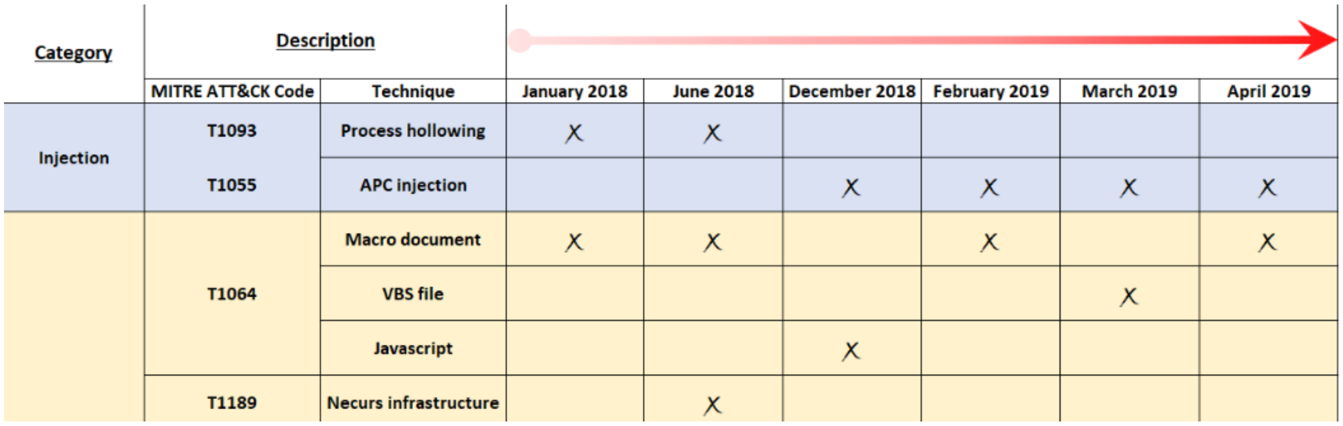
Ursnif: The Latest Evolution of the Most Popular Banking Malware

blog.yoroi.company/research/ursnif-the-latest-evolution-of-the-most-popular-banking-malware/

April 5, 2019



Category	Description							
	MITRE ATT&CK Code	Technique	January 2018	June 2018	December 2018	February 2019	March 2019	April 2019
Injection	T1093	Process hollowing	X	X				
	T1055	APC injection			X	X	X	X
	T1064	Macro document	X	X		X		X
		VBS file					X	
		Javascript			X			
	T1189	Necurs infrastructure		X				

04/05/2019

Introduction

Few days ago, the researchers of ZLab Yoroi-Cybaze dissected another attack wave of the infamous Ursnif malware, also known as Gozi ISFB, an offspring of the original Gozi which source code was leaked in 2014. Ursnif/Gozi is active from over a decade and was one of the most active malwares listed in 2017 and 2018. Today it constantly reaches several organization across Italy presenting itself in several ways, for instance as a malicious document delivered through email.

The malware has evolved over time and has added functionality, in fact, apart from collecting banking credentials it is also able to collect keystrokes, cryptocurrencies, screenshots, webmail, integrating spyware features together with banking Trojans features.

During their investigations, researchers of ZLab Yoroi-Cybaze intercept a new variant of this malware delivered through malspam campaign towards Italian companies. This latest Ursnif variant shows the same modus operandi: a malicious document in which is embedded an highly obfuscated VBA macro that acts as a first stage dropper.

The Ursnif Threat Evolution

According to Microsoft since its appearance in 2009, Ursnif has shown incredible capabilities to steal users' credentials, credentials for local webmail, cloud storage, cryptocurrency exchange platforms and e-commerce sites while remaining more stealthiness as possible. It uses many advanced trick to evade several sandboxes environment and today is the most popular malware spreading in the wild. ZLab researchers have studied many samples in the past to profile the techniques used by the malware, to track its evolution and sophistication over time.

Table 1: Ursnif techniques evolution
First analyzed sample backs to January 2018. That Ursnif variant has delivered through a macro document and consist of a few obfuscated stage and a process hollowing injection technique to execute its payload. After few months, in June 2018, we find evidence that Ursnif was delivered through Necurs Botnet. The latter is one of the most famous botnets known nowadays and it has been used to deliver this Ursnif variant. The hidden link among necurs and ursnif has been discovered by ZLab researchers as explained in this link. In December 2018, a first shift is about the implementation of many dropper stages, in order to hide the final payload; moreover, in order to execute its payload, the malware does not perform a classical process injection as in the previous samples but an APC injection, not yet seen as payload injection trick used by Ursnif.

The sample spread in February 2019 use two new features: the first one is a several obfuscated powershell stages in order to evade AVs and reduce its detection, the second one is the use of steganography technique. The latter permit to hide code into a legit image manipulating specific bits. Next, another code perform a decryption and execution of malicious code into the victim machine.

In March 2019 another weaponized variant of Ursnif has been detected: in this case, to spread the malicious software, a google drive document combined with an obfuscated VBA Script is used over steganography. The last sample shown in previous table is similar to February's sample but include another interesting feature: in this case a first VBS stage is encrypted using the Vigenere cipher; this allow to hide its malicious code and evade many sandboxes environment. We are observing a continuous evolution due to several features added in few months, this is an indicator that this malware is still in development and, observing also features fragmentation among variants lets us think, with high confidence, that there are various fork of the same codebase spreading in the wild.

Technical Analysis

Sha 256	34669dde1e33ec96147540433f60e90056d38df1e3bb952fdc600e979d74f690
Threat	Ursnif dropper
Descrizione Breve	Excel with macro
ssdeep	1536:hn1DN3aMePUKccCEW8yjJTdrBX/3t4k3hOdsylKlgryzc4bNhZFGzE+cL4LgldAK:hn1DN3aM+UKc

Table 2: information about Ursnif dropper

The most widespread infection vector observed were the macro enabled office documents, and this variant uses the same technique too. The malicious document looks like an invoice that requires enabling macros in order to proper view its contents.

Figure 1: excel document requiring macro enabling
The whole infection chain begins when the macro is enabled. This Ursnif variant presents a macro protection technique technique that it's not present in previous variants, in order to make the analysis hard avoiding manipulation and extraction. After extraction of OLE object inside the document we are able to see the content of macros and their associated name, as shown in the following figure:

Figure 2: macros isolation
Now it is possible to isolate an interesting macro in order to further analyze it in detail. It contains a piece of VBA that was extracted.

Figure 3: VB macro source code
In a different way than the past waves, the malware author added a "*VigenereDo*" function to decrypt and reconstruct the initial infection step, using an algorithm based on the Vigenère cipher, a classical polyalphabetic cipher.

The resulting command text is obtained combining the obfuscated strings defined in "*jeneric*" function with other strings (not visible in figure) and after further some manipulations is possible to spot the whole script will be executed. When user enable macros, the "*wmic.exe*" process run the following code through the "*wmic 'PRocesS'   "Call" 'CREATe"*" command.

Figure 4: the powershell script (crypted)
So, at this point, several powershell deobfuscation steps occurs. First of all, every value ("*${1F}*") defined in the ps string is replaced with content stored into "*$1F*" variable corresponding to "," (comma) character. After having replaced these values, the script is run through "iex" primitive invoked by "*.($psHomE[4]+$pshOMe[34]+'X')*" and next through "*(". (`$ShELLid[1]+`$shelLID[13]+'X')*". The complete deobfuscated script is the following.

Figure 5: the first powershell script (decrypted)

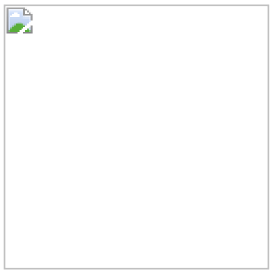


Figure 6: image with malicious embedded powershell script

First of all the malware checks the current *TimeZone* in order to verify if it is set on +01:00. If true, it download the next stage from “hxxps://i[.]imgur[.]com/TVkWKQa[.]png”. As well as in other recent attacks, the downloaded image hides another powershell stage leveraging steganography techniques.

The malware code iterates over each pixel of the image and through several mathematical binary operation converts grabs the two Least Significant Bits of every byte of the picture, concatenating them with other LSBs to produce a complete Powershell code.

Figure 7: second powershell script extracted from the steganographic image
Et voilà, another URL is found but, before download the next stage from it, the malware perform a further checking in order to evaluate the value returned by “*CurrentCulture*”.

Figure 8: CurrentCulter verification in powershell
If check is verified, once again through the “IEX” primitive it try to download other components named “ose000000.exe” from “hxxps://nuovalo[.]site/RGI82B3.-tmp-tmp”, saving it into “%TEMP%” folder. In the following table are shown the information about sample.

Sha 256	0f2245eec921949d9a1d8e13cab747a7fbb137aaee3b9ddacee0681c8b2e4fa8
Threat	Ursnif
Descrizione Breve	Final payload of Ursnif banking malware
ssdeep	6144:LCLAh6EzJYJtmavTXyulcNcyuo8PGJMewXo79y:L54EzetmCb3cNc3o0PR4

Table 3: information about Ursnif final payload

Conclusion

This latest Ursnif wave keeps showing a complex infection process. The starting point of the entire chain was the usual Visual Basic macro, this time protecting its code with a Vigenère cipher, responsible of the decryption of the additional Powershell stage launched abusing the Windows Management Infrastructure (WMI) functionalities, decoupling it to the original infection tree and then completing the infection chain exploiting steganography techniques to bypass network detection and several environmental check, to ensure the malware is running into expected machines confirming the highly evasive trend of this aggressive malware threat.

Indicator of Compromise

Hashes:

- 34669dde1e33ec96147540433f60e90056d38df1e3bb952fdc600e979d74f690
- 0f2245eec921949d9a1d8e13cab747a7fbb137aaee3b9ddacee0681c8b2e4fa8

Dropurl:

- hxxps://i[.]imgur[.]com/TVkWKQa[.]png
- hxxps://nuovalo[.]site/RGI82B3.-tmp-tmp

C2:

hxxp://nuovalo[.]icu/images/VYs5UVqTBYpg/NLIzhwT9sEr/k5V7J2HpECsh9Q/PEQNY9sYYk6JJwA_2F2Bo/pJhVLIob8XfZrIAv/Z4Kfzn93mPtOoBm/XSvYLd89VPxLpVwdy4/wEwkKFt2N/HAOr6bgXfRaPw2gN3hlS/qqBsTeYAfyj/qODGq[.]avi

Yara Rules

```
import "pe"
rule Ursnif_documentDropper_201904 {
meta:
        description = "Yara rule for Ursnif loader - April 2019 version"
        author = "Yoroi - ZLab"
        last_updated = "2019-04-02"
        tlp = "white"
        category = "informational"
strings:
        $a1 = { 43 6F 77 61 6E 64 43 6F 77 }
        $a2 = { 56 69 67 65 6E 65 72 65 44 6F }
        $b1 = "xlTickLabelPositionNextToAxis" ascii wide

condition:
        all of them
}

rule Ursnif_201904 {
meta:
        description = "Yara rule for Ursnif - April 2019 version"
        author = "Yoroi - ZLab"
        last_updated = "2019-04-02"
        tlp = "white"
        category = "informational"
strings:
        $a1 = "PECompact2" ascii wide

condition:
        all of them and pe.imphash()=="09d0478591d4f788cb3e5ea416c25237"
}
```

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