Exploring the Long Tail of (Malicious) Software Downloads

Babak Rahbarinia*, Marco Balduzzi⋆, Roberto Perdisci‡
*Dept. of Math and Computer Science, Auburn University at Montgomery, Montgomery, AL
⋆Trend Micro, USA ‡Dept. Computer Science, University of Georgia, Athens, GA
brahbari@aum.edu, marco_balduzzi(at)trendmicro.com, perdisci@cs.uga.edu

***Abstract*—In this paper, we present a large-scale study of global trends in software download events, with an analysis of both benign and malicious downloads, and a categorization of events for which no ground truth is currently available. Our measurement study is based on a unique, real-world dataset collected at *Trend Micro* containing more than 3 million *in-the-wild* web-based software download events involving hundreds of thousands of Internet machines, collected over a period of seven months.**

Somewhat surprisingly, we found that despite our best efforts and the use of multiple sources of ground truth, more than 83% of all downloaded software files remain *unknown*, i.e. cannot be classified as benign or malicious, even two years after they were first observed. If we consider the number of machines that have downloaded at least one unknown file, we find that more than 69% of the entire machine/user population downloaded one or more unknown software file. Because the accuracy of malware detection systems reported in the academic literature is typically assessed only over software files that can be labeled, our findings raise concerns on their actual effectiveness in large-scale real-world deployments, and on their ability to defend the majority of Internet machines from infection.

To better understand what these *unknown* software files may be, we perform a detailed analysis of their properties. We then explore whether it is possible to extend the labeling of software downloads by building a rule-based system that automatically learns from the available ground truth and can be used to identify many more benign and malicious files with very high confidence. This allows us to greatly expand the number of software files that can be labeled with high confidence, thus providing results that can benefit the evaluation of future malware detection systems.

I. Introduction

Most modern malware infections are caused by web-driven software download events, for example via drive-by exploits [6] or social engineering attacks [11]. In response to the growth of infections via software downloads, the security community has conducted a wealth of research, the majority of which is dedicated to detection and remediation efforts [2], [7], [14]–[16], [20]. Some recent studies have focused on measuring specific infection vectors. For instance, Caballero et al. [1] have studied the business infrastructure of malware distribution networks, while Rossow et al. [17] and Kwon et al. [10] have focused their attention towards malware droppers, and provide detailed measurements that aim to better understand how dropper-driven infections work.

In this paper we aim to provide a broader, large-scale study of global trends in software download events, with an analysis of both benign and malicious downloads, and a categorization of events for which no ground truth is currently available. Our measurement study is based on a unique, real-world dataset we obtained from *Trend Micro* – a leading anti-malware vendor (which we refer to as AMV). This dataset contains detailed (anonymized) information about 3 million *in-the-wild* web-based software download events involving over a million of Internet machines, collected over a period of seven months. Each download event includes information such as a unique (anonymous) global machine identifier, detailed information about the downloaded file, what process initiated the download and the URL from which the file was downloaded. To label benign and malicious software download events and study their properties, we make use of multiple sources of ground truth, including information from VirusTotal.com and AMV's private resources. This ground truth was collected over several months, both at a time close to the software download events as well as many months after the collection of our dataset, so to account for the time typically needed by anti-malware vendors to develop new malware signatures.

Somewhat surprisingly, we found that despite our best efforts, we were able to label only less than 17% of the 1,791,803 software files contained in our dataset. In other words, more than 83% of all downloads remain *unknown*, even two years after they were first observed. Most of these files have very low prevalence. Namely, when considered independently from one another, each file is downloaded by only one (or few) machines overall. Therefore, one may think that these files are uninteresting, and the fact that they remain unknown is understandable, since if they were malicious they would impact a negligible number of machines. However, if we consider the number of machines that have downloaded at least one unknown file, we find that more than 69% of the entire machine population downloaded one or more unknown software file, during our observation period. This is a significant result, in that it highlights a major challenge faced by the malware research community. In fact, most malware detection and classification systems proposed in the scientific literature are naturally evaluated only on samples (i.e., executable files) for which ground truth is available. Unfortunately, because the accuracy of these systems can only be assessed over a small minority of in-the-wild software downloads, this raises concerns on their actual effectiveness in large-scale real-world deployments, and on their ability to defend the majority of Internet machines from infection.

To better understand what these *unknown* software files may look like, we perform a detailed analysis of their properties. We then explore whether it is possible to extend the labeling of software downloads by building a rule-based system that automatically learns from the available ground truth. Specifically, we aim to generate human-readable classification rules that can accurately identify benign and malicious software using a combination of simple features, while keeping the false positive rate to a low target rate of 0.1%, which is a common threshold in the anti-malware industry. For instance, we show that features such as software signing information can be leveraged to improve file labeling. In particular, unlike studies that focus primarily on potentially unwanted programs [8], [9], [19], we show that software signing information is present in other types of malware, contrast them with signed benign programs, and leverage this information for labeling purposes. These automatically extracted rules allow us to increase the number of samples labeled by 233% (a 2.3x increase) with high confidence, compared to the available ground truth. Furthermore, each newly labeled sample can be traced back to the human-readable rule that assigned the label, thus providing a way for analysts to interpret and verify the results. By providing a way to significantly expand the labeling of software files, our rule-based system can therefore benefit the evaluation of future malware detection systems.

In summary, our paper makes the following contributions:

- We explore trends in the software downloads collected *in-the-wild* from over a million machines from a leading anti-malware provider, and study the proprieties of benign, malicious, and unknown software.
- We report on the importance of considering low prevalence files, which in aggregate are run by almost 70% of the monitored machines and whose true nature tends to remain *unknown* to AV vendors even two years after they were first observed;
- We present a novel rule-based classification system that learns *human-readable* file classification rules from easy-to-measure features, such as the process used to download a file and the software file signer. We then show that this system can be used to significantly increase the number of software files that can be labeled, compared to available ground truth, thus providing results that can benefit the evaluation of future malware detection systems.

II. Data Collection and Labeling

A. Software Download Events

To collect in-the-wild software download events, we monitor more than a million machines of a well-known leading anti-malware vendor (we only monitor download events from customers who have approved sharing this information with AMV). Each customer machine runs a monitoring software agent (SA), which is responsible for identifying web-based software downloads and reporting these events to a centralized data collection server (CS). Each download event is represented by a 5-tuple, (f, m, p, u, t), where f is the downloaded file, m is the machine that downloaded f, p is the process on a the user's machine that initiated the download, u is the download URL, and t is a timestamp. The downloaded files and client processes are uniquely identified by their respective file hash, whereas the machines are uniquely identified by an anonymized global unique ID (generated by AMV's software agent installation). In addition, for every downloading process and downloaded file we have their (anonimyzed) path on disk, including file names.

While each SA captures all web-based download events observed on the system, for efficiency reasons only events considered of interest are reported to the CS. Specifically, our dataset contains only software download events that satisfy the following conditions:

- The newly downloaded file is *executed* on the user's machine. Namely, software files that are downloaded from the web but remain "inactive" (i.e., are not executed on the system) are not reported.
- The current *prevalence* of the downloaded file is below a predefined threshold, σ. For instance, consider a newly downloaded software file f observed by a monitored machine m at time t. This new event is reported by m to the CS only if the number of distinct machines that downloaded the same file (as determined based on its hash) before time t is less than σ.
- The URL from which the file is downloaded is not whitelisted. For instance, software updates from Microsoft or other major software vendors are not collected.

Overall, the rules described above aim to reduce the system-overhead and bandwidth consumption needed to transfer the download events from millions of monitoring agents to the collection server.

During our data collection period, σ was set to 20. Namely, each file could be reported up to 20 times, if it occurred in 20 different download events. It is possible that a file will reach a true prevalence higher than 20, though this will not be reflected in the dataset we analyze. At the same time, if the final prevalence of a file (i.e., at the end of the collection period) is less than 20, this means that the file was actually downloaded by less than 20 of the monitored machines, as reported in our measurements. Of all the files we observed, we found that 99.75% have a prevalence of less than 20. Namely, our prevalence measurements are capped at 20 for only less than 0.25% of all the downloaded files we observed (see Section IV-A for more details).

B. File Labeling

For every software file, we gather related ground truth using multiple sources. Specifically, to label benign software files we use a large commercial whitelist and NIST's software reference library[1]. Note that this information is gathered for both downloaded files and downloading processes. We also make use of VirusTotal.com (VT). Specifically, given a software file f, we query VT both close to the time of download and then again almost two years after the data collection. We let this large amount of time pass before re-querying VT, so to give plenty of time for VT to collect and process (via crowdsourced submissions) files that we observed, and for anti-viruses to develop new detection signatures.

We label a file as *benign* if either it matches our whitelists, or if all anti-virus engine (AV) on VT still classify the file as benign, even after almost two year from collection. We label a file as *likely benign* if it is classified as benign by VT but the time difference between first and last scans is less then 14 days. To label malicious files, we adopt the following approach. Of the more than 50 anti-virus (AV) engines on VT, we consider two groups: a group of "trusted" AVs that includes ten among the most popular AV vendors (i.e., Symantec, McAfee, Microsoft, Trend Micro, etc.), and a group containing all other available AVs, which tend to produce somewhat less reliable detection results. Then we label a file as *malicious* if at least one of the ten "trusted" AVs assigns it an AV label. On the other hand, if

[1]http://www.nsrl.nist.gov

none of the ten "trusted" AV vendors assigns an AV label to the file but at least one of the remaining less popular AVs detects the file as malicious, we assign a *likely malicious* label. The downloading processes are also labeled similarly. Files (processes) for which no ground truth can be found are labeled as *unknown*. For every file, including *unknown* files, we obtain additional details, such as their file size, their prevalence across all machines of AMV, if the file carries a valid software signature, if it is packed and with what packer, etc.

To label the URLs from which files are downloaded, we use AMV's internal URL whitelists and blacklists, the list of most popular domains according to Alexa.com, and Google Safe Browsing (GSB) [5]. Specifically, to label a URL as *benign*, we maintain a list of domains that consistently appeared in the top one million Alexa sites for about a year. To further mitigate possible noise in the Alexa list, we consult multiple whitelists and adjust the labels as follows. If the effective second-level domain (e2LD) of a URL appears in the Alexa.com list and the URL also matches our private curated whitelist (provided by Trend Micro), the URL will be labeled as benign. On the other hand, a URL will be labeled as *malicious* if it matches GSB and our private URL blacklist.

C. Malicious File Types

To shed light on what kind of malware are involved in the software download events we observed, we attempt to group known malicious files into *types*. To this end, for each malicious file we use multiple AV labels to derive their *behavior type* (e.g., fakeAV, ransomware, dropper, etc.) and their *family* (e.g., Zbot, CryptoLocker, etc.). While we acknowledge that AV labels are often noisy and sometimes inconsistent, we use a best effort approach, similar to previous work [12], [18]. For instance, to derive the *family* labels from AV labels, we simply use a recently proposed system called `AVclass` [18]. As we are not aware of any similar tool that can derive the *behavior type*, we have developed the labeling scheme described below, which is based on AV label mappings provided by Trend Micro and on our own empirical experience.

To determine the behavior type (or simply *type*, for brevity) of a malicious file, we consider the AV labels assigned to the file by a subset of five leading AV engines[2], for which we have obtained a "label interpretation map" provided by Trend Micro (ref. Table II). By leveraging this map, we identified a set of behavior type keywords used by these leading AVs, such as *fake-av*, *ransomware*, *bot*, etc. For instance, an AV label such as `TROJ_FAKEAV.SMU1` assigned by Trend Micro indicates a *fake-av* malware type. However, because different AVs may disagree on the label to be assigned to a specific malicious file, we designed a set of simple rules to resolve such conflicts:

1) *Voting*: Given a malicious file f, we first map each label into its respective *type*. We then assign to f the type label with the highest count. In case of two or more type labels receive an equal number of votes, we break the tie using the second rule.
2) *Specificity*: If among the types considered for a malicious file, there is one type that is more "specific" than the rest, that specific type is assigned. For example, if AV labels for a file report conflicting types, such as *banker* and *trojan*, we will select *banker* as the final label because it identifies a more specific type keyword than *trojan* (notice that AV engines often use *trojan* or *generic* to flag malicious file whose true behavior/class is unknown).

In some rare cases where the above two rules still cannot resolve a conflict, we derive the final type via manual analysis.

As an example of the results given by rule 1), consider a malicious file with four AV labels (i.e., one out of the five leading AVs we consider for type labeling did not report the file as being malicious): Symantec=`Trojan.Zbot`, McAfee=`Downloader-FYH!6C7411D1C043`, Kaspersky=`Trojan-Spy.Win32.Zbot.ruxa`, and Microsoft=`PWS:Win32/Zbot`. The type *banker* can be derived from three of the AV labels (Zbot is programmed to steal banking information[3]), while McAfee's AV label indicates a *dropper* (i.e., `Downloader` is mapped to the *dropper* behavior type). In this case, the final type we assign will be *banker*. Now consider an example of rule 2) where the following AV labels are assigned to a malicious file: Kaspersky=`Trojan-Downloader.Win32.Agent.heqj` and McAfee=`Artemis!DEC3771868CB`. In this case, Kaspersky's label indicates a *dropper* behavior, while McAfee's label is a generic one (Artemis refers to a heuristics-based detection approach). Since *dropper* indicates a more specific behavior, we assign it as the final type.

For 44% of all malicious downloaded files and client processes, we were able to assign a type label without encountering any conflicts (i.e., the AVs fully agreed on the type). In about 28% of cases, the type label was assigned using the Voting rule, whereas the Specificity rule was applied in 23% or the cases. In the remaining 5% of the cases, the type label was resolved via manual analysis. To foster reproducibility of these results, we provide our malicious type extractor tool as an open source tool at gitlab.com/pub-open/AVType.

III. Dataset Overview

In this section, we provide an overview of our dataset, including the exact number of machines we monitored during the data collection period, the number of software download events we observed, how many of these events we were able to label, the malware types and families included in the dataset, etc. More detailed measurements are provided in Sections IV and V.

Our observation period spans seven months, from January 2014 to August 2014. During this time, we observed 3,073,863 software download events triggered by 1,139,183 machines. The software files were downloaded from 1,629,336 distinct URLs, across 96,862 different domain names. Out of 1,791,803 downloaded files, we labeled 9.9% as *malicious* and 2.3% as *benign*. We also labeled 4.8% as either likely benign or malicious. Note that although some ground truth is available for likely benign and likely malicious files, we exclude them from the rest of our study due to our lack of confidence if they are truly benign or malicious, and the possibility that they introduce noise into results.

The remaining 83% of downloaded files were *unknown*, i.e. no ground truth exists for them. The software download events were initiated by 141,229 different download processes (identified by

[2]Microsoft, Symantec, TrendMicro, Kaspersky, and McAfee

[3]https://www.symantec.com/security_response/writeup.jsp?docid=2010-011016-3514-99

their hash). Of these processes, 18.5% were labeled as *malicious* and 7.6% as *benign*.

These results are summarized in Table I, whereas Figure 1 and Table II summarize the distribution of malware *families* and *types*, respectively, for the downloaded files that were labeled as malicious. As mentioned in Section II, we obtained the malware family names in Figure 1 by running `AVclass` [18] on our dataset of known malicious files. The figure only shows the top 25 families by number of samples. Overall, our dataset contains malware from 363 different families, according to `AVclass`. However, for 58% of the samples `AVclass` was unable to derive a family name. We provide a brief description of malware types in our dataset in Table II, too. Among all malware types droppers are the most common type in our dataset. Also note the "undefined" type which refers to those malicious files that were assigned generic AV labels (e.g. `Artemis` by McAfee) or labels for which we did not have any mappings available.

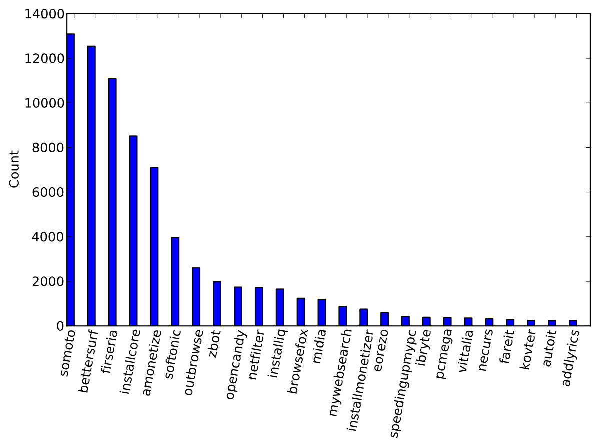

Fig. 1: Distribution of malware families (top 25)

IV. Analysis of Software Download Events

In this section we present an in-depth analysis of the trends we observed in our collection of software download events. We will focus mainly on *what files* are downloaded, and *from where*, leaving a more detailed analysis of *how* files are downloaded (i.e., by what downloading processes and machines) to Section V.

A. File Prevalence

Figure 2 reports the *prevalence* of the downloaded files. We define the prevalence of a downloaded file as the total number of distinct machines that downloaded the file. As can be seen, the prevalence distribution for all downloaded files has a very long tail. It should be noted that this is in part due to the fact that, as discussed in Section II-A, highly popular (i.e., high-prevalence) software files are not collected by AMV's software agents. Also, in Section II-A we explained that file download events are reported only until their prevalence exceeds 20 and if they are executed.

Nonetheless, it is remarkable that among all downloaded files, almost 90% are *downloaded and executed* by only one machine. We can notice from Figure 2 that the long-tail of the prevalence distribution is driven by *unknown* files (i.e., files for which no ground truth is available), which have an extremely low prevalence, compared to benign and known malicious files. We also explored

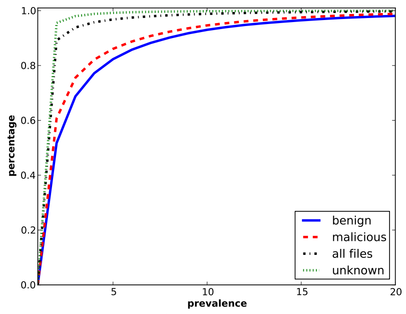

Fig. 2: Prevalence of the downloaded software files

the distribution of different malware types and found that they are very similar to each other.

In aggregate, these *unknown* files have been downloaded and run by 69% of the 1.1 million machines we monitored. Clearly, if a large percentage of the *unknown* files was malicious, it would affect a very large fraction of machines. It is therefore important to study this long tail, given the large number of machines involved.

B. Analysis of Download URLs

Table III reports most contacted effective second-level domains (e2LDs) from which software files were downloaded, according to different criteria. Here we define the popularity of a domain by the *total number of unique machines that contacted the domain to download a file*. The "Overall" column reports the most popular domains in general; and "Benign" and "Malicious" columns report the most popular domains from which benign and malicious files were downloaded, respectively.

Table III shows that many file hosting services, such as softonic.com, cloudfront.com, and mediafire.com, are used both for distributing legitimate software as well as abused by malware distributors. This represents a challenge for malware detection systems that rely on a notion of reputation for the download server/URL (e.g., CAMP [16] and Amico [20]), because the mixed the reputation of the domains/IPs that serve both benign and malicious downloads could cause a significant number of false positives or negatives.

Also, from Table IV, which reports the domains in our dataset that serve the highest number of unique downloaded benign and malicious files, we can notice that there is again a notable overlap among the domains listed under different columns. For example, domains such as softonic.com and mediafire.com host the highest number of both benign and malicious files. This suggests that files downloaded from these software hosting websites are not entirely trustworthy. In fact, comparing the distribution of the Alexa ranks of domains from which benign and malicious files are downloaded, shown in Figure 3 suggests that malicious files aggressively use higher Alexa ranked domains for distribution, such as file hosting services mentioned above.

Table V reports a break-down by malicious file type of the number of files served per domain. From Table V, we can make some interesting observations. Some malicious file types, such as *dropper*, rely heavily on file hosting services to spread, while other types, such as *bot*, seem to employ other sources for their distribution. Also, we can notice that domains used to distribute *fakeavs*, such as 5k-stopadware2014.in, sncpwindefender2014.in,

TABLE I: Monthly summary of data collected by the anti-malware vendor (AMV)

Month	# of Machines	# of Download Events	Download Processes					Downloaded Files					Download URLs		
			Total	Benign	Likely Benign	Malicious	Likely Malicious	Total	Benign	Likely Benign	Malicious	Likely Malicious	Total	Benign	Malicious
January	292,516	578,510	27,265	15.8%	8.4%	16.2%	4.8%	366,981	2.9%	2.8%	7.9%	2.8%	318,834	30.2%	11.6%
February	246,481	470,291	25,001	15.4%	8.2%	16.8%	4.8%	296,362	3.1%	3.1%	8.9%	3.1%	258,410	30.0%	12.2%
March	248,568	493,487	25,497	15.7%	9.1%	16.2%	4.6%	312,662	3.0%	3.1%	9.6%	2.9%	282,179	33.0%	12.3%
April	215,693	427,110	23,078	16.3%	9.3%	19.4%	4.5%	258,752	3.6%	3.4%	12.6%	3.2%	250,634	31.8%	11.3%
May	180,947	351,271	20,071	17.3%	9.5%	19.3%	4.7%	218,156	3.7%	3.5%	12.5%	3.2%	206,095	29.9%	18.9%
June	176,463	351,509	23,799	14.3%	8.1%	20.9%	3.8%	206,309	3.8%	3.4%	14.0%	3.5%	201,920	29.5%	23.0%
July	157,457	323,159	26,304	12.2%	7.2%	16.6%	3.3%	188,564	4.0%	3.7%	12.6%	3.6%	187,315	29.3%	17.9%
Overall	1,139,183	3,073,863	141,229	7.6%	6.6%	18.5%	3.1%	1,791,803	2.3%	2.5%	9.9%	2.3%	1,629,336	29.8%	15.1%

TABLE II: Breakdown of downloaded malicious files per type

Type	Total	Description
Droppers	22.7%	Malware specialized in dropping other files like second-stage malware
PUPs	16.8%	Potentially unwanted program that is distributed as bundled in a benign application
Adware	15.4%	Malicious software specialized in rendering ads without the consent of the user
Trojan	11.3%	Generic name for malware that disguises as benign application and does not propagate
Bankers	0.9%	Malware targeting online banking and specialized in stealing banking credentials
Bots	0.6%	Remotely controlled malware
FakeAVs	0.5%	Malware distributed in form of concealed antivirus software
Ransomware	0.3%	Malware specialized in locking an endpoint (or files) and on asking for a ransom
Worms	0.1%	Malware that auto-replicates and propagates through a victim network
Spyware	0.04%	Malicious software specialized in monitoring and spying on the activity of users
Undefined	31.3%	Generic or unclassified malicious software

TABLE III: Domains with highest download popularity

Overall	# machines	Benign	# machines	Malicious	# machines
softonic.com	64,300	softonic.com	64,300	softonic.com	64,300
inbox.com	49,481	inbox.com	49,481	inbox.com	49,481
humipapp.com	30,966	cloudfront.net	20,065	humipapp.com	30,966
bestdownload-manager.com	30,376	amazonaws.com	17,702	freepdf-converter.com	25,858
freepdf-converter.com	25,858	driverupdate.net	17,505	cloudfront.net	20,065
cloudfront.net	20,065	arcadefrontier.com	15,738	soft32.com	18,241
soft32.com	18,241	mediafire.com	14,336	amazonaws.com	17,702
amazonaws.com	17,702	uptodown.com	13,431	arcadefrontier.com	15,738
driverupdate.net	17,505	ziputil.net	12,972	free-fileopener.com	15,179
arcadefrontier.com	15,738	rackcdn.com	12,893	mediafire.com	14,336

TABLE IV: Number of files served per domain (top 10 domains)

Benign downloads	#files	Malicious downloads	#files
cnet.com	1,574	softonic.com	21,355
sourceforge.net	1,357	nzs.com.br	8,009
mediafire.com	774	cloudfront.net	7,416
informer.com	749	baixaki.com.br	4,564
softonic.com	569	cdn77.net	4,043
wildgames.com	503	mediafire.com	3,857
lenovo.com	432	softonic.com.br	3,251
naver.net	361	files-info.com	2,559
ricoh.com	327	v47installer.com	2,545
tistory.com	305	downloadaixeechahgho.com	2,266

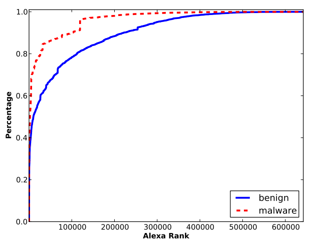

Fig. 3: Distribution of the Alexa ranks of domains hosting benign and malicious files

webantiviruspro-fr.pw, etc., embed social engineering tactics in the domain name themselves. Another interesting point, which seems to confirm findings reported in [13], is that adware usually spreads by utilizing free live streaming services, such as media-watch-app.com, trustmediaviewer.com, vidply.net, and etc.

C. File Signers and Packers

The use of a simple static analysis of the downloaded files can in some cases provide valuable information about their true nature. In this section, we explore if downloaded software is typically signed and by what signers[4]. Furthermore, we analyze what files are packed, and with what packing software. The information about software signatures and packer identification have been obtained from both VirusTotal.com as well as from AMV's internal software analysis infrastructure.

[4]https://msdn.microsoft.com/en-us/library/ms537361(v=vs.85).aspx

TABLE V: Popular download domains per type of malicious file

Bot	# files	Dropper	# files	Adware	# files	FakeAV	# files
mediafire.com	70	softonic.com	4,599	media-watch-app.com	1,936	rackcdn.com	685
4shared.com	35	files-info.com	2,072	media-buzz.org	1,911	5k-stopadware2014.in	4
naver.net	34	mediafire.com	845	trustmediaviewer.com	1,620	sncpwindefender2014.in	3
ge.tt	23	softonic.com.br	732	media-view.net	1,608	webantiviruspro-fr.pw	3
sharesend.com	13	d0wnpzivrubajjui.com	601	pinchfist.info	1,080	12e-stopadware2014.in	3
co.vu	12	vitkvitk.com	489	media-viewer.com	919	zeroantivirusprojectx.nl	3
gulfup.com	11	cloudfront.net	414	dl24x7.net	848	wmicrodefender27.nl	3
hinet.net	10	softonic.fr	356	zrich-media-view.com	749	qwindowsdefender.nl	3
wipmsc.ru	10	softonic.jp	334	vidply.net	722	updatestar.com	3
f-best.biz	9	downloadnuchaik.com	302	mediaply.net	654	alphavirusprotectz.pw	3

Table VI reports the percentage of *benign*, *unknown*, and *malicious* files that are signed. According to Table VI some malicious file types, such as *dropper* and *pup*, tend to carry a valid software signature, while some others, such as *bot* and *banker*, are rarely signed. This might be because malware types such as *dropper* and *pup* are usually the initiators of infections and are often directly downloaded via a web browser with user consent (e.g. via social engineering attacks). Signing these malicious files may be a way to persuade the users about their legitimacy, and perhaps also to thwart AV detection. To verify this intuition, the "From Browsers" column reports the percentage of signed files that are downloaded via popular web browsers. A row by row comparison reveals that malicious files that are directly downloaded by browsers are more likely to be signed. This is also true for benign and unknown files.

Another interesting observation is the percentage of signed malicious files is much higher than signed benign software. This again may be due to the fact that malware distributors try hard to trick users into running their software and evade AV detection.

Table VII shows the number of unique signers that signed different types of malicious files. We also compare the signers of different types of malicious files with benign files. The "In common with benign" column shows the number of common signers between malicious and benign files. For example, out of 248 signers that signed different droppers, 46 also signed some benign files and consequently 202 exclusively signed malicious files only. We further provide examples of these signers in Table VIII. The "Top Signers" column list the names of the top 3 signers for each type of file. For different types of malware, the table reports the top 3 signers that are in common with benign files as well as top 3 signers that exclusively signed malware files. Similar information is also provided for benign files. One interesting case is the droppers' top signer being "Softonic International", which shows that some popular software download websites may distribute bundled applications that include malicious software. Table VIII also shows some of the top signers that exclusively signed either malware or benign files as well as the number of files signed by each signer. Note that file signer information could be utilized to gain more insight on the true nature of completely unknown files. In Section VI, we present a system that uses signers data (along side other information) to label unknown files.

A more detailed view of what signers are in common between malicious and benign files is given in Figure 4. The figure includes a count of how many malicious/benign files are signed by each signer. Among the interesting results are malicious files that are signed by seemingly reputable signers such as AVG Technologies and BitTorrent, which further manual analysis revealed that they are mostly PUPs.

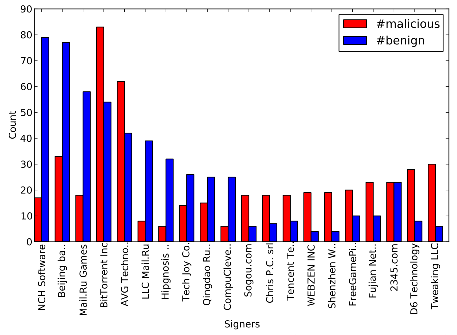

Fig. 4: Common signers between malicious and benign files

TABLE VI: Percentage of signed benign, unknown, and malicious files

Type	Overall		From Browsers	
	# of Files	Signed	# of Files	Signed
Trojan	22,413	59.9%	12,827	81.3%
Dropper	43,423	85.6%	33,820	95.4%
Ransomware	563	44.4%	313	68.7%
Bot	1,092	1.5%	268	2.2%
Worm	201	5.5%	57	12.3%
Spyware	80	21.2%	40	25.0%
Banker	1,719	1.2%	272	1.8%
FakeAV	987	2.8%	446	4.5%
Adware	29,345	43.1%	8,792	91.8%
PUP	31,018	76.0%	21,792	79.6%
Undefined	60,609	65.1%	42,614	71.3#
Benign	43,601	30.7%	30,346	32.1%
Unknown	1,626,901	38.4%	1,227,241	42.1%
Malicious	191,450	66%	121,241	81%

TABLE VII: Common signers among malicious file types

Type	# Signers	In common with benign
Trojan	426	71
Dropper	248	46
Ransomware	14	4
Banker	11	2
Bot	15	3
Worm	7	1
Spyware	9	4
FakeAV	14	4
Adware	532	77
PUP	691	108
Undefined	1,025	339
Total	1,870	513

We also investigate file packers. Interestingly, our analysis reports that both benign and malicious downloaded files are equally packed, with respectively 54% and 58% of them processed with a known packing software. In addition, similarly to what previously discussed with the signers, many packers are used to concurrently pack both benign and malicious software: out of 69 unique packers adopted by our collection of software downloads, more than half of them (35) are equally used in both benign and malicious cases. For example, we observed many benign and malicious files that are packed by INNO, UPX, AutoIt, and etc. This makes detection systems that solely rely on packing information falling short in terms of accuracy. Among the packers that are exclusively used on malicious files, we observed Molebox, NSPack, Themida, for example. In addition, a simple breakdown of packers per type of malicious files does not reveal a discriminating factor among them because files appear to be commonly packed by similar software.

V. Downloading Processes and Machines

In this section, we study what type of files are typically downloaded by different processes. For instance, we are interested in answering questions such as: What category of processes (e.g. browsers, windows processes, etc.) contribute more to malicious downloads? What files are typically downloaded by benign software?, and etc.

A. Analysis of Benign Processes that Download Executables

For our first measurements in this section, we focus on different categories of file downloading processes. We group the client processes into five broad classes, namely *browsers*, *windows processes*, *Java* processes (i.e., Java runtime environment software), *Acrobat Reader* processes, and *all other* processes. The reason we consider Java and Acrobat Reader processes separately is that these two software are notoriously vulnerable and have been exploited by malware distributors many times in the past (e.g., via exploit kits like Nuclear, Fiesta or Angler[5]).

To label a process according to the above labels, we leverage the name of the executable file on disk from which the process was launched. For instance, any process with the name of `firefox.exe` is labeled as the Firefox web browser. To this end, we compiled a list of different file names observed in the wild for each process category. At the same time, we need to take into account the fact that malware may in some cases disguise itself as a legitimate process. Therefore, in our measurements we will focus on the download behavior of *known benign* processes only, whose related executable file hash matches our whitelist.

Table X reports, per each category, the number of distinct process versions (counted as the number of distinct hashes for the files from which the processes are launched), the overall number of machines on which those processes were run, the number of executable files downloaded (and then executed) by those processes, the number of machines that became infected due to malicious file downloads initiated by the processes, and the distribution of malicious types for the downloaded file.

From Table X, we can immediately notice that most files downloaded by Java and Acrobat Reader are in fact malicious, and cause the related downloading machines to become infected. Specifically, of the 1,080 machines that run an instance of Acrobat Reader that was observed to initiate an executable file download, 78.52% downloaded and executed at least one of the 696 malicious files, thus becoming infected. We can also see that none of the executable files downloaded by Acrobat Reader processes could be labeled as benign, and that 264 files could not be labeled with existing ground truth, thus remaining *unknown*. However, it is likely that the vast majority (if not all) of these files are also malicious.

Similarly, Java processes mostly download malicious file. The 25 benign downloads shown in Table X appeared to be outliers, which we therefore investigated more closely. From a manual investigation these appeared to be licit bundled software like Java applets for sound recording or custom calenders.

Windows system processes can also initiate the download (and execution) of new malicious file. Because, as mentioned earlier, we only consider known benign processes, we suspect that the malicious downloads are due to these processes being exploited (either remotely or locally). The number of machines affected by these malicious downloads is quite significant. In fact, of the 429,593 machines on which an executable file download was initiated by a Windows process, 27.71% downloaded and executed at least one of the 68,767 malicious files we observed overall. This tends to suggest that a consistent number of Windows machines seem to run not properly patched Windows processes, representing then a primary form of infection.

[5]https://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-evolution-of-exploit-kits.pdf

TABLE VIII: Top signers of different file types

Type	**Top signers**	**Top common signers with benign files**	**Top signers exclusive to malware files**
trojan	Somoto Ltd., Somoto Israel, RAPIDDOWN	Open Source Developer, Binstall, Rspark LLC	Somoto Ltd., Somoto Israel, RAPIDDOWN
dropper	Softonic International, Somoto Israel, Sevas-S LLC	Softonic International, RBMF Technologies LLC, Open Source Developer	Somoto Israel, Sevas-S LLC, SecureInstall
ransomware	ISBRInstaller, WorldSetup, UpdateStar GmbH	WorldSetup, UpdateStar GmbH, AppWork GmbH	ISBRInstaller, Trusted Software Aps, The Nielsen Company
bot	Benjamin Delpy, Supersoft, Flores Corporation	Nir Sofer	Benjamin Delpy, Supersoft, Flores Corporation
worm	70166A21-2F6A-4CC0-822C-607696D8F4B7, JumpyApps, Xi'an Xinli Software Technology Co.		70166A21-2F6A-4CC0-822C-607696D8F4B7, JumpyApps, Xi'an Xinli Software Technology Co.
spyware	Refog Inc., R-DATA Sp. z o.o., Mipko OOO	Refog Inc., Video Technology, Valery Kuzniatsou	R-DATA Sp. z o.o., Mipko OOO, Ts Security System - Seguranca em Sistemas Ltda
banker	WEBPIC DESENVOLVIMENTO DE SOFTWARE LTDA, JDI BACKUP LIMITED, Wallinson	Open Source Developer, TLAPIA	WEBPIC DESENVOLVIMENTO DE SOFTWARE LTDA, JDI BACKUP LIMITED, Wallinson
fakeav	UpdateStar GmbH, Webcellence Ltd., ISBRInstaller	UpdateStar GmbH, The Phone Support Pvt. Ltd., 2345.com	Webcellence Ltd., ISBRInstaller, William Richard John
adware	Apps Installer SL, SITE ON SPOT Ltd., Open Source Developer	SITE ON SPOT Ltd., Open Source Developer, Binstall	Apps Installer SL, Tuto4PC.com, ClientConnect LTD
pup	Binstall, Somoto Ltd., SITE ON SPOT Ltd.	Binstall, SITE ON SPOT Ltd., Perion Network Ltd.	Somoto Ltd., Amonetize ltd., Firseria
undefined	ISBRInstaller, JumpyApps, Somoto Israel	Binstall, UpdateStar GmbH, BoomeranGO Inc.	ISBRInstaller, JumpyApps, Somoto Israel
malicious (total)	Softonic International, Binstall, Somoto Ltd.	Softonic International, Binstall, SITE ON SPOT Ltd.	Somoto Ltd., ISBRInstaller, Somoto Israel
Type	**Top signers**	**Top common signers with malware files**	**Top signers exclusive to benign files**
benign	Lenovo Information Products (Shenzhen) Co., MetaQuotes Software Corp., Rare Ideas	Lenovo Information Products (Shenzhen) Co., MetaQuotes Software Corp., Rare Ideas	TeamViewer, Blizzard Entertainment, Lespeed Technology Ltd.

TABLE IX: Top signers that exclusively signed benign or malicious files

Benign	**# Files**	**Malware**	**# Files**
TeamViewer	209	Somoto Ltd.	5,652
Blizzard Entertainment	77	ISBRInstaller	5,127
Lespeed Technology Ltd.	71	Somoto Israel	5,062
Hamrick Software	66	Apps Installer SL	5,049
Dell Inc.	59	SecureInstall	2,694
Google Inc	59	Firseria	2,474
NVIDIA Corporation	58	Amonetize ltd.	1,932
Softland S.R.L.	52	JumpyApps	1,896
Adobe Systems Incorporated	48	ClientConnect LTD	1,761
Recovery Toolbox	43	Media Ingea SL	1,671

As expected, the vast majority of web-based executable file downloads are initiated by browsers (see "Browsers" in Table X). Table XI reports the number and type of files downloaded by popular browsers. Somewhat surprisingly, these results show that Internet Explorer (IE) could be considered as the "safest" browser, judging by the percentage of malicious downloads it initiated and the percentage of infected machines. In fact, of the 411,138 machines that used IE to download one or more executable files, only 18% became infected due to an IE-initiated malicious file download. On the other hand, of the 344,994 machines that were observed using Chrome to download an executable file, 31.92% became infected, which represents the highest rate of infection across all popular browsers. We should notice, though, that these results are based on the *known malicious* files, and that the large number of *unknown* file downloads by both Chrome and IE could tilt the scale, if complete ground truth was available. Nonetheless, it is significant that known malicious software tends to affect more Chrome users, than IE users.

From Table X we can also see that the most represented malicious file type (if we exclude *undefined* malicious files) downloaded by browsers is *droppers*. This can be explained by the fact that droppers are first-stage malware, which are typically leveraged to download additional malware once the machine is infected. This observation is also in accordance with the results we presented in Table VI, which shows that 85.6% of droppers have a valid software signatures, which is likely used as a way to evade current malware defenses and persuade users into running the software.

B. Analysis of Malicious Processes

To extend our experiments of download behavior of processes, now we turn our attention to download behavior of malicious processes. In particular, we categorize the malicious processes according to their malware types and demonstrate what is typically downloaded by each malware type. Table XII has a similar structure as Table X, but instead of process categories, it explores different malware types. In this case the "Processes" column reports the number of processes associated to each malware type.

The results in Table XII indicates that processes of a specific malware type download other malwares of the same type in majority of cases. However, there are some unexpected behaviors in terms of download behaviors of different malware types. For example, many malware types, even the most specific ones, such as ransomware, fakeav, etc., seem to download other completely different malware types. The reason behind this depends on how the malware operates on the system and what is its intention. For example, a fakeav could lure victims into buying other things, but it could simultaneously drop another piece of malware to take full advantage of the victim. One thing that is clear, however, is that if a machine is infected with somewhat less dangerous malware initially, such as adwares and PUP applications, there is a good chance that the machine gets infected with more aggressive and damaging malware.

From Adware/PUP to Malware

Adware and PUPs are often considered as "less damaging" malware. In fact, PUP stands for *potentially unwanted* program (sometimes also called potentially unwanted application, or PUA). However, some studies (e.g., [21]) have suggested that running adware/PUPs increases the chances that a machine will be later infected with more damaging malware (e.g., ransomware, bots, etc.). In this section, we provide measurements that aim to quantitatively support this suspicion.

First, we can analyze the results reported in Table XII. As we can see, both adware and PUP processes tend to mostly download other adware or PUP software. However, we can also see that, for both adware and PUP processes, more than 6% of the downloaded executable files are trojans. In addition, almost 3% of the files downloaded by adware are droppers, whereas the same figure goes up to 4.57% for PUPs. Furthermore, both adware and PUPs in some cases directly download ransomware, bankers, and other highly damaging malware.

Besides direct downloads, adware/PUP process could also be the cause of indirect infections. For instance, adware processes often display ads from low-reputation ad networks, thus exposing users to malvertisement [21]. Consequently, if a user clicks on a malicious ad, she may be redirected, via her default web browser, to downloading other malware [11]. To include these indirect downloads into our analysis, we proceed as follows. Let m be a machine that has downloaded and executed an adware/PUP at time t_1. We then check if, after t_1, m downloads and executes other types of malicious software (thus excluding other adware, PUP, and *undefined* malicious files). We repeat this process for each

TABLE X: Download behavior of benign processes (divided by process category)

Processes	Machines	Downloaded files			Infected Machines	Malware type of downloaded malicious files
		unknown	benign	malicious		
Browsers						
1,342	799,342	1,120,855	28,265	113,750	24.44%	dropper=28.05%, pup=18.55%, trojan=10.48%, adware=7.36%, fakeav=0.35%, ransomware=0.27%, banker=0.23%, bot=0.22%, worm=0.05%, spyware=0.03% (undefined=34.43%)
Windows Processes						
587	429,593	368,925	23,059	68,767	27.71%	dropper=25.42%, pup=17.75%, trojan=11.75%, adware=5.80%, banker=1.23%, bot=0.73%, ransomware=0.37%, fakeav=0.11%, worm=0.08%, spyware=0.06% (undefined=36.7%)
Java						
173	2,977	227	25	488	33.36%	trojan=45.29%, bot=15.78%, dropper=12.30%, banker=6.97%, ransomware=4.30%, pup=1.02%, worm=0.82% (undefined=12.54%)
Acrobat Reader						
9	1,080	264	0	696	78.52%	trojan=39.51%, dropper=23.71%, banker=15.80%, bot=8.19%, ransomware=3.74%, fakeav=1.44%, spyware=0.43%, worm=0.29% (undefined=6.89%)
All other processes						
8,714	112,681	68,334	5,642	15,440	31.24%	pup=22.57%, dropper=17.22%, trojan=11.34%, adware=8.38%, fakeav=5.03%, banker=1.20%, bot=0.79%, ransomware=0.44%, worm=0.30%, spyware=0.02% (undefined=32.71%)

TABLE XI: Download behavior of benign browser processes

Browser	# Processes	# Machines	Unknown Files	Benign Files	Malicious Files	Infected Machines
Firefox	378	86,104	104,237	7,411	21,443	26.00%
Chrome	528	344,994	460,214	17,623	73,806	31.92%
Opera	91	4,337	4,749	534	1,567	27.83%
Safari	17	1,762	2,579	117	422	18.56%
IE	307	411,138	561,769	13,801	48,206	18.09%

machine m that ran adware/PUP software, and compute the time delta between the adware/PUP infection and the download of other types of malware. Figure 5 shows a CDF for the obtained results. As we can see, more than 40% of these machines download and execute other malware on the same day (day 0) in which they downloaded and executed the adware/PUP software. After only five days from the execution of adware/PUP, the number of those machines infected with other malware types exceeds 55%. On the contrary, let's consider the same measurements for machines that at a given time t_1 download a benign software (and that have not been observed to download malicious files in the past). What we aim to show is that if a machine does not run adware/PUPs, it is much less likely to download malware in the immediate future. On the same Figure ("benign" line), after five days from the benign software download event, only 20% machines downloaded malicious files (excluding adware and PUPs, for comparison with "PUP" and "adware" lines).

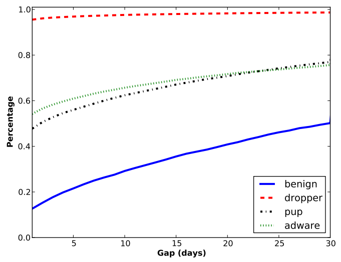

Fig. 5: Time delta between downloading benign/adware/pup/dropper and other malware

Dopper-driven Malware Infections

Droppers play a significant role in malware infections [10]. To provide additional information on the behavior of malicious dropper processes, we proceed in a way similar to Section V-B. For instance, we measure how long it takes for droppers to infect users. To this end, we compute the time gap between the first time a machine downloads (and executes) a dropper and a subsequent malware download. Notice that we exclude adware, PUPs, and undefined from this measurement, so that we can compare the results directly to the transition between adware/PUPs to other malware types discussed above.

Figure 5 (dashed red line) reports our results. As anticipated, a machine that is infected with a dropper is almost certain to download and execute malware in the following days. In particular, by comparing the dropper, adware, and pup curves in Figures 5, we can see that there is a much shorter time gap between downloading a dropper and another malware, than downloading an adware/PUP and then another malware.

VI. Exploring and Labeling Unknown Files

As reported in Section II (see Table I), the majority of file downloads (about 83%) are *unknown*, in that no ground truth is available about their true nature, even two years after they were first observed. As these unknown files involve a significant number of users who downloaded them (69% of all machines in our data downloaded some unknown files), it is of utmost importance to be able to reason at least about some of them. In fact, if these *unknown* files were malicious, they would infect the vast majority of the machine population. Therefore, in this section we explore the characteristics of unknown files. In addition, we aim to build a rule-based classifier that is able to accurately label a significant fraction of these unknown files as either *malicious* or *benign*.

A. Exploring the Characteristics of Unknown Files

Table XIII shows the top 10 domains from which unknown files were downloaded, whereas Figure 6 plots the distribution of the Alexa rank of all domains hosting unknown files. Table XIV shows what benign processes tend to download most of these files. Naturally, most unknown files are downloaded via web browsers. However, we can see that a large number of unknowns are downloaded by Windows processes as well. This is alarming, if we consider that Table X also shows that a large majority of downloaded files by Windows processes for which ground truth is available are actually malicious. Take Acrobat Reader as an extreme example (again, from Table X). Of the 960 downloaded files, 696 are known to be malicious and none are known benign. This means that all of the remaining 264 unknowns, reported in Table XIV, are also highly likely malicious.

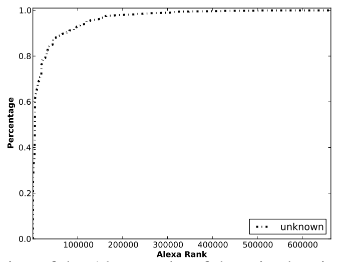

Fig. 6: Distribution of the Alexa ranks of domains hosting unknown files

B. Labeling Unknown Files

During our analysis, we noticed that in some cases a simple analysis of the properties of unknown files would allow us to identify, with high confidence, their true nature. For instance, an executable file that is signed by a software signer that in the past has

TABLE XII: Download behavior of different types of malicious processes

Processes	Machines	Downloaded files			Type of downloaded malicious files
		unknown	benign	malware	
Trojan					
3,442	11,042	1,265	73	4,168	trojan=51.90%, adware=11.80%, dropper=10.94%, pup=8.25%, banker=4.25%, bot=0.89%, ransomware=0.34%, fakeav=0.12%, worm=0.10% (undefined=11.42%)
Dropper					
4,242	10,453	1,565	267	2,992	dropper=39.10%, trojan=16.78%, pup=10.26%, adware=8.46%, banker=7.59%, bot=1.34%, ransomware=0.47%, worm=0.30%, fakeav=0.20%, spyware=0.07% (undefined=15.44%)
Ransomware					
136	332	7	0	147	ransomware=80.95%, trojan=9.52%, dropper=3.40%, banker=1.36% (undefined=4.76%)
Bot					
323	689	81	2	394	bot=64.72%, trojan=15.99%, dropper=4.57%, banker=4.31%, pup=2.54%, ransomware=1.27%, worm=0.51%, adware=0.25%, fakeav=0.25% (undefined=5.58%)
Worm					
67	164	4	0	69	worm=72.46%, banker=8.70%, trojan=4.35%, dropper=4.35%, bot=1.45%, pup=1.45% (undefined=7.25%)
Spyware					
7	19	2	1	6	spyware=66.67%, trojan=16.67% (undefined=16.67%)
Banker					
484	1,146	47	5	525	banker=76.00%, trojan=14.48%, dropper=4.00%, worm=0.57%, fakeav=0.38%, ransomware=0.19%, bot=0.19%, adware=0.19% (undefined=4.00%)
Fakeav					
43	81	1	0	53	fakeav=56.60%, trojan=22.64%, banker=9.43%, dropper=7.55% (undefined=3.77%)
Adware					
2,862	16,509	2,934	98	6,078	adware=66.24%, pup=9.97%, trojan=6.65%, dropper=2.91%, banker=0.13%, bot=0.03% (undefined=14.07%)
PUP					
5,597	32,590	6,757	199	16,957	adware=58.64%, pup=22.91%, trojan=6.30%, dropper=4.57%, ransomware=0.02%, bot=0.01%, banker=0.01%, fakeav=0.01% (undefined=7.54%)
Undefined					
8,905	29,216	6,343	499	8,329	adware=6.52%, pup=5.53%, dropper=3.77%, trojan=3.36%, banker=0.36%, bot=0.22%, worm=0.06%, ransomware=0.04%, spyware=0.04%, fakeav=0.01% (undefined=80.09%)
Overall					
26,108	93,644	18,473	1,044	36,402	adware=39.04%, pup=14.18%, trojan=10.97%, dropper=7.14%, banker=1.94%, bot=0.90%, ransomware=0.39%, worm=0.18%, fakeav=0.11%, spyware=0.02% (undefined=25.13%)

TABLE XIII: Top 10 Download Domains

Domain	# downloads
inbox.com	75,946
humipapp.com	43,365
bestdownload-manager.com	37,398
freepdf-converter.com	32,276
coolrom.com	27,833
soft32.com	27,229
gamehouse.com	24,498
arcadefrontier.com	24,191
driverupdate.net	21,370
zilliontoolkitusa.info	19,550

TABLE XIV: Categories of Download Processes

Downloading process type	#Unknowns
Browser	1,120,855
Windows	368,925
Java	227
Acrobat Reader	264
Other benign processes	36,059
Total	1,486,961

signed many malicious files but no benign software is also likely malicious. Conversely, an executable file that is signed by a reputable software developer, which has exclusively signed benign files in the past, is highly likely benign. Similarly, a file that is packed with a packer/obfuscation tool that is known to be used exclusively to protect malicious files from AV detection is highly likely malicious. Overall, we have identified a set of eight intuitive and easy-to-measure features, summarized in Table XV, that we can use to label many in-the-wild unknown file downloads with high accuracy. In the following, we present a novel rule-based classification system that uses these features to mine past file download events and automatically extract simple *human-readable* file classification rules.

C. Generating Human-Readable Classification Rules

Recently, authors of [3] explored the importance of interpretability in machine learning systems and suggested that the decisions of such systems should be explainable. To this end, we aim to generate simple human-readable classification rules and proceeded as follows. First, we use past file download observations whose ground truth is known as a training dataset. Then, we use the PART rule learning algorithm [4] to derive a set of human-readable classification rules based on the features reported in Table XV. Finally, we prune the classification rules output by PART to only retain highly accurate rules (i.e., rules with low error rate). Unlike other machine learning algorithms (e.g., support vector machines (SVMs), neural networks, etc.), this approach generates easy-to-interpret classification rules that can be reviewed and modified by threat analysts. The following is an example of a simple classification rule based on the described features:

IF (file's signer is "Shanghai Gaoxin Computer System Co.") AND (file is packed by "NSIS") $\rightarrow$ file is malicious.

This rule was learned from more than 50 instances of malicious files downloads, and does not match any of the tens of thousands of benign downloads we observed.

D. Evaluation of Classification Rules

To systematically evaluate the efficacy of the human-readable classification rules, we proceeded as follows. We first describe how we prepared the evaluation data, and then explain how we filtered the generated rules to select only the rules with low error rates.

- *Training dataset:* To produce the rules, a training dataset of labeled feature vectors is generated over all known benign and malicious files from download events observed during a training time window T_{tr} (e.g. 30 days).
- *Testing dataset:* The performance and accuracy of the rules are evaluated using a test dataset. The test dataset contains known benign and malicious files from download events gathered from a test time window T_{ts} that immediately follows the training time window T_{tr}. Importantly, we ensure that the intersection between training and test file download events is empty, so none of the samples from testing dataset are ever used for extracting the rules. Furthermore, this perfectly simulates how the system is used in operational environments; rules generated based on past events are used to classify new, unknown events in the future.
- *Unknown files dataset:* The goal is to utilize the extracted rules to classify previously unknown files. Therefore, we extract the truly unknown files during T_{ts} and generate a dataset of unknown files. Obviously, there is no ground truth available whatsoever about any of the files in this dataset. We use the rule-based classifier to reduce the number of unknowns in this dataset by classifying them as either benign or malicious. Due to lack of ground truth, the correctness of classification of unknown files cannot be verified. However, we measure their properties, and manually analyze some of the samples to attempt to determine the correctness of their new labels.

We now present our evaluation results. To this end, we consider a month of download events as our training time window and extract the classification rules. Then we evaluate the performance of these rules in terms of true positives (TP) and false positives (FP). Finally, we report the number of completely unknown files that the rules classify during T_{ts}.

We evaluated the rule-based classification system on different T_{tr} and T_{ts} periods. Table XVI reports a summary on the number of

TABLE XV: Features Description

Feature	Explanation
File's signer	The entity who signed a downloaded file.
File's CA	The certification authority in the chain of trust of signers for the downloaded file
File's packer	The packer software used to pack the downloaded file, if any
Process's signer	The signer of the process that downloaded the file
Process's CA	The CA of the process that downloaded the file
Process's packer	The packer software used to pack the downloading process
Process's type	The type of process that downloaded the file (browser, windows process, etc.)
Download domain's Alexa rank	The Alexa rank of the domain from which the file was downloaded

TABLE XVI: Statistical information about extracted rules during different T_{tr}

T_{tr}	Overall # of rules	τ	Selected rules	Rules composition	
				# of benign	# of malicious
Feb	1,766	0.0%	1,020	889	131
		0.1%	1,031	894	137
Mar	1,680	0.0%	1,148	970	178
		0.1%	1,162	976	186
Apr	1,272	0.0%	1,054	872	182
		0.1%	1,070	875	195
May	1,476	0.0%	974	791	183
		0.1%	986	793	193
Jun	944	0.0%	740	577	163
		0.1%	753	585	168
Jul	1,376	0.0%	937	755	182
		0.1%	953	763	190

extracted rules per different training time. As mentioned before, we use a subset of all rules generated by the PART algorithm [4], i.e. by including only those rules with error rates less than a maximum (configurable) error threshold τ. The value of τ should be properly chosen as it impacts the performance of the classifier. To compare the results, for every T_{tr}, we extract the rules based on different configurations for τ during training. For example, for month of March as T_{tr} and by choosing the rules that have no training error ($\tau = 0.0\%$), 1,148 rules (out of 1,680 rules) will be selected. The detection results of these different settings are then compared to each other. Column "rules composition" shows the number of rules that result in a *benign* or *malicious* label, among the 1,148 selected rules.

By increasing τ, the number of rules and samples that match them increases, at the expense of the trade-off between TPs and FPs. Therefore, we limit ourselves to experimenting with low values of τ. Table XVI shows the results for different number of rules extracted per month for $\tau = 0.0\%$ and $\tau = 0.1\%$. The evaluation results for these two different rule sets are reported in Table XVII.

In this table, each row corresponds to an experiment in which rules are extracted according to a specific configuration (see Table XVI) from download events during a month T_{tr}. The rules are then tested against samples in the test dataset from T_{ts} (see column "test dataset"). More specifically, under "test dataset", columns "# malicious" and "# benign" report the size of the benign and malicious test samples which matched the rules. Note that those test samples that do not match any rules are not considered, because the rule-based classifier cannot label them. Therefore, the TP and FP rates are computed only over the test samples that actually match at least one rule. Column "# FP Rules" reports the number of rules that cause FPs. We will discuss these rules in Section VII.

The rule-based classifier also needs to deal with cases in which conflicts occur among multiple rules that match the feature vector of a file. In this situation, some rules identify the file as benign while some other conflicting rules classify the same file as malicious. In our rule-based classification system, should a conflict occur when classifying a file, we "reject" the file and do not provide any classification to avoid inaccurate results. This is another advantage of using our system over regular decision trees in which rejecting some classification decisions of the decision tree is not an intuitive task. Rejecting a file in case of conflicting rules helps in reducing the errors (FPs), as we will demonstrate shortly.

As it can be seen from Table XVII, rules extracted with maximum error rate of $\tau = 0.1\%$ consistently produced accurate detection results in terms of combination of TPs and FPs during all T_{ts}. Overall, using this setting, the rule-based classifier achieved $TP > 95\%$ and $FP < 0.32\%$ on test datasets. Please note that due to rejecting conflicting and inaccurate classifications, in some cases during the same T_{ts}, the number of rules that produce FPs decreases even after selecting more rules by increasing τ. Furthermore, "unknowns dataset" column in Table XVII reports the percentage of completely unknown files from period T_{ts} that match the extracted rules, and hence are now classified ("matched" column). Exact numbers of matched unknown files classified as benign or malicious are also shown.

Also, note the percentage of truly unknown files that match the extracted rules in each τ setting. More rules are chosen as τ increases, and consequently, more unknown files match the rules. However, as discussed before, after a certain τ value, adding more rules causes deterioration of TPs and FPs. This is because if too many inaccurate rules with higher error rates are added to the set of extracted rules, they could lead to misclassifications. In addition, the possibility that files match conflicting rules increases and the classifier rejects these files. So even though we can label more truly unknown files with more rules, the final classification of these files might not be very accurate. As $\tau = 0.1\%$ produced the best performance on the test dataset, hence we use the same setting for classifying the unknown files.

As mentioned before, this is one of the advantages of our rule-based classification system over regular decision trees, as the whole decision tree, which contains some less accurate branches, does not need to be used. Overall, from February to August, 406,688 previously unknown files were classified as either benign or malicious by the system. This number accounts for 28.30% of total unknown files observed during this period.

TABLE XVII: Evaluation results and classification of unknown files using rule-based classifier (conflicts are handled by rejecting the test and unknown files)

T_{tr} - T_{ts}	τ	Test dataset (extracted during T_{ts})					Unknowns dataset (extracted during T_{ts})			
		# malicious	TP	# benign	FP	# FP rules	# unknowns	matches	# malicious	# benign
Jan - Feb	0.0%	3,590	96.72%	1,401	0.07%	1	292,793	24.08%	68,200	2,312
	0.1%	3,647	96.45%	2,718	0.00%	0		24.14%	68,368	2,312
Feb - Mar	0.0%	3,045	97.59%	2,051	0.39%	8	301,715	29.22%	68,165	20,005
	0.1%	3,070	97.60%	2,830	0.32%	9		29.22%	68,165	20,005
Mar - Apr	0.0%	4,793	97.98%	1,367	0.37%	6	242,810	22.06%	51,096	2,470
	0.1%	4,842	99.61%	2,315	0.30%	8		22.23%	51,504	2,467
Apr - May	0.0%	3,001	92.01%	1,873	0.05%	1	197,526	36.92%	46,651	26,266
	0.1%	7,203	96.96%	2,267	0.13%	2		38.03%	49,014	26,108
May - Jun	0.0%	3,834	90.53%	2,038	0.15%	4	191,574	32.05%	40,600	20,794
	0.1%	7,895	96.64%	2,597	0.12%	4		34.46%	43,175	22,846
Jun - Jul	0.0%	7,200	95.39%	2,414	0.25%	7	177,255	30.71%	35,530	18,906
	0.1%	7,202	95.28%	2,837	0.18%	6		31.54%	35,693	20,207

VII. Discussion

As mentioned earlier, one of the advantages of our rule-based classification system is that the rules are human-readable and can be easily reviewed by an analyst. In the following, we report few example rules that led to the most true positives, as well as those rules that sometimes caused misclassifications. Below, we list three example rules that are responsible for correctly labeling many malicious downloads:

1) IF (file's signer is "SecureInstall") $\rightarrow$ file is malicious.
2) IF (file's signer is "Apps Installer S.L.") AND (downloading process's signer is "Microsoft Windows") AND (file's CA is "thawte code signing ca - g2") $\rightarrow$ file is malicious.
3) IF (file is not signed) AND (downloading process is "Acrobat Reader") $\rightarrow$ file is malicious.

The above-mentioned rules follow our reported measurement results. For example, Table X showed that many malware files are downloaded by benign Windows processes. It also reported that files downloaded by Acrobat Reader are malware.

Rules that produce some false positives include the following:

1) IF (file's signer is "mail.ru games") $\rightarrow$ file is malicious.
2) IF (file is not signed) AND (downloading process's signer is "Amonetize ltd.") AND (file's packer is "NSIS") $\rightarrow$ file is malicious.
3) IF (file is not signed) AND (Alexa rank of file's URL is between 10,000 to 100,000) AND (downloading process is benign) AND (file's packer is "aspack") $\rightarrow$ file is malicious.

It should be noted that some of the classifications that we count as false positives may actually be due to the presence of noise in our ground truth. For instance, let us consider rule (2) above. "Amonetize ltd" is related to a family of adware and PUP software. Therefore, executable files downloaded from a process signed by "Amonetize ltd" may in fact be themselves malicious.

Additionally, 33% of benign (according to our ground truth) test samples were downloaded by malware processes or from malicious URLs. These may therefore be false positives due to noise in the whitelist. Overall, these observations indicate that it is possible that the false positives we obtained may be somewhat overestimated.

Our evaluation results indicate that signers of downloaded files play an important role in our rule-based classifier. In fact, the *file signer* feature appeared in 75% of all rules. The other three most useful features, in order, are the file's packer, downloading process type, and downloading process's signer, which appeared in 8%, 5%, and 4% of all rules. Another interesting observation is that our classifier does not heavily rely on the feature related to the Alexa rank of the domains, as it appeared in 1.4% of the rules. This is in accordance with our previous measurement analysis that showed many benign file hosting websites tend to host malicious files along side benign files. Also, we noticed that simple rules containing one feature are less error prone and composed 89% of rules, for $\tau = 0.1\%$.

Analysis of Test Dataset Results. Among the correctly classified malicious test samples, 45% of files are droppers, 38% are trojans, 3.5% are bankers, and the remaining are divided among other malicious file types. The following sample rules were the most successful in detecting different types of malware:

- *bankers and bots*: IF (downloading process is "Acrobat Reader") $\rightarrow$ file is malicious.
- *droppers*: IF (file's signer is "Somoto ltd.") $\rightarrow$ file is malicious.
- *fakeavs*: IF (file is not signed) AND (Alexa rank of file's URL is above 100K) AND (downloading process is benign) AND (downloading process's signer is "Microsoft Windows") $\rightarrow$ file is malicious.

Expanding Available Ground Truth by Labeling Unknown Files. As mentioned earlier, the set of rules we learned were able to label 28.30% of all 1,436,829 previously unknown files from February to August, which represents a 233% increase in labeled files, compared to the available ground truth. These 28.30% of unknown files were downloaded by as many as 294,419 machines, or 31% of all machines, and have therefore a significant penetration across the machine population (notice that the overall number of machines that downloaded any of the 1,436,829 unknown files between February and August is 457,756).

These results indicate that our rule-based classification method would enable a significant expansion of the labeling of software files, compared to the ground truth available from multiple anti-virus sources. Ultimately, this would allow researchers to evaluate the accuracy of their malware detection systems over a much larger labeled dataset, including challenging cases of low-prevalence malicious files that in aggregate tend to impact a large population of machines.

Evading Detection. Evasion is certainly possible for most statistical detection models. Malware developers could change signer information by acquiring new signing certificates. However, valid certificates are not cheap. Therefore, it would be expensive to create polymorphic malware variants with always different signatures. Also, stealing a benign certificate is possible (though not easy); however, once this is detected by the true certificate owners, the certificate could be revoked. Using "benign" packers would make it easier to unpack and analyze the code. Therefore, malware often uses custom/hard-to-reverse packers. Thus, even though it is technically possible to evade our system, it would not be very practical in real-world.

VIII. Related Work

In this work, we focus on a specific class of software downloads that we believe been neglected in the past, namely *low-prevalence* downloads. With respect to previous work investigating malicious software downloads, we report the following. Rossow et al. [17] analyzed a limited number of about twenty dropper families for aspects such as their network infrastructure, infection, propagation and persistence on infected machines. More recently, Kwon et al. [10] extended this research by looking into the download chains that occur after infection. In comparison, we provided a comprehensive break-down of different types of malware, beside droppers, and analyzed their characteristics from various aspects, namely their signers, downloading URLs, transitions from one type to another, and etc. More importantly, [10] does not discuss the evaluation of their classifier on files for which no ground truth is available whatsoever although these files seem to comprise a significant portion (82%) of their dataset.

A second corpus of literature consists of papers focusing on potentially unwanted programs [8], [9]. Kotzias et al. [8], for example, looked into the who-installs-who relationships of PUPs and reported similar findings to ours with respect to PUPs delivering PUPs after first infection. Similarly, we identified this behavior on other types of malware, e.g. ransomware transitioning to ransomware in 80% of cases. Interestingly, our results also suggested that seemingly less harmful malware such as adware and PUP tend to leave machines vulnerable to other malware (Section V-B)

The same authors in [9] looked at PUPs from the perspective of code signing. Their analysis showed that most signed samples are PUPs and that other malware is not commonly signed. We also looked into this phenomenon, and our work tends to suggest that possibly-malicious software normally downloaded by browsers like droppers and PUPs tend to be correctly signed – probably as a need to send the code to execution on modern operating systems (Section IV-C). We leveraged this and other features identified in our measurement study to efficiently report *unknown* software downloads as malicious.

Kurt et al. [19] and Caballero et al. [1] explored the ecosystem of pay-per-install campaigns (PPI) and their role in the proliferation of PUPs by uncovering the operational organization and ecosystem of bundled software at back-end. In contrast, our evaluation is ran at front-end on a population of over a million end-point machines. We reported on the importance to consider low-prevalence software downloads as generating files with no ground truth for a total of 69% of the entire machine population.

In Section VI, we proposed a rule-based classifier that helped us reducing the large number (83%) of unknown that we observed in our population of software download. Only about 0.25% of the files that we observed during our measurement period had prevalence of more than 20. While similar classification systems have been proposed in the past (e.g., Polonium [2], Amico [20], CAMP [16], and Mastino [14]), they appear to be somewhat limited in scope when dealing with low-prevalence software files. Polonium [2], for example, reports 48% detection rate on files with prevalences of 2 and 3, and it does not work on files seen on single machines – overall accounting for 94% of the dataset in [2]. Other systems [14], [16], [20] could potentially mistake low-prevalence benign files as malware. Also, these systems rely on prevalence of the downloading URLs to provide classifications, which as explained in Section IV-B, could cause issues for them.

IX. Conclusions

We have presented a large-scale study of global trends in software download events, with an analysis of both benign and malicious downloads, and a categorization of events for which no ground truth is currently available. Our measurement study, which is based on a real-world dataset containing more than 3 million *in-the-wild* web-based software download events involving hundreds of thousands of Internet machines, shows that more than 83% of all downloaded software files remain *unknown* to the anti-malware community even two years after they were first observed. To better understand what these *unknown* software files may be, and their potential impact on real-world Internet machines, we have performed a detailed analysis of their properties. We then built a rule-based classifier to extend the labeling of software downloads. This system can be used to identify many more benign and malicious files with very high confidence, allowing us to greatly expand the number of software files that can be used to evaluate the accuracy of malware detection systems.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation (NSF) under grant No. CNS-1149051. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. This work is also partially supported by a grant from the Auburn University at Montgomery Research Grant-in Aid Program. Additional acknowledgments go to Trend Micro's Forward-Looking Threat Research (FTR), SPN and Machine Learning teams who supported the research in different forms.

References

[1] J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring pay-per-install: The commoditization of malware distribution. In *Usenix security symposium*, 2011.

[2] D. H. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos. Polonium: Tera-scale graph mining for malware detection. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2010.

[3] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. arXiv:1702.08608v2[stat.ML].

[4] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In J. Shavlik, editor, *Fifteenth International Conference on Machine Learning*, pages 144–151. Morgan Kaufmann, 1998.

[5] Google. Google Safe Browsing. https://www.google.com/transparencyreport/safebrowsing/.

[6] C. Grier, L. Ballard, J. Caballero, N. Chachra, C. J. Dietrich, K. Levchenko, P. Mavrommatis, D. McCoy, A. Nappa, A. Pitsillidis, N. Provos, M. Z. Rafique, M. A. Rajab, C. Rossow, K. Thomas, V. Paxson, S. Savage, and G. M. Voelker. Manufacturing compromise: The emergence of exploit-as-a-service. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 821–832, New York, NY, USA, 2012. ACM.

[7] A. Kapravelos, Y. Shoshitaishvili, M. Cova, C. Kruegel, and G. Vigna. Revolver: An automated approach to the detection of evasive web-based malware. In *USENIX Security*, pages 637–652. Citeseer, 2013.

[8] P. Kotzias, L. Bilge, and J. Caballero. Measuring pup prevalence and pup distribution through pay-per-install services. In *Proceedings of the USENIX Security Symposium*, 2016.

[9] P. Kotzias, S. Matic, R. Rivera, and J. Caballero. Certified PUP: Abuse in Authenticode Code Signing. In *ACM Conference on Computer and Communication Security*, 2015.

[10] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitras. The dropper effect: Insights into malware distribution with downloader graph analytics. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1118–1129. ACM, 2015.

[11] T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad. Towards measuring and mitigating social engineering software download attacks. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, 2016.

[12] R. Perdisci et al. Vamo: towards a fully automated malware clustering validity analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 329–338. ACM, 2012.

[13] M. Z. Rafique, T. Van Goethem, W. Joosen, C. Huygens, and N. Nikiforakis. Its free for a reason: Exploring the ecosystem of free live streaming services. 2016.

[14] B. Rahbarinia, M. Balduzzi, and R. Perdisci. Real-time detection of malware downloads via large-scale url$\rightarrow$file$\rightarrow$ machine graph mining. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '16, pages 783–794, New York, NY, USA, 2016. ACM.

[15] B. Rahbarinia, R. Perdisci, and M. Antonakakis. Segugio: Efficient behavior-based tracking of malware-control domains in large isp networks. In *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*, pages 403–414. IEEE, 2015.

[16] M. A. Rajab, L. Ballard, N. Lutz, P. Mavrommatis, and N. Provos. Camp: Content-agnostic malware protection. In *NDSS*, 2013.

[17] C. Rossow, C. Dietrich, and H. Bos. Large-scale analysis of malware downloaders. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 42–61. Springer, 2012.

[18] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero. Avclass: A tool for massive malware labeling. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 230–253. Springer, 2016.

[19] K. Thomas, J. A. E. Crespo, R. Rasti, J.-M. Picod, C. Phillips, C. Sharp, F. Tirelo, A. Tofigh, M.-A. Courteau, L. Ballard, et al. Investigating commercial pay-per-install and the distribution of unwanted software. In *USENIX Security Symposium*, 2016.

[20] P. Vadrevu, B. Rahbarinia, R. Perdisci, K. Li, and M. Antonakakis. Measuring and detecting malware downloads in live network traffic. In *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, pages 556–573, 2013.

[21] X. Xing, W. Meng, U. Weinsberg, A. Sheth, B. Lee, R. Perdisci, and W. Lee. Unraveling the relationship between ad-injecting browser extensions and malvertising. In *Proceedings of the International Conference on the World Wide Web*, 2015.