IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

Costly freeware: a systematic analysis of abuse in download portals

ISSN 1751-8709
Received on 24th November 2017
Revised 3rd May 2018
Accepted on 16th May 2018
E-First on 12th July 2018
doi: 10.1049/iet-ifs.2017.0585
www.ietdl.org

Richard Rivera[1,2] ✉, *Platon Kotzias*[1,2], *Avinash Sudhodanan*[1], *Juan Caballero*[1]

[1]*IMDEA Software Institute, Madrid, Spain*
[2]*Universidad Politécnica de Madrid, Madrid, Spain*
✉ *E-mail: richard.rivera@imdea.org*

Abstract: Freeware is proprietary software that can be used free of charge. A popular vector for distributing freeware is *download portals*, i.e. websites that index, categorise, and host programs. Download portals can be abused to distribute potentially unwanted programs (PUP) and malware. The abuse can be due to PUP and malware authors uploading their ware, by benign freeware authors joining as affiliate publishers of pay-per-install (PPI) services and other affiliate programs, or by malicious download portal owners. The authors perform a systematic study of abuse in download portals. They build a platform to crawl download portals and apply it to download 191 K Windows freeware installers from 20 download portals. They analyse the collected installers and execute them in a sandbox to monitor their installation. They measure an overall ratio of PUP and malware between 8% (conservative estimate) and 26% (lax estimate). In 18 of the 20 download portals examined the amount of PUP and malware is below 9%. However, they also find two download portals exclusively used to distribute PPI downloaders. Finally, they detail different abusive behaviours that authors of undesirable programs use to distribute their programs through download portals.

1 Introduction

Freeware is proprietary software that can be used without monetary cost. Freeware is distributed in binary form and should not be confused with open-source software that is also free but provides access to its source code. A related model is shareware where the software is initially free to use, but users are expected to pay to continue using it. In contrast, freeware is free to use for unlimited time. While freeware can be used free of charge, authors may want to cover their development costs and benefit from their freeware. This can be achieved through freemium models where the user pays for advanced functionality, voluntary user donations, advertisements, and by offering third-party software through commercial pay-per-install (PPI) services [1, 2]. For example, Skype uses a freemium model where users pay for calling phone numbers and Sun's Java offers users to also install the Yahoo toolbar.

A popular vector for distributing freeware is *download portals* which are websites that index, categorise, and host programs. Download portals such as *cnet* [3], *softonic* [4], or *tucows* [5] are a meeting point for freeware authors that want to advertise their programs and for users looking for a specific program or functionality. Users can leverage download portals for searching for popular freeware in a category (e.g. video software, security tools, and Windows themes), browsing through the program metadata (e.g. version, author, platform), reading program reviews, and eventually downloading the chosen programs. Download portals enable freeware authors to distribute their programs, increasing their user base. Using download portals, the freeware author can save on advertisements costs required to let users know about the freeware's existence. Authors on a low budget can also avoid setting up a webpage for the freeware. The download portals invest in advertising and have a motivation to rank highly on search engine results to attract users that can be monetised through advertisements and PPI schemes.

Download portals can be abused as a distribution vector for potentially unwanted programs (PUP) and malware. PUP are a category of undesirable software, that while not outright malicious like malware, contain behaviours considered harmful by many users. While the boundary between PUP and malware is sometimes blurry, prior work has tried to delineate what constitutes PUP [6–8] and companies such as Google [9], Microsoft [10], and MalwareBytes [11] have policies for defining what behaviours make a program PUP. Two types of programs that are widely considered PUP are *adware* that aggressively pushes advertisements and *rogueware* that scares users into buying a software license, despite its limited functionality.

Download portals may be used to distribute PUP and malware in three ways. First, download portals can be abused by PUP and malware authors to distribute their programs, by uploading their undesirable software and disguising it as potentially useful freeware. Second, authors of benign freeware may become affiliate publishers of commercial PPI services, bundling their freeware with a PPI downloader and uploading the bundle to a download portal. When installing the bundle, users will be offered third-party advertiser programs, which may be PUP. In fact, prior work has measured that at least 25% of PUP is distributed through 24 commercial PPI services [1]. Third, download portal owners may be untrustworthy and use their download portals to purposefully distribute undesirable software to visitors.

While several blog posts point to download portals being bloated with PUP [12–14], their conclusions are based on *ad hoc* measurements performed on the top downloaded programs of a few download portals. In this work, we perform a systematic study of abuse in download portals. We build a platform to crawl download portals. We use our platform to download all Windows freeware offered by 20 download portals. This enables examining PUP and malware prevalence beyond that of the most popular downloads. Our crawling downloads 191 K programs from the 20 download portals, which correspond to 157 K unique files with a cumulative size of 2.5 TB. We analyse the collected freeware to identify PUP and malware and execute the programs in a sandbox to analyse what modifications they perform to the system, e.g. changing a browser's homepage and installing browser modifications. Our analysis addresses the following three main questions:

- *What percentage of programs in download portals are PUP and malware?* We use two policies to quantify undesirable (i.e. PUP or malware) programs in download portals. Our conservative policy identifies as undesirable any program flagged by more

than three AV engines, while our lax policy identifies as undesirable any program flagged by at least one AV engine. We measure an overall ratio of undesirable programs across all download portals analysed ranging between 8% (conservative) and 26% (lax). Among the undesirable programs PUP (76%) dominates malware (24%). For 18 of the 20 download portals examined, the amount of PUP and malware is moderate ranging between 8.5% and a low as 0.2%. These ratios are significantly lower than those reported in prior works that only examine the top downloads [12–14]. We believe our measurements are more accurate since we examine all programs indexed by a download portal.

- *Are there download portals that are clearly abusive?* We identify two download portals, run by the same company, which serve 100% PUP. Those download portals are exclusively used to distribute a PPI downloader. Regardless what program the user chooses to download, he is always provided with a customised PPI downloader. We provide a detailed analysis of the operation leveraging those two download portals and identify another 12 similar download portals from the same owners.
- *How are download portals abused?* Our analysis uncovers different abusive behaviours that authors of undesirable programs employ to distribute their programs through download portals. We observe authors uploading the same file as different programs in the same download portal, as well as across multiple download portals. We identify some authors using external links to bypass security checks by download portals. We show that the failure to identify repetitive abusers is widespread across download portals, rather than limited to a few careless download portals. Finally, we observe impersonation of benign popular authors by other authors that want to leverage their reputation, e.g. to disguise their undesirable programs as innocuous.

2 Download portals

Download portals index large amounts of programs from different authors. To enable users finding the program they are interested in, or a program that matches a specific functionality, programs are typically grouped into categories and indexed using keywords. Download portals have existed for at least 20 years with popular download portals such as *cnet* and *softonic* being launched in 1996 and 1997, respectively. The download portals may host the programs themselves, i.e. the file is downloaded from domains owned by the download portal, may link to the author's webpage where the program is hosted; or may provide both types of downloads.

Most download portals accept submissions from software developers through forms where a developer specifies information about its software such as program name, version, description, and program's URL. Each download portal requires different information from the developers. Some download portals also support submissions through the portable application description (PAD) standard, an XML schema introduced in 1998 by the Association of Software Publishers to standardise software metadata [15].

Download portals face challenges in determining that a submitted program matches its description and that it is uploaded by its real author. Some download portals may take steps towards reducing abuse, e.g. analysing submitted files using online services such as VirusTotal (VT) [16]. Recently, some download portals like *filehippo* and *softonic* have stopped accepting submissions by developers. These download portals analyse the freeware ecosystem themselves to select new programs to add.

PPI: A popular software monetisation mechanism are PPI agreements where an advertiser, i.e. a software publisher interested in distributing its program to users, pays a third-party to help with the distribution. PPI agreements can be bilateral between two software publishers, e.g. Oracle distributing the Ask toolbar with its Java platform [17]. They can also take the form of commercial PPI services that connect multiple advertisers interested in distributing their programs with multiple affiliate publishers willing to offer those advertised programs to users that install the affiliate's program [1, 2]. Affiliate publishers are often freeware authors that own programs that users want to install. They bundle their freeware with a PPI downloader and distribute the bundle, e.g. by uploading the bundle to download portals, in exchange for a commission paid for each installation. When a user installs the bundled freeware, the PPI downloader offers to the user the advertised programs. If the user installs an advertised program, the advertiser pays the PPI service for the installation and the affiliate publisher receives a commission. Some download portals such as *cnet* run their own commercial PPI service to supplement their advertisement income. Freeware authors uploading their programs to the download portal are invited to join as publishers of the download portal's PPI service to monetise their programs.

An alternative PPI model is for software publishers to directly recruit affiliates to distribute their software without using a PPI service. Some large PUP publishers have affiliate programs such as Mindspark, Babylon, Systweak, and Spigot [1]. Freeware authors can sign up as affiliates of such programs, obtain a bundle and distribute it, e.g. by uploading to download portals.

Installers: Most programs need to be installed before they can be executed. The installation process may, among others, check if system requirements and dependencies are met, setup program files into a specific folder structure, configure services that run automatically, download resources from the Internet, and make the program easy to launch (e.g. adding shortcuts and entries to the start menu). To ease installation, programs are often distributed as *installers*, i.e. auxiliary programs (e.g. setup.exe) responsible for installing a target program. Most files downloaded from download portals correspond to installers.

Analysed download portals: We analyse 20 download portals that offer Windows programs. The selected portals may offer programs for other platforms as well, but we crawl only the Windows programs they offer. We have selected download portals that target different geographic locations and of different popularity (according to their Alexa ranking [18]) because there may be differences in behaviours between those types, e.g. how they vet publishers leading to different amounts of abuse. Table 1 shows the list of 20 download portals analysed in this work. For each download portal, it shows the abbreviated name we use to refer to the download portal, its URL, the Alexa ranking, whether the download portal accepts software submissions using forms and PAD files, and the platforms for which it indexes programs. Download portals that do not accept submissions through forms nor PAD may accept them through email or not accept submissions at all.

3 Approach

Our approach processes one download portal at a time and comprises four steps: preparation, crawling, file processing, and execution. During preparation, an analyst manually generates one portal metadata file for the download portal, which contains all the needed information to crawl the download portal. The crawling, file processing, and execution steps are automated and illustrated in Fig. 1. The crawling takes as input a portal metadata file, and automatically navigates a selenium-based crawler [36] to download all the programs the download portal offers. It outputs the downloaded files and saves all information about the crawling into a central database. The file processing extracts information statically from the downloaded files (e.g. filename, filetype), collects the file report from VT [16], extracts executable files from archives, and checks the authenticode digital signature (if signed). The execution takes as input the programs, runs them in a sandbox, generates an execution report, and saves the report data in the central database. Each of these four steps is detailed next in its own subsection.

3.1 Preparation

Download portals share the goal of enabling users to find the software that they are interested in. This creates similarities in their structure and design. However, each download portal is different. For example, download portals differ in their layout and the

Table 1 Download portals analysed in this work, their Alexa ranking (from 10 October 2016), whether they accept software submissions through forms or PAD files, and the platforms covered (Windows, Android, MacOS, Linux). GP means it redirects to Google Play

Portal name		Submission		Platforms			
	Alexa	Form	PAD	Win.	And.	Mac	Lin.
uptodown [19]	155	✓	✗	✓	✓	✓	✓
cnet [3]	164	✓	✓	✓	✓	✓	✗
softonic [4]	200	✗	✗	✓	GP	✓	✗
filehippo [20]	615	✗	✗	✓	✗	✗	✗
softpedia [21]	1589	✓	✓	✓	✓	✗	✗
soft112 [22]	4672	✗	✓	✓	GP	✓	✓
majorgeeks [23]	6206	✗	✗	✓	GP	✗	✗
soft32 [24]	6640	✓	✓	✓	✓	✓	✗
eazel [25]	8760	✗	✗	✓	✓	✓	✗
fileforum [26]	9449	✓	✗	✓	✗	✓	✓
filehorse [27]	9980	✗	✗	✓	✗	✓	✗
portalprogramas [28]	12,171	✓	✗	✓	✓	✓	✗
freewarefiles [29]	13,556	✓	✗	✓	✗	✗	✗
tucows [5]	25,084	✓	✓	✓	✗	✓	✓
snapfiles [30]	33,545	✓	✗	✓	✗	✗	✗
filecluster [31]	56,379	✓	✓	✓	✗	✗	✗
descargarmp3 [32]	104,352	✗	✗	✓	GP	✗	✗
download3000 [33]	230,115	✓	✗	✓	✗	✗	✗
fileguru [34]	308,929	✓	✓	✓	✗	✗	✗
geardownload [35]	466,545	✓	✓	✓	✗	✗	✗

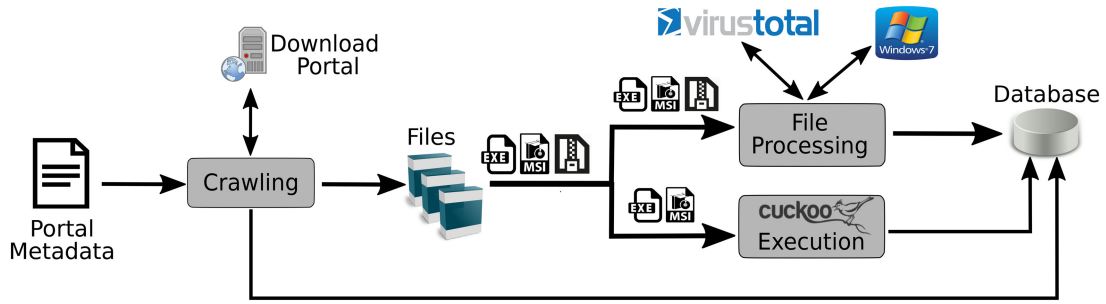

Fig. 1 *Summary of our approach for one download portal, excluding the preparation step*

information they collect about the programs. This makes it challenging to crawl a large number of download portals to analyse the software they index. To address this issue, at the beginning of this project we studied multiple download portals to identify a common abstraction that worked for all of them and enables adding a new download portal to be crawled with very limited effort.

All the download portals we have examined share a basic structure in which every program in the download portal has its own *program page*, which presents the data that the download portal has collected on the program. The program page may contain, among others, the program name, version, description, license, platform, language, size, author, number of downloads, reviews, date of publication, date of last update, screenshots, previous versions, and download links. The specific program attributes in the program page vary across download portals. Different programs within the same download portal may have different program attributes, e.g. if an upload form has optional fields that some authors fill and others do not. However, we have been able to identify a subset of six program attributes that are available for all programs in all download portals analysed: name, version, platform, size, author, and download link.

The output of the preparation step is a portal metadata file that has all the information needed for the crawling. Specifically, the portal metadata file contains three parts: how to list all the program pages, how to identify the six program attributes from the program page, and how to download the program file from the download link or button in the program page. We describe them next.

Listing the program pages: We can classify download portals into two classes regarding how to list all program pages in the download portal. Class 1 download portals allow us to list all programs directly and Class 2 download portals allow us to list all programs in each category. Class 1 download portals may offer a direct link to list all programs or may enable searching with wildcard identifiers through which we can list all programs. For Class 1 download portals, we can extract a URL template such as http://portal.com/software/?page=X, where *X* is a positive integer that can be monotonically increased to iterate over the search results. Every page in this iteration contains, among other content, a list of program entries, each with a *program URL* that points to the program page. To identify the program URLs inside the search results pages, we use the path in the page's DOM. For Class 2 download portals, we need to first obtain the list of all program categories. Then, we can define a URL template such as http://portal.com/<CATEGORY>/?page=X and we can iterate over the search results of each category by monotonically increasing *X* and extracting the program URLs similar to Category 1 download portals. The analyst provides in the portal metadata file: an URL template for Class 1 and Class 2 download portals, and a list of all software categories for Class 2 download portals.

Identifying the program attributes: To identify the program attributes in the program page (name, version, platform, size, author, and download link), the analyst provides in the portal metadata file a DOM path for each program attribute, which uniquely identifies the position of a page element that contains the attribute.

Downloading the program files: The program page always contains an element to download the program, e.g. a download link or a download button. However, that element may not link directly

to the file to download (e.g. executable or archive). Instead, it may open a download page with some advertisements and one or more download links, e.g. from different mirrors. We call *download URL* to the URL from where our crawler downloads the program file, i.e. the URL that points to the program file. The analyst provides in the portal metadata file the sequence of clicks, i.e. DOM paths to buttons to be clicked, starting with the click on the download element in the program page that the crawler needs to perform to arrive and click on the download URL of the latest program version.

We developed the process above iteratively. Once we converged on this processing, generating a portal metadata file for a new download portal took us 2–3 h.

3.2 Crawling

The crawling takes as input the portal metadata file. It outputs the program files downloaded from the portal and saves into a central database the crawling information such as timestamps, URLs visited, and program attributes collected from the download portal. We use a crawler based on selenium WebDriver [36] with Mozilla Firefox. The crawler follows the instructions in the portal metadata file to list all program pages. For each program page, it identifies the program attributes using the DOM paths in the portal metadata file and stores the attributes. If the program is a Windows freeware, it clicks on the download link, and follows the instructions in the portal metadata file to locate the download URL.

Once the download URL is clicked, an attempt to download the program file is performed. If the download has not started after 30 s, the crawler tries again. A file download may fail for the following five reasons: (i) the download link is broken; (ii) the download has not completed in 5 min, a timeout we chose to limit the maximum size of a downloaded file. This timeout provides significant storage savings and only affects files 200 MB–2 GB depending on the bandwidth of the download portal (even with this limit the downloaded programs use 2.5 TB disk storage); (iii) any web page has not finished loading within 30 s; (iv) the download link redirects our crawler to an external web page and does not point directly to a program file, i.e. the user is expected to locate the right download link in the publisher's webpage; (v) the download portal refreshes the webpage with which our crawler is interacting, e.g. to change an advertisement.

After each download attempt, whether successful or not, the crawler outputs a tuple with: timestamp, download portal identifier, error code, program URL, download URL, four program attributes from the program page (program name, size, version, and author), and a file identifier if the download was successful.

We are interested in whether the download portals host the downloaded programs onsite or simply redirect users to the publisher's webpage. For this, we manually build a mapping, using Whois and DNS information, of which domains in the download URLs belong to each download portal. For example, cnet uses two effective second-level domains for hosting programs: *cnet.com* and *downloadnow.com*.

3.3 File processing

The file processing step statically analyses the files and saves all information in the central database. It first processes each downloaded file to obtain: MD5, SHA1, SHA256, size on disk, filename, and filetype. Then, it attempts to decompress archives to extract any executables inside. Note that a downloaded archive may contain other archives, so this is a recursive process, which we detail in Section 4.1. Next, the file hash of each executable, directly downloaded or extracted from an archive, is used to query VT [16], an online service that examines user-submitted files with a large number of security tools. At the time of writing this paper, VT analyses submitted files using 70 AV engines including all major AV vendors (https://www.virustotal.com/en/about/credits/). Engines are frequently updated and the list of engines evolves over time. If the file is known to VT, a report is downloaded that contains, among others, the number of AV engines that detect the file, the timestamp when the file was first submitted, and file metadata. We submit to VT all files downloaded from the download portals that are smaller than 30 MB. This threshold is due to the VT API, which has a limit of 32 MB. We reduced the limit to 30 MB because we observed errors with files near the limit.

We use two maliciousness policies: conservative and lax. The conservative policy is to consider a program undesirable (i.e. PUP or malware) if it detected by more than three AV engines in the VT report. This policy is designed to minimise false positives due to a few AVs committing an error in the detection. The lax policy considers undesirable any program detected by at least one AV engine. We use the lax policy as an upper bound. For programs distributed as archives, we consider them undesirable if the archive itself, or any file inside the archive, satisfies the policy.

To classify an undesirable executable as either malware or PUP, and to determine its family, we use AVClass [37], a malware labelling tool. AVClass takes as input the AV labels in a VT report; removes noise from AV labels by addressing label normalisation, generic token detection, and alias detection; and outputs for each sample whether it is PUP or malware, its most likely family name, and a confidence factor based on the agreement across AV engines.

The final file processing step is to analyse the signed executables. Code signing is a technique that embeds a digital signature in an executable, which enables verifying the program's integrity and authenticating its publisher. Prior work has shown that properly signed executables detected by AVs are predominantly PUP, since it is challenging for malware to obtain a valid code signing certificate from a Certification Authority (CA) [38]. The file processing component validates the authenticode signature in executable files. For this, it uploads all executables to a Windows VM and uses the Microsoft-provided validation tool to check if the executable is signed and whether the signature validates using different policies (e.g. default and kernel driver). Signed executables are further processed by our own code to retrieve the X.509 leaf certificate and extract, among others: Subject CN, Issuer CN, PEM and DER hashes, validity period, signing hash, digital signature algorithm, signed file hash (called Authentihash), and public key. For executables that are signed and whose signature validates, we can confidently identify the publisher's identity.

3.4 Execution

We run downloaded executables in the Cuckoo Sandbox [39]. Cuckoo receives an executable, assigns it to a VM for execution, and generates a behavioural report for the execution. We have implemented a few extensions to Cuckoo to better fit our needs. These include adding some anti-anti-evasion techniques to harden the sandbox [40, 41], extending the Graphical User Interface (GUI) exploration, and adding signatures for specific events that we want to be notified about. These modifications are detailed below.

The vast majority of executables downloaded from the download portals correspond to installers (e.g. firefox_setup.exe) that will install the real program binaries on the end host (e.g. firefox.exe) upon execution. Such installers are typically GUI-based and require user interaction to complete the installation. Cuckoo provides functionality to identify buttons in windows launched during the execution, and to automatically click buttons labelled with keywords such as ‘Next’ or ‘Confirm’ simulating the default user behaviour of accepting all windows to quickly install the program. However, the list of keywords used to identify those buttons is pretty small and limited to English. Thus, we extended the keyword list and translated the keywords into popular languages such as German and Spanish.

We also extended the signatures module of Cuckoo, which enables defining signatures for events of interest that Cuckoo can directly report. This module provides performance improvements to identify those events. For example, we could scan the list of registry modifications provided by Cuckoo to see if a specific key that stores Internet Explorer's homepage has been modified. However, it is significantly more efficient to build a signature for that registry key and let Cuckoo automatically report its modification. Our signatures include events such as whether browser extensions have been installed, and whether some browser settings have been altered.

We configure Cuckoo to use 30 VirtualBox VMs on a single host running Ubuntu 16.04 LTS. Each VM is configured with 1 GB of RAM and 20 GB hard disk. The VMs run Windows 7, which is still the most popular OS [42]. Our VM image has a large number of popular programs installed such as Internet Explorer, Chrome, Firefox, Opera, Java, Adobe Reader, Adobe Flash player, and the.NET framework. This is done so that we can observe modifications that the executed programs may perform on those programs.

4 Evaluation

This section evaluates our approach. Section 4.1 presents the results of crawling the download portals, Section 4.2 examines the prevalence of undesirable programs in download portals, Section 4.3 compares the PUP, malware, and benign programs behaviours, Section 4.4 summarises the execution results, and Section 4.5 provides a case study on download portals associated with PPI services.

4.1 Download portals crawling statistics

In this section, we present some general statistics on the crawling. We analyse the security aspects such as prevalence of undesirable programs and abusive behaviours in the next sections.

Table 2 summarises the crawling results. For each download portal it presents: the date when it was crawled (all dates from 2016), the number of Windows programs offered, the number of successfully downloaded programs, the number of unique downloaded files by hash, the size of the downloaded files (in GB), the split of the unique files by type (EXE, ZIP, RAR, MSI, and other), and the percentage of unique files hosted onsite (i.e. on domains that belong to the download portal).

Overall, we downloaded 191 K programs from the 20 download portals, corresponding to 157 K unique files with a cumulative size on disk of 2.5 TB. The downloaded files correspond to 65% of the 325 K offered programs. Section 3.2 details the reasons why a download can fail. The largest download portals are *soft112* and *softpedia* with 107 and 69 K offered programs, respectively. We downloaded the most from *softpedia* with 48 K unique files, followed by *soft112* with 43 K. The smallest download portals were *download3000* and *filehorse* with less than a thousand programs offered each, and 275–350 unique files downloaded. Note that the download portals are sorted by Alexa ranking (same order as Table 2), showing that popularity does not have a direct correspondence with size. For example, *uptodown*, *cnet*, *softonic*, and *filehippo* all have higher Alexa ranking than the two largest download portals.

File types: Most downloaded files are executables (48%) and archives (46%). More specifically, of the unique files downloaded 48% are EXEs, 40% ZIP archives, 3% MSI installers, 2.7% RAR archives, 1.6% JAR archives, another 2% other types of archives (.gzip, .bz2, .7z, and .cab) and the remaining are comprised of a long tail of over 70 filetypes including JPEG, text files, ISO images, Office files, PDFs, and source files (e.g. PHP, Python, C). We automatically decompress archives, finding an additional 10 M files (170 K executables) inside.

Signed executables: Of the 75,615 downloaded executables, 39% (29,228) are signed. Of those signed executables, 76% validate correctly on Windows, 20% have expired certificates, 1% have revoked certificates, and the remaining 3% generate various validation errors. There are two download portals (*descargarmp3* and *eazel*) that sign all their executables, and each of those two download portals uses a single code signing certificate for signing the executables. We perform an in-depth analysis of these two download portals in Section 4.5.

4.2 Undesirable programs in download portals

In this section, we examine the prevalence of undesirable programs in download portals. As explained in Section 3, we submit to VT all downloaded files larger than 30 MB (89% of all downloaded files). According to the lax policy (i.e. a file is undesirable if at least one AV engine flags it), 41,664 of the files are undesirable. According to the conservative policy (i.e. undesirable if flagged by more than three AV engines), 12,340 files are undesirable. Thus, the overall ratio of undesirable programs across all download portals ranges between 8% (conservative) and 26% (lax).

We apply AVClass on the 12,340 files flagged as undesirable by the conservative policy in order to classify them as PUP/malware and to label them with a family. Of those, 9376 (76%) are PUP and 2955 (24%) are malware. These numbers show that PUP is more than three times more common than malware in download portals.

Table 3 ranks the download portals by percentage of undesirable programs. For each download portal, it first shows the ratio for all undesirable programs and is split into PUP and malware using the conservative policy. Then, it shows the overall ratio using the lax policy. Two download portals (*eazel* and

Table 2 Download portals crawling results

Portal	Programs					File type					Hosting
	Date	Offered	Downl.	Unique	Size, GB	EXE	ZIP	RAR	MSI	Other	Onsite
uptodown	06/14	8115	7071	7066	227.8	4882	1747	166	161	110	99.8%
cnet	06/26	6814	5220	5161	67.7	3432	1350	0	0	379	100.0%
softonic	11/05	23,737	14,575	14,487	225.3	8139	5075	292	342	639	98.3%
filehippo	06/15	1274	1167	1163	38.8	973	125	2	0	63	100.0%
softpedia	09/09	69,738	48,747	48,247	386.0	20,438	21,855	1226	170	4558	39.3%
soft112	10/11	107,642	44,110	43,078	287.7	8908	24,958	2457	838	5917	0.0%
majorgeeks	06/25	4712	4227	4223	63.9	2498	1574	0	19	132	14.6%
soft32	09/02	8563	698	671	14.5	345	287	3	3	33	99.7%
eazel	07/29	2444	2397	2397	2.1	2397	0	0	0	0	0.0%
fileforum	08/31	6141	1917	1902	13.4	1236	525	5	4	132	0.0%
filehorse	08/04	435	351	350	15.1	315	17	0	11	7	99.1%
portalprogramas	10/04	9223	7140	7102	187.2	3720	2514	221	252	395	99.9%
freewarefiles	09/03	17,083	7162	7108	94.2	3824	2858	59	140	227	0.0%
tucows	09/03	22,695	22,187	22,153	206.1	17,835	3869	0	92	357	99.3%
snapfiles	08/29	3998	3651	3648	42.5	2387	1118	0	110	33	18.6%
filecluster	09/03	11,782	7421	7300	172.4	4894	1923	17	310	156	100.0%
descargarmp3	09/03	3551	3530	3530	3.2	3530	0	0	0	0	0.0%
fileguru	08/31	5552	1653	1632	13.5	532	814	53	18	215	100.0%
download3000	09/03	967	281	275	2.1	207	55	0	4	9	0.0%
geardownloads	09/06	11,194	7364	7197	58.5	4913	1979	37	23	245	0.0%
	total	325,660	190,869	156,954	2585.5	75,615	62,487	4298	4727	9827	

descargarmp3) have a ratio of 100% undesirable programs. We examine these two portals in more detail in Section 4.5. The other 18 portals have at most 8.5% undesirable programs.

Table 4 shows the top ten PUP and malware families output by AVClass. For each family it shows its rank, family name, and the number of files in the family. For PUP families, we also provide a type. Of the top ten PUP families, five are PPI downloaders, two are generic PUP classes such as password tools (pswtool) and software that changes browsing preferences (prefchanger), one is a label for software downloaded from a specialised download portal (securityxploded), another is a PUP running an affiliate program (spigot), and the last is a marketing tool used to monitor users' Internet habits (relevantknowledge). The lower ranks of malware families, as well as the presence of multiple not-so-popular malware families, also point to PUP abusing download portals for distribution much more often than malware.

Overall, our results identify two portals that are clearly malicious with all programs being considered PUP, and that the amount of undesirable programs in the rest is moderate ranging between 8.5% and as low as 0.2%. Among the undesirable programs PUP (76%) dominates malware (24%). These prevalence numbers are in contrast with prior reports that measure much higher rates of undesirable programs among the top downloads and promoted programs [12–14, 43]. We believe that our analysis, which takes into account the complete list of programs in a download portal, rather than only the top downloads, provides a more accurate estimate of the prevalence of undesirable programs in download portals.

4.3 Abusive behaviours

In this section, we describe several abusive behaviours that we have observed in our analysis.

Same file as different programs: One behaviour that we observe is the same file (i.e. same hash) appearing as different programs in different download portals, and even within the same download portal. In some cases, these are benign programs such as the same Realtek audio driver being offered for different HP notebook models. However, oftentimes these are undesirable programs that fraudulently advertise themselves as different tools. For example, in *soft112* one file is offered as 47 different programs including 'start up business advisor', 'consumer credit authorisation', 'debt collection service london', 'outsourcing service wimbledon', and 'fsa compliance'. Similarly, in *uptodown* there is a file offered under six different author and program pairs such as 'bittorrent sync by Bittorrent Inc', 'mobaxterm by Mobatek', and 'photofilm by KC Software'. We argue that these cases where the same file is registered as different programs within the same download portal are easy to identify by the download portal and such registrations should be blocked as they are misleading to users.

The case where the same file is advertised as different programs in different download portals is harder to protect against unless download portals share program information or monitor each other. An example of this case is a file advertised as ten different programs in five download portals including 'scftp' in *softpedia*, 'ftpright' in *freewarefiles*, 'esftp' in *geardownload*, 'robust ftp & download manager' in *fileforum*, and 'free ftp and download manager' in *download3000*.

External program hosting: A surprising observation is that half of the download portals host less than half of the programs they index, and 35% of download portals do not host any files. This is shown in the rightmost column in Table 2, which captures for each download portal, the percentage of onsite hosted programs. Of the 20 download portals, 10 host nearly all programs (over 98%) onsite, 7 host no programs onsite, and the other 3 host 14–40% programs onsite. External hosting of programs allows a malicious publisher to abuse time-of-check to time-of-use (TOCTOU) conditions by submitting a program URL that initially points to a benign file, but later points to an undesirable file. Overall, of the 12,340 undesirable files, 33% are hosted onsite and 67% offsite. However, if we exclude the two download portals that exclusively serve PUP from an external site, the remaining undesirable files are hosted 63% onsite and 37% offsite. Thus, while we do observe instances of this behaviour, it does not seem that attackers currently are widely abusing such TOCTOU conditions. Still, we advise download portals to periodically scan the externally hosted files.

Impersonating benign authors: The program author advertised in a download portal may not necessarily be the true author since it is hard for download portals to verify authorship in the general case. Thus, some authors may be tempted to impersonate popular benign authors to make their programs more attractive, including malicious authors that want to make their undesirable programs look innocuous. On the other hand, impersonating an author with low reputation does not provide a benefit to the real author.

Table 5 shows the top ten undesirable authors with more than 50 files across the 20 download portals. For each author, it shows the number of download portals where it appears, the number of files and signed executables, the number of publishers signing the executables, the total percentage of undesirable downloaded files and classified as PUP or malware.

The table includes two benign software publishers: Microsoft and Adobe. A closer look shows that a significant fraction of the files that claim authorship from Adobe and Microsoft are not signed, which is rare for those publishers. This likely indicates other authors trying to impersonate Microsoft and Adobe. Of the programs that claim the author is Adobe or Microsoft 17 and 4% are undesirable, respectively. The majority of those are signed by *Delivery Agile*, a PUP company analysed in Section 4.5.

Repetitive abusers: Another observation from Table 5 is that there exist authors that repeatedly abuse download portals (as mentioned earlier, it does not seem likely that other authors are impersonating authors with low reputation). It should be easy for download portals to identify such repetitive abusers, e.g. through the use of blacklists. We observe that the failure to identify them is widespread, rather than being specific to a few careless download portals. The data shows that the majority of those authors have programs in multiple download portals, and only *freeridegames* and *siteken* abuse a single download portal (namely *tucows*).

4.4 Program execution

We configured Cuckoo to run programs for 180 s. Determining if an installation has successfully completed is challenging as we do not even know how many programs (or which programs) will be installed. We consider an installation successful if the original executable run by Cuckoo (i.e. the installer) writes to disk at least another executable with a different hash, i.e. installs at least one program. Overall, 68% of the executions install at least one program.

We observe a significant number of modifications to installed browsers. Our VM has both Internet Explorer (IE) and Firefox installed with IE set as the default browser. We observe 1399 installers that modify the start page of IE. Of those, 77% set the homepage to http://www.ihotsee.com/ and 4% to http://search.conduit.com/. We also observe nine installers that change the default browser, eight of them to the Avant Browser [44] and one to the now defunct Sundial browser.

We also observe 551 installers that install additional browser functionality. More specifically, 445 installers add an IE toolbar, 20 add a Firefox extension, 5 add an IE bar, 178 add a default URLSearchHook for IE (which hooks any URL typed by the user without a protocol, e.g. to redirect to a hijacker's webpage), and 21 install an IE Browser Emulation. The large difference between IE and Firefox modifications is likely due to IE being the default browser.

4.5 Case study: Vittalia download portals

The download portals *eazel* and *descargarmp3* have noticeable similarities among themselves that distinguish them from the other download portals. Specifically, both download portals offer only executables (no archives or other filetypes), each downloaded file is unique (no file is downloaded twice in those portals), each file downloaded from the same download portal has the same size (the size differs among the two download portals), all executables are signed, and the same code signing certificate is used to sign all executables from each download portal.

The fact that each downloaded file is unique and has the same size points to the all downloads being polymorphic variants of the same program. We confirm this by running the executables in our sandbox. For both download portals, all files show the typical behaviour of a PPI downloader [2]. First, it asks the user if he wants to download the original program that the user thought it was downloading from the download portal. If the user declines to install at this point, nothing is installed. Second, it offers to install third party programs such as McAfee WebAdvisor, Avast antivirus, and Pro PC cleaner. For each offer, the user has the option to accept or decline the installation. If the user accepts the offer, another offer is displayed, until no more offers are left. If the user declines one offer, no more offers are shown. Third, all accepted offers are downloaded showing a progress bar. When the download finishes, there is an additional offer. Fourth, the user is asked if he wants to perform the installation of all the accepted offers or do it later. If the user postpones installation, the PPI downloader is still installed and will periodically remind the user to install the programs.

After the process above, the user ends up with the following programs installed: the original program it wanted to download, any offers that it accepted to install, and the PPI downloader. Note that the PPI downloader was never offered to the user, so at a minimum the user always gets an additional program that it did not desire or accepted to install.

Table 3 Percentage of undesirable programs in each download portal

RK	Portal	AV > 3			AV > 0
		All, %	PUP, %	Mal., %	All, %
1	eazel	100	100.0	0.0	100.0
2	descargarmp3	100	100.0	0.0	100.0
3	geardownloads	8.5	5.6	2.9	33.4
4	uptodown	8.3	5.0	3.3	32.1
5	tucows	7.0	4.7	2.3	35.4
6	download3000	5.9	4.4	1.5	30.2
7	filehorse	5.2	4.3	0.9	20.3
8	fileforum	5.1	3.0	2.1	33.4
9	softonic	4.9	1.9	3.0	30.1
10	majorgeeks	4.8	2.6	2.2	28.8
11	filehippo	4.3	3.3	1.0	21.6
12	softpedia	4.1	1.7	2.4	25.1
13	cnet	3.5	1.3	2.2	25.5
14	filecluster	3.3	2.1	1.2	25.0
15	freewarefiles	2.8	1.4	1.4	25.5
16	snapfiles	2.8	1.8	1.0	26.3
17	soft112	2.3	1.1	1.2	13.9
18	soft32	1.6	0.4	1.2	16.8
19	fileguru	1.4	0.5	0.9	15.6
20	portalprogramas	0.2	0.1	0.1	16.5

Table 4 Top ten PUP and malware families

Rank	PUP			Malware		
	Family	Files	Type	Rank	Family	Files
1	installcore	6033	PPI	15	delf	50
2	opencandy	757	PPI	17	autoit	38
3	securityxploded	202	DP	18	zbot	32
4	pswtool	148	Generic	23	joke	30
5	spigot	100	Aff.	29	scar	23
6	prefchanger	95	Generic	32	bumble	19
7	relevantknowledge	89	Marketing	34	crawler	19
8	installmonetizer	87	PPI	36	rbot	17
9	installmonster	77	PPI	37	atraps	16
10	outbrowse	72	PPI	38	ircbot	16

Table 5 Top ten undesirable authors with more than 50 files

Rank	Name	DP	Files	Signed	Pub.	All, %	PUP, %	Mal., %
1	zebnet	6	74	74	2	68	15	53
2	myplaycity	7	100	67	1	49	49	0
3	securityxploded	11	397	1	1	48	39	9
4	freeridegames	1	73	73	1	48	47	0
5	siteken	1	314	0	0	41	31	10
6	xilisoft	10	142	56	2	31	31	0
7	adobe	10	127	48	3	17	17	0
8	nirsoft	16	438	33	2	16	13	3
9	mediafreeware	4	85	9	1	15	12	4
10	microsoft	17	1930	1156	21	4	4	<1

All executables downloaded from these two download portals were unknown to VT when first downloaded, which can be expected since they seem to be generated on the fly as the user (or our crawler) requests them. However, once submitted to VT all of them were flagged as undesirable by more than three AV engines. Furthermore, if we run the executables from these download portals on a Cuckoo sandbox with no anti-anti-evasion techniques, they exhibit a different behaviour. In this situation, executables downloaded from *eazel* install a program called *Remebeher* and those from *descargarmp3* a program called *Liret*. No third-party offers are shown to the user. This change of behaviour indicates anti-sandboxing checks.

Both download portals state in their terms and conditions that they belong to *Vittalia Internet*, a known PUP publisher that used to run a PPI program called OneInstaller [1], which seems defunct. We query the IP addresses of those two portals using VT and discover another 12 Vittalia download portals hosted on those IP addresses such as solodrivers.com, filewon.com, fileprogram.net, and downloadsoft.nl. The PPI downloader offered by the Vittalia download portals is not the one from the OneInstaller PPI that Vittalia used to run. Instead, the AV engines identify them as the PPI downloader for InstallCore, an Israeli PPI program [1]. In addition, executables downloaded from *eazel* are signed by 'FunnelOpti (Alpha Criteria Ltd.)' and those downloaded from *descargarmp3* are signed by 'Delivery Agile (New Media Holdings Ltd.)'. Both New Media Holdings Ltd. and Alpha Criteria Ltd. are companies part of the IronSource group, who owns the InstallCore PPI program.

Finally, we observe that all files downloaded from *eazel* are hosted offsite at www.sendchucklebulk.com and those downloaded from *descargarmp3* come from three domains: www.sendcapitalapplication.com, www.quickbundlesnew.com, and www.guardmegahost.com. Those four domains all resolve to the same set of six IP addresses and are registered by the same privacy protection service in Israel. We believe that these domains belong to InstallCore. When a user requests to download a file, the download portals request InstallCore's API to generate on the fly a user-customised PPI downloader.

To summarise, our investigation shows that Vittalia has moved away from its own PPI service and instead has signed up as a publisher to the more popular InstallCore PPI service. When a user tries to download any program from one of Vittalia's download portals, they are instead provided an InstallCore PPI downloader generated on the fly for the user. The user may decide to install some offers from third-party advertisers who pay InstallCore for distribution, and Vittalia gets a payment for each installation it enables. The user ends up installing not only the original program that it wanted but also the PPI downloader, and any offers it accepts. This case study illuminates how some download portals are exclusively used to assist in the distribution of PPI downloaders and PUP products.

5 Related work

Download portals: Security vendors have analysed the top downloads of download portals and concluded that they are bloated with PUP [12–14]. In concurrent and independent work, Geniola *et al.* [43] collect 800 installers of promoted applications from eight download portals. They execute them in a sandbox and find that 1.3% of those installers drop well-known PUP to the system and 10% install a browser or a browser extension. One main goal of this work is measuring the amount of abuse in download portals, i.e. the percentage of PUP and malware. The main limitation of prior works towards that goal is that they analyse only the top downloaded programs or the promoted applications, which may not be representative of all distributed programs. In contrast, we have downloaded *all* the Windows programs offered by 20 download portals. We have collected 75,615 unique executables, almost two orders of magnitude more than prior works. Our results show an overall ratio of PUP and malware between 8 and 26%, significantly higher than the 1.3% reported by Geniola *et al.* Our analysis also identifies two download portals, part of a PPI distribution service, which serve 100% PUP. Finally, we have identified abusive behaviours PUP authors employ to distribute their programs through download portals.

PUP: Early work on PUP focuses on what constitutes PUP [6–8] and its deceptive methods [45–47]. Research on PUP has recently revived with a number of papers examining PUP prevalence and its distribution through commercial PPI services. Thomas *et al.* [48] measured that ad-injectors, a type of PUP that modifies browser sessions to inject advertisements, affect 5% of unique daily IP addresses accessing Google. Kotzias *et al.* [38] studied abuse in Windows Authenticode by analysing 356 K samples from malware feeds. They found that PUP has quickly increased in so-called malware feeds since 2010, that the vast majority of properly signed samples are PUP, and that PUP publishers use high file and certificate polymorphism to evade security tools and CA defenses such as identity validation and revocation. In a separate work, Kotzias *et al.* [1] used AV telemetry of 3.9 M real hosts for analysing PUP prevalence and its distribution through commercial PPI services. They found PUP installed in 54% of the hosts and identified 24 commercial PPI services that distribute over a quarter of all the PUP in their 2013–2014 dataset. They also determined that commercial PPI services used to distribute PUP are disjoint from underground PPI services used to distribute malware [49]. In simultaneous and independent work, Thomas *et al.* [2] analysed the advertiser software distributed to US hosts by four commercial PPI services. They used SafeBrowsing data to measure that PPI services drive over 60 million download events every week in the second half of 2015, nearly three times that of malware. Nelms *et al.* [50] analysed web-based advertisements that use social engineering to deceive users to download PUP. They found that most programs distributed this way are bundles of free software with PUP. This work differs from the above in that it analyses PUP prevalence in download portals.

Sandboxing: Many works have proposed sandboxing platforms for malware analysis [51–54]. Those may use in-guest tracing of Windows API calls [54], emulation [51], hardware-supported virtualisation [52], and bare machines [53]. In this work, we use the open-source Cuckoo sandbox [39].

6 Conclusion

In this work, we have performed a systematic study of abuse in download portals, which index freeware from multiple authors. We have built a platform to crawl download portals and have applied it to download 191 K Windows freeware installers from 20 download portals. We have analysed the collected installers and executed them in a sandbox. We measure an overall ratio of PUP and malware between 8% (conservative) and 26% (lax). In 18 of the 20 download portals the amount of PUP and malware is moderate, i.e. below 9%. However, we also find two download portals exclusively used to distribute PPI downloaders. We have performed a thorough analysis of those two download portals. Finally, we have detailed different abusive behaviours that authors of undesirable programs use to distribute their programs through download portals such as uploading the same file as different programs, using external links to bypass security checks, and impersonating benign popular authors.

7 Acknowledgments

Richard Rivera was supported by a SENESCYT fellowship from the Ecuador Government. This research was also partially supported by the Regional Government of Madrid through the N-GREENS Software-CM project S2013/ICE-2731, the Spanish Government through the DEDETIS grant no. TIN2015-7013-R, and the European Union through the ElasTest project ICT-10-2016-731535. All opinions, findings and conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the sponsors.

8 References

[1] Kotzias, P., Bilge, L., Caballero, J.: 'Measuring PUP prevalence and PUP distribution through pay-per-install services'. USENIX Security Symp., Austin, TX, USA, August 2016

[2] Thomas, K., Crespo, J.A.E., Rastil, R., *et al.*: 'Investigating commercial pay-per-install and the distribution of unwanted software'. USENIX Security Symp., Austin, TX, USA, August 2016
[3] CNET: http://download.cnet.com, accessed November 2017
[4] Softonic: https://www.softonic.com/, accessed November 2017
[5] Tucows: http://tucows.com, accessed November 2017
[6] Bruce, J.: 'Defining rules for acceptable adware'. Virus Bulletin Conf., Dublin, Ireland, 2005
[7] McFedries, P.: 'Technically speaking: the spyware nightmare', *IEEE Spectr.*, 2005, **42**, (8), pp. 72–72
[8] Pickard, C., Miladinov, S.: 'Rogue software: protection against potentially unwanted applications'. Int. Conf. Malicious and Unwanted Software, Fajardo, PR, USA, 2012
[9] Google: 'Unwanted software policy, 2018'. Available at https://www.google.com/about/unwanted-software-policy.html
[10] Microsoft: 'How Microsoft antimalware products identify malware and unwanted software', 2018. Available at https://www.microsoft.com/en-us/wdsi/antimalware-support/malware-and-unwanted-software-evaluation-criteria
[11] MalwareBytes: 'PUP reconsideration information – how do we identify potentially unwanted software?', 2018. Available at https://www.malwarebytes.com/pup/
[12] EmsiSoft: 'Mind the PUP: top download portals to avoid'. Available at http://blog.emsisoft.com/2015/03/11/mind-the-pup-top-download-portals-to-avoid/, accessed December 2016
[13] Heddings, L.: 'Here is what happens when you install the top 10 download.com apps'. Available at http://www.howtogeek.com/198622/heres-what-happens-when-youinstall-the-top-10-download.com-apps, accessed November 2017
[14] Heddings, L.: 'Yes, every freeware download site is serving Crapware'. Available at https://www.howtogeek.com/207692/yes-every-freeware-download-site-is-serving-crapware-heres-the-proof/, accessed November 2017
[15] ASP, The Association of Software Professionals: http://padsites.org, accessed November 2017
[16] Virustotal: https://virustotal.com/, accessed October 2017
[17] O. Java: 'What are the ask toolbars?'. Available at https://www.java.com/en/download/faq/ask_toolbar.xml
[18] Alexa Website Ranking: http://www.alexa.com/, accessed November 2017
[19] Uptodown: https://www.uptodown.com, accessed November 2017
[20] FileHippo: http://filehippo.com, accessed November 2017
[21] Softpedia: http://www.softpedia.com/, accessed November 2017
[22] Soft112: http://soft112.com, accessed November 2017
[23] MajorGeeks: http://majorgeeks.com, accessed November 2017
[24] Soft32: http://soft32.com, accessed November 2017
[25] Eazel: http://eazel.com, accessed November 2017
[26] FileForum: http://fileforum.betanews.com, accessed November 2017
[27] FileHorse: http://filehorse.com, accessed November 2017
[28] PortalProgramas: http://portalprogramas.com, accessed November 2017
[29] FreewareFiles: http://freewarefiles.com, accessed November 2017
[30] SnapFiles: http://snapfiles.com, accessed November 2017
[31] FileCluster: http://filecluster.com, accessed November 2017
[32] DescargarMP3: http://descargar.mp3.es, accessed November 2017
[33] Download3000: http://download3000.com, accessed November 2017
[34] FileGuru: http://fileguru.com, accessed November 2017
[35] GearDownload: http://geardownload.com, accessed November 2017
[36] Selenium WebDriver: http://www.seleniumhq.org/projects/webdriver/
[37] Sebastián, M., Rivera, R., Kotzias, P., *et al.*: 'Avclass: a tool for massive malware labeling'. Symp. Research in Attacks, Intrusions, and Defenses, Evry, France, 2016
[38] Kotzias, P., Matic, S., Rivera, R., *et al.*: 'Certified PUP: abuse in authenticode code signing'. ACM Conf. Computer and Communication Security, Denver, Colorado, USA, 2015
[39] Cuckoo Sandbox: https://cuckoosandbox.org, accessed October 2017
[40] Hardening Cuckoo Sandbox against VM aware malware: https://www.alienvault.com/blogs/labs-research/hardening-cuckoo-sandbox-against-vm-aware-malware
[41] Ferran, O.: 'How to detect the Cuckoo Sandbox and hardening it?'. EICAR Annual Conf., Hannover, Germany, 2013
[42] W3C: 'OS statistics', 2017. Available at https://www.w3schools.com/browsers/browsers_os.asp
[43] Geniola, A., Antikainen, M., Aura, T.: 'A large-scale analysis of download portals and freeware installers'. Nordic Conf. Secure IT Systems, Tartu, Estonia, 2017
[44] Avant Browser: http://www.avantbrowser.com/, accessed November 2017
[45] Edelman, B.: 'Spyware installation methods'. Available at http://www.benedelman.org/spyware/installations/
[46] Good, N., Dhamija, R., Grossklags, J., *et al.*: 'Stopping spyware at the gate: a user study of privacy, notice and spyware'. Symp. Usable Privacy and Security, Pittsburgh, PA, USA, 2005
[47] Good, N., Grossklags, J., Mulligan, D.K., *et al.*: 'Noticing notice: a large-scale experiment on the timing of software license agreements'. SIGCHI Conf. Human Factors in Computing Systems, San Jose, CA, USA, 2007
[48] Thomas, K., Bursztein, E., Grier, C., *et al.*: 'Ad injection at scale: assessing deceptive advertisement modifications'. IEEE Symp. Security and Privacy, San Jose, CA, USA, May 2015
[49] Caballero, J., Grier, C., Kreibich, C., *et al.*: 'Measuring pay-per-install: the commoditization of malware distribution'. USENIX Security Symp., San Francisco, CA, USA, August 2011
[50] Nelms, T., Perdisci, R., Antonakakis, M., *et al.*: 'Towards measuring and mitigating social engineering malware download attacks'. USENIX Security Symp., Austin, TX, USA, 2016
[51] Anubis: http://anubis.iseclab.org/, accessed November 2017
[52] Dinaburg, A., Royal, P., Sharif, M., *et al.*: 'Ether: malware analysis via hardware virtualization extensions'. ACM Conf. Computer and Communications Security, Alexandria, VA, USA, 2008
[53] Kirat, D., Vigna, G., Kruegel, C.: 'Barecloud: bare-metal analysis-based evasive malware detection'. USENIX Security Symp., San Diego, CA, USA, 2014
[54] Willems, C., Holz, T., Freiling, F.: 'Toward automated dynamic malware analysis using CWSandbox', *IEEE Secur. Priv.*, 2007, **5**, (2), pp. 32–39