Prueba de Vulnerabilidad por Inyección SQL en DVWA

Introducción

Este reporte documenta un experimento de ciberseguridad realizado en un entorno controlado, utilizando una máquina virtual con Debian y la herramienta Damn Vulnerable Web Application (DVWA). El objetivo fue identificar una vulnerabilidad de tipo Inyección SQL (SQL Injection) y observar sus efectos dentro de una aplicación web intencionalmente insegura.

Damn Vulnerable Web Application (DVWA) es una aplicación web escrita en PHP y respaldada por MySQL creada intencionalmente para contener vulnerabilidades de seguridad clásicas. Su propósito es brindar un campo de prácticas seguro para estudiantes, desarrolladores y profesionales de ciberseguridad que deseen perfeccionar técnicas de ataque y defensa sin comprometer sistemas reales.

¿Qué es una Inyección SQL?

La Inyección SQL (SQL Injection) es una técnica de ataque que explota vulnerabilidades en la capa de acceso a bases de datos de una aplicación. Ocurre cuando los datos ingresados por un usuario no son debidamente validados o filtrados, permitiendo que un atacante inserte código SQL malicioso en una consulta.

Este tipo de ataque puede permitir desde el acceso no autorizado a información confidencial hasta la modificación o eliminación de datos, e incluso el control total de la base de datos en casos graves. La inyección SQL es una de las vulnerabilidades más comunes y peligrosas en aplicaciones web, especialmente cuando no se usan consultas preparadas ni mecanismos de sanitización de datos.

Descripción del Incidente

Durante la práctica, se ejecutó una inyección SQL simple en el módulo de autenticación de DVWA. El ataque consistió en introducir un payload en el campo de nombre de usuario que permitió eludir la autenticación sin credenciales válidas. Este tipo de vulnerabilidad representa un riesgo crítico en aplicaciones reales.

Proceso de Reproducción

1. **Verificar la configuración antes de iniciar:**
    - Verificar la correcta instalación de MySQL (MariaDB), Apache y PHP (LAMP Stack).
    - Establecer la contraseña del usuario root de MariaDB:

        sudo mysql -u root

        ALTER USER 'root'@'localhost' IDENTIFIED BY 'Password';

        FLUSH PRIVILEGES;

        EXIT;

2. **Configurar DVWA dentro de la máquina virtual Debian:**

    - Descargar y descomprimir el archivo DVWA.zip.
    - Copiar config.inc.php.dist a config.inc.php y editar las credenciales (root / Password).
    - Crear la base de datos dvwa:

```
debian@debian:/var/www/html/DVWA/config$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 5
Server version: 10.11.11-MariaDB-0+deb12u1 Debian 12

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

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

MariaDB [(none)]> CREATE DATABASE dvwa;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> EXIT;
```

    - Asignar permisos al directorio de DVWA:

      sudo chown -R www-data:www-data /var/www/html/DVWA/

      sudo chmod -R 755 /var/www/html/DVWA/

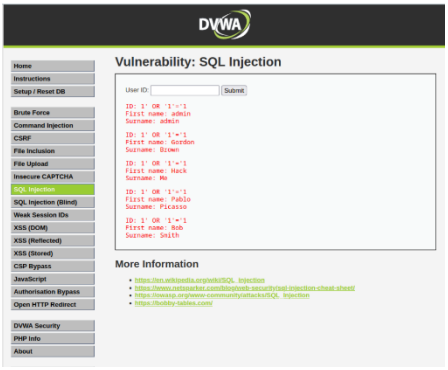
3. **Acceder a la interfaz web de DVWA** usando las credenciales predeterminadas:

    - *Usuario: admin*
    - *Contrasena: password*

4. **Ejecutar la inyección SQL** en el campo de usuario ingresando el PAYLOAD:

    1' OR '1'='1

5. **DVWA aceptó la autenticación** y mostró los registros de la base de datos, confirmando la vulnerabilidad.



Impacto del Incidente

La inyección SQL permitió eludir completamente la autenticación, comprometiendo la integridad y confidencialidad del sistema. En un entorno de producción, un atacante podría acceder, manipular o destruir información crítica, además de escalar privilegios dentro de la infraestructura.

Recomendaciones

- Implementar consultas preparadas (prepared statements) con parámetros.
- Validar y sanear todos los datos ingresados por el usuario.
- Evitar construir directamente consultas SQL con datos externos.
- Emplear un ORM (Object-Relational Mapping) para abstraer las consultas SQL.

Conclusión

Este ejercicio demuestra la importancia de mitigar vulnerabilidades como la inyección SQL. Aunque se realizó en un entorno seguro y controlado, el escenario resalta el peligro potencial en aplicaciones del mundo real si no se implementan medidas de seguridad adecuadas desde el desarrollo.