A Deep Dive into the VirusTotal File Feed

Kevin van Liebergen[1(✉)], Juan Caballero[1], Platon Kotzias[2], and Chris Gates[2]

[1] IMDEA Software Institute, Madrid, Spain
{kevin.liebergen,juan.caballero}@imdea.org
[2] Norton Research Group, Tempe, USA
{platon.kotzias,chris.gates}@nortonlifelock.com

Abstract. Online scanners analyze user-submitted files with a large number of security tools and provide access to the analysis results. As the most popular online scanner, VirusTotal (VT) is often used for determining if samples are malicious, labeling samples with their family, hunting for new threats, and collecting malware samples. We analyze 328M VT reports for 235M samples collected for one year through the VT file feed. We use the reports to characterize the VT file feed in depth and compare it with the telemetry of an AV vendor. We answer questions such as How diverse is the feed? How fresh are the samples it provides? What fraction of samples can be labeled on first sight? How different are the malware families in the feed and the AV telemetry?

1 Introduction

Online scanners analyze artifacts (i.e., files, URLs, domains, IPs) submitted by users using a large number of security tools, and provide access to the analysis results through free and commercial APIs. The most popular online scanner is VirusTotal [42] (VT), which is widely used by security analysts, and acts as a de-facto central sharing service for the security community. Detection labels in VT reports are routinely used for determining if an artifact is malicious by either applying a threshold on their count (e.g., [27,29,44]) or feeding them to machine-learning models [34,39], as well as for identifying the family of malicious files [16,35,36]. Prior work has shown that VirusTotal can be used to identify new malware before it is released, since malware developers often leverage VT during development to check if their samples are detected and, if so, revise them until they become *fully undetected* (FUD) [13,14,43]. VirusTotal is also commonly used as a source for collecting malware samples [3,12,14,25,26].

Amongst its commercial services, VT offers feeds, i.e., streams of analysis reports for all submissions of a type [1]. VT offers separate feeds for files, URLs, and domains. In this work, we perform what we believe is the first characterization of the VT file feed (or simply the feed). The VT file feed includes reports for new files (i.e., first submission to VT), resubmissions of previously submitted files, and re-scans requested by users. Each *report* in the VT file feed contains detailed information about the analysis of a sample (i.e., file). The report contains, among others, file metadata (e.g., hashes, size), certificate metadata for

D. Gruss et al. (Eds.): DIMVA 2023, LNCS 13959, pp. 155–176, 2023.
https://doi.org/10.1007/978-3-031-35504-2_8

signed samples (e.g., thumbprint, subject), VT specific data (e.g., time of first submission to VT, submission filenames), and the list of detection labels assigned by up to 70 antivirus (AV) engines used to scan the file. The VT file feed service also allows unlimited downloads of the samples submitted in the last seven days.

We collect reports from the VT file feed for one year, from December 21st, 2020 to December 20th, 2021. During the first 11 months we collect reports where the sample is detected by at least one AV engine, while in the last month we collect all feed reports, regardless of the number of detections. Overall, we collect 328M reports for 235M samples. We analyze the collected reports to characterize the VT file feed as a source for collecting malicious samples and for identifying new threats. Samples from the feed can be used for building labeled malware datasets such as those required by machine learning (ML) based malware detection (e.g., [4,15,17,32,37]) and family classification (e.g., [15,33]). We investigate fundamental questions for such use including How diverse is the feed? Does it allow building malware datasets for different filetypes? How fresh are the samples it provides? What is the distribution of malware families it sees? The feed can also be a source for malware triage and malware hunting approaches (e.g., [10,18]). For this use, we investigate what fraction of the feed samples are variants of known malware families that analysts may not need to investigate. In particular, we measure what fraction of the samples in the VT file feed can be detected as malicious on first sight, what fraction can be labeled with a family on first sight, and what fraction of malicious samples are originally fully undetected but later become detected by multiple AV engines. We complement our characterization of the VT file feed with a comparison with telemetry data collected in a privacy-sensitive manner from tens of millions of Windows devices of clients of a large antivirus vendor. The comparison allows us to investigate how different are the views of the malware landscape observed by both datasets and which dataset observes samples faster.

To improve family labeling, we have more than doubled the size of the AVCLASS [36] taxonomy and tagging rules. We have contributed our updates to the AVCLASS repository and they have been integrated into AVCLASS 2.8.0. The following are some of the most significant insights we gain:

- The VT file feed is a great source for malicious samples with a much higher maliciousness ratio than the AV telemetry. Still, the VT file feed is not a malware feed since half of its volume is for benign samples. Thus, it can be used to build both malicious and benign file datasets for supervised ML approaches.
- The feed is diverse with a wealth of filetypes and 4.9K families with at least 100 samples. However, the diversity is largely due to Windows and Android families.
- The feed is fresh: it receives an average of 732K new malicious samples each day and malicious samples appear a median of 4.4 h after they are seen in user devices. 39% of new malicious samples appear in the VT file feed earlier than in the AV telemetry, allowing AV engines to leverage the VT file feed to build detections for samples before they affect their customers.

Table 1. Dataset collected from VT file feed between 2020/12/21 and 2021/12/20.

Data	All	peexe	apk	other
Reports	328.3M	220.3M	15.9M	92.0M
Samples	235.7M	155.5M	8.2M	72.0M

- On first sight, 62% of the samples can be labeled as variants of known families, and thus could be ignored when hunting for new threats.
- We identify 600K originally FUD samples. These samples have no detections on first sight, but are later detected by multiple AV engines.
- The AV telemetry and VT file feed observe largely disjoint sets of malicious samples with minimal overlap (1.2%–1.8%).
- The most popular families in the VT file feed by number of samples widely differ from the families affecting most devices in the AV telemetry.

2 Datasets

We use two datasets in this work. We collect reports of files that appear in the VT file feed for one year. We also examine the Windows telemetry of an AV vendor over the same time period, which contains the metadata (e.g., file hash, file type) of the files present in tens of millions of Windows devices that opted-in to the data collection. Both datasets include benign and malicious files of different file types.

VT File Feed. The VT file feed contains analysis reports for files submitted to VT, regardless of the file type and platform (e.g., Windows executables, Android APKs, Linux ELF executables, PDF and Microsoft Office documents). Other artifacts submitted to VT like URLs, domains, and IPs have their own separate feeds that we do not analyze. The VT file feed includes reports for new files (i.e., first submission to VT), resubmissions of previously submitted files, and user-requested re-scans of previously submitted files. Throughout the paper we use *sample* and *file* indistinctly. Multiple reports may appear in the feed for the same file. In general, we focus on the last report we collected for each sample because it should provide the most up-to-date information (e.g., updated AV labels). However, when interested in what happened to a sample when first submitted to VT (e.g., whether it was detected or labeled), we examine instead its first report.

We collect reports from the feed every minute. To keep the storage manageable, we do not download the samples from the feed, only the reports. In the first 11 months, we only collected reports where at least one AV engine detected the file as malicious, which (as later shown) roughly corresponds to half of all reports in the feed. On November 19th, 2021, we started collecting all reports in the feed regardless of the number of detections, i.e., including reports with zero detections. Overall, as summarized in Table 1, over one year between December 21st, 2020 and December 20th, 2021, we collected 328M reports for 235M samples (by unique file SHA256).

Table 2. Features used.

Feature	Scope	Type	peexe	apk
cert_issuer	sample	string	✓	✓
cert_subject	sample	string	✓	✓
cert_thumbprint	sample	cryptohash	✓	✓
cert_valid_from	sample	timestamp	✓	✓
cert_valid_to	sample	timestamp	✓	✓
exiftool_filetype	sample	string	✓	✓
fseen_date	sample	timestamp	✓	✓
md5	sample	cryptohash	✓	✓
package_name	sample	string	✗	✓
sha1	sample	cryptohash	✓	✓
sha256	sample	cryptohash	✓	✓
trid_filetype	sample	string	✓	✓
detection_labels	scan	string list	✓	✓
scan_date	scan	timestamp	✓	✓
sig_verification_res	scan	string	✓	✗
vt_meaningful_name	scan	string	✓	✓
vt_score	scan	integer	✓	✓
avc_family	derived	string	✓	✓
avc_tags	derived	string list	✓	✓
avc_is_pup	derived	bool	✓	✓
filetype	derived	string	✓	✓

Telemetry. The telemetry comprises metadata of files present in tens of millions of real Windows devices in use by customers of an AV engine. It does not contain the samples, only their metadata. The customers opted-in to sharing their data and the devices are anonymized to preserve customer privacy. The AV engine queries a central service with file hashes observed on the device to obtain file reputation information. Each query for a file hash sent by a device is an *event*. An event comprises a timestamp, the anonymous identifier of the device, a file hash, a filename, and the signer key if the file is signed (i.e., the SHA256 of the public key in the file's certificate). The telemetry contains events for both benign and malicious files present on the devices. Those files may be of different types including Windows PE executables (e.g., .exe, .dll, .sys, .ocx), PDF documents, and Microsoft office files. We also obtain information from the AV vendor on the subset of telemetry files for which the AV engine threw an alert, i.e., the detected samples. We examine telemetry events over the same one year period we monitored the VT file feed.

3 Features

Since we do not download the samples, we need to restrict our analysis to features available in the reports, or that can be derived from the reports. We focus on a selected set of 21 features: 17 from the VT reports and 4 derived from those (e.g., filetype and malware family). Features are summarized in Table 2. We define three scopes for a feature: sample, scan, and derived. Sample features should have

the same value across all scans of a sample. On the other hand, scan features may differ across scans of the same sample, i.e., they evolve over time. For example, the hash of the certificate of a signed sample (*cert_thumbprint*) should always be the same. But, whether the signature of a signed sample validates (*sig_verification_res*) can change across scans, e.g., if the certificate expires or is revoked. Features may be extracted only for a subset of filetypes, e.g., be specific to Windows PE executables or Android APKs, and may be null for some samples (e.g., certificate features are not available for unsigned Windows executables). We detail the VT report features in Sect. 3.1 and the derived features in Sect. 3.2.

3.1 VT Report Features

Of the 17 features from the VT report, 3 are cryptographic hashes over the whole file used to identify the sample (*sha256*, *sha1*, *md5*), 5 are related to code signing, 2 capture the file type, another 2 capture the program name, and 5 are specific to the scan. The code signing features are available for a variety of file types including Android APKs, iOS apps, signed Windows executables, and signed Windows MSI installers.

Timestamps. We obtain four timestamps from a VT report. The *scan_date* when the sample is analyzed, which is always within our collection period. The VT first seen date (*fseen_date*) when the sample was first submitted to VT. For signed samples, we also obtain the certificate's validity period defined by the *cert_valid_from* and *cert_valid_to* dates.

AV Scans. VT scans each submitted sample with a large number of AV engines. We extract the number of engines that detected the sample (i.e., gave it a non-NULL label) (*vt_score*) and the list of *detection_labels*. The labels are used to derive three classification features, as detailed in Sect. 3.2.

Program Names. We use two features that capture the program a sample corresponds to. The *package_name* is the package identifier for Android apps and *vt_meaningful_name* is the most meaningful filename VT selects for a sample (e.g., among all filenames of the sample when submitted to VT).

3.2 Derived Features

Filetype. Determining the filetype of the sample in a report is not straightforward because VT reports do not have a single field for it. Instead, there are multiple fields that provide, possibly contradictory, filetype information. We derive a unique *filetype* feature for each report by performing a majority voting on three fields: *trid_file_type*, *vt_tags*, and *vt_meaningful_name*. *trid_file_type* captures the filetype identified by the TrID tool [31], which has very fine-grained granularity (e.g., over 90 *peexe* subtypes). We build a mapping from TrID filetypes to coarser-grained filetypes such as grouping all Windows PE files (e.g., EXE, DLL, OCX, CPL) under *peexe* and all Word files (DOC, DOCX) under *doc*. *vt_tags* provides a list of tags assigned by VT to enable searching for samples across different dimensions. Some of the tags such as *apk*, *peexe*, and *elf*

provide filetype information. When *vt_meaningful_name* is available, we extract the extension from the filename and map the extension to a filetype.

AVClass Features. We feed the *detection_labels* to the AVCLASS malware labeling tool [36]. AVCLASS outputs a list of tags (*avc_tags*) for the sample that include its category, behaviors, file properties, and the most likely family (*avc_family*). It also provides whether the sample is considered potentially unwanted or malware (*avc_is_pup*). AVCLASS uses a taxonomy to identify non-family tokens that may appear in the AV labels such as malware classes (e.g., *CLASS:virus*), behaviors (e.g., *BEH:ddos*), file properties (e.g., *FILE:packed:asprotect*), and generic tokens (e.g., *GEN:malicious*). It also uses tagging rules to identify aliases between families (e.g., *zeus* being an alias to *zbot*). In this work we apply AVCLASS to 328M VT reports, eight times more than the largest to date work [36]. Thus, our AVCLASS results include a wealth of new tags, including new aliases and non-family tokens. We have used the AVCLASS update module and extensive manual validation to identify new tagging rules that capture previously unknown aliases, as well as new taxonomy entries for tokens appearing in over 100 samples. This process has resulted in more than doubling the AVCLASS taxonomy and tagging rules. We have contributed our updates to the AVCLASS repository.

4 Feed Analysis

This section characterizes the VT file feed, answering the following questions: (1) How large is the VT file feed? (2) How fresh are samples in the feed? (3) How diverse is the feed in terms of filetypes? (4) What fraction of samples are signed? (5) What fraction of samples can be detected as malicious on first scan? (6) What fraction of malicious samples are fully undetected on first scan? (7) How diverse is the feed in terms of families? (8) What fraction of samples can be labeled on first sight?

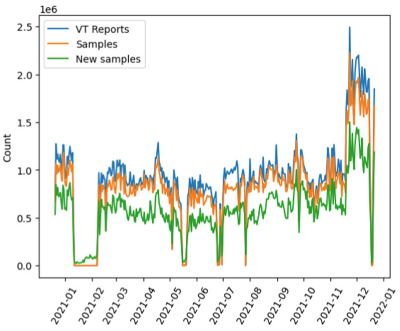

Fig. 1. Number of daily VT reports and samples collected.

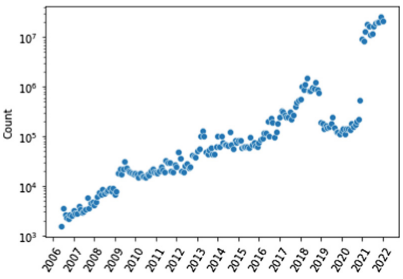

Fig. 2. Number of samples first seen by VT on each month. y-axis is in logarithmic scale.

Table 3. Daily statistics when collecting all reports (from 2021/11/21 to 2021/12/20).

	Mean	Median	Stdev	Max
Reports	1,786,565	1,879,952	482,286	2,492,454
Samples	1,586,750	1,680,520	424,590	2,223,638
New samples	1,092,640	1,120,242	299,645	1,504,174

Volume. Figure 1 shows for each day in the collection period, the number of reports in the feed, the number of unique samples in the daily reports, and the number of samples first seen by VT on that day. The figure shows a few gaps when the collection infrastructure was not working, the longest taking place between January 11th and February 7th. The volume of reports and hashes significantly increases once we started collecting samples with no detections. We compute the daily statistics, excluding days in the collection gaps, split into two periods: before November 21st, 2021 when we were collecting only reports with at least one detection, and after that date when we were collecting all reports. We say that a sample is *new* only on the first day that it is submitted to VT. Table 3 shows the daily stats when collecting all reports: the average number of daily reports is nearly 1.8M, the average number of samples nearly 1.6M, and the average number of new samples nearly 1.1M. When only collecting reports with at least one detection the daily averages were 913K reports, 823K samples, and 580K new samples. Thus, approximately half of the reports (51%), samples (51%), and new samples (53%) in the feed are for undetected samples.

Takeaway 1

At the end of 2021, the VT file feed had daily averages of 1.8M reports, 1.6M samples, and 1.1M new samples. The VT file feed is a file feed rather than a malware feed. Half of its volume in terms of reports, samples, and new samples is for undetected samples.

Freshness. The same sample may appear in the VT file feed multiple times, e.g., because different users submit it at different times. On average, 69% of the files observed in one day are new (i.e., previously unknown to VT) and 31% correspond to re-submissions or re-scans of already known files. Over the one year analysis period, 89% (209M) of the samples had a VT first seen date later than our collection start date. This ratio increases over time as every day the influx of new samples (69%) is larger than that of already seen samples (31%).

The previously seen samples that re-appear in the feed may be fairly recent or really old. The VT first seen date provides a lower bound for a sample's lifetime, i.e., the sample could be older if it took some time for it to be submitted to VT. The oldest sample observed in our collection period was first seen by VT on May 22nd, 2006. Figure 2 shows the number of samples (in logarithmic scale) whose VT first seen date is on each month, capturing how old are the samples already known to VT. The shape of the figure captures the volume increase in samples

Table 4. Top 20 filetypes for all samples observed. *peexe* includes all Windows PE files (EXE, DLL, CPL, OCX, ...) *doc* and *xls* include also *docx* and *xlsx*, respectively. *NULL* corresponds to samples for which a filetype could not be determined.

#	Filetype	Samples	Perc
1	peexe	155,526,594	65.97%
2	javascript	21,048,404	8.93%
3	html	12,540,571	5.32%
4	pdf	11,346,815	4.81%
5	apk	7,992,206	3.40%
6	text	5,149,050	2.18%
7	NULL	4,128,183	1.75%
8	zip	3,934,987	1.67%
9	dex	3,015,650	1.28%
10	gzip	2,926,739	1.24%
11	lnk	2,718,635	1.15%
12	elf	942,148	0.40%
13	rar	516,514	0.22%
14	jar	448,324	0.19%
15	doc	429,794	0.18%
16	xls	428,057	0.18%
17	macho	409,399	0.17%
18	php	352,143	0.15%
19	xml	335,962	0.14%
20	powershell	321,178	0.14%
	Other	1,233,754	0.52%
	ALL	235,745,107	100.0%

submitted to VT over time until 2019, followed by a decrease in 2019–2021. The reduction could be due to some vendors reducing their sharing from 2019.

Takeaway 2

On average, 69% of the samples observed in one day are new, i.e., previously unknown to VT, and the feed provides over a million new samples each day. Thus, the VT file feed is a great source of fresh samples.

Filetypes. Table 4 shows the top 20 filetypes for all samples observed. The feed is dominated by Windows PE files (EXE, DLL, OCX, CPL, ...) that correspond to 66% of the samples. Far behind are other filetypes like JavaScript (8.9%), HTML (5.3%), PDF (4.8%), and Android applications (3.4%). The top 5 filetypes cover 88.4% of all samples. We could not obtain a filetype for 1.7% of samples as they had no TrID information, no VT filetype-related tags, were not signed, and had no most meaningful filename with extension. This highlights the lack of a unified filetype field and the limitation of the tools VT uses for filetype determination.

Ugarte-Pedrero et al. [40] reported that 51% of an AV feed were PE executables. The larger VT file feed ratio may be due to users contributing more frequently PE executables to VT, avoiding other filetypes like HTML or text files that may contain more private data.

Takeaway 3

Two thirds of feed samples are Windows PE files, but the feed is a good source of samples for a large variety of filetypes. The feed lacks a unified filetype field and filetype identification is challenging for a significant number of samples.

Code Signing. VT extracts code signatures from multiple filetypes. The collected reports contain 13.3M samples (5.6% of all samples) for which VT extracted code signing certificates. Of the signed samples, 55.9% are Android APKs, 43.4% are Windows PE files, and 0.7% are other filetypes such as Microsoft Installers (.msi) and patches (.msp), Mach-O executables, iOS applications, Apple image files (.dmg), and some archive formats (e.g., .zip, .cab). PDF is one popular filetype for which VT does not currently extract signatures. 91.3% of all *apk* samples, 3.7% *peexe*, 31.4% *msi*, and 7.6% *macho* are signed. APKs have to be signed in order for the Android OS to install them in a device. The 8.7% of unsigned APKs is due to apps under development being uploaded to VT, possibly to check if any AV engine detects them or as part of continuous delivery pipelines.

Takeaway 4

VT supports the extraction of code signatures for a variety of filetypes, but only a small fraction (5.6%) of all feed samples, and 3.7% of the *peexe* samples, have a code signing signature.

4.1 AV Detections

A common approach for detecting malicious samples is to apply a threshold on the number of detections in a VT report [44]. We use this approach to quantify the percentage of malicious samples in the feed. We focus on the last month when collecting all feed reports. Figure 3 shows the distribution of the number of AV detections for all reports collected starting 2021/11/21. The figure shows that 51% of the reports in the last month have no detections and 7% have one detection. But, there are 9.6M samples with at least 40 detections.

We also examine the number of detections the first time a sample is submitted to VT. Figure 4 shows the complementary CDF of VT scores for the first report of each new sample since 2021/11/21. The figure captures the fraction of malicious

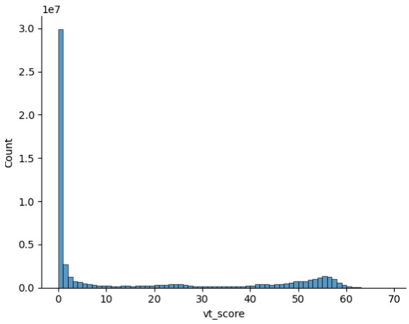

Fig. 3. Number of detections distribution for all reports since 2021/11/19.

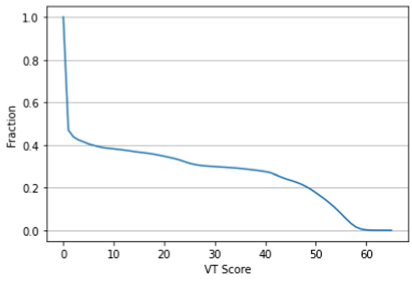

Fig. 4. Reverse ECDF for the first report of each new sample since 2021/11/19.

samples in the feed depending on the selected detection threshold. 53% of the samples have zero detections on their first observation. This percentage includes truly benign programs as well as malicious samples that go fully undetected. If we set the detection threshold on at least one detection, 47% of the samples would be considered malicious. If the threshold is set higher to minimize false positives, that reduces the fraction of malicious samples, e.g., 41% if we set it to at least four detections as done in several related works [19–21].

Takeaway 5

On first sight, 41% of samples are detected as malicious by at least 4 AV engines, and 47% by at least one AV engine. These malicious samples share traits with previously seen malware (i.e., match existing signatures).

Originally FUD Malware. It is possible that a malicious sample is fully undetected when first submitted to VT, but a later report classifies it as malicious. To detect originally FUD samples, we measure the number of samples that satisfy three conditions: (1) they are first observed by VT during our collection period; (2) their last report has at least 4 detections; and (3) their first report had zero detections *or* their VT first seen date is not in a data collection gap and is before their first observation. The last condition is a disjunction to address that we only collected reports with zero detections in the last month. During the first 11 months we can know if a sample had zero detections in their first scan because their VT first seen date is in our collection period and happens before the earliest scan date collected for the sample. The exception are samples first seen during a collection gap, for which a delayed scan date does not necessarily imply zero detections on the first scan.

We identify 637K samples satisfying those conditions. However, the time difference between the first seen date and the first report with at least four detections, indicates that 37K samples change from zero to at least four detections within 5 min of their first VT observation. We exclude those 37K samples as we observe that the distribution stabilizes afterwards (i.e., after 15 min only an extra 1K samples flip classification).

Thus, we identify 600K originally FUD samples that had no detections on their first scan, but were later considered malicious by at least 4 AV engines. Increasing the detection threshold would decrease the percentage, but the detection rate of a malicious sample tends to increase over time and for 82% of samples we only have one report. Thus, we believe our FUD rate estimation is conservative. The median time to flip classification is 7 days, (mean of 23.8 days) with 12% of the samples flipping classification in less than one day.

Of the 600K originally FUD samples, 60% are *peexe*, followed by 11% *pdf*, and 8% *javascript*. PDFs are more than twice as likely to be FUD than expected since they comprise only 4.8% of all feed samples. Malicious PDFs typically contain exploits and are used in spearphishing attacks. These numbers point to malicious PDFs being harder to detect.

Takeaway 6

Over the one year analyzed, we identify 600K samples that are originally FUD, i.e., they have zero detections on the first VT observation, but later are considered malicious by at least 4 engines. PDF documents are more likely to be FUD than other filetypes.

4.2 Family Labeling

We obtain a sample's family by feeding to AVCLASS the last report of each sample in our dataset, which should have the most up-to-date labels. AVCLASS labels 151.7M (64.3%) of the samples with 74,360 distinct family names. However, many families output by AVCLASS are rare. In particular, 41.4K (55.8%) of all families have only one sample, 14K (19.5%) have at least 10 samples, 4.9K (6.7%) have at least 100 samples, 1.5K (2.1%) have at least 1K samples, 526 (0.7%) have at least 10K samples, 147 families (0.2%) have at least 100K samples, and only 32 families (0.04%) have at least 1M samples.

Despite more than half of the families having only one sample, the fact that there are 4.9K families with more than 100 samples shows that the feed is diverse and is not dominated by a few highly polymorphic families (e.g., file infectors). However, the diversity is largely due to Windows families. By filetype, the number of families with more than 100 samples is led by *peexe* with 3.8K families, followed far behind by *apk* (447), *html* (129), *javascript* (116), *doc* (53), *macho* (52), *xls* (47), *elf* (37), and *pdf* (15). Thus, by monitoring the feed it is possible to build datasets with a large number of families for Windows and Android malware. But, for other filetypes like *macho* and *elf*, even after collecting for a year, we could only obtain 52 and 37 families with at least 100 samples, respectively.

AVCLASS outputs as family the top-ranked tag that is either a family in the taxonomy or unknown (i.e., not in the taxonomy). Of the 74,360 families output by AVCLASS, 2,391 (3.2%) are in the updated taxonomy, which contains a total of 2,451 families (i.e., 97.5% of taxonomy families appear in one year of feed reports). However, the families in the updated taxonomy contribute 90.6% of the labeled samples, only 9.4% of the samples are labeled with unknown families. This indicates that the most popular families are in the updated taxonomy, which is expected as it is common for analysts like us to add the most popular previously unknown families to the taxonomy. In fact, of the families with at least 1M samples, only 3% are unknown, increasing to 15% for families with 100K samples, 43% for those with 1K samples, and 85% for those with 10 samples. Unknown families can be due to two main reasons. One are tags that it is unclear if they are a family name or another category such as a behavior or a file property (e.g., *lnkrun*, *refresh*). The other are tags that correspond to random-looking signature identifiers or family variants (e.g., *aapw*, *dqan*). We manually examine the top 1K families and identify that 89% of the unknown families correspond to the first case and 11% to the latter. We repeat this check on 200 randomly sampled unknown families with only one sample and the result is the opposite: 11% corresponding to the first case and 89% to the latter. Thus,

Table 5. Peexe top 10 families.

Family	Class	Samples
FAM:berbew	backdoor	19,371,273
FAM:dinwod	downloader	9,398,314
FAM:virlock	virus	7,921,534
FAM:pajetbin	worm	7,164,373
FAM:sivis	virus	6,222,693
FAM:lamer	virus	4,074,441
FAM:salgorea	downloader	3,737,865
FAM:vobfus	worm	3,415,996
FAM:drolnux	worm	2,858,975
FAM:griptolo	worm	2,407,104

Table 6. Apk top 10 families.

Family	Class	Samples
FAM:smsreg	pup	616,406
FAM:ewind	pup:adware	430,531
FAM:hiddad	pup:adware	219,577
FAM:fakeadblocker	pup:adware	82,715
FAM:airpush	pup:adware	80,704
FAM:revmob	pup:adware	78,495
FAM:dowgin	pup:adware	68,522
FAM:dnotua	pup	65,330
FAM:kuguo	pup:adware	63,262
FAM:mobidash	pup:adware	40,016

Table 7. Elf top 10 families.

Family	Class	Samples
FAM:xorddos	ddos	287,631
FAM:mirai	backoor	163,525
FAM:gafgyt	backoor	59,348
FAM:tsunami	backoor	3,381
FAM:hajime	downloader	2,499
FAM:mozi	backdoor	1,996
FAM:setag	backdoor	1,454
FAM:dofloo	backdoor	890
FAM:fakecop	pup	805
FAM:ladvix	virus	580

Table 8. Mach-O top 10 families.

Family	Class	Samples
FAM:flashback	downloader	33,087
FAM:mackontrol	backdoor	15,459
FAM:mackeeper	pup	15,017
FAM:evilquest	ransomware	7,070
FAM:cimpli	pup:adware	5,444
FAM:gt32supportgeeks	pup	3,453
FAM:genieo	pup:adware	3,339
FAM:bundlore	pup:adware	3,142
FAM:installcore	pup:adware	1,543
UNK:fplayer	pup:adware	905

Table 9. Macros (doc & xls) top 10 families.

Family	Class	Samples
FAM:emotet	infosteal	26,430
UNK:sneaky	downloader	23,521
FAM:qbot	downloader	22,416
FAM:squirrelwaffle	downloader	18,230
FAM:valyria	downloader	16,256
FAM:sagent	downloader	13,298
FAM:zloader	downloader	12,371
FAM:sload	downloader	10,923
UNK:encdoc	downloader	5,703
FAM:thus	virus	4,917

Table 10. Javascript top 10 families.

Family	Class	Samples
FAM:faceliker	clicker	2,288,894
FAM:facelike	–	952,180
FAM:coinhive	miner	766,087
FAM:cryxos	–	744,894
FAM:smsreg	pup	415,669
UNK:gnaeus	–	400,570
FAM:fakejquery	downloader	330,792
UNK:hidelink	–	210,306
UNK:agentwdcr	–	87,101
FAM:inor	downloader	83,694

for less prevalent families AVCLASS may output a name that corresponds to a signature identifier or variant. While those random-looking names are not very descriptive for analysts, they are still valid cluster identifiers, i.e., samples with the same name should belong to the same family. Based on the above, we estimate that over the whole year a total of 33.8K (41.4K * 0.11 + 32.9K * 0.89) families of all filetypes have been observed in the feed.

We also obtain the family using the first report for samples first seen during our monitoring period. AVCLASS is able to label on first sight 62.3% of samples,

Table 11. Html top 10 families.

Family	Class	Samples
UNK:refresh	–	882,026
FAM:cryxos	–	363,821
FAM:faceliker	clicker	312,563
FAM:smsreg	pup	201,253
UNK:redir	–	200,926
FAM:coinhive	miner	152,968
UNK:generickdz	–	121,975
UNK:pushnotif	–	120,085
FAM:ramnit	virus	80,044
UNK:fklr	rogueware	79,353

Table 12. Pdf top 10 families.

Family	Class	Samples
UNK:fakeauthent	phishing	194,963
UNK:minerva	phishing	15,527
FAM:pdfka	exploit	13,618
UNK:pidief	exploit	6,319
FAM:alien	downloader	6,137
UNK:gorilla	phishing	4,749
UNK:talu	phishing	2,379
UNK:gerphish	phishing	1,558
UNK:urlmal	phishing	1,469
FAM:rozena	backdoor	839

slightly less than the 64.3% using the last collected report. The fact that 62% of samples can be attributed on first sight to a family indicates they correspond to variants of known families with accurate signatures. This result shows that AVCLASS can be used during triage as a filter to remove 62% of samples from well-detected families so that analysts can focus on the 38% unlabeled samples.

Prior work has applied AVCLASS to *peexe*, *apk*, and *elf* files (e.g., [12,36]). However, AVCLASS can be applied on AV labels regardless of platform or filetype. Tables 5, 6, 7 and 8 show the top 10 families for the four executable filetypes. The largest families overall are for Windows led by *berbew* with 19.4M samples, followed by *dinwod* (9.4M), and *virlock* (7.9M). We use AVCLASS to output a relations file on the whole feed. We identify a family's class checking the strongest CLASS relation for each family with a strength of at least 0.2. The top 10 *peexe* families are dominated by 4 worm and 3 virus families due to their high polymorphism. However, as already discussed, overall the feed is not dominated by file infectors and worms. For Android, the top 10 families are all PUP and 8 of them are adware. The top Linux families are dominated by backdoors including *mirai* derivatives (*gafgyt*, *hajime*, *mozi*). For macOS, seven top families are PUP and five of those adware. Table 9 shows the top 10 families for Microsoft Office macros including both Word and Excel files. Malicious macros are dominated by downloaders. Tables 10, 11 to 12 show the top families for three other popular filetypes (JavaScript, HTML, PDF) for which we observe that top families output by AVCLASS contain many unknown tokens that may correspond instead to other categories (e.g., *redir* may indicate injections that redirect the user). We also observe overlaps between JavaScript and HTML families (e.g., *cryxos*, *facelike*) and that for 9/30 families we cannot identify a class. We conclude that for these three filetypes the concept of a family is not as well defined and that AV labels for these filetypes capture instead behaviors such as phishing, injections, and exploitation.

Table 13. Top 10 families (>10K samples) sorted by ratio of originally FUD samples.

			FUD	
Family	**Class**	**Type**	**Samp.**	**Ratio**
pcacceleratepro	pup	peexe	1,749	9.5%
sagent	down.	macro	2,141	9.3%
dstudio	down.	peexe	1,255	6.2%
pasnaino	down.	peexe	613	5.9%
opensupdater	pup	peexe	2,051	4.8%
mobtes	down.	apk	967	4.6%
hesv	pup	peexe	849	4.4%
asacub	infosteal	apk	833	4.1%
agentino	down.	peexe	649	4.0%
fakecop	pup	apk	672	3.6%

Table 14. Top 10 families for feed samples in the telemetry ranked by number of infected devices.

Family	**Class**	**Dev.**	**Samp.**
winactivator	pup	2.0M	10,871
utorrent	pup	1.6M	1,366
installcore	pup	1.5M	46,758
webcompanion	pup	1.4M	2,569
dotsetupio	pup	1.1M	198
iobit	pup	898K	4,321
opensupdater	pup	692K	14,918
opencandy	pup	579K	9,346
offercore	pup	555K	363
driverreviver	pup	545K	615

Takeaway 7

The feed is diverse. Over one year, 33K families are observed with 4.9K families having at least 100 samples. However, the diversity is largely due to *peexe* and *apk* families. For those two filetypes, the feed is a good source to build datasets for large-scale family classification. AVCLASS labels 62% of samples on first sight. Thus, it can be used in triage to remove samples from well-detected families so that analysts can focus on the 38% unlabeled samples.

Originally FUD Families. Using their last report, AVCLASS outputs a family for 62.5% of the 600K originally FUD samples, which is in line with the overall labeling rate, indicating a similar fraction of well-known families among originally FUD samples. However, some families have larger fractions of originally FUD samples, and thus are harder to detect. Table 13 shows the top 10 families with at least 10K samples sorted by the ratio of originally FUD samples over all family samples. These include 6 families for Windows, 3 for Android, and one family of Microsoft Office macros. All of them have FUD ratios 6–16 times higher than the 0.59% average over all families with at least 10K samples.

5 Comparison with Telemetry

This section compares the VT file feed with the AV telemetry. We compare the total volume and percentage of malicious files, compute the intersection of malicious files, examine the family distribution, and measure which dataset observes malicious files faster.

Total and Malicious Volume. We first compare the total volume of both datasets over the one month when we were collecting all VT reports. Over that month, the VT file feed contains reports for 39.8M samples, while the telemetry

contains events for 686.5M samples. Both numbers include all samples observed over that month in each dataset, regardless of the filetype, if the samples are old or new, and whether they are benign or malicious. Thus, the telemetry volume is 17 times larger than the VT file feed volume. The AV vendor has other file datasets available beyond the Windows telemetry (e.g., Android telemetry), thus its total file volume is even larger.

Over that month, the AV engine threw alerts for 1.9M malicious files in 905K devices, 0.3% of all samples seen in the telemetry over that month. In comparison, the VT file feed contains 14.8M samples with at least four detections (37.3%) and 17.5M with at least one detection (43.9%). Thus, the ratio of malicious files in the VT feed is 126–146 times larger than in the telemetry. This is likely due to two reasons. First, prior work has shown that AV telemetry is largely dominated by rare benign files, i.e., 94% of files in AV telemetry are observed only in one device and the ratio of benign to malicious such files is 80:1 [23]. Second, the VT file feed is likely biased towards malicious samples, as VT contributors may only submit suspicious samples to be analyzed, while avoiding to submit samples known to be benign.

Over the whole year, the AV engine detected 12.9M files as malicious. In comparison, the VT file feed contains 187.0M samples with at least four detections and 212.2M with at least one detection. Thus, over the course of the year the VT file feed observes 16–17 times more malicious files. Of the 12.9M detected files in the telemetry, 5.2M (40.3%) have extensions corresponding to peexe files (*.exe*, *.dll*, *.sys*, *.cpx*, *.ocl*, *.scr*), followed by *.tmp* temporary files (17.8%) and *.lnk* link files (9.7%).

Takeaway 8

While massive, the total VT file feed volume is 17 times lower than the Windows telemetry of a AV vendor. However, despite the much lower volume, the VT file feed contains 16–17 times more malware than the telemetry, making it a great source of malicious samples.

Intersection. We compute the intersection between both datasets over the whole year. Given the massive size of the telemetry (i.e., $> 10^8$ events), to make the query scale, we focus the intersection on malicious *peexe* files and ignore other filetypes and benign executables. Thus, we query the telemetry using the 151.7M *peexe* file hashes from the VT file feed with at least one detection. For each file hash found in the telemetry, we collect the anonymized identifiers of the devices where it was observed and the telemetry first seen time, i.e., the earliest time, within our collection period, a feed sample was queried by an endpoint to obtain its reputation.

The intersection contains 3.8M samples with at least one detection (1.8% of feed samples with one detection) and 2.2M (1.2%) with at least four detections. The small intersection indicates that the telemetry and the VT file feed observe largely disjoint sets of malicious samples. Prior work has observed that public and commercial threat intelligence feeds have small overlap [9,41]. However,

those works focus on IP addresses [41] or work on APT-focused commercial TI feeds [9]. As far as we know, no prior work has checked the overlap between large (malicious) file hashes datasets. Our results show that even the largest (malicious) file hashes datasets are largely disjoint with minimal overlap. This is likely caused by a huge space of malicious samples of which each vendor only sees a small portion.

Of the 12.9M files detected as malicious by the AV vendor over the year, 11.9M (92.2%) are not observed in the VT file feed. These files are either never submitted to VT or their last VT report was before our collection start. Quantifying this requires querying the 11.9M files to VT which due to API restrictions is not possible. Instead, we estimate these figures by querying a subset of 1M randomly selected hashes. Only 10.9% of those are known to VT, while 89.1% have never been submitted. This shows that security vendors may only share a fraction of their malicious samples with VT. Sharing decisions by the AV vendor are transparent to us.

Takeaway 9

The telemetry and VT file feed observe largely disjoint sets of malicious samples (1.2%–1.8% of feed samples in common). Thus, even the largest file datasets only see a small portion of the whole space of malicious samples.

Family Distribution. Table 14 shows the top 10 families in the intersection sorted by number of telemetry devices where the samples of the family are observed. All these families are PUP. Instead, when we ranked families by number of samples observed in the VT file feed (*peexe* families in Table 5), the top families were dominated by virus and worm families. From the top 10 VT file feed families by number of samples, *vobfus* and *virlock* are the two families that affect most devices in the telemetry found on 25.9K and 3.3K devices, respectively, 1–2 orders of magnitude less devices than the families in Table 14. The remaining 8 families are ranked below the 1,000th position affecting each less than 2K devices. These results indicate that the top families in the VT file feed, i.e., those with the most samples submitted by contributors, may be biased towards highly-polymorphic families such as viruses and worms and may not correspond to the families that affect most user devices, which according to the telemetry are PUP families.

Takeaway 10

The top families by number of samples collected is biased towards highly polymorphic families such as viruses and worms, and may significantly differ from the top families by number of infected devices.

Observation Delay. The telemetry first seen timestamp for a sample, i.e., the earliest time within our collection period a feed sample was queried by an

endpoint, is an upper bound on the earliest time the AV vendor observed the sample. For example, a sample first seen by the AV vendor in 2010 may appear in the telemetry subset we analyze as first queried on December 22nd, 2020. We calculate the delay to observe a sample as the VT first seen timestamp minus the telemetry first seen timestamp, but only for the 2.1M samples first observed by VT during our analysis period and that are in the intersection with the telemetry. Of those 2.1M samples, 2.5M (61%) are first observed by the telemetry (i.e., positive difference) while 816K (39%) are first observed by VT (i.e., negative difference). The median delay for VT to observe the sample is 4.4 h. Thus, real devices observe the sample a few hours earlier than VT. However, the mean delay is 21 days because 12% of these samples are first submitted to VT at least 3 months after they appear in the telemetry, compared to 3% being observed by VT 3 months earlier than in the telemetry. It is important to note that since the telemetry first seen is an upper bound for the AV vendor first seen, the VT delay may be actually larger.

Takeaway 11

Malicious samples are first seen a median of 4.4 hours earlier in the telemetry. Still, 39% of samples are first seen by VT before they are first seen in user devices. Thus, VT may provide useful early alerts to AV vendors.

6 Discussion

The section discusses the implications of our results for future works, limitations, threats to validity, and avenues for improvement.

Result Implications. Our results have implications for researchers analyzing the malware ecosystem. We show that the most popular Windows families widely differ between the VT file feed and the AV telemetry. Top families in the feed correspond to highly polymorphic malware such as viruses and worms. In contrast, families affecting most user devices are PUP. Thus, the most popular feed families may not be those that impact end users most, but rather those for which samples are easier to collect (e.g., due to their many polymorphic variants). Focusing only on the top feed families might ignore popular families that affect many user devices. Those families are also found in the VT file feed, but with lower volumes, so researchers may need to dive deeper into the feed beyond the top families.

Our results have implications for researchers that need to build malware datasets for ML approaches. The VT file feed is a great source for malware (and also benign) files, due to its large volume, filetype diversity, and freshness of samples. However, the diversity largely centers on Windows and to a smaller degree Android samples. For other platforms, even collecting samples for one year, would only provide a handful of families with at least 100 samples (e.g.,

52 for Mac OS and 37 for Linux), which we consider the minimum for training, validating, and testing ML family classification models.

Our results have implications for researchers building detection models on the VT file feed. Pendlebury et al. [28] argued that the goodware/malware ratio expected in ML testing datasets should be matched when training the model. They measured this ratio was 90/10 for AndroZoo [2]. Previous work has also shown that this ratio is roughly 99/1 in AV telemetry [23]. In contrast, we observe a ratio of nearly 50/50 for the VT file feed, indicating VT users are more likely to submit malicious samples. Accounting for this ratio is important for applying ML models on the VT file feed. To avoid temporal bias, Pendlebury et al. [28] also recommend that samples in the testing dataset have timestamps larger than any sample in the training dataset and that in every testing slot, all samples come from the same time window. For the VT file feed, this separation should use the VT first seen date because we show that 31% of the daily samples are re-submissions of older samples which may break these properties.

While the dynamics of detections labels have been studied before [8,44], our work is the first one that can analyze them on samples that are not selected apriori and re-scanned daily by the authors. This allows us to identify 600K originally FUD samples that initially escaped detection until multiple AV vendors realized their maliciousness a median of 7 days later (mean of 23.8 days). This raises the question of how many other malicious files may remain undetected in the feed.

Data Collection Issues. Longitudinal analyses often face unexpected data collection issues that create gaps in the temporal data sequence. Such issues prevented us from collecting VT reports on 39 days, most notably over 27 days between January 11th and February 7th, 2022. Thus, our dataset contains data for 326 days, rather than a whole year. We account for these gaps throughout the measurements, e.g., we do not provide yearly volume statistics, but provide daily statistics that exclude data gaps.

AV Telemetry Comparison. Our work shows that the VT file feed has little overlap with the telemetry of a large AV vendor and that the most popular families largely differ in both datasets. Results could differ for the telemetry of other AV vendors. However, we believe this is unlikely given the large size of both datasets. Furthermore, our results match those observed in smaller APT-focused file datasets [9] and in datasets of other malicious indicators such as IP addresses [41]. We believe the different results in this area indicate that feeds (even those that aggregate data from multiple other feeds) achieve limited coverage of indicators, thus highlighting the need for further aggregation and cooperation. It would be interesting to examine whether different AV vendors observe very different top malware families as well, but getting access to the telemetry of multiple AV vendors is challenging.

Family Labeling. Our malware labeling is based on AV labels processed by AVCLASS. Thus, it inherits the limitations of both the AV labels and the tool. For example, our results show that AV labels for document filetypes such as HTML and PDF often contain behaviors rather than family names. If the AV labels do not contain a family name, possibly because the AV vendors do not

have a good definition of family for those filetypes, then AVCLASS cannot output a family. There are also cases where AVCLASS identifies as a family a token that is not a family (e.g., looks randomly generated). These may be due to new AV engines or changes to AV label format since AVCLASS was released. We will report them to the developers so that they can be addressed.

Filetype Identification. VT reports lack a unified filetype field. Instead, they provide the output of different filetype identification tools, which may not agree. To handle disagreements and minimize the number of samples without a filetype, we combine multiple filetype-related fields in the VT reports. Still, we cannot infer the filetype for 1.75% of samples indicating that further research on filetype identification is needed. Furthermore, filetype identification tools should output hierarchical filetypes allowing users to aggregate results as they prefer. For example, a DLL is also a PE executable and an APK is also a ZIP archive. Whether to count DLLs and APKs as their own filetypes or as part of their parent filetypes should be up to the user.

7 Related Work

Most related is the work by Ugarte-Pedrero et al. [40] that analyzes 172K PE executables that a large AV vendor collects through multiple sources on one day. In contrast, we examine one year of a file feed with 235M samples of multiple filetypes and compare it to the telemetry of a large AV vendor. Other works have performed large scale longitudinal malware analysis on Windows [7,22], Android [24,38], and Linux [3,12]. In contrast, our work examines malware for multiple platforms including Windows, Android, Linux, macOS, Microsoft Office macros, PDF documents, and Web content.

Detection labels such as those available in VT reports have been widely studied. Early works showed how different AV engines disagree on labels for the same sample [5,11]. Still, AV labels have been widely used to build training datasets and evaluate malware detection and clustering approaches (e.g., [5,6,30]). Recent works have examined the dynamics of detection labels [8,44] and have proposed to replace the traditional threshold-based detection approach on the number of detections, which we use in this paper, with machine-learning models [34,39]. We plan to explore these approaches in future work.

8 Conclusions

We have characterized the VirusTotal file feed by analyzing 328M reports for 235M samples collected during one year, and have compared the feed with the telemetry of a large AV vendor. Among others, we show that despite having a volume 17 times lower than the AV telemetry, the VT file feed observes 8 times more malware. The feed is fresh with 69% of daily samples being new and samples appear a median of 4.4 h after they are seen in user devices. The feed is diverse containing 4.9K families with at least 100 samples. However, the

diversity largely focuses on Windows and Android families. The AV telemetry and VT file feed observe largely disjoint sets of malicious samples (1.2%–1.8% overlap). We identify 600K originally FUD samples that have no detections on first scan, but are later considered malicious by at least 4 AV engines.

Acknowledgment. This work has been partially supported by the PRODIGY Project (TED2021-132464B-I00) funded by MCIN/AEI/10.13039/501100011033/ and EU NextGeneration funds.

References

1. Virustotal API 2.0 reference: File feed. http://developers.virustotal.com/v2.0/reference/file-feed
2. Allix, K., Bissyandé, T.F., Klein, J., Le Traon, Y.: AndroZoo: collecting millions of android apps for the research community. In: International Conference on Mining Software Repositories (2016)
3. Alrawi, O., et al.: The circle of life: a large-scale study of the IoT malware lifecycle. In: USENIX Security Symposium (2021)
4. Arp, D., Spreitzenbarth, M., Huebner, M., Gascon, H., Rieck, K.: Drebin: efficient and explainable detection of android malware in your pocket. In: Network and Distributed System Security (2014)
5. Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F., Nazario, J.: Automated classification and analysis of internet malware. In: International Symposium on Recent Advances in Intrusion Detection (2007)
6. Bayer, U., Comparetti, P.M., Hlauschek, C., Kruegel, C., Kirda, E.: Scalable, behavior-based malware clustering. In: Network and Distributed System Security (2009)
7. Bayer, U., Habibi, I., Balzarotti, D., Kirda, E., Kruegel, C.: A view on current malware behaviors. In: LEET (2009)
8. 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)
9. Bouwman, X., Griffioen, H., Egbers, J., Doerr, C., Klievink, B., Van Eeten, M.: A different cup of TI? The added value of commercial threat intelligence. In: USENIX Security Symposium (2020)
10. Buyukkayhan, A.S., Oprea, A., Li, Z., Robertson, W.K.: Lens on the endpoint: hunting for malicious software through endpoint data analysis. In: International Symposium on Research in Attacks, Intrusions, and Defenses (2017)
11. Canto, J., Dacier, M., Kirda, E., Leita, C.: Large scale malware collection: lessons learned. In: IEEE SRDS Workshop (2008)
12. Cozzi, E., Graziano, M., Fratantonio, Y., Balzarotti, D.: Understanding Linux malware. In: IEEE Symposium on Security and Privacy (2018)
13. Graziano, M., Canali, D., Bilge, L., Lanzi, A., Balzarotti, D.: Needles in a haystack: mining information from public dynamic analysis sandboxes for malware intelligence. In: USENIX Security Symposium (2015)
14. Huang, H., et al.: Android malware development on public malware scanning platforms: a large-scale data-driven study. In: International Conference on Big Data (2016)
15. Huang, W., Stokes, J.W.: MtNet: a multi-task neural network for dynamic malware classification. In: Detection of Intrusions and Malware, and Vulnerability Assessment (2016)

16. Hurier, M., et al.: Euphony: harmonious unification of cacophonous anti-virus vendor labels for android malware. In: IEEE/ACM International Conference on Mining Software Repositories (2017)
17. Jindal, C., Salls, C., Aghakhani, H., Long, K., Kruegel, C., Vigna, G.: Neurlux: dynamic malware analysis without feature engineering. In: Annual Computer Security Applications Conference (2019)
18. Kaczmarczyck, F., et al.: Spotlight: malware lead generation at scale. In: Annual Computer Security Applications Conference (2020)
19. Kotzias, P., Bilge, L., Caballero, J.: Measuring PUP prevalence and PUP distribution through pay-per-install services. In: USENIX Security Symposium (2016)
20. Kotzias, P., Caballero, J., Bilge, L.: How did that get in my phone? Unwanted app distribution on android devices. In: IEEE Symposium on Security and Privacy (2021)
21. Kotzias, P., Matic, S., Rivera, R., Caballero, J.: Certified PUP: abuse in authenticode code signing. In: ACM Conference on Computer and Communication Security (2015)
22. Lever, C., Kotzias, P., Balzarotti, D., Caballero, J., Antonakakis, M.: A lustrum of malware network communication: evolution and insights. In: IEEE Symposium on Security and Privacy (2017)
23. Li, B., Roundy, K., Gates, C., Vorobeychik, Y.: Large-scale identification of malicious singleton files. In: ACM Conference on Data and Application Security and Privacy (2017)
24. Lindorfer, M., Neugschwandtner, M., Weichselbaum, L., Fratantonio, Y., Van Der Veen, V., Platzer, C.: Andrubis-1,000,000 apps later: a view on current android malware behaviors. In: International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (2014)
25. Maffia, L., Nisi, D., Kotzias, P., Lagorio, G., Aonzo, S., Balzarotti, D.: Longitudinal study of the prevalence of malware evasive techniques. arXiv preprint arXiv:2112.11289 (2021)
26. Mantovani, A., Aonzo, S., Ugarte-Pedrero, X., Merlo, A., Balzarotti, D.: Prevalence and impact of low-entropy packing schemes in the malware ecosystem. In: Network and Distributed Systems Security Symposium (2020)
27. Masri, R., Aldwairi, M.: Automated malicious advertisement detection using VirusTotal, UrlVoid, and TrendMicro. In: International Conference on Information and Communication Systems (2017)
28. Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., Cavallaro, L.: Tesseract: eliminating experimental bias in malware classification across space and time. In: USENIX Security Symposium (2019)
29. Peng, P., Yang, L., Song, L., Wang, G.: Opening the blackbox of VirusTotal: analyzing online phishing scan engines. In: Internet Measurement Conference (2019)
30. Perdisci, R., Lee, W., Feamster, N.: Behavioral clustering of HTTP-based malware and signature generation using malicious network traces. In: USENIX Symposium on Networked Systems Design and Implementation (2010)
31. Pontello, M.: TrID - File Identifier (2021). http://mark0.net/soft-trid-e.html
32. Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., Nicholas, C.K.: Malware detection by eating a whole EXE. In: Workshops at the AAAI Conference on Artificial Intelligence (2018)
33. Rieck, K., Holz, T., Willems, C., Düssel, P., Laskov, P.: Learning and classification of malware behavior. In: Detection of Intrusions and Malware, and Vulnerability Assessment (2008)

34. Salem, A., Banescu, S., Pretschner, A.: Maat: automatically analyzing VirusTotal for accurate labeling and effective malware detection. ACM Trans. Privacy Secur. **24**(4), 1–35 (2021)
35. Sebastian, M., Rivera, R., Kotzias, P., Caballero, J.: AVClass: a tool for massive malware labeling. In: Research in Attacks, Intrusions, and Defenses (2016)
36. Sebastián, S., Caballero, J.: AVClass2: massive malware tag extraction from AV labels. In: Annual Computer Security Applications Conference (2020)
37. Smutz, C., Stavrou, A.: Malicious PDF detection using metadata and structural features. In: Annual Computer Security Applications Conference (2012)
38. Suarez-Tangil, G., Stringhini, G.: Eight years of rider measurement in the android malware ecosystem. IEEE Trans. Depend. Secure Comput. (2020)
39. Thirumuruganathan, S., Nabeel, M., Choo, E., Khalil, I., Yu, T.: SIRAJ: a unified framework for aggregation of malicious entity detectors. In: IEEE Symposium on Security and Privacy (2022)
40. Ugarte-Pedrero, X., Graziano, M., Balzarotti, D.: A close look at a daily dataset of malware samples. ACM Trans. Privacy Secur. **22**(1), 1–30 (2019)
41. Li, V.G., Dunn, M., Pearce, P., McCoy, D., Voelker, G.M., Savage, S.: Reading the Tea leaves: a comparative analysis of threat intelligence. In: USENIX Security Symposium (2019)
42. VirusTotal. http://www.virustotal.com/
43. Yuan, L.-P., Wenjun, H., Ting, Yu., Liu, P., Zhu, S.: Towards large-scale hunting for android negative-day malware. In: International Symposium on Research in Attacks, Intrusions and Defenses (2019)
44. Zhu, S., et al.: Measuring and modeling the label dynamics of online anti-malware engines. In: USENIX Security Symposium (2020)