UNIVERSIDAD AUTÓNOMA "TOMAS FRÍAS"

FACULTAD DE CIENCIAS PURAS

CARRERA DE INGENIERIA INFORMATICA

PRACTICA Nº 2

Universitario (a): Natalia Coro Vasquez

Carrera: Ingeniería Informática

Docente: M. Sc. Huáscar Fedor Gonzales Guzmán

Auxiliar: Univ. Job Molina

Materia: Seguridad de Software

ANÁLISIS DE VULNERABILIDADES EN APLICACIONES WEB

Objetivo:

Comprender y analizar las vulnerabilidades más comunes en aplicaciones web, usando el estándar OWASP Top Ten, para proponer soluciones que prevengan estos riesgos y fomenten el desarrollo seguro.

Instrucciones:

Lectura y Análisis: Lee el material proporcionado sobre las vulnerabilidades comunes en aplicaciones web, prestando especial atención a las siguientes categorías del OWASP Top Ten:

- **Inyección (SQL, NoSQL, OS, LDAP**): Consiste en introducir código malicioso a través de entradas no seguras para ejecutar comandos en bases de datos o sistemas operativos. Por ejemplo, las inyecciones SQL permiten acceder o manipular datos.
- **Cross-Site Scripting (XSS):** Permite a un atacante inyectar scripts maliciosos que se ejecutan en el navegador de la víctima.
- **Cross-Site Request Forgery (CSRF):** Forza a usuarios autenticados a realizar acciones no deseadas.
- **Broken Authentication:** Fallos en la autenticación que permiten acceder a cuentas.
- **Insecure Direct Object References (IDOR**): Exposición de identificadores sensibles que permiten acceder a información de otros usuarios.
- **Security Misconfiguration:** Configuraciones inseguras que permiten a los atacantes explotar brechas de seguridad.
- **Sensitive Data Exposure:** Exposición de datos críticos debido a la falta de cifrado o controles de acceso.

Investigación de Casos Reales: Selecciona dos vulnerabilidades del listado y busca casos documentados de aplicaciones o sitios web populares que hayan sido afectados por estas vulnerabilidades. Describe brevemente:

- La aplicación o sitio web afectado.

- La vulnerabilidad explotada.
- Las consecuencias del ataque.
- Las medidas correctivas que se implementaron posteriormente.

Caso 1: Inyección SQL en LinkedIn (2012)

- **Aplicación afectada:** LinkedIn
  - **Vulnerabilidad explotada:** Inyección SQL

❖ **Consecuencias del ataque:** Los atacantes explotaron una vulnerabilidad de inyección SQL en el sistema de autenticación de LinkedIn, obteniendo acceso a una base de datos que contenía contraseñas de millones de usuarios. Esto provocó la filtración de más de 6.5 millones de contraseñas, exponiendo la privacidad y seguridad de las cuentas afectadas. Muchas de las contraseñas no estaban debidamente cifradas, lo que facilitó su exposición.

❖ **Medidas correctivas:** LinkedIn implementó consultas parametrizadas para evitar la inyección de SQL y fortaleció el cifrado de contraseñas mediante el uso de algoritmos de hashing robustos, como bcrypt. Además, mejoró la infraestructura de seguridad para prevenir futuros ataques y estableció controles de acceso más estrictos en sus bases de datos.

Caso 2: Cross-Site Scripting (XSS) en Twitter (2020)

- **Aplicación afectada:** Twitter
  - **Vulnerabilidad explotada:** Cross-Site Scripting (XSS)

❖ **Consecuencias del ataque:** Una vulnerabilidad XSS permitía a los atacantes inyectar scripts maliciosos en los perfiles de los usuarios. Este tipo de ataque tenía como objetivo realizar acciones como el robo de sesiones y la redirección de usuarios a sitios de phishing. La vulnerabilidad afectó a numerosos usuarios, lo que generó preocupación por la seguridad de las cuentas.

❖ **Medidas correctivas:** Twitter aplicó una política de Content Security Policy (CSP) para restringir la ejecución de scripts no autorizados y adoptó un filtrado más estricto de las entradas de usuario para evitar que los scripts se reflejaran en las páginas.

También reforzó su sistema de validación y sanitización de datos para proteger contra futuros ataques de XSS.

Análisis Crítico:

- Explica por qué crees que estas vulnerabilidades siguen siendo comunes en las aplicaciones web.
- Proporciona un análisis de los métodos de prevención que podrías implementar en el desarrollo de una aplicación web para evitar cada una de las dos vulnerabilidades que seleccionaste en el punto anterior.

❖ R**azón de persistencia:** Estas vulnerabilidades persisten debido a la falta de actualización en prácticas de desarrollo seguro y la complejidad de la protección en aplicaciones modernas.

❖ **Prevención en desarrollo:**

- ➢ **Inyección SQL:** Utilizar sentencias preparadas, validación de entradas y separación de datos y código en consultas.
- ➢ **XSS:** Implementar políticas de seguridad de contenido (CSP) y sanitizar todas las entradas de usuario.