**Práctica 3: Implementación de Prácticas de Seguridad en el Desarrollo de Aplicaciones Web**

**1. Validación y Sanitización de Entradas**

**Descripción de los pasos para validar y sanitizar entradas:**

1. **Validación**:
   * Verificar que los datos cumplen con el formato esperado (ejemplo: correos electrónicos, números, etc.).
   * Utilizar expresiones regulares para comprobar patrones específicos.
   * Rechazar datos que no cumplan los criterios establecidos.
2. **Sanitización**:
   * Eliminar caracteres peligrosos que puedan ser utilizados en ataques XSS o inyecciones.
   * Utilizar funciones integradas del lenguaje como htmlspecialchars o librerías para manejar contenido HTML.

**Ejemplo en PHP: Validación de correos y sanitización de texto:**

<?php

// Validación de correo electrónico

function validarEmail($email) {

if (filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

echo "Correo válido: " . htmlspecialchars($email) . "<br>";

} else {

echo "Correo no válido.<br>";

}

}

// Sanitización de texto para prevenir XSS

function sanitizarTexto($texto) {

return htmlspecialchars($texto, ENT\_QUOTES, 'UTF-8');

}

// Ejemplo

$email = "usuario@dominio.com";

$texto = "<script>alert('XSS');</script>";

validarEmail($email);

echo "Texto sanitizado: " . sanitizarTexto($texto);

?>

**2. Control de Acceso y Autenticación Segura**

**Beneficios de la MFA y autenticación basada en tokens (JWT):**

* **MFA**:
  + Incrementa la seguridad combinando múltiples factores (contraseña + código enviado al móvil).
  + Previene el acceso incluso si una contraseña es comprometida.
* **JWT**:
  + Es un estándar ligero para transmitir datos entre el cliente y el servidor.
  + Facilita la autenticación sin necesidad de almacenar sesiones en el servidor.

**Ejemplo en PHP: Creación de un JWT:**

<?php

require 'vendor/autoload.php';

use Firebase\JWT\JWT;

// Llave secreta

$llave = "clave\_secreta";

// Datos del usuario

$datos = [

'id' => 1,

'email' => 'usuario@dominio.com',

'rol' => 'admin'

];

// Generar JWT

$jwt = JWT::encode($datos, $llave, 'HS256');

echo "JWT generado: " . $jwt;

?>

**3. Gestión de Sesiones y Cookies**

**Atributos de seguridad de cookies:**

* **HttpOnly**: Previene que las cookies sean accesibles desde JavaScript, protegiendo contra ataques XSS.
* **Secure**: Asegura que las cookies solo se envíen a través de conexiones HTTPS.

**Configuración segura de sesiones en PHP:**

<?php

// Configuración segura de cookies

session\_set\_cookie\_params([

'lifetime' => 3600, // Tiempo de vida de la cookie

'path' => '/',

'domain' => '',

'secure' => true, // Solo en conexiones HTTPS

'httponly' => true, // No accesible desde JavaScript

'samesite' => 'Strict' // Previene CSRF

]);

session\_start();

$\_SESSION['usuario'] = 'usuario@dominio.com';

echo "Sesión configurada de forma segura.";

?>

**4. Protección de Datos Sensibles**

**Diferencia entre hashing y cifrado:**

* **Hashing**:
  + Convierte datos en un valor irreversiblemente único (hash).
  + Uso: Contraseñas, firmas digitales.
* **Cifrado**:
  + Transforma datos para hacerlos ilegibles, pero permite revertirlos con una clave.
  + Uso: Transmisión de datos confidenciales.

**Ejemplo en PHP: Uso de password\_hash y password\_verify:**

<?php

// Crear hash de contraseña

$contraseña = "mi\_contraseña\_secreta";

$hash = password\_hash($contraseña, PASSWORD\_BCRYPT);

echo "Hash de contraseña: " . $hash . "<br>";

// Verificar contraseña

if (password\_verify($contraseña, $hash)) {

echo "Contraseña válida.";

} else {

echo "Contraseña no válida.";

}

?>