**Resolución de la Práctica 2**

La tarea requiere el análisis de dos vulnerabilidades del OWASP Top Ten y la elaboración de un informe basado en estas.

**1. Selección de Vulnerabilidades**

Del estándar **OWASP Top Ten**, seleccionaremos:

* **Inyección SQL (SQL Injection)**.
* **Cross-Site Scripting (XSS)**.

**2. Investigación de Casos Reales**

**Caso 1: Inyección SQL**

* **Aplicación o sitio afectado**: Yahoo (2014).
* **Vulnerabilidad explotada**: Una inyección SQL en sus bases de datos permitió a los atacantes extraer información confidencial de más de 450,000 cuentas.
* **Consecuencias del ataque**:
  + Exposición de datos de usuario como contraseñas en texto claro.
  + Daño reputacional y pérdida de confianza por parte de los usuarios.
* **Medidas correctivas**:
  + Implementación de consultas preparadas con parámetros.
  + Auditoría de seguridad en las bases de datos y el código fuente.
  + Validación estricta de entradas del usuario.

**Caso 2: Cross-Site Scripting (XSS)**

* **Aplicación o sitio afectado**: Twitter (2010).
* **Vulnerabilidad explotada**: Un ataque XSS almacenado permitió la inyección de scripts maliciosos en tweets. Cuando otros usuarios visualizaban el tweet afectado, el script se ejecutaba en su navegador.
* **Consecuencias del ataque**:
  + Compromiso de cuentas de usuario.
  + Redirección a sitios de phishing y propagación del ataque.
* **Medidas correctivas**:
  + Implementación de un sistema de validación y sanitización de entradas.
  + Uso de políticas de seguridad de contenido (CSP).
  + Corrección del código afectado para evitar el uso de innerHTML.

**3. Análisis Crítico**

**Razones por las que siguen siendo comunes:**

1. **Inyección SQL**:
   * Falta de concienciación sobre la necesidad de utilizar consultas preparadas.
   * Reutilización de código inseguro por desarrolladores.
   * Falta de auditorías de seguridad regulares en el software.
2. **XSS**:
   * Dependencia excesiva de los datos proporcionados por los usuarios sin validación.
   * Falta de implementación de mecanismos de protección como CSP.
   * Complejidad en el manejo del DOM del lado cliente.

**4. Métodos de Prevención**

**Para Inyección SQL:**

* Usar consultas preparadas y sentencias parametrizadas:
* $stmt = $conn->prepare("SELECT \* FROM users WHERE username = ?");
* $stmt->bind\_param("s", $username);
* $stmt->execute();
* Implementar validaciones estrictas en los campos de entrada:
  + Rechazar caracteres especiales (', ;, etc.).
* Limitar los privilegios del usuario de base de datos:
  + Solo acceso de lectura/escritura necesario.

**Para XSS:**

* Validar y sanitizar todas las entradas del usuario:
  + Utilizar librerías como DOMPurify en el frontend.
* Implementar políticas de contenido:
* Content-Security-Policy: script-src 'self';
* Evitar el uso de innerHTML y funciones similares que ejecuten código dinámico.