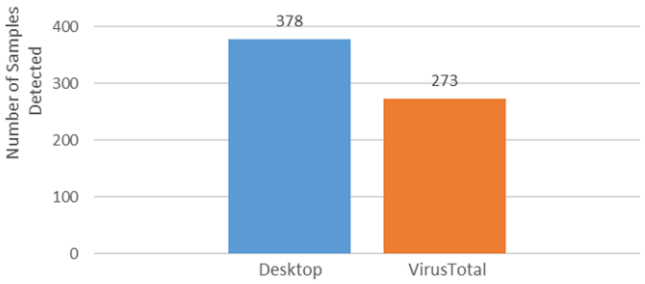

Fig. 1: First Scenario - Overall Malware Detection on Desktop and VirusTotal

More specifically, from Fig. 1, we can observe that the AV engines on VirusTotal (the 12 selected AV engines), managed to detect in total 273 out of 600 compared to the 378 that the equivalent desktop engines detected. That being said, we have to take into account that the desktop products did not have their cloud capabilities enabled which would likely result in higher detection rates for the desktop products. This will be examined below in the second scenario of our experiments.

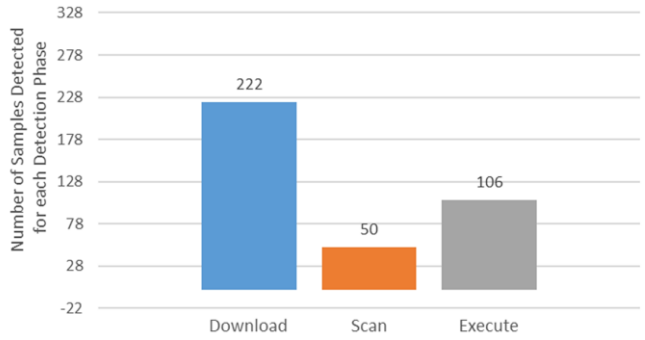

Fig. 2: First Scenario - Malware Detection Phase on Desktop

Regarding the detection phase of the desktop AVs, out of the 378 detected samples, 222 were detected at download, 50 at scanning and 106 at execution (see Fig. 2). When malware is detected at download or scanning, we can assume that it was detected with signature based detection techniques. On the other hand, when malware is detected at runtime, it is probably due to behavioral or heuristic based detection techniques. All in all, roughly 1/5 of the samples were detected by behavioral or heuristic means. We cannot reproduce the same results for VirusTotal, as the latter does not provide any information regarding the detection method.

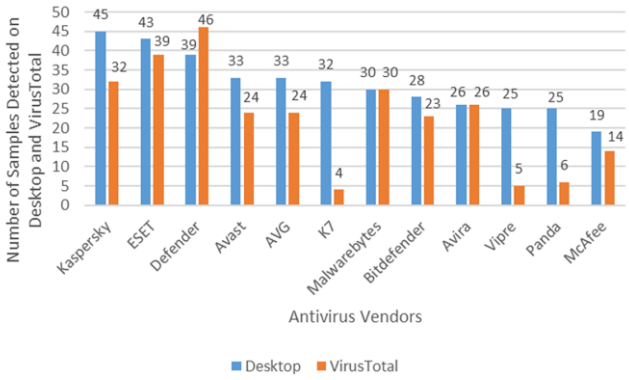

Fig. 3: First Scenario - Malware Detection for each Engine on Desktop and VirusTotal

Next, we elaborate on the detection rate of each AV individually both on desktop and VirusTotal. Fig. 3 demonstrates the malicious files each AV engine managed to detect both on desktop and on VirusTotal (the number of files detected out of the 50 files in total). It is important to mention here that the evaluation of the detection capabilities between the AV engines is out of scope of the paper. Instead, the goal here is to identify and demonstrate the differences between the engines on VirusTotal and desktop engines.

As seen in Fig. 3, a total number of 10 out of 12 vendors demonstrate inconsistencies between their VirusTotal and desktop AV engines. From those 10 vendors that demonstrated irregularities, only 1 product, Windows Defender, had higher detection rate on VirusTotal, with all the other vendors showing lower detection rates on VirusTotal. The only exception was Avira and Malwarebytes that had the same detection rate on both desktop and VirusTotal. ESET had the lowest reduction in the detection rate on VirusTotal, since its VirusTotal engine detected only four files less than its desktop counterpart. On the other hand, K7 had the highest reduction in the detection rate, since its VirusTotal engine detected a hopping 28 samples less compared to its desktop counterpart. Along with K7, Vipre and Panda complete the top three engines with the highest irregularities. These three engines while detecting around half of the samples on desktop, they managed to detect 10% or less of the overall samples on VirusTotal.

We continue by pinpointing the possible reasons of the presented irregularities between online (VirusTotal) and desktop engines. A careful examination of the results reveals that K7 seems to primarily use behavioral or heuristic based detection, since it detected most samples at execution phase. In particular, out of the 50 malicious files, K7 detected 32, with 28 of those being detected at execution (through behavioral or heuristic based detection). The rest of the detected malicious files (i.e., four malicious files) were

detected at download. These four malicious files were the only ones that were detected by K7 in VirusTotal. This behavior was exhibited by almost every desktop AV engine that was tested. That is, if a malicious file was detected at execution, it was not detected by the corresponding VirusTotal AV engine, while every file detected at download or scanning by the desktop engines (through signatures), was detected by the corresponding VirusTotal engines. The above corroborates also the fact that there were no irregularities in Avira and Malwarebytes, simply because these two AVs do not employ behavioral or heuristic based detection (at least they do not without their cloud capabilities).

Another interesting outcome was that Windows Defender was the only AV to have increased detection on VirusTotal. We speculate that Windows Defender relies heavily on cloud detection nowadays and this is the reason why its VirusTotal version had higher detection rates. We validate this assumption in our second scenario of experiments in which we have three AVs connected to the internet, Windows Defender, Avira and Bitdefender. Keep in mind, that this set of experiments was performed one day after our 50 malware files were submitted to VirusTotal during our first scenario of experiments. The reason was that it is widely speculated that VirusTotal cooperates with AV vendors by providing signatures of malicious files, therefore detection results can be different from the moment that a file is submitted to VirusTotal.

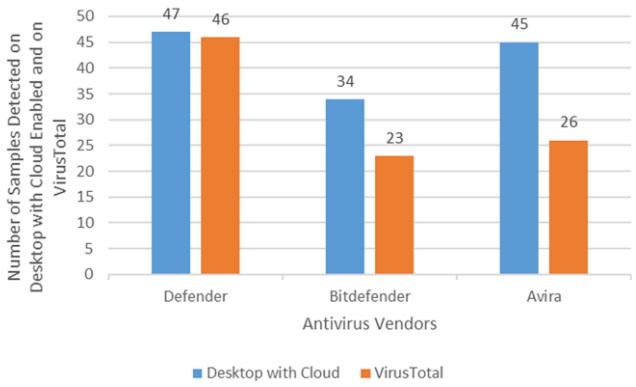

Fig. 4: Second Scenario - Malware Detection on Desktop and VirusTotal with Cloud Enabled

Fig. 4 demonstrates the differences regarding malware detection between VirusTotal engines and desktop engines when Internet access was enabled. Notice that Defender's desktop engine with cloud detected one more file than its VirusTotal counterpart. This result is very different compared to our first scenario of experiments where Windows Defender had higher detection rate on VirusTotal. We also see differences with Avira. On our first set of experiments without internet connectivity, Avira did not demonstrate any inconsistency between its desktop and VirusTotal engine. From our second scenario of experiments, we observe that the desktop engine detected 19 more files than the corresponding VirusTotal engine. This means Avira's VirusTotal engine did not use the cloud while its desktop counterpart has this capability to improve detection. Bitdefender follows the same pattern since its desktop engine detected 11 files more than its VirusTotal counterpart compared to the five files in our previous experiments. This leads us to the observation that some VirusTotal engines do not utilize the cloud for detection as desktop AV products do. It also seems that cloud provides critical information that significantly affects the detection capabilities of AVs.

V. Discussion

From the above outcomes, the following observations can be derived. First, it is evident that there is a lower malware detection capability on VirusTotal compared to the desktop AVs, because the platform runs stripped-down versions of the AV engines relying mostly on signature-based detection. The decreased detection varies between the AVs, but almost every AV displayed some irregularity between its desktop engine and the VirusTotal engine. The only exception was Windows Defender which seems to be configured in VirusTotal to utilize the cloud. Thus, the detection rate of Windows Defender is similar in VirusTotal and in desktop. It is important to mention that VirusTotal acknowledges the fact that AV vendors are free to configure their products as they wish to when deployed in VirusTotal [10]. This means that VirusTotal engines can be shipped with different settings from the commercial versions.

The discrepancies discovered between VirusTotal and desktop engines do not change the fact that VirusTotal is a valuable tool for the security community and will continue to be used for research purposes. After all, it is the AV vendors that decide with which settings the engines are going to be used by the online scanning platforms [9]. In any case, researchers should always be aware that VirusTotal results may be more 'pessimistic' in the sense that the detection outcomes of VirusTotal does not represent the true detection capabilities of desktop AVs. Researchers should adopt a more rigorous methodology to evaluate malware by complementing the results of VirusTotal with desktop AVs.

Finally, it is important to mention that our experiments have certain limitations. We compared malware detection differences using 12 vendors (out of the 36 that also have desktop versions). Our second scenario of experiments, with internet connectivity, considered three AV vendors. As a final remark, this study did not examine false positives. It is possible that by scanning more aggressively to obtain a higher detection rate, desktop AVs exhibit higher false positives than VirusTotal.

VI. Conclusions

Due to VirusTotal having seen such an extensive usage, it is imperative to find whether there are discrepancies in malware detection between the AV products on the VirusTotal platform and their corresponding desktop versions, and how large those discrepancies are. In this paper, we examined the reliability of the VirusTotal platform. First, we created our dataset which was comprised of 50 malicious files and used it to compare the detection rate of the AV products on VirusTotal with their desktop counterparts. Results showed that there is inconsistency between the engines on VirusTotal and their desktop versions. That is, VirusTotal exhibited lower malware detection rates for most AV engines compared to their desktop counterparts. While some AVs such as Avira demonstrate small inconsistencies, others such as K7 exhibit large differences between their VirusTotal and desktop engines. We concluded that this may be attributed to the fact that AV engines on VirusTotal do not include cloud-based

detection, or they are shipped with different settings compared to their desktop counterparts.

ACKNOWLEDGMENT

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 883273 (AI4HEALTHSEC).

REFERENCES

[1] VirusTotal. https://www.virustotal.com/. [Last accessed on May 2022].

[2] Hybrid-Analysis. https://www.hybrid-analysis.com/. [Last accessed on May 2022].

[3] AntiScan. https://antiscan.me/. [Last accessed on May 2022].

[4] VirSCAN. https://antiscan.me/. [Last accessed on May 2022].

[5] Jotti's malware scan. https://virusscan.jotti.org/. [Last accessed on May 2022].

[6] VirusTotal - Contributors. https://support.virustotal.com/hc/en-us/articles/115002146809-Contributors. [Last accessed on May 2022].

[7] VirusTotal - Frequently Asked Questions (FAQ). https://support.virustotal.com/hc/en-us/articles/115002122285-AV-product-on-VirusTotal-detects-a-file-and-its-equivalent-commercial-version-does-not. [Last accessed on May 2022].

[8] VirusTotal API. https://developers.virustotal.com/reference/overview. [Last accessed on May 2022].

[9] Abrams, R. VirusTotal Tips, Tricks and Myths. https://www.virusbulletin.com/uploads/pdf/magazine/2017/VB2017-Abrams.pdf [Last accessed on May 2022].

[10] VirusTotal Article. https://support.virustotal.com/hc/en-us/articles/11500 2122285-AV-product-on-VirusTotal-detects-a-file-and-its-equivalent-commercial-version-does-not. [Last accessed on May 2022].

[11] Avet - AntiVirus Evasion Tool. https://github.com/govolution/avet. [Last accessed on May 2022].

[12] Phantom-Evasion. Python antivirus evasion tool. https://github.com/oddcod3/Phantom-Evasion. [Last accessed on May 2022].

[13] Shellter Tool - AV Evasion Artware. https://www.shellterproject.com/. [Last accessed on May 2022].

[14] Veil Tool - Metasploit Payload Generator. https://github.com/Veil-Framework/Veil. [Last accessed on May 2022].

[15] TheFatRat. https://github.com/screetsec/TheFatRat. [Last accessed on May 2022].

[16] WinPayloads - Undetectable Windows Payload Generation. https://github.com/nccgroup/Winpayloads. [Last accessed on May 2022].

[17] Xeexe - AV Evasion. https://github.com/ProTechEx/Xeexe-TopAVEvasion. [Last accessed on May 2022].

[18] Unicorn - PowerShell downgrade attack and shellcode injection. https://github.com/trustedsec/unicorn. [Last accessed on May 2022].

[19] Chimera - PowerShell obfuscation script. https://github.com/tokyoneon/Chimera. [Last accessed on May 2022].

[20] MsfMania. https://github.com/G1ft3dC0d3/MsfMania. [Last accessed on May 2022].

[21] PEzor - Open-Source Shellcode & PE Packer. https://github.com/phra/PEzor. [Last accessed on May 2022].

[22] RapidPayload - Metasploit Payload Generator. https://github.com/AngelSecurityTeam/RapidPayload. [Last accessed on May 2022].

[23] FourEye - AV Evasion Tool For Red Team Ops. https://github.com/lengjibo/FourEye. [Last accessed on May 2022].

[24] Crypter – Obfuscate and encrypt the python source code to bypass AntiVirus. https://github.com/PushpenderIndia/crypter. [Last accessed on May 2022].

[25] Onelinepy - Python Obfuscator enerate One-Liners and FUD Payloads. https://github.com/spicesouls/onelinepy. [Last accessed on May 2022].

[26] Hercules - special payload generator that can bypass antivirus softwares. https://github.com/EgeBalci/HERCULES. [Last accessed on May 2022].

[27] Zhu, S., Shi, J.Z., Yang, L., Qin, B., Zhang, Z., Song, L., & Wang, G. (2020). Measuring and Modeling the Label Dynamics of Online Anti-Malware Engines. USENIX Security Symposium.

[28] Peng, P., Yang, L., Song, L., & Wang, G. (2019). Opening the Blackbox of VirusTotal: Analyzing Online Phishing Scan Engines. Proceedings of the Internet Measurement Conference.

[29] Menéndez, H.D., Clark, D., & Barr, E.T. (2021). Getting Ahead of the Arms Race: Hothousing the Coevolution of VirusTotal with a Packer. Entropy.

[30] Kwon, B.J., Mondal, J., Jang, J., Bilge, L., & Dumitras, T. (2015). The Dropper Effect: Insights into Malware Distribution with Downloader Graph Analytics. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security.

[31] Wang, H., Si, J., Li, H., & Guo, Y. (2019). RmvDroid: Towards A Reliable Android Malware Dataset with App Metadata. 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR), 404-408.

[32] Zhu, S., Zhang, Z., Yang, L., Song, L., & Wang, G. (2020, October). Benchmarking Label Dynamics of VirusTotal Engines. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (pp. 2081-2083).

[33] Teeraratchakarn, V., & Limpiyakorn, Y. (2020, April). Automated monitoring and behavior analysis for proactive security operations. In Proceedings of the 2020 2nd International Conference on Management Science and Industrial Engineering (pp. 105-109).

[34] Salem, A., Banescu, S., & Pretschner, A. (2021). Maat: Automatically analyzing virustotal for accurate labeling and effective malware detection. ACM Transactions on Privacy and Security (TOPS), 24(4), 1-35.