

Paso 4: Reporte del Incidente

✓ Cumple la Estructura del Reporte

- Título del Reporte
- Introducción
- Descripción del Incidente
- Proceso de Reproducción
- Impacto del Incidente
- Recomendaciones
- Conclusión

SQL injection 1' OR '1'='1

Introduccion

Se estará realizando un sql injection para verificar la seguridad dentro de la máquina debian. Lo que se quiere lograr es comprender los conceptos de seguridad y entender como se reporta un incidente para futuras iteraciones.

Descripción del incidente

Se plantea usar el servicio de dvwa para emular vulnerabilidades dentro de la base de datos de mariaDB. Una vez configurado se plantea hacer un sql injection 1' OR '1'='1 para obtener las cuentas de una base de datos.

Pasos a seguir

1. cd /var/www/html/
2. sudo apt-get install wget unzip
3. sudo git clone https://github.com/digininja/DVWA.git /var/www/html/dvwa
4. sudo chmod -R 755 /var/www/html/dvwa
5. cd dvwa/config/
6. sudo mv config.inc.php.dist config.inc.php
7. sudo nano config.inc.php

CONFIGURAR DE LA SIGUIENTE MANERA

```
$_DVWA[ 'db_server' ] = getenv('DB_SERVER') ?: '127.0.0.1';
```

```
$_DVWA[ 'db_database' ] = getenv('DB_DATABASE') ?: 'dvwa';
```

```
$_DVWA[ 'db_user' ]    = getenv('DB_USER') ? 'dvwa';
$_DVWA[ 'db_password' ] = getenv('DB_PASSWORD') ? 'pass';
$_DVWA[ 'db_port' ]    = getenv('DB_PORT') ? '3306';
```

8. `sudo mysql -u root -p`
9. `CREATE DATABASE dvwa;`
10. `CREATE USER 'dvwa'@'127.0.0.1' IDENTIFIED BY 'pass';`
11. `GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwa'@'127.0.0.1';`
12. `exit;`
13. Abra un navegador en su máquina virtual y vaya a `http://127.0.0.1/dvwa/setup.php`

14. Revise la configuración y haga clic en «Crear/Restablecer base de datos».

15. En la Nueva Ventana `http://127.0.0.1/dvwa/login.php`

Username: admin

Password: password

16. Una vez logeado. Vaya a la pestaña «DVWA Security» y seleccione el nivel de seguridad «Bajo» para facilitar la explotación.

DVWA Security

Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.

Low Submit

Additional Tools

- [View Broken Access Control Logs](#) - View access logs for the Broken Access Control vulnerability

17. Se procede a la sección SQL injection

18. En la parte superior escribimos `1' OR '1'='1`

Vulnerability: SQL Injection

User ID:

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

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

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

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

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

Explicación

El payload `1' OR '1'='1` se usa cuando un atacante introduce esto en un campo como un 'User ID', la aplicación vulnerable lo "pega" en su código, transformando una consulta que buscaba un usuario (ej. `...WHERE UserID = '1'`) en una que busca al usuario 1 O cualquier fila donde `'1'='1'`. Dado que `'1'='1'` es una condición que siempre es verdadera, la base de datos devuelve todas las filas de la tabla, permitiendo al atacante ver toda la información o saltarse una pantalla de login.

Impacto

De esta manera el atacante puede tener acceso a múltiples cuentas en la base de datos. Lo cual es una brecha de seguridad enorme para las bases de datos y datos de los usuarios.

Se plantea contrarrestar con las siguientes instrucciones :

Para evitar este ataque específico, la defensa más efectiva es nunca construir consultas "pegando" la entrada del usuario directamente en el código SQL. En su lugar, se deben utilizar consultas parametrizadas (o prepared statements). Con este método, la consulta se envía a la base de datos como una plantilla (ej. `SELECT * FROM users WHERE UserID = ?`) y la entrada del usuario (`1' OR '1'='1`) se envía por separado como un parámetro. La base de datos trata esta entrada como un simple valor de texto, no como código ejecutable; intentará buscar a un usuario cuyo ID sea literalmente la cadena `"1' OR '1'='1"`, lo cual fallará y neutraliza el ataque por completo.