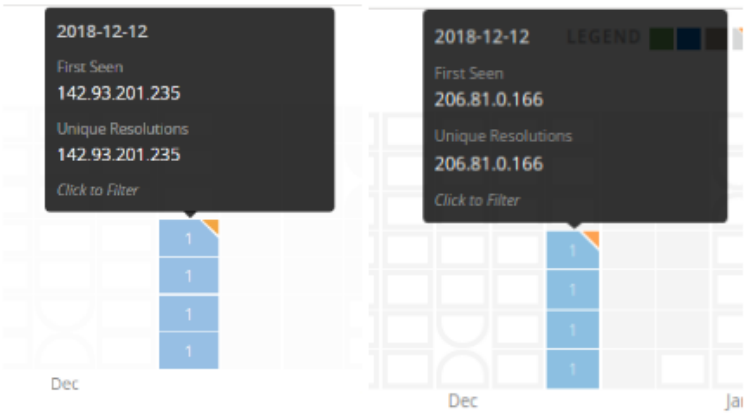
The “AVE_MARIA” Malware

blog.yoroi.company/research/the-ave_maria-malware/

January 11, 2019



01/11/2019

The Cybaze-Yoroi ZLab researchers analyzed phishing attempts spreading in the last days of the past year against an italian organization operating in the Oil&Gas sector. The malicious emails try to impersonate a supplier’s sales office sending invoices and shipping orders confirmations. As usual, the mail conveys malicious Excel files exploiting the popular CVE-2017-11882 vulnerability to run an executable retrieved from a malicious website, previously compromised by the attackers.

The domains used to vehicle the malicious messages remained active only for few days in the middle of December, just the time needed to spread phishing emails.

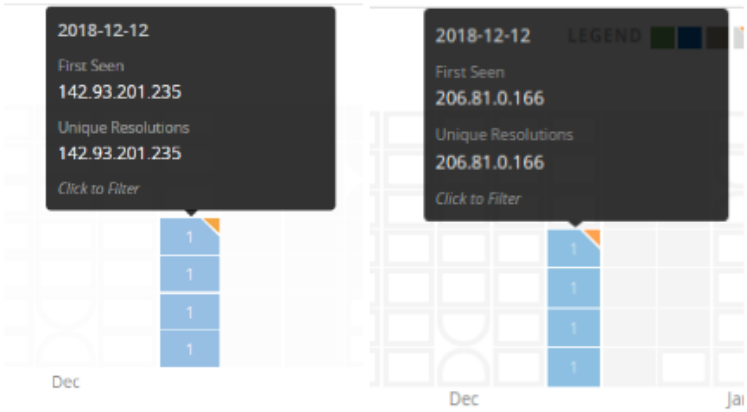


Figure 1. The sender’s domains were active from 12 to 14 Dec 18

The Cybaze-Yoroi ZLab analyzed and dissected the payload delivered during these days.

Technical analysis

The actual infection chain starts from the self-extracting archive (SFX) dropped by after the opening of the malicious Office document. The sample contains the image of Kagamine Rin as icon, a character belonging to the singing voice synthesizer software dubbed VOCALOID.

The file is a WinRAR self-extractor configured to unpack its contents into the temporary folder "*%TEMP%\04505187"* and then silently run a specific setup routine:

Figure 2. Configuration of the SFX extractor
The timestamp of the compressed files shows the attacker weaponized the archive at 22:56 of 13th December 2018, within the domain activation time-span.

Figure 3. Files extracted by SFX executable
All the files have misleading extension to confuse the analysis and most of them are text files containing junk data. But three of these files deserve further attention:

- *xfi.exe*: a legit AutoIt interpreter; it is able to execute a specified AutoIt script.
- *hbx=lbl*: the first AutoIt script; it is obfuscated hiding the instructions among an huge number of comments.
- *uaf.icm*: file containing all the malware settings such as the installation folder, the interpreter name and other parameters used in the next stages; it is structured according to the "*INI file format*".

Figure 4. Appearance of the first AutoIt script (called "hbx-lbl").
Figure 5. uaf.icm's structure.
Similar packing of AutoIT code have been observed even by Juniper back in 2016, where SFX files were abused this way to deliver scripts used as first stage of the malware. As shown in the configuration in Figure 2, the sample able to run the first script using the command:

> **$> xfi.exe hbx=lbl**

At this point, using the encoded data contained into *"uaf.icm"* between the string pattern *"[sData]"* and *"[esData]"*, the first script creates a second one, with a random name (es. *"ZZQLZ"*), and runs it using "*xfi.exe"* engine.

Figure 6.The second script is binary-encoded and hidden into the uaf.icm file between "[sData]" and "[esData]".
The second script is heavily obfuscated using binary-encoding. After deobfuscation, it reveals interesting capabilities. First of all, there are different evasion techniques, such as a check about the current running processes: if there is a process related to some virtualization software, like Virtualbox, the malware kills itself.

Figure 7.Example of malware evasion.
The main purpose of the second script is to decrypt and execute the final payload hidden inside *"[Data]"* and *"[eData]"* delimiter strings of the "*uaf.icm*" file. The data is decrypted using the "Advapi32.dll!CryptDecrypt" Microsoft function, which is dynamically invoked into the AutoIt script through the high-level API "DllCall". The decryption key is retrieved from the usual settings file.

It is interesting the way used by the AutoIt script to run the just extracted payload. In the first instance, the malware creates a copy of legit *Regsvcs.exe*, the .NET Services Installation Tool, into %TEMP% folder and runs it. Then, it performs a process injection in order to start the malicious payload behind the Regsvcs process.

In the following figure, it is shown the routine to extract, decrypt and inject the malicious binary stored into *"uaf.icm"* settings file.

Figure 8.Example of malware evasion.
The malware uses the *CallWindowProcW* Windows function as process injection technique, through *DllCall* AutoIt API.

Figure 9.Function to decrypt and inject malicious payload into legit process.
The malware author used a custom shellcode stored into *$ASM* variable to correctly inject the binary payload into the running *regsvcs* process.

Finally, the second AutoIt script provides to set persistence onto the victim's machine writing the registry key *HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run*.

Figure 10.Registry key set by the malware.
The registry key's name corresponds to the value extracted from *"uaf.icm"* settings file at the section *"Key"*.

The Payload: AVE_MARIA Stealer

The payload injected into legit .NET process shows a typical bot behavior: it contacts a C2 hosted on *anglekeys.warzonedns[.]com* and retrieves the next action to perform. The attacker' server is currently down, so it is not possible to obtain further stages of the commands.

A static investigation shows the malware looks for the installed e-mail client, like Microsoft Exchange Client or Outlook, to exfiltrate victim's credentials.

Figure 11.Research of the installed email client software.
Moreover, the bot is able to decrypt all the credentials stored by Firefox browser. These sensitive data are protected using PK11 encryption from Mozilla Network Security Services, so the malware is weaponized with all the necessary functions decrypt them.

The malware writer re-used publicly available code to implement this functionality. The following screen shows part of the execution flow (on the left) and a piece of code belonging to a KeePass plugin (on the right) published on github; these two flows are very similar.

Figure 12.Malware's piece of code (on the left); KeePass plugin's piece of code (on the right).
In addition, the malware embeds an utility able to bypass the User Access Control within the resource section. It abuses a vulnerability of the "*pkgmgr.exe"* Windows tool; many resources related to this exploit are publicly available on the internet.

Figure 13.Workflow of the UAC bypass utility.
Despite the wide malware's capabilities, the writer left some evidences referring to his environment into the malicious code.

Figure 14.Probable address path of the malware writer's workspace.
Finally, another strange string is emerged from the executable: *"AVE_MARIA"*. Which is used as HELLO message when the malware correctly contacts the C2. This particular string has been elected as common malware name by many researchers of the InfoSec community.

Figure 15.The characteristic string sent by the malware.

Conclusion

The first stages of the malware, including the AutoIt scripts, are very similar to another malware waves analyzed few years ago by third party security researchers: the malware logic, based on an INI settings file, and some pieces of AutoIt code are the same but the final payload is different.

It's possible the author of these malware is the same, showing an increasing complexity of implant, or also the first stage of the malware may have been purchased in the dark markets and the author of the "AVE_MARIA" malware composed a new stealer using publicly available code, forgetting to wipe the information related to him.

Indicator of Compromise

FileName	sha256	description
DSK.exe	4576d9940db9a748378a7e7d8c0edc048529ed72ef5161ed4a75c5612da3d5d9	SFX dropper
hbx=lbl	6fff30ad7d09e11e85614de11ea3607ed39c2c6ed2cca481d7e54b506c423707	AutoIt script dropper 1
xfi.exe	237d1bca6e056df5bb16a1216a434634109478f882d3b1d58344c801d184f95d	Legit AutoIt engine
uaf.icm	7b5a8198138abc2436d92dfcd16f0be26e8783a51e42d2a4ad5334686f4c9140	Malware settings file
ZZQLZ	02cb295e95881abca2fe85fadc4228a932a12ea0d1fa6b961a38d789e7b8287f	AutoIt script dropper 2
payload	81043261988c8d85ca005f23c14cf098552960ae4899fc95f54bcae6c5cb35f1	AveMaria payload
uac_bypass	021d01fe3793879f57a2942664fc7c096710e94e87ad13dc21467c12edf61546	UAC_bypass utility

- Malspam
  Sending domains
  - sentinelx[.tk
  - xinchingho[.ml
- Droprul
  hxxps://www.masaimaranationalparkkenya[.com/wp-admin/js/jsk/DSK.exe
- C2:
  - anglekeys.warzonedns[.com
  - 192.3.162[.161

Yara Rules

```
rule SFX_AutoIt_dropper_09_01_2019{

    meta:
      description = "Yara Rule for SFX dropper"
      author = "Cybaze Zlab_Yoroi"
      last_updated = "2019_01_09"
      tlp = "white"
      category = "informational"

    strings:
        $a1 = "CryptProtectMemory"
        $a2 = {2A 3F ?? ?? ?? ?? 72 61 72}
        $b = {4D 5A}
        $c = {CE E8 DC F8 FF FF 56 E8 A3 80 FF FF 59 33 C0 5E}
        $d = "publicKeyToken=\"6595b64144ccf1df\""
        $e = {7B 65 32 30 31 31 34 35 37 2D 31 35 34 36}

    condition:
        $b and $c and $d and $e and 1 of ($a*)

}


rule AveMaria_infostealer_09_01_2019{

    meta:
      description = "Yara Rule for AveMaria infostealer"
      author = "Cybaze Zlab_Yoroi"
      last_updated = "2019_01_09"
      tlp = "white"
      category = "informational"

    strings:
        $a1 = "PK11SDR_Decrypt"
        $a2 = {70 69 6E 67 2E 65 78 65}
        $a3 = {4D 5A}
        $a4 = {31 32 37 2E 30 2E 30 2E 32}
        $a5 = {4D D0 8B 46 08 33 C2 23 C7 C1 CF}
        $a6 = "AVE_MARIA"

    condition:
        all of them

}
```

This blog post was authored by Antonio Farina, Luca Mella, Antonio Pirozzi of Cybaze-Yoroi Z-LAB