UNIVERSIDAD LATINO

Hacking Ético

ACTIVIDAD 3

Alumnos:
Cristian Jesús Castro Loría
Genesis Valeria Aguilar Ortegón
Esteban Almeida Alcocer

Profesor: Alfonso Segovia

Manual de Práctica

Explotación y Mitigación de SQL Injection en DVWA

1. Introducción

El presente manual documenta el desarrollo de una práctica de seguridad informática enfocada en la explotación de vulnerabilidades de **SQL Injection** utilizando el entorno **DVWA (Damn Vulnerable Web Application)** desplegado mediante **Docker**, con un atacante basado en **Kali Linux**.

El objetivo principal es **comprender el ciclo completo de un ataque**, desde la preparación del entorno hasta la explotación, y finalmente analizar el problema desde una **perspectiva Blue Team**, proponiendo medidas de mitigación reales.

2. Objetivo General

- Desplegar un entorno vulnerable utilizando Docker.
- Configurar DVWA con un nivel de seguridad bajo.
- Verificar conectividad entre atacante y víctima.
- Explotar una vulnerabilidad de SQL Injection mediante sqlmap.
- Analizar el ataque y proponer soluciones de mitigación.

3. Entorno de Trabajo

Herramientas utilizadas:

- Docker y Docker Compose
- DVWA (Damn Vulnerable Web Application)
- Kali Linux
- sqlmap
- Navegador web con herramientas de desarrollador (F12)

4. Evidencia 1 – Preparación del Entorno

Descripción:

En esta fase se levantó el entorno vulnerable utilizando Docker Compose. El comando ejecutado fue:

docker-compose up -d

Este comando permitió iniciar los contenedores definidos en el archivo docker-compose.yml, dejando los servicios ejecutándose en segundo plano.

Evidencia requerida:

- Captura de pantalla donde se observe el comando docker-compose up -d.
- Visualización de los contenedores en estado *running*.

Resultado esperado:

El entorno DVWA y el contenedor atacante se encuentran activos y listos para su uso.

5. Evidencia 2 – Configuración de DVWA

Descripción:

Una vez desplegado DVWA, se accedió a la aplicación web y se configuró el nivel de seguridad en **Low**, lo cual desactiva protecciones básicas y permite la explotación directa de vulnerabilidades.

Posteriormente, se utilizó la herramienta de desarrollador del navegador (F12) para inspeccionar las cookies de sesión.

Evidencia requerida:

- Captura de DVWA con el nivel de seguridad configurado en "Low".
- Captura de la herramienta de desarrollador mostrando la cookie PHPSESSID.

Importancia:

La cookie PHPSESSID identifica la sesión activa del usuario y puede ser utilizada por herramientas automáticas para explotar la aplicación.

6. Evidencia 3 – Conexión entre Atacante y Víctima

Descripción:

Desde el contenedor Kali Linux se verificó la conectividad hacia la máquina víctima utilizando el comando ping.

Comando ejecutado:

ping <IP_víctima>

Evidencia requerida:

- Captura de la terminal mostrando respuestas exitosas al ping.

Resultado esperado:

La comunicación entre ambos contenedores es correcta, confirmando que el atacante puede interactuar con la víctima.

7. Evidencia 4 – Explotación (SQL Injection)

Descripción:

En esta fase se realizó la explotación de la vulnerabilidad de SQL Injection presente en DVWA utilizando la herramienta sqlmap desde Kali Linux.

sqlmap automatiza la detección y explotación de inyecciones SQL, permitiendo extraer información sensible de la base de datos.

Ejemplo de comando utilizado:

sqlmap -u "http://<IP_víctima>/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" \

--cookie="PHPSESSID=XXXX; security=low" --dump

Evidencia requerida:

- Captura de la terminal de Kali mostrando el resultado final de sqlmap.
- Visualización de tablas, usuarios o datos volcados (*dump*).

Resultado obtenido:

La herramienta logró extraer información sensible de la base de datos, demostrando que la aplicación es vulnerable a SQL Injection.

8. Reto Final – Propuesta de Solución

Mitigación de Vulnerabilidades

Desde una **perspectiva Blue Team**, el ataque ocurrió principalmente debido a:

- Falta de validación y sanitización de las entradas del usuario.
- Uso de consultas SQL construidas dinámicamente mediante concatenación de cadenas.
- Ausencia de mecanismos de protección a nivel de base de datos y aplicación.

Cuando una aplicación inserta directamente los valores introducidos por el usuario dentro de una consulta SQL, un atacante puede modificar la lógica de la consulta y ejecutar comandos no autorizados.

Implementación de Prepared Statements

Una de las medidas más efectivas para evitar este tipo de ataques es el uso de **Prepared Statements (consultas preparadas)**.

¿Por qué funcionan?

- Separan la lógica de la consulta de los datos introducidos por el usuario.
- El motor de base de datos interpreta primero la estructura SQL y después los valores.
- Evitan que el código malicioso sea ejecutado como parte de la consulta.

Ejemplo conceptual:

En lugar de construir consultas así:

SELECT * FROM users WHERE id = '" + userInput + "';

Se utilizan consultas preparadas:

SELECT * FROM users WHERE id = ?;

El valor del parámetro se envía de forma segura, impidiendo la inyección de código SQL.

Medidas Adicionales de Mitigación

- Validar y sanitizar todas las entradas del usuario.
- Usar niveles de seguridad adecuados en producción.
- Limitar los privilegios del usuario de la base de datos.

- Implementar monitoreo y registros de actividad sospechosa.
- Realizar pruebas de seguridad periódicas.

9. Conclusión

Esta práctica permitió comprender cómo una mala implementación de consultas SQL puede comprometer completamente un sistema. Asimismo, se evidenció la importancia de adoptar buenas prácticas de desarrollo seguro, como el uso de **Prepared Statements**, para prevenir ataques y proteger la información crítica de una organización.