A Large-Scale Analysis of Download Portals and Freeware Installers

Alberto Geniola[1], Markku Antikainen[2(✉)], and Tuomas Aura[1]

[1] Aalto University, Espoo, Finland
[2] Helsinki Institute for Information Technology, University of Helsinki, Helsinki, Finland
markku.antikainen@helsinki.fi

Abstract. We present a large-scale study of Windows freeware installers. In particular, we look for potentially unwanted programs (PUP) and other potentially unwanted modifications to the target system made by freeware installers. The analysis is based on almost 800 installers gathered from eight popular software download portals. We measure how many of them drop PUP, such as browser plugins, or make other modifications to the system. In addition to these results, we find that most installers that download executable files over the network are vulnerable to man-in-the-middle attacks, which in the worst cases may be used to execute arbitrary code with elevated privileges on the target system. Moreover, serious man-in-the-middle vulnerabilities are found in application managers provided by download portals.

1 Introduction

Most computer users download and install some freeware applications from the Internet. The source is often one of the many download portals, which aggregate software packages and also offer locations for hosting them. It is common concern that the downloaded software might be infected with malware or have other unwanted side effects. Freeware installers are also known for dropping potentially unwanted programs (PUP) to the user's computer. PUP and other unwanted system modifications to desktop computers can be considered a security threat [5,21]. This phenomenon is partly caused by the *pay-per install* (PPI) business model where freeware software developers monetize their software by bundling it with other third-party applications or by promoting some software and services by changing the user's default settings. This business model is not always illegal as the application installer may inform the users about the third-party software and even allow them to opt-out. However, this is often done in a way that the user is not fully aware of the choices made.

In this paper, we set out to analyze nearly 800 popular software installers from download portals. We do this with an automated analysis system that downloads and installs the applications in a sandbox while monitoring the target system. The sandbox emulates the behavior of a lazy user who tries to complete the installation process with the default settings of the installer. That is, we assume

H. Lipmaa et al. (Eds.): NordSec 2017, LNCS 10674, pp. 209–225, 2017.
https://doi.org/10.1007/978-3-319-70290-2_13

that the user wants to finish the installation as fast as possible and is habituated to accept the default settings and to bypass warnings.

Our study differs from earlier research [4,20] in several respects. First, we try to better understand the prevalence of any problems by gathering large quantities of software from the most popular download portals. Second, we do not differentiate between legitimate and malicious actions, which would easily lead to complicated legal and moral arguments, but instead try to cover all potentially unwanted changes to the system. Thirdly, our research methodology provides insights to software installers and download portals in general.

The most important findings from our study are following. We find that, while the most popular download portals do not distribute malware, some (1.3%) of the studied installers drop a well-known PUP to the target system. Furthermore, nearly 10% of the installers came with a with a third-party browser (e.g. Chrome) or a browser extension. On the positive side, we find no evidence that download portals would themselves bundle significant amounts of potentially unwanted content to the downloads – the PUPs seem to come from the original freeware authors. When analyzing the installers, we also find prevalent vulnerabilities. The installers often download the application binaries over HTTP, and over half of the installers that do so, do not verify the integrity of the binary and are thus vulnerable to man-in-the-middle (MitM) attacks. We also spot serious MitM vulnerabilities in update managers of two major download portals, which allow an attacker to underhandedly advertise malicious binaries as software updates.

The rest of this paper is organized as follows. Section 2 reviews related work. In Sect. 3, we describe the methodology and then briefly explain the analysis system. Analysis results are presented in Sect. 4 and further discussed in Sect. 5. Section 6 concludes the paper.

2 Background

This section describes the related work and ideas on which our research is based.

Downloading applications from the Internet can be dangerous, and this also applies to download portals [9,10]. The applications might come with unwanted features that range from clearly malicious, such as bundled malware and spyware, to minor nuisances like changing the browser's default search engine. Such software is often referred to as *potentially unwanted programs* (PUP)[1]. We use the broad definition of Goretsky [8], which states that a PUP is an application or a part of an application that installs additional unwanted software, changes the behavior of the device, or perform other kinds of activities that the user has not approved or does not expect. PUP often functions in a legal and moral gray area. The threat of legal action from PUP authors has been suggested as the reason why anti-malware labels it as "potentially unwanted" rather than "malicious" [2,13], and this was also confirmed by anti-malware developers who gave feedback on our research.

[1] Potentially Unwanted Application (PUA) is another often used term.

Recent studies have shown that freeware installers only rarely come bundled with critical malware [11]. More often, the system modifications are just unnecessary and unexpected. The user may even be informed about them, e.g., in the EULA, or the installer may allow a careful user to opt out of unwanted features. Users, however, do not always read EULAs and may be habituated to accept default settings and ok any warnings [1,17]. This *rushing-user* behavior leads the user to giving *uninformed consent* to the system modifications. While solutions have been proposed, they have not been widely adopted [2]. Moreover, PUP installers often come with a complex EULAs [7], which users are likely to accept blindly [3].

One root cause for the problem of unwanted software is the pay-per-install (PPI) business model. PPI is a monetization scheme where a software developer or distributor gets payed for dropping unrelated third-party applications to the target computer. This may be done with or without the user's consent. Recent research publications have studied the PPI business model [4,11,20]. The PPI application installer typically downloads the third-party software from a PPI distributor. Caballero et al. [4] reverse engineered protocols used by PPI distributors and found that the choice of applications depends on the target computer's geolocation. Another result is that, while PPI distributors do spread some known malware, this is not a very prevalent phenomenon—probably because blacklisting by anti-virus vendors would hurt the PPI business [11]. Another related paper analyzed black-market PPI that installs third-party applications silently in the background [20]. In the current research, we consider commercial PPI that does not necessarily try to hide its actions but rather takes advantage of the rushing user behavior to maximize the number of installs. We also analyze other unwanted side effects of the installers even if not part of the PPI business.

In summary, while there is plenty of anecdotal evidence showing that download portals distribute PUP [10,18], probably due to the PPI business model, the true extent of this problem has not been studied methodically. We aim to fill this gap by providing a comprehensive analysis of nearly 800 application installers retrieved from the most popular download portals. While the PUP phenomenon is not limited to a single operating system or platform, we focus purely on Microsoft Windows, which still is the most popular OS on desktop and laptop computers (84% market share at the time of writing [19]).

3 Methodology

This section describes the methodology and the analysis system used in our study. While the analysis system is rather complex, we describe them only briefly because the focus of this paper is on the analysis results.

3.1 Analysis System Overview

Our goal is to implement automated analysis of large numbers of Windows freeware installers. For this, we need an infrastructure that automatically downloads,

executes and analyzes the application installers. On a high level, the analysis system (1) crawls selected download portals for Windows freeware installers, (2) automatically runs them in guest machines with emulated user interaction, (3) monitors the modifications made to the guest machine as well as network communication, and (4) saves the results for later use.

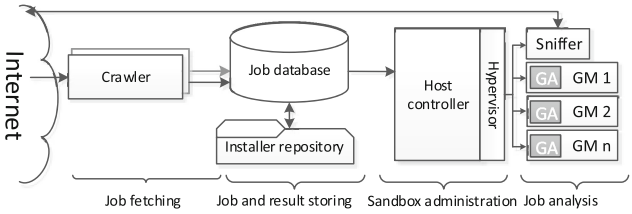

Fig. 1. Analysis system architecture

The architecture of the analysis system is shown in Fig. 1. First of all, we implemented crawlers for the download portals. The actual analysis is orchestrated by the host controller. It handles the life cycle of the guest machines, in which the installers are executed. This essentially means that the host controller is responsible for (1) fetching a job from the database, (2) initializing a guest machine and serving it the installer binary, (3) pre-processing and storing data about the installation process, and finally (4) cleaning up the guest machine. In each guest machine, there is a guest agent that pulls the installer from the host controller and drives its execution by launching it and interacting with its UI. The agent also monitors any filesystem and registry modifications and reports these to the host controller. The network traffic to the guest machines is routed through a network sniffer, which captures it. While the analysis system is modular and can support any guest OS, we have so far implemented the guest agent only for 32-bit Windows 7 guest machines.

The installers require user attention. Therefore, we implemented a heuristic interaction system which emulates the behavior of a lazy user during the installation process. When the installer runs, the guest agent tries to detect when it is waiting for user input and then sends the input event that is most likely to cause progress. The guest agent does this by observing screenshots that are taken periodically from the installer UI: the installer is likely to wait for user input if it is not performing any I/O operations and if the screenshots are stable for some time. The next input is chosen using heuristics that, for example, prefer rectangular shapes containing text such as “OK”, “Install”, or “Next”. The graphical screenshot approach was taken because most installers do not make use of the standard Windows UI components. The UI interaction heuristics in the guest agent were optimized for Windows; however, they could easily be adapted to other operating systems.

3.2 Installer Crawling

We chose eight download portals based on their Alexa rankings (Table 1). While some of these sites also provide other content than application downloads, the ranking gives a rough picture of their popularity and perceived trustworthiness.

Each studied download portal promotes a list of the most popular applications on its front page, except Softpedia which promotes recent downloads. We decided to focus on the promoted applications and set a crawler to download up top 200 installers from each portal. When possible, it applied a filter for 32-bit Windows or Windows 7 freeware. With some portals, there were fewer than 200 actual downloads, mainly because of the limitations of the web interface. Table 1 summarizes portals chosen for our study and the number of downloaded files.

In addition to crawling, we also manually downloaded installers for the most popular freeware applications directly from the developers' websites. We used Alexa rankings of top freeware applications as well as Google Trends for the most popular searches that include the words "software download". The manual download was done to compare the behavior of the installers published directly by the shareware authors with those distributed through the download portals. However, it should be emphasized that we only downloaded 20 installers manually. More extensive comparison between the portals and "original" software would not scale because it cannot be automated, and it would also be complicated by the fact that many authors use one of the portals as their main distribution point.

Table 1. Download portals studied in this paper

Download portal	Alexa rank Oct.2016	Filters	Downloaded files	Successfully analyzed
download.cnet.com	159	Win,free	200	146
softonic.com	285	Win7,free	170	126
filehippo.com	662	Win	90	64
informer.com	881	Win,free	200	117
softpedia.com	1732	Win,free	200	148
majorgeeks.com	6077	Win,free	55	37
soft32.com	7279	Win,free	200	113
brothersoft.com	8600	Win,free	41	26
manual download	–	–	20	15
			1177	792

We were not able to automatically analyze every installer. First, almost 10% of the crawled files failed either because the application was not an installer in the first place (e.g. a stand-alone application) or because of missing hardware,

software dependency, product key, or a similar reason. Additionally, 23% of the installers failed because the automated UI interaction was not smart enough. The reason was mostly complex interaction, such as selecting the directory to which the program should be installed. Another reason was that the installer used some other language than English. Nevertheless, a relatively high percentage of the installers (67%) completed. This was the result of iterative improvements to the UI automation heuristics and other parts of the analysis system.

The results discussed in the rest of this paper were obtained from the 792 installers completed successfully. Of these files, 751 were unique. We nevertheless consider even the installers with the same hash as distinct because some download portals have in the past served identical installers for several applications[2]. In these cases, the installer executable determines the further files to download and install based on its own filename.

4 Results

We present the results of our analysis in two parts. Section 4.1 describes what we can learn simply by looking at the files served by the download portals. Then, Sect. 4.2 presents the results of dynamic analysis. All the results are based on the 792 installers that were successfully executed. Some of the results are not directly related to security but are of general interest and serve as background information.

4.1 Static Properties of the Installers

This section describes some of the basic properties of the analyzed installers.

Analyzed Applications: We first compare the applications promoted on different portals. This helps to understand the data and is interesting in itself. We manually grouped the different versions of the same applications. Table 2 shows the overlap in applications at different portals. The number of distinct applications served by each portal is on the diagonal.

Our first observation is that the portals serve quite different sets of applications. Those promoted by CNET, FileHippo, Informer and Soft32 overlap the most. On the other hand, Softonic and Softpedia tend to promote applications that are not on the other portals. In the case of Softpedia, the reason may be that it does not promote the most popular software but the latest downloads. Finally, some portals use only the last week's downloads for the popularity ranking. This metric is susceptible to manipulation and short-term fluctuation, e.g. when an update is published. For these reasons, one has to be very careful when comparing different download portals based on our data.

[2] CNET's downloader VT report available at https://virustotal.com/it/file/9961ebc9782037f68b73096bcff3047489039d6dc5c089f789b3dbff4109e21b/analysis/.

Table 2 also shows the median ages of the installers served by each portal. Software age may be an indication of how seriously the publisher or download portal take security. We obtained the application ages from VirusTotal. Our assumption is that popular software tends to be submitted to VirusTotal soon after release. Although the first-seen date obtained from VirusTotal does not precisely tell how old a binary file is, it gives an independent indication of when the software began spreading more widely.

The overall observation is that much of the popular freeware is not frequently updated, and many installers are several years old. This can be a cause for concern. The collected data also shows that CNET, MajorGeeks and Softpedia serve relatively recent software installers while the rest of the portals serve considerably older binaries. In addition to the actual age of the software, the results could be explained by differences in which software the sites promote and the type of software that each portal distributes. For instance, there may be value to archiving popular legacy software that is no longer updated. But even considering such alternative explanations, we can still assert that the most popular download site CNET distributes relatively recent software: its installer ages align closely to those of manually downloaded files, which can serve as a reference metric.

Table 2. Number of common applications served by each download portal pair (different versions of same application have been combined). *Age* shows the median age in days of the installers served by each portal.

	Brothersoft	CNET	FileHippo	Informer	MajorGeeks	Soft32	Softonic	Softpedia	manual	age
Brothersoft	26	1	3	2	0	0	1	0	0	953
CNET	1	144	19	22	6	21	7	0	4	111
FileHippo	3	19	64	18	6	15	4	1	4	160
Informer	2	22	18	117	7	14	3	0	5	604
MajorGeeks	0	6	6	7	35	3	1	0	1	8
Soft32	0	21	15	14	3	112	6	2	2	573
Softonic	1	7	4	3	1	6	125	2	0	723
Softpedia	0	0	1	0	0	2	2	148	0	18
manual	0	4	4	5	1	2	0	0	15	117
# distinct files	*26*	*146*	*64*	*117*	*37*	*112*	*126*	*148*	*15*	

Application Signing: Our first security-related question was whether the installer binaries are signed. Recent research showed that while malware is generally not signed, potentially unwanted programs are [12]. We wanted to know where software distributed by the download portals stands.

The application signature verification results can be seen in Table 3. While most of the analyzed binaries (64%) had a valid signature, 30 (3.8%) cases did not verify correctly. Publisher certificate expiration was the most common cause

Table 3. Signature verification of analyzed installers

Verification outcome	# .EXE	# .MSI	# Total
Signed and verified	486	23	509
Verification error	26	4	30
Unsigned	239	14	253

of failure (24 cases). Other causes were explicit revocation (1 case) and untrusted root CA (5 case). The remaining 32% of the analyzed installers were unsigned.

Interestingly, there were differences between the download portals. CNET, FileHippo and Informer had about 80% correctly signed code while Soft32, Softonic and Softpedia had lower rates (62%, 61%, 44%, respectively). The other portals appeared to belong to the latter group, but there were too few installers for a fair comparison. The high percentage of signed files in three of the four most popular download sites seems to indicate that there is value for the publishers in code signing even though the portals do not require it.

4.2 Dynamic Analysis of Installers

This section presents results from the dynamic execution and monitoring of the installers.

Network Traffic Analysis: The sniffed traffic was analyzed with the Tshark and Bro protocol analyzers. We also implemented custom Python scripts for extracting further information.

We begin the discussion by looking at the network protocols (Table 4). Most of the traffic is HTTP and HTTPS over TCP (99%). The most frequent UDP packets were for UPnP, SSDP and DNS. Our script was unable to classify some of the UDP packets. Manual investigation revealed that such traffic mainly belongs to the BitTorrent protocol, legitimately used by torrent-based installers. In three cases, we identified JSON encoded text over UDP, which is used by content-sharing applications for advertising themselves on the local network. In one case, the installer used a variant of the GVSP video streaming protocol, presumably to show a video to the user.

Next, we focus on HTTP, which constitutes the bulk of the network traffic. Figure 2 shows the domains that are contacted by most installers and from which the installers download most of the data. It can be seen that more than 80% of the HTTP downstream traffic is from well-known CNDs. Akamai and Google are the two most-contacted ones. The figure also reveals that quite a few installers contact Google but only download small volumes of data. A close investigation revealed that 23 installers made least one HTTP request for the Google Analytics web beacon and 29 installers downloaded the Google Analytics JavaScript library. This may indicate that many freeware authors benefit from its value-added services such as user tracking and demographic data.

Table 4. Breakdown of network traffic (inbound and outbound), total for all analyzed installers

Transport layer	Application protocol	MB	(%)
TCP	HTTP	6567.84	95.22
	TLS/SSL	328.63	4.76
	Others	1.25	0.02
		6897.72	99.25
UDP	UPnP	18.03	34.51
	SSDP	17.25	33.02
	DNS	4.39	8.40
	Others	2.90	5.55
	Unknown	9.67	18.52
		52.24	0.75
ICMP		0.01	0.00
Total		6949.96	100

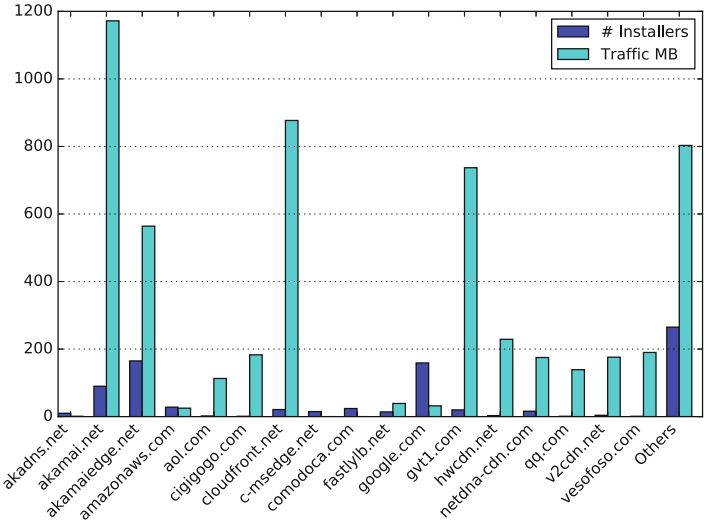

Fig. 2. HTTP downstream traffic breakdown by top domains. The blue bars (left) represent downstream traffic volume, while the cyan bars (right) indicate the number of installers that contacted the domain (Color figure online)

We reassembled and inspected the HTTP streams for binary content. Table 5 shows the results. Executable files and binary payloads constitute most of the traffic. This indicates that many of the installers (348) behave as install-time downloaders. To see if there is a clear distinction, we plotted the installer binary size and the downloaded traffic volume in Fig. 3. We have visually classified the

Table 5. HTTP downloads by MIME type

MIME type	Downloaded data (MB)	# Installers
application/x-dosexec	1879.17	96
application/octet-stream	1808.99	227
application/vnd.ms-cab-compressed	475.02	25
application/gzip	462.82	11
application/zip	267.32	32
application/vnd.ms-office	257.15	8
application/x-7z-compressed	228.09	4
application/x-bzip2	29.51	4
text/plain	16.90	787
text/html	12.33	138
others	18.13	155
Total	5455.42	788 (Distinct)

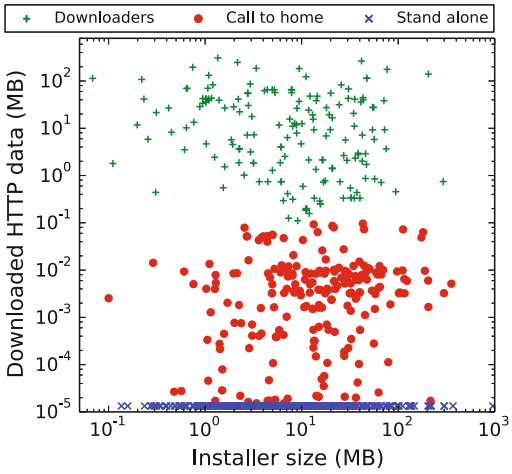

Fig. 3. Scatter plot of installer size vs downstream traffic volume. The points were divided to the three classes based on visually observed grouping

installers into three classes: downloaders, installers that call home but do now download significant amounts of data, and stand-alone installers. It can be seen from the figure that there is no apparent correlation between the installer size and the amount of data it downloads.

Man-in-the-middle Vulnerability: As seen in Table 4, installers tend to download binary files over insecure HTTP connections. These files are typically

executed within the installation process, possibly with high system privileges. In such a context, it is essential that the installers authenticate the downloaded files, e.g. with a digital signature. To check if they do that, we implemented an automated MitM attack against the installers. This was done with a transparent HTTP proxy in the sniffer component of the analysis system, which replaced executable files in HTTP responses (but not HTTPS) with its own. The malicious binary (EXE or MSI file) that was fed to the installer simply took note of its running privileges and terminated. The malicious binary was injected in the following cases:

1. Request URL ended with .EXE or .MSI
2. Response MIME type matched executable or MSI
3. First bytes of the HTTP response body matched magic numbers for EXE or MSI.

Among the 792 analyzed installers, the MitM attack was triggered 100 times. Amazingly, more than half of the attempted attacks (55%) led to immediate execution of the attacker's binary file, meaning that no authentication or integrity check was done for the downloaded binaries. Only 8 installers refused to execute the tampered file and removed it right away. In the remaining 37 cases, the attacker's file was not executed, yet it was found on the disk after the installation. 17 of these were saved in temporary system folders (subject to later removal upon system cleanup) while 20 were stored in persistent file system locations, such as under the *Program Files* directory. The latter cases leave the system open to a delayed attack when the application is used.

The MitM attack is particularly dangerous because the attacker's file is executed with the same privileges as the installer application. In 75 out of the 100 successful attacks, the malicious binary was executed with elevated privileges.

There are straightforward ways of mitigating the MitM vulnerability. One approach would be to use HTTPS for the download. Another possibility is to use asymmetric signature verified by the installer application with a static publisher public key. Clearly, there is no good technical excuse to be vulnerable.

It is worth noticing that most of the download portals distribute installers via HTTP in the first place, and the installer itself could be fake. The user, however, has the opportunity to check that the installer binary is signed by the correct publisher. In comparison, the MitM attack succeeds even if the user takes care to only execute legitimate, signed installer binaries from the correct publisher.

File System Analysis for Malware Drops: We collect the hashes of all files that are temporarily or permanently stored on the guest machine as well as hashes of files reconstructed from network traces including HTTPS connections. We looked up all the collected file hashes in VirusTotal, which aggregates results from various virus scanners. This was done two months after running the installers to leave time for new malware variants to be detected.

The number of positive results was high (235 files) but most of these were reported by only one scanner and, most likely, were false positives. Only 1.3% of

the installers contained files detected as malware by six or more scanners. More importantly, majority of such the positives were labeled as PUP. There was only one detected critical threat, and it was an Android rootkit that does not infect Win32 systems. The files with highest detection rates are listed in Table 6.

The analysis shows that download portals are not used for blatant malware distribution. The portals probably perform scans of the binaries before publishing them. On the other hand, the presence of PUP related to the well-known InstallCore PPI network indicates that download portals could still implement stricter countermeasures against grayware and bundled unwanted applications.

Table 6. Threats ranked by VirusTotal detection rate.

File name (truncated)	# Positive scanner(s)	# Inst	Description	Source portal
rootf.apk	30	1	Android rootkit	Soft32
fusion.dll	17	1	PUP InstallCore	CNET
videora.exe	15	1	PUP OpenCandy	Softonic
...Setup.exe	14	1	Adware Mobogenie	CNET
fusion.dll	14	1	PUP InstallCore	CNET
fusion.dll	12	1	PUP InstallCore	Soft32
...0061e.exe	12	1	PUP Montiera toolbar	Soft32
...B2C6B.dll	8	1	PUP Conduit	Brothersoft
...126C1.exe	6	1	PUP Zugo Toolbar	Softonic

Registry Modifications: We tracked the registry modifications made by the installers and analyzed changes to the following:

- Automatic program startup
- Default browser
- Browser plugins

There are many ways for a program to start automatically in Windows including registry keys [16] and specific file system folders. We found that 88 installers (12%) configured the installed software to automatically run at system startup.

Similar analysis was done on default browser changes: 26 installers replaced the default browser. Interestingly, 11 of these are not browser installers. Table 7 reports the details. Google Chrome turned to be the only third-party browser installed by non-browser installers. Manual investigation revealed that, in all cases, Google Chrome installation is optional but pre-selected by default.

Our analysis continued with the inspection of installed third-party browser modules. We focused on Internet Explorer, which was the only browser installed by default on the fresh Windows 7 guest machine. There were 69 registry modifications regarding browser extensions by 38 installers. As shown in Table 8, the

Table 7. Installers bundling unrelated third party browsers

Product name	Publisher	Bundled browser
Adobe Shockwave Player	Adobe Systems Inc	Google Chrome
CCleaner	Piriform Ltd	Google Chrome
Defraggler	Piriform Ltd	Google Chrome
PhotoScape	Mooii	Google Chrome
Recuva	Piriform Ltd	Google Chrome
Speccy	Piriform Ltd	Google Chrome
SUPERAntiSpyware Free	SUPERAntiSpyware	Google Chrome

Table 8. Third-party plugins dropped on IE

Portal	#Installers	Toolbar	Menu extensions	BHO	Total	%
Brothersoft	26	2	0	4	6	23.1
CNET	146	4	7	7	18	12.3
FileHippo	64	7	0	9	16	25.0
Informer	117	4	1	7	12	10.3
MajorGeeks	37	0	0	0	0	0.0
Soft32	113	2	0	4	6	5.3
Softonic	126	1	1	2	4	3.2
Softpedia	148	1	0	3	4	2.7
manual	15	1	0	2	3	20.0
Total	792	22	9	38	69	

predominant type of installed extension is the *Browser Helper Object* (BHO), which is the most powerful and potentially most dangerous IE component type because it runs in the same memory context as the browser and has access to the user's browsing data [6].

In the case of browser extensions, there were differences between the portals. MajorGeeks installers did not bundle any browser plugins, and Softpedia registered a total of just 4 dropped items over 148 installers (2.7%). Softonic and Soft32 had also relatively low rates. CNET and Informer, on the other hand, dropped considerably more browser plugins, and FileHippo topped the league with 25% drop rate. Interestingly, even manual downloaded installers bundled browser plugins. This could indicate that the plugins are bundled by the original software vendors and not by the portals.

Installer Best-Practices Compliance: Microsoft advises vendors to follow certain best practices for installers [15]. Firstly, each installed application should provide a consistent uninstall feature. For this, the installer should populate two registry keys on the system, one with the program's human-readable name

and the other with a path to the uninstaller binary. If one of these two values is missing, removal of the application becomes cumbersome. Of the analyzed installers, 82 failed to specify both the program name and uninstaller path. Another 5 only stored the product name without specifying an uninstaller binary.

Secondly, Microsoft requires installers to specify valid *ProductName* property in their metadata, which is usually placed within the resource section of the executable file. It is exposed to the user in a properties dialog [14]. 174 of the analyzed installers failed to provide this information.

4.3 App Managers and Software Updates

Some download portals provide app-manager clients for simplifying software downloads and updates. By default, app managers run at system startup and regularly check for application updates. We analyzed three app managers (FileHippo, Informer and MajorGeeks).

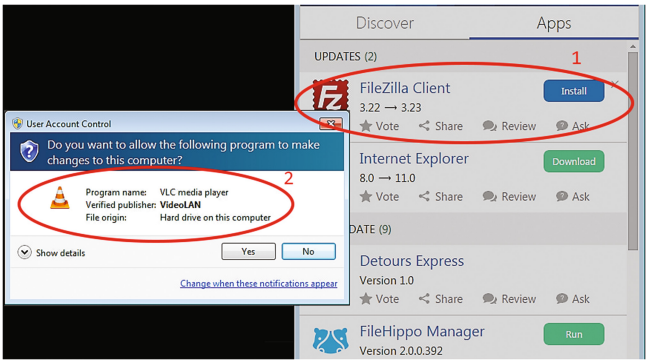

Fig. 4. Example of a successful MitM attack. User decides to update FileZilla (1), but is served a VLC installer instead (2)

Two of the analyzed app managers (FileHippo and Informer) are vulnerable to MitM attack. More specifically, they communicate with the portal using HTTP and do not check the integrity of the served binaries in any way. We were able to make modifications to the list of available updates, such as changing the download URL or adding new entries to the list of available software. Furthermore, both vulnerable app managers by default periodically check for software updates. This makes the attack even more serious in two respects: first, the attacker can push arbitrary binaries to the victim's computer without any initial user action and, second, the MitM attack can be mounted when a mobile user is visiting an untrusted access network such as wireless hotspot.

Figure 4 illustrates the MitM attack. Windows UAC still asks user's permission to run the attacker-selected application, which is either unsigned or has a different name from what the use should expect, but the user may not pay attention to such details—especially after explicitly deciding to install the update.

5 Discussion and Future Work

This research was motivated, in part, by the suspicion that download portals might distribute malware or bundle excessive amounts of unwanted programs to freeware downloads. The results of our analysis do *not* support these suspicions. No serious infections by known malware were detected, and the bundled PUPs seem to have been mostly included by the original freeware authors. Thus, freeware distribution does not appear to be such a Wild West as has been suggested in the past.

From the security viewpoint, the most important negative discoveries were:

- The median age of installers varies notably among portals, and the distributed freeware versions are often not the latest.
- Some portals host installers that bundle known PUP.
- The most common types of PUP bundled with freeware are third-party browser plugins and the Google Chrome browser.
- Many installers that download executable files are vulnerable to MitM attacks that enable code injection to the client machine.
- Some app managers provided by the portals are similarly vulnerable to MitM attacks and code injection.

While we make some comparisons between the portals throughout the paper, it would not be possible to make fair ranking of the portals regarding security or PUP. The portals differ in the types and quantity of software available. While Softpedia does well on all the metrics, it promotes a different set of software than the other portals (based on recent downloads rather than popularity), and thus the results may not be comparable. Further, our study is based on a snapshop and is limited to a single point in time. A longitudinal study would be needed for comparing different portals.

Our methodology does have some limitations. Firstly, our UI automation could still be more reliable. Manual inspection of installation screenshots confirmed that our UI engine was able to correctly automate 67% of the installers. There clearly is still scope for improvement. Secondly, our current analysis system only supports 32-bit Windows 7 on the guest machines. This is because of the free availability of an API hooking library for this platform. Thirdly, it is possible that some installers drop a plugin to an already installed third-party application (e.g. a Google Chrome plugin). Because the guest machines were initialized with a fresh OS, we have not spotted these. Fourthly, we have focused on grayware PUP and explicit changes to the system. We did not emphasize stealthiness in the design of our analysis system, due to which evasive malware may have escaped our analysis. We also cannot exclude the possibility of custom backdoors, spyware features, or other malicious behavior in the software.

6 Conclusion

We present analysis results from almost 800 freeware application installers obtained from download portals. The results indicate that portals do not actively

distribute known malware. Many freeware installers, however, come bundled with unwanted programs and have links to PPI networks. The most dangerous detected security flaw is the lack of authentication for downloads made by installers and app managers. They are vulnerable to man-in-the-middle attacks that enable code injection to the client machines. We expect to extend the scope and coverage of the analysis and will share our data with the research community.

References

1. Böhme, R., Köpsell, S.: Trained to accept? A field experiment on consent dialogs. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2010, pp. 2403–2406. ACM, New York (2010)
2. Boldt, M., Carlsson, B.: Privacy-invasive software and preventive mechanisms. In: International Conference on Systems and Networks Communications, ICSNC 2006, p. 21, October 2006
3. Bruce, J.: Defining rules for acceptable adware. In: Proceedings of the Fifteenth Virus Bulletin Conference (2005)
4. Caballero, J., Grier, C., Kreibich, C., Paxson, V.: Measuring pay-per-install: the commoditization of malware distribution. In: Proceedings of the 20th USENIX Conference on Security, SEC 2011, USENIX Association, Berkeley (2011)
5. Emm, D., Unuchek, R., Garnaeva, M., Ivanov, A., Makrushin, D., Sinitsyn, F.: It threat evolution in Q2 2016. Technical report (2016). https://securelist.com/files/2016/08/Kaspersky_Q2_malware_report_ENG.pdf
6. Esposito, D.: Browser helper objects: the browser the way you want it. https://msdn.microsoft.com/en-us/library/bb250436(v=vs.85).aspx. Accessed 29 Dec 2016
7. Good, N., Dhamija, R., Grossklags, J., Thaw, D., Aronowitz, S., Mulligan, D., Konstan, J.: Stopping spyware at the gate: a user study of privacy, notice and spyware. In: Proceedings of the 2005 Symposium on Usable Privacy and Security, SOUPS 2005, pp. 43–52. ACM, New York (2005)
8. Goretsky, A.: Problematic, unloved and argumentative: what is a potentially unwanted application (PUA)? Technical report, November 2011. Accessed 03 June 2016
9. Heddings, L.: Stop testing software on your PC: use virtual machine snapshots instead. http://www.howtogeek.com/206286/stop-testing-software-on-your-pc-use-virtual-machine-snapshots-instead/
10. Heddings, L.: Yes, every freeware download site is serving crapware (here's the proof). Technical report. http://www.howtogeek.com/207692/yes-every-freeware-download-site-is-serving-crapware-heres-the-proof/
11. Kotzias, P., Bilge, L., Caballero, J.: Measuring PUP prevalence and PUP distribution through Pay-Per-Install services. In: Proceedings of the USENIX Security Symposium (2016)
12. Kotzias, P., Matic, S., Rivera, R., Caballero, J.: Certified PUP: abuse in authenticode code signing. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 465–478. ACM (2015)
13. McFedries, P.: Technically speaking: the spyware nightmare. IEEE Spectr. **42**(8), 72–72 (2005)
14. Microsoft: VERSIONINFO resource. https://msdn.microsoft.com/en-us/library/aa381058.aspx. Accessed 05 Jan 2017

15. Microsoft: Windows installer and logo requirements. https://msdn.microsoft.com/en-us/library/windows/desktop/aa372825(v=vs.85).aspx. Accessed 30 Dec 2016
16. Microsoft: Run, RunOnce, RunServices, RunServicesOnce and Startup, November 2006. Accessed 08 Dec 2016
17. Motiee, S., Hawkey, K., Beznosov, K.: Do windows users follow the principle of least privilege? Investigating user account control practices. In: Proceedings of the Sixth Symposium on Usable Privacy and Security, p. 1. ACM (2010)
18. Slade: Mind the PUP: top download portals to avoid. Technical report, March 2015. http://blog.emsisoft.com/2015/03/11/mind-the-pup-top-download-portals-to-avoid/
19. Statcounter: Desktop operating system market share worldwide, June 2017. http://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-201706-201706-bar
20. Thomas, K., Crespo, J.A.E., et al.: Investigating commercial pay-per-install and the distribution of unwanted software. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 721–739. USENIX Association, Austin, August 2016
21. Wood, P., Nahorney, B., Chandrasekar, K., Wallace, S., Haley, K., et al.: Symantec internet security threat report trends for 2016. Technical report, April 2016