Implementación de Laboratorio de Seguridad Web en Entorno Local (XAMPP)

Objetivo del laboratorio

Desplegar un entorno local en Windows con un servidor web Apache + MySQL (XAMPP) para ejecutar un laboratorio funcional que permita simular, mitigar y validar ataques de tipo **SQL Injection (SQLi)**, **Cross-Site Scripting (XSS)** y **Cross-Site Request Forgery (CSRF)** a través de un formulario web en PHP conectado a base de datos.

Herramientas utilizadas

- **XAMPP para Windows v3.3.0** (incluye Apache, MySQL, PHP)
- **phpMyAdmin** (incluido en XAMPP)
- **Navegador web moderno (Chrome)**
- **Editor de código** (Visual Studio Code o Bloc de notas)
- **Terminal integrada de XAMPP (Shell)**

Paso a paso de implementación

1. Descarga e instalación de XAMPP

- Se descargó XAMPP desde el sitio oficial: https://www.apachefriends.org/es/index.html

Se instaló normalmente en la ruta:

C:\xampp\

-
- Desde el **XAMPP Control Panel**, se activaron los servicios:
    - Apache
    - MySQL

2. Subida del laboratorio al entorno local

Se descargó y descomprimió el paquete `seguridad_lab.zip`, que contenía los siguientes archivos:

```
seguridad_lab/
├── index.php
├── comentarios.php
├── conexion.php
├── init.php
└── README.txt
```

Esta carpeta fue copiada en el directorio raíz del servidor Apache de XAMPP:

```
C:\xampp\htdocs\seguridad_lab\
```

3. Configuración de la base de datos

- Se accedió a `phpMyAdmin` a través de: http://localhost/phpmyadmin
- Se creó la base de datos y tabla necesarias para el laboratorio:

```
CREATE DATABASE seguridad_lab;

USE seguridad_lab;


CREATE TABLE comentarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100),
  comentario TEXT
);
```

4. Configuración del archivo `conexion.php`

Se creó/modificó el archivo `conexion.php` para conectarse correctamente a la base de datos con los parámetros utilizados por XAMPP:

```
<?php

$conexion = new mysqli("127.0.0.1", "root", "", "seguridad_lab", 3307);


if ($conexion->connect_error) {

    die("Conexión fallida: " . $conexion->connect_error);

}

?>
```

> Se utilizó el **host** `127.0.0.1` en lugar de `localhost` para evitar conflictos de socket y se especificó el puerto **`3307`**, ya que el puerto `3306` se encontraba ocupado.

5. Configuración de `phpMyAdmin`

Se editó el archivo de configuración ubicado en:

C:\xampp\phpMyAdmin\config.inc.php

Y se realizaron los siguientes cambios:

```
$cfg['Servers'][$i]['auth_type'] = 'config';

$cfg['Servers'][$i]['user'] = 'root';

$cfg['Servers'][$i]['password'] = '';

$cfg['Servers'][$i]['AllowNoPassword'] = true;

$cfg['Servers'][$i]['host'] = '127.0.0.1';

$cfg['Servers'][$i]['port'] = '3307';
```

// Se comentaron las siguientes líneas para evitar errores por usuarios no existentes:

// $cfg['Servers'][$i]['controluser'] = 'pma';

// $cfg['Servers'][$i]['controlpass'] = '';

6. Ejecución del laboratorio

- Se accedió al laboratorio desde:
  http://localhost/seguridad_lab/index.php

- Se utilizó un formulario en HTML para enviar datos (`nombre` y `comentario`) a través de `POST`, el cual:

    - **Escapaba caracteres** usando `htmlspecialchars()` para evitar XSS.

    - **Utilizaba `prepared statements`** para prevenir SQLi.

    - **Incluía un token CSRF** generado y validado desde `init.php`.

7. Validaciones de seguridad implementadas

Inyección SQL (SQLi) mitigada con:

$stmt = $conexion->prepare("INSERT INTO comentarios (nombre, comentario) VALUES (?, ?)");

$stmt->bind_param("ss", $nombre, $comentario);

XSS mitigado con:

htmlspecialchars($_POST['comentario'], ENT_QUOTES, 'UTF-8');

CSRF mitigado con token generado y validado:

init.php:

```
session_start();
if (!isset($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
```

HTML:

```
<input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
```

PHP (comentarios.php):

```
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die("Token CSRF inválido.");
}
```

Archivos originales modificados

Archivo	Modificación realizada
`conexion.php`	Se configuró conexión con `127.0.0.1`, usuario `root`, sin contraseña, puerto `3307`.
`config.inc.php` (`phpMyAdmin`)	Se desactivó `controluser`, se forzó conexión por TCP a puerto 3307.
`init.php`	Se generó el token CSRF.
`comentarios.php`	Se implementaron validaciones contra XSS, SQLi y CSRF.

`index.php`	Se construyó el formulario HTML con medidas de seguridad.

Conclusión

Este laboratorio representa un entorno realista y controlado para el estudio de vulnerabilidades web comunes. La implementación demuestra el ciclo completo de configuración de un entorno PHP + MySQL, así como la aplicación efectiva de **buenas prácticas de desarrollo seguro**.