

eman ta zabal zazu



Universidad
del País Vasco

Euskal Herriko
Unibertsitatea

Sistemas de Gestión de Seguridad de Sistemas de Información

Ingeniería Informática de Gestión y Sistemas de
Información

Sistema Web

Autores:

Xabier Gabiña
Ainhize Martinez
Marcos Martín

17 de noviembre de 2023

Índice general

1. Introduccion	3
2. Primera auditoria	4
2.1. ZAP	4
2.2. sqlmap	5
2.3. nmap	6
2.4. Metaexploit	7
3. Vulnerabilidades	8
3.1. Rotura de control de acceso	8
3.1.1. Control de acceso	8
3.2. Fallos criptográficos	12
3.2.1. Cifrado de extremo a extremo	12
3.2.2. Almacenamiento de contraseñas	14
3.2.3. Configuración errónea de las Cookies	16
3.3. Inyección	17
3.3.1. SQL Injection	17
3.3.2. Cross-Site Scripting (XSS)	18
3.3.3. Cross-Site Request Forgery	19
3.4. Diseño inseguro	20
3.4.1. Reutilización de códigos seguros	20
3.5. Configuración de seguridad insuficiente	22
3.5.1. Despliegue seguro	22
3.5.2. Cabecera CSP	24
3.5.3. Cabecera Cache-Control	25
3.5.4. Cabecera HSTS	26
3.5.5. Cabecera X-XSS-Protection	27
3.5.6. Cabecera X-Content-Type-Options	28
3.5.7. Cabecera X-Frame-Options	29
3.5.8. Cabecera X-Permitted-Cross-Domain-Policies	30
3.5.9. Cabecera X-DNS-Prefetch-Control	31
3.5.10. Cabecera X-Download-Options	32
3.5.11. Cabecera Referrer-Policy	33
3.5.12. Cabecera Feature-Policy	34

3.5.13.	Cabecera Expect-CT	35
3.5.14.	Configuracion PHP	36
3.5.15.	Configuracion Apache	37
3.6.	Componentes vulnerables y obsoletos	38
3.6.1.	Control de versiones de los componentes	38
3.7.	Fallos de identificación y autenticación	39
3.7.1.	Captcha	39
3.7.2.	Contraseñas debiles o por defecto	40
3.7.3.	Invalidacion de sesiones	41
3.8.	Fallos en la integridad de datos y software	42
3.8.1.	Copias de seguridad	42
3.8.2.	CI/CD	43
3.9.	Fallos en la monitorizacion de la seguridad	44
3.9.1.	Auditorias de seguridad	44
3.9.2.	Configuracion de logs del sistema	45
4.	Segunda auditoria	46
4.1.	ZAP	46
4.2.	sqlmap	47
4.3.	nmap	48
4.4.	Metaexploit	49
5.	Conclusiones	50
6.	Bibliografia	51

Introduccion

La finalidad de esta etapa del proyecto consiste en garantizar la seguridad y robustez de la página web desarrollada en la fase inicial del proyecto. Con el propósito de lograr este objetivo, se recurrirá a la aplicación de diversas herramientas y técnicas que han sido abordadas y exploradas a lo largo del curso. El enfoque principal de esta fase se centra en identificar y abordar posibles vulnerabilidades y debilidades de seguridad en el sistema, a fin de implementar medidas correctivas y fortalecer la integridad de la plataforma web.

Este proceso implica la utilización de herramientas especializadas, como ZAP, SQLMap, Nmap y Metasploit, que permiten evaluar y poner a prueba la infraestructura de seguridad de la página web. A través de su empleo, se llevará a cabo un exhaustivo análisis que permitirá descubrir posibles fallos o vulnerabilidades que puedan ser aprovechadas por actores maliciosos, poniendo en riesgo la integridad de los datos y la privacidad de los usuarios.

El objetivo último es la identificación, documentación y solución de posibles vulnerabilidades, lo que contribuirá a elevar la seguridad y confiabilidad de la página web. A medida que se avance en este proceso, se aplicarán estrategias y medidas correctivas para abordar los hallazgos identificados y se establecerán salvaguardias que minimicen el riesgo de futuros incidentes de seguridad.

Este proyecto representa un ejercicio académico valioso que no solo demuestra la comprensión de los conceptos de seguridad informática, sino que también ofrece la oportunidad de aplicar de manera práctica las habilidades adquiridas en la detección y mitigación de riesgos en entornos web. A través de esta experiencia, se espera adquirir un conocimiento más profundo y aplicable en el campo de la seguridad informática y estén mejor preparados para enfrentar desafíos en el mundo real.

Primera auditoria

La idea de esta auditoria es la de encontrar los fallos de seguridad que tiene nuestro sistema web para en el posterior capitulo de este documento comentarlos y solucionarlos.

2.1. ZAP

Para empezar con la primera auditoria ejecutaremos el proxy ZAP como se nos ha propuesto en clase. Al ejecutarla en nuestra pagina web nos encontramos con el siguiente listado de errores:

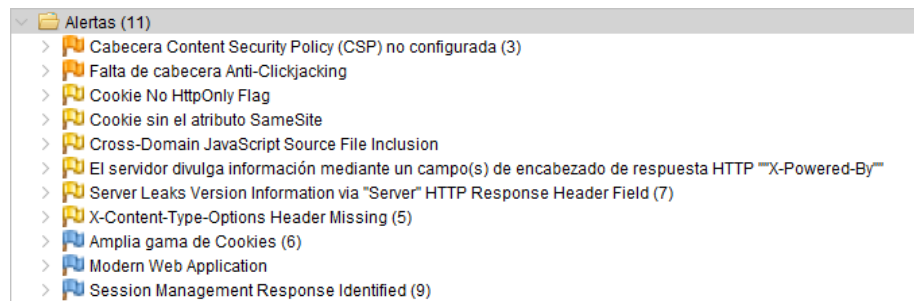


Figura 2.1: Listado de errores de la primera auditoria

Como podemos ver en la imagen, tenemos un total de 11 errores, los cuales iremos comentando uno a uno en los siguientes apartados y solucionando.

2.2. sqlmap

2.3. nmap

2.4. Metaexploit

Vulnerabilidades

3.1. Rotura de control de acceso

La rotura de control de acceso es una vulnerabilidad que permite a un atacante acceder a recursos restringidos o privilegiados, ya sea por un error en la implementación de la autenticación y autorización o por un error en la lógica de control de acceso.

3.1.1. Control de acceso

Descripción

En nuestro sistema, un usuario puede modificar sus datos personales, pero también puede modificar los datos de otros usuarios. Esto es un fallo de rotura de control de acceso ya que un usuario no debería poder modificar los datos de otro usuario.

PoC

Pongamos en el ejemplo que tenemos dos usuarios, Admin y Xabier con sus respectivas IDs

id	nombre
1	Xabier
10	Admin

Figura 3.1: Datos de Usuarios

Si desde inspeccionar elementos hacemos click sobre el boton 'Perfil' este nos mostrara el link el cual se ve asi:

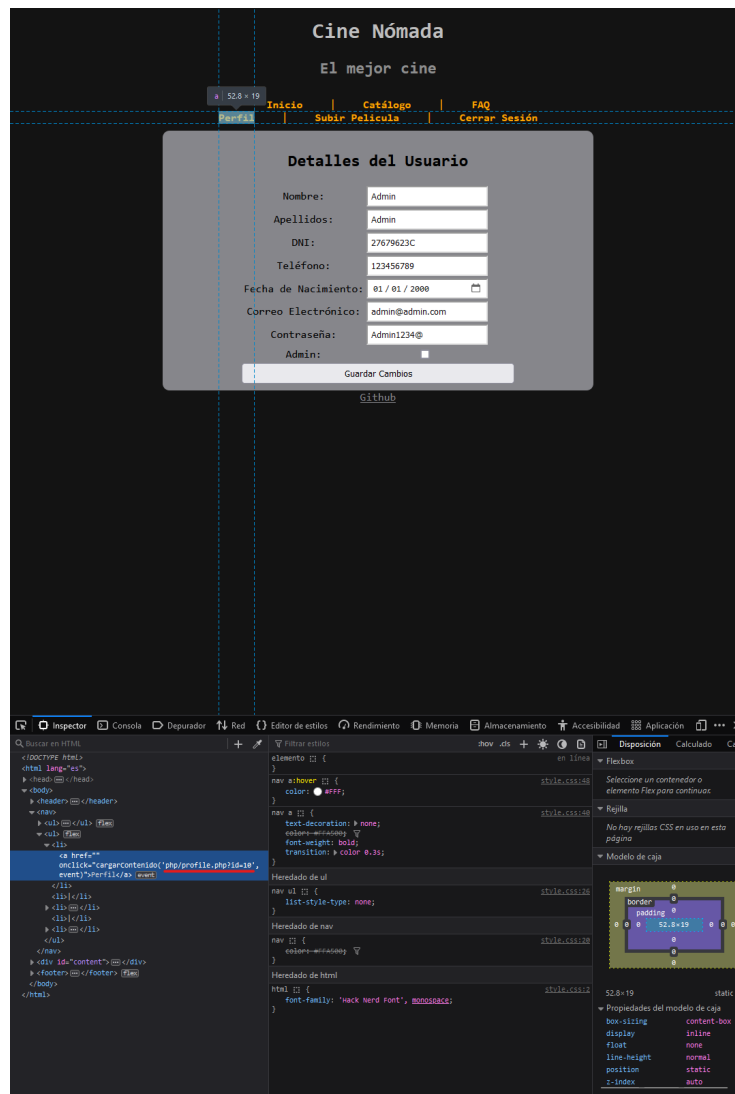


Figura 3.2: Perfil de Admin

Si alteramos el valor de `?id=X` por en este caso la id de Xabier (La ID 1) podemos acceder a sus datos

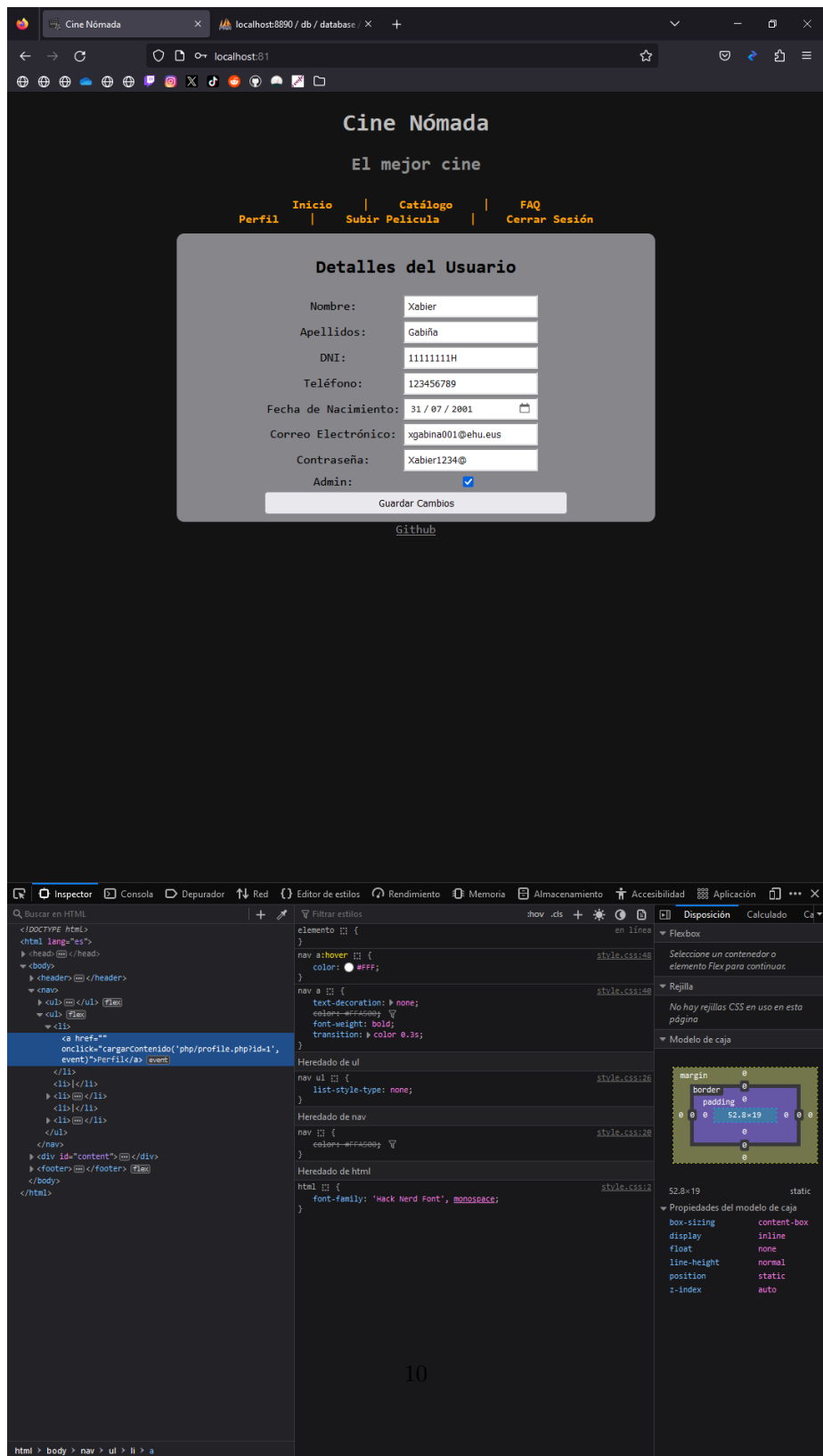


Figura 3.3: Perfil de Xabier

Solución

Para solucionar este problema, hemos añadido una comprobación en el código que comprueba que el usuario que está intentando modificar los datos es el mismo que el usuario que está logueado en el sistema.

```
// Verificar si se recibió un ID válido a través de la URL
if (isset($_GET['id']) && is_numeric($_GET['id']))
{
    // Verificar si el usuario es el mismo que el de la sesión
    if ((int)$_GET['id'] == (int)$_SESSION['user_id'])
    {
        $userId = $_SESSION['user_id'];
    }
}
```

Figura 3.4: Comprobación de usuario

Esta misma error tambien ha sido corregido en el catalogo.

3.2. Fallos criptográficos

Los "fallos criptográficos" se refieren a debilidades o errores en el diseño, implementación o uso de algoritmos criptográficos que comprometen la seguridad de la información protegida mediante técnicas de cifrado y protección.

3.2.1. Cifrado de extremo a extremo

Descripción

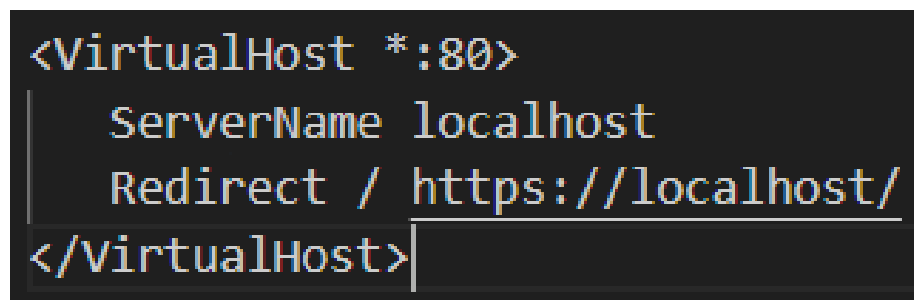
El cifrado de extremo a extremo es un método de seguridad informática en el que la información se cifra en el punto de origen y solo se descifra en el punto de destino. Esto significa que los datos están protegidos durante su transmisión y solo son legibles para la parte destinataria que posee la clave de descifrado correspondiente.

Solución

Para solucionar este problema configuraremos nuestro servidor para que cifre y redirija todo el tráfico a HTTPS.

Creamos un certificado SSL autofirmado dentro del Dockerfile.

Creamos una regla para redirigir el tráfico entrante del puerto 80 al puerto 443.



```
<VirtualHost *:80>
    ServerName localhost
    Redirect / https://localhost/
</VirtualHost>
```

Figura 3.5: Redirección de tráfico

También debemos decir al puerto 443 que use el certificado SSL que hemos creado.

```
<VirtualHost *:443>
    ServerName localhost
    DocumentRoot /var/www/html

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt
    SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key
</VirtualHost>
```

Figura 3.6: Certificado SSL

3.2.2. Almacenamiento de contraseñas

Descripción

En nuestro sistema, no se almacenan las contraseñas de los usuarios de forma segura, lo que permite que un atacante pueda obtener las contraseñas de los usuarios.

PoC

Si un atacante consigue acceso a la base de datos, puede obtener las contraseñas de los usuarios en texto plano.

email	passwd
xabierland@gmail.com	Xabier1234@
admin@admin.com	Admin1234@

Figura 3.7: Contraseñas en texto plano

Solución

Para solucionar este problema de la mejor forma posible debemos tener tres puntos bien definidos:

1. Configurar el factor de costo apropiado
 - El factor de costo (work factor) en bcrypt determina el número de iteraciones utilizadas en el cálculo del hash. Un valor mayor implica una contraseña más segura, pero también requiere más tiempo para calcular el hash. Un valor razonable es 12 o más, dependiendo del hardware y las necesidades de rendimiento.
2. Usar un algoritmo de cifrado seguro.
 - En nuestro caso usaremos BCrypt. CRYPT_BLOWFISH se usa para crear el hash. Producirá un hash estándar compatible con crypt() utilizando el identificador "\$2y\$". El resultado siempre será un string de 60 caracteres, o false en caso de error.
3. Generar un semilla aleatoria para cada usuario.

- BCrypt ya gestiona las semillas de forma automática y en el manual de PHP no recomiendan su uso de otra manera. Más información en <https://www.php.net/manual/es/function.password-hash.php>

```
// Hashear la contraseña  
$options=['cost'=>12,];  
$hashedPassword = password_hash($passwd, PASSWORD_BCRYPT, $options);
```

Figura 3.8: Contraseñas encriptadas

3.2.3. Configuración errónea de las Cookies

Descripción

La configuración errónea de las cookies se refiere a la práctica de establecer parámetros o atributos de las cookies de manera inadecuada, lo que puede tener consecuencias negativas en términos de seguridad, privacidad y funcionalidad en una aplicación web.

Solución

Para solucionar este problema, hemos añadido una configuración segura a las cookies de nuestro sitio web.

```
<?php
session_start([
    'cookie_lifetime' => 0,           // La sesión expira cuando se cierra el navegador.
    'cookie_path' => '/',            // Disponible en todo el dominio.
    'cookie_secure' => true,         // Solo se envía la cookie sobre conexiones HTTPS.
    'cookie_httponly' => true,       // La cookie solo es accesible a través de HTTP.
    'cookie_samesite' => 'Lax',      // Define la política de SameSite (puede ser 'Lax' o 'Strict').
]);
?>
```

Figura 3.9: Configuración de las cookies

3.3. Inyección

En el ámbito de la seguridad informática y el desarrollo de software, el término “inyección” se refiere a una clase de vulnerabilidades que permite a un atacante introducir y ejecutar código malicioso en una aplicación o sistema.

3.3.1. SQL Injection

Descripción

La inyección SQL es una vulnerabilidad de seguridad informática que permite a un atacante modificar las consultas SQL que se ejecutan en una base de datos subyacente. Esto puede permitir a un atacante obtener información confidencial, alterar o eliminar datos, o incluso comprometer completamente el sistema.

PoC

Solución

Para solucionar este problema, hemos modificado el código que procesa las consultas SQL para que no se puedan inyectar consultas SQL.

```
// SQL seguro utilizando consultas preparadas
$sql = "INSERT INTO usuarios (nombre, apellidos, passwd, dni, fechaN, email, telefono) VALUES (?, ?, ?, ?, ?, ?, ?)";

if ($stmt = $conn->prepare($sql)) {
    // Vincula las variables a los marcadores de posición
    $stmt->bind_param("sssssss", $nombre, $apellidos, $hashedPassword, $dni, $fechaNacimiento, $email, $telefono);

    // Ejecuta la consulta preparada
    if ($stmt->execute()) {
        echo "Registrado con éxito";
    } else {
        echo "Error al ejecutar la consulta: " . $stmt->error;
    }
} else {
    echo "Error al preparar la consulta: " . $conn->error;
}
```

Figura 3.10: Parametrizar consulta SQL

3.3.2. Cross-Site Scripting (XSS)

Descripción

El Cross-Site Scripting (XSS) es una vulnerabilidad de seguridad informática que permite a un atacante inyectar código malicioso (por lo general JavaScript) en páginas web que son vistas por otros usuarios. El XSS permite a los atacantes ejecutar scripts en el navegador de un usuario, lo que puede secuestrar sesiones de usuario, desfigurar sitios web, robar información confidencial y distribuir malware.

Solución

La solución a este problema se verá más adelante en la sección 3.5.5

3.3.3. Cross-Site Request Forgery

Descripción

El Cross-Site Request Forgery (CSRF) es una vulnerabilidad de seguridad informática que permite a un atacante forzar a un usuario autenticado a ejecutar acciones no deseadas en una aplicación web en la que confía. El CSRF puede utilizarse para realizar acciones como transferencias de fondos, cambios de contraseñas, compras, etc.

Solución

Para solucionar este problema, hemos añadido un token CSRF a nuestro sitio web.

```
// Generar token CSRF si no existe
if (!isset($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
```

Figura 3.11: Token CSRF

```
// Verificar el token CSRF
if (isset($_POST['csrf_token']) && $_POST['csrf_token'] === $_SESSION['csrf_token']) {
    // Token CSRF válida, continúa con el registro
}
```

Figura 3.12: Comprobacion de token CSRF

3.4. Diseño inseguro

El "diseño inseguro" se refiere a la presencia de debilidades fundamentales en el diseño de una aplicación web que pueden ser explotadas por atacantes para comprometer la seguridad.

3.4.1. Reutilización de códigos seguros

Descripción

La reutilización de código seguro se refiere a la práctica de aprovechar componentes de software previamente probados y seguros en lugar de escribir código desde cero. Esto no solo puede acelerar el desarrollo de software, sino que también puede reducir el riesgo de introducir vulnerabilidades de seguridad.

Solución

En nuestro proyecto se han implementado estas prácticas mediante la reutilización del archivo `validar.js` en el que se encuentran las funciones de validación de todos los formularios de nuestro sitio web.

```
// validaciones.js
function validarNumeroTelefono(telefono) {
    // Expresión regular para validar números de teléfono en formato de 9 dígitos
    var regex = /^[0-9]{9}$/;

    // Utiliza test() para verificar si el número cumple con la expresión regular
    return regex.test(telefono);
}

function validarCorreoElectronico(correo) {
    // Expresión regular para validar direcciones de correo electrónico
    var regex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

    // Utiliza test() para verificar si el correo cumple con la expresión regular
    return regex.test(correo);
}

function validarContrasena(contrasena) {
    // Expresión regular para validar contraseñas
    var regex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*])[A-Za-z\d!@#$%^&]{8,}$/;

    // Utiliza test() para verificar si la contraseña cumple con la expresión regular
    return regex.test(contrasena);
}

function validarDNI(nif) {
    // Expresión regular para validar un NIF español
    var regex = /^[0-9]{8}[TRWAGMYFPDXBNJZSQVHLCKE]$/i;

    // Utiliza test() para verificar si el número cumple con la expresión regular
    if (!regex.test(nif)) {
        return false; // El formato del NIF es incorrecto
    }

    // Obtiene los números del NIF (los primeros 8 caracteres)
    var numerosNIF = nif.substr(0, 8);

    // Obtiene la letra proporcionada en el NIF (el último carácter)
    var letraProporcionada = nif.charAt(8).toUpperCase();

    // Array con las letras correspondientes a los números del NIF
    var letrasNIF = 'TRWAGMYFPDXBNJZSQVHLCKE';

    // Calcula la letra correcta para los números del NIF
    var letraCorrecta = letrasNIF[numerosNIF % 23];

    // Compara la letra proporcionada con la letra correcta
    return letraProporcionada === letraCorrecta;
}
```

Figura 3.13: Validación de formularios

3.5. Configuración de seguridad insuficiente

La configuración de seguridad incorrecta o insuficiente se refiere a ajustes o parámetros de seguridad que no están adecuadamente configurados para proteger un sistema, aplicación, red o servicio. Puede haber varias áreas en las que la configuración de seguridad puede ser insuficiente o incorrecta, lo que podría dejar un sistema vulnerable a amenazas y ataques.

3.5.1. Despliegue seguro

Descripción

Es importante que a la hora de montar nuestro servidor web no haya archivos que puedan ser accesibles desde el exterior y que puedan contener información sensible como contraseñas. Esto a nosotros nos ocurre en el `docker-compose.yml`

Solución

Hemos creado un `.env` donde se almacenan las contraseñas y demás información sensible y hemos añadido el archivo al `.gitignore` para que no se suba al repositorio. Dado que esto es un trabajo el `.env` será incluido para poder comprobar que funciona correctamente.

```
db:
  container_name: db
  image: mariadb:latest
  restart: always
  volumes:
    - ./mysql:/var/lib/mysql
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
  ports:
    - "8889:3306"

phpmyadmin:
  container_name: phpmyadmin
  image: phpmyadmin/phpmyadmin:latest
  links:
    - db
  ports:
    - 8890:80
  environment:
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
```

Figura 3.14: docker-compose sin informacion sensible

3.5.2. Cabecera CSP

Descripción

Las cabeceras CSP son una medida de seguridad utilizada en la programación web para mitigar los riesgos asociados con ataques de Cross-Site Scripting (XSS) y otros tipos de ataques de inyección de código malicioso en páginas web.

Solución

Para solucionar este problema, hemos añadido una cabecera 'Content-Security-Policy' con los siguientes valores:

```
<meta http-equiv="Content-Security-Policy" content="
default-src 'self';
script-src 'self' https://code.jquery.com/ 'unsafe-inline';
style-src 'self';
img-src 'self' data;;
">
```

Figura 3.15: Cabecera CSP

3.5.3. Cabecera Cache-Control

Descripción

Las cabeceras 'Cache-Control' son utilizadas en las respuestas HTTP enviadas por un servidor web para controlar cómo los recursos web deben ser almacenados en la memoria caché del navegador o en servidores intermedios (como proxies) y cómo se deben comportar en términos de almacenamiento y actualización. Estas cabeceras son fundamentales para gestionar la eficiencia de la carga de páginas web, la seguridad y la privacidad.

Solución

Para solucionar este problema crearemos una cabecera 'Cache-Control' con el valor 'no-store' para que el navegador no almacene en cache la pagina.

```
<!DOCTYPE html>
<html Lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  ⚡ <meta http-equiv="Cache-Control" content="no-store, no-cache, must-revalidate">
  <title>Cine Nómada</title>
  <link rel="stylesheet" href="css/style.css">
  <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
```

Figura 3.16: Cabecera Cache-Control

3.5.4. Cabecera HSTS

Descripción

Las cabeceras HSTS (HTTP Strict Transport Security) son una medida de seguridad utilizada en la programación web para garantizar que las comunicaciones entre un navegador web y un sitio web se realicen a través de una conexión segura y encriptada utilizando el protocolo HTTPS (HTTP Secure). HSTS ayuda a prevenir ataques de tipo man-in-the-middle (MitM) que podrían exponer datos sensibles o comprometer la seguridad de la comunicación.

Solución

Para solucionar este problema, hemos añadido una cabecera 'Strict-Transport-Security' con los siguientes valores:

```
# Cabecera HSTS
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"
```

Figura 3.17: Cabecera HSTS

3.5.5. Cabecera X-XSS-Protection

Descripción

Las cabeceras "X-XSS-Protection" son una medida de seguridad utilizada en la programación web para ayudar a prevenir ataques de tipo Cross-Site Scripting (XSS). Los ataques XSS ocurren cuando un atacante inyecta código JavaScript malicioso en una página web, que luego se ejecuta en el navegador de un usuario sin su conocimiento. Estas cabeceras se utilizan para controlar y mitigar estos ataques.

Solución

Para solucionar este problema, hemos añadido una cabecera 'X-XSS-Protection' con los siguientes valores:

```
# Cabecera X-XSS-Protection
Header always set X-XSS-Protection "1; mode=block"
```

Figura 3.18: Cabecera X-XSS-Protection

3.5.6. Cabecera X-Content-Type-Options

Descripción

Las cabeceras "X-Content-Type-Options" son una medida de seguridad utilizada en la programación web para mitigar ciertos tipos de ataques, como ataques de tipo MIME-sniffing. Estas cabeceras se utilizan para controlar cómo el navegador web interpreta y muestra el contenido de una página web.

Solución

Para solucionar este problema, hemos añadido una cabecera 'X-Content-Type-Options' con los siguientes valores:

```
# Cabeceras X-Content-Type-Options
Header always set X-Content-Type-Options "nosniff"
```

Figura 3.19: Cabecera X-Content-Type-Options

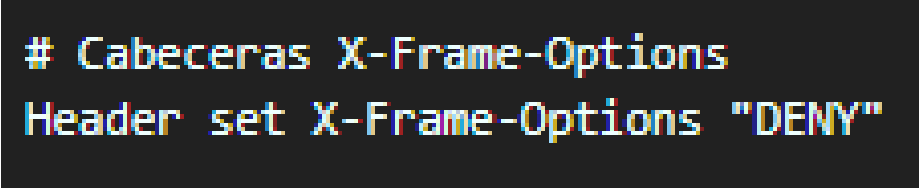
3.5.7. Cabecera X-Frame-Options

Descripción

Las cabeceras "X-Frame-Options" son una medida de seguridad utilizada en la programación web para controlar si una página web puede ser incrustada o mostrada dentro de un marco (frame) de otro sitio web. Estas cabeceras se utilizan para prevenir ataques de clickjacking, en los cuales un atacante puede ocultar contenido malicioso detrás de una página legítima y engañar a los usuarios para que hagan clic en elementos sin su consentimiento.

Solución

Para solucionar este problema, hemos añadido una cabecera 'X-Frame-Options' con los siguientes valores:



```
# Cabeceras X-Frame-Options  
Header set X-Frame-Options "DENY"
```

Figura 3.20: Cabecera X-Frame-Options

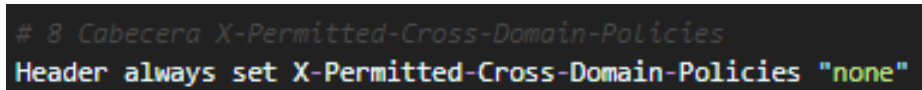
3.5.8. Cabecera X-Permitted-Cross-Domain-Policies

Descripción

La cabecera X-Permitted-Cross-Domain-Policies permite a los propietarios del contenido controlar cómo los documentos son tratados en contextos de navegación cruzada.

Solución

Para solucionar este problema crearemos una cabecera 'X-Permitted-Cross-Domain-Policies' con el valor 'none' para que el navegador no permita que se usen características y APIs en los contenidos de una página.

A screenshot of a code editor with a dark background. It shows two lines of code. The first line is a comment: `# 8 Cabecera X-Permitted-Cross-Domain-Policies`. The second line is an instruction: `Header always set X-Permitted-Cross-Domain-Policies "none"`.

```
# 8 Cabecera X-Permitted-Cross-Domain-Policies
Header always set X-Permitted-Cross-Domain-Policies "none"
```

Figura 3.21: Cabecera X-Permitted-Cross-Domain-Policies

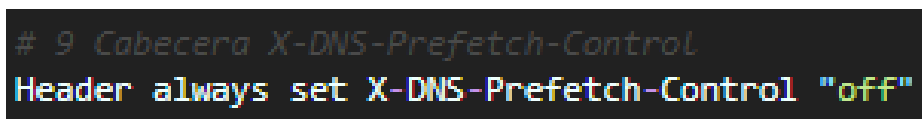
3.5.9. Cabecera X-DNS-Prefetch-Control

Descripción

La cabecera X-DNS-Prefetch-Control es una cabecera HTTP que se utiliza para controlar la funcionalidad de prefetching DNS en los navegadores. El prefetching DNS es una técnica que los navegadores utilizan para anticiparse a las solicitudes DNS antes de que un usuario haga clic en un enlace. Esto puede acelerar la carga de páginas al preresolver los nombres de dominio antes de que se soliciten explícitamente. Esto puede provocar que el navegador realice peticiones a dominios que no son de confianza, dar información de dominios sensibles, etc.

Solución

Para solucionar este problema crearemos una cabecera 'X-DNS-Prefetch-Control' con el valor 'off' para que el navegador no permita que se precargue contenido de la página.



```
# 9 Cabecera X-DNS-Prefetch-Control  
Header always set X-DNS-Prefetch-Control "off"
```

Figura 3.22: Cabecera X-DNS-Prefetch-Control

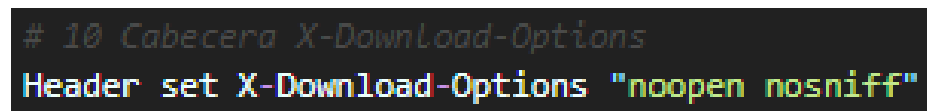
3.5.10. Cabecera X-Download-Options

Descripción

La cabecera X-Download-Options es una cabecera HTTP que se utiliza para controlar cómo ciertos navegadores manejan la descarga de archivos. Esta cabecera tiene un propósito específico en el contexto de Internet Explorer (IE) y se utiliza para mitigar el riesgo de ejecución automática de archivos descargados que podrían ser maliciosos.

Solución

Para solucionar este problema crearemos una cabecera 'X-Download-Options' con el valor 'noopen' para que el navegador no permita que se ejecuten los archivos descargados.



```
# 10 Cabecera X-Download-Options  
Header set X-Download-Options "noopen nosniff"
```

Figura 3.23: Cabecera X-Download-Options

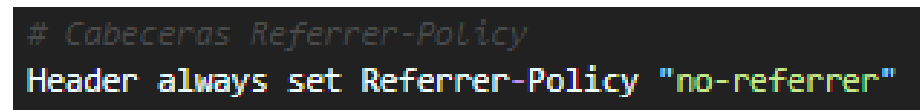
3.5.11. Cabecera Referrer-Policy

Descripción

Esta cabecera controla cómo se incluye el encabezado Referer en las solicitudes. Puedes configurarlo para reducir la cantidad de información sensible enviada en las solicitudes de referencia.

Solución

Para solucionar este problema crearemos una cabecera 'Referrer-Policy' con el valor 'no-referrer' para que el navegador no envíe información sensible en las solicitudes de referencia.

A screenshot of a code editor with a dark background. It shows two lines of code. The first line is a comment: `# Cabeceras Referrer-Policy`. The second line is a directive: `Header always set Referrer-Policy "no-referrer"`. The text is in a monospaced font with syntax highlighting: the comment is grey, and the directive uses various colors (blue, green, red, yellow) for different parts of the code.

```
# Cabeceras Referrer-Policy
Header always set Referrer-Policy "no-referrer"
```

Figura 3.24: Cabecera Referrer-Policy

3.5.12. Cabecera Feature-Policy

Descripción

La cabecera 'Feature-Policy' permite que un sitio controle qué características y APIs pueden ser utilizadas por los contenidos de una página.

Solución

Para solucionar este problema crearemos una cabecera 'Feature-Policy' con el valor 'none' para que el navegador no permita que se usen características y APIs en los contenidos de una página.

```
# Cabeceras Feature-Policy  
Header always set Feature-Policy "geolocation 'self'; camera 'none'";
```

Figura 3.25: Cabecera Feature-Policy

3.5.13. Cabecera Expect-CT

Descripción

Las cabeceras 'Expect-CT' ayuda a proteger al usuario contra ataques de certificate transparency.

Solución

Para solucionar este problema crearemos una cabecera 'Expect-CT' con el valor 'max-age=86400' para que el navegador no permita que se usen características y APIs en los contenidos de una pagina.

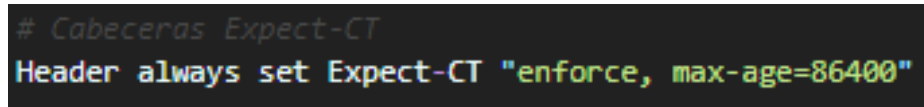
A screenshot of a code editor with a dark background. The first line is a comment: `# Cabeceras Expect-CT`. The second line is a directive: `Header always set Expect-CT "enforce, max-age=86400"`. The text is color-coded: the comment is grey, and the directive is in blue, green, and yellow.

Figura 3.26: Cabecera Expect-CT

3.5.14. Configuración PHP

Descripción

Cuando usamos PHP en un servidor web es importante el configurarlo de forma segura para evitar que se filtre información que pueda ayudar a los atacantes a encontrar una vulnerabilidad en nuestro sistema.

Solución

Hemos ocultado la versión de PHP en las peticiones a nuestro servidor para evitar que un atacante pueda usar esta información para encontrar una vulnerabilidad en nuestro sistema. Para ello hemos creado el `php.ini`

```
; Decides whether PHP may expose the fact that it is installed on the server
; (e.g. by adding its signature to the Web server header). It is no security
; threat in any way, but it makes it possible to determine whether you use PHP
; on your server or not.
; https://php.net/expose-php
expose_php = Off
```

Figura 3.27: Ocultar versión de PHP

3.5.15. Configuración Apache

Descripción

Al igual que con PHP en el punto anterior es importante configurar Apache para que muestre la mínima información al exterior.

Solución

Para ello, al igual que con PHP, hemos ocultado las versiones del servidor para evitar en la medida de lo posible que el atacante encuentre vulnerabilidad en nuestro servidor. Para ello hemos editado el `apache2.conf`

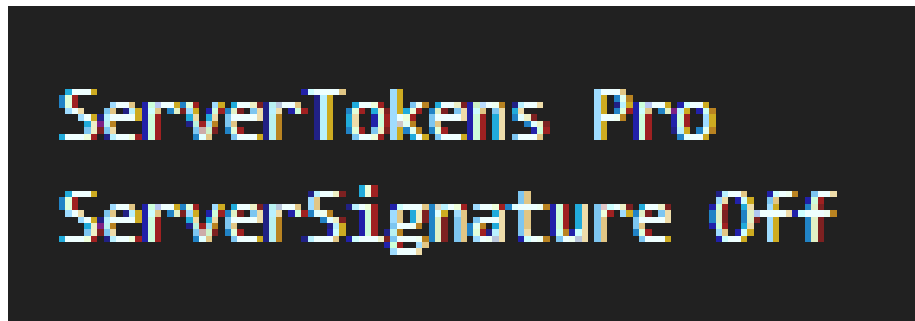


Figura 3.28: Ocultar versión de Apache

3.6. Componentes vulnerables y obsoletos

Se refiere a partes o elementos dentro de un sistema, software o hardware, que presentan vulnerabilidades de seguridad debido a su antigüedad o a la falta de actualizaciones.

3.6.1. Control de versiones de los componentes

Descripción

Es crítico mantener actualizados los sistemas operativos y el software de seguridad con las últimas actualizaciones y parches de seguridad.

Solución

Para solucionar este problema, hemos actualizado todos los componentes a sus ultimas versiones.

- PHP 7.2.2 → 8.2
- MariaDB 10.8.2 → 11.1.2
- phpMyAdmin 5.1.1 → 5.1.1
- jQuery 3.6.0 → 3.7.1

Es recomendable en estos casos no usar latest ya que en el pasado ha sido un problema de seguridad al obtener un atancante acceso a la imagen y alterandola provocando que todo el que use latest se descargase la version infectada.

3.7. Fallos de identificación y autenticación

Los fallos de identificación y autenticación se refieren a deficiencias o problemas en los procesos y mecanismos diseñados para verificar la identidad de un usuario y asegurar que la persona o entidad que intenta acceder a un sistema o recurso es realmente quien afirma ser.

3.7.1. Captcha

Descripción

Un captcha es un tipo de prueba de Turing que se utiliza para determinar si el usuario es o no humano. Los captchas se utilizan para prevenir ataques de tipo bot, como el spam y el abuso de servicios web.

Solución

Para solucionar este problema, hemos añadido un captcha a nuestro formulario de registro e inicio de sesión. Hemos añadido para nuestro proyecto el reCAPTCHA v3 de Google ya que es el más fácil de implementar y el que menos molesta al usuario. La puntuación se basa en las interacciones con su sitio y le permite tomar las medidas adecuadas para su sitio. Es por esto, que ha diferencia de otros captchas, este no muestra ninguna imagen al usuario pero si que se ejecuta en segundo plano y si detecta que el usuario es un bot, se le bloquea el acceso.

3.7.2. Contraseñas debiles o por defecto

Descripción

Las contraseñas débiles o por defecto son contraseñas que son fáciles de adivinar o que se utilizan en muchos sistemas diferentes. Esto puede permitir a un atacante obtener acceso no autorizado a un sistema o aplicación.

Solución

Para solucionar este problema, hemos hecho dos cosas:

- Añadir un JS que asegura que las contraseñas de nuestros usuarios cumplan con los requisitos minimos de seguridad.

```
function validarContrasena(contrasena) {  
  // Expresión regular para validar contraseñas  
  var regex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*])[A-Za-z\d!@#$%^&*]{8,}$/  
  
  // Utiliza test() para verificar si la contraseña cumple con la expresión regular  
  return regex.test(contrasena);  
}
```

Figura 3.29: JS de validacion de contraseñas

- Cambiar las contraseñas de nuestros servicios como phpMyAdmin, MariaDB, etc.

```
MYSQL_ROOT_PASSWORD: root  
MYSQL_USER: admin  
MYSQL_PASSWORD: test  
MYSQL_DATABASE: database
```

Figura 3.30: Contraseña antes del cambio

- Estas contraseñas se ven dentro del .env el cual no se debería subir pero al ser un trabajo se ha subido al repositorio.

3.7.3. Invalidacion de sesiones

Descripción

La invalidación de sesiones se refiere a la práctica de terminar una sesión de usuario cuando el usuario sale de una aplicación o sitio web. Esto puede ayudar a prevenir ataques de tipo secuestro de sesión, en los que un atacante puede tomar el control de una sesión de usuario activa.

Solución

En este punto tambien hemos puesto dos soluciones al problema:

1. Hemos añadido un boton de cerrar sesion en la pagina de perfil de usuario.
2. La cookie se cierra automaticamente al cerrar el navegador.

Con estas dos soluciones, si un usuario se olvida de cerrar sesion, el sistema lo hara automaticamente y si un atacante consigue la sesion de un usuario, esta se invalidara de igual forma.

3.8. Fallos en la integridad de datos y software

Los fallos en la integridad de datos y software pueden tener graves consecuencias en la seguridad y confiabilidad de sistemas y aplicaciones. Estos fallos pueden ocurrir debido a diversas razones, y es importante abordarlos adecuadamente para mantener la integridad de los datos y el funcionamiento del software.

3.8.1. Copias de seguridad

Descripción

Las copias de seguridad son una medida de seguridad que permite a los usuarios y administradores de sistemas restaurar datos y sistemas a un estado anterior en caso de que se produzca una pérdida de datos o un fallo del sistema. Las copias de seguridad pueden ser útiles para recuperar datos perdidos debido a errores humanos, ataques de malware, fallos de hardware, desastres naturales, etc.

Solución

Para solucionar este problema, demos de realizar una copia de seguridad de nuestra base de datos cada cierto periodo de tiempo. En nuestro caso accedemos a la base de datos mediante el phpMyAdmin y exportaremos la base de datos en un archivo .sql

3.8.2. CI/CD

Descripción

CI/CD es un conjunto de prácticas y herramientas que permiten a los equipos de desarrollo de software entregar cambios de código de forma rápida y fiable. CI/CD es un acrónimo de Integración Continua (CI) y Despliegue Continua (CD). CI/CD es una práctica fundamental para DevOps.

Solución

Para solucionar este problema podemos generar un pipeline de CI/CD en GitHub Actions mediante los Workflows.

```
name: Test

on: push

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3
        with:
          context: ./Docker
```

Figura 3.31: Pipeline de CI/CD

Este no es un pipeline completo con todas las pruebas unitarias ni la creación de las imágenes ni un despliegue como el que podría ser Kubernetes pero sirve como ejemplo para este trabajo.

3.9. Fallos en la monitorizacion de la seguridad

Los fallos en la monitorización de la seguridad se refieren a deficiencias o problemas en los sistemas y procesos diseñados para vigilar y evaluar la seguridad de una red, sistema informático, aplicación o entorno en general.

3.9.1. Auditorias de seguridad

Descripción

Una auditoría de seguridad es un proceso sistemático de evaluación y revisión de sistemas, redes, aplicaciones o entornos tecnológicos con el propósito de identificar vulnerabilidades, debilidades y riesgos de seguridad. El objetivo principal de una auditoría de seguridad es garantizar que los controles de seguridad estén implementados adecuadamente y cumplan con los estándares de seguridad, y proporcionar recomendaciones para mejorar la protección de activos y datos contra amenazas y ataques cibernéticos.

Solución

La solución a este problema es realizar una auditoría de seguridad de forma periódica para detectar posibles fallos de seguridad.

3.9.2. Configuración de logs del sistema

Descripción

Los logs del sistema son una herramienta de seguridad que permite a los administradores de sistemas y a los equipos de seguridad supervisar y analizar la actividad de los sistemas y las redes. Los logs del sistema pueden utilizarse para detectar y analizar incidentes de seguridad, así como para identificar y prevenir posibles amenazas.

Solución

Para solucionar este problema, hemos implementado una lógica del log del lado del servidor para guardar los logs de los usuarios al cometer errores.

```
<?php
// Ruta del archivo de registro
$logFile='../log/log.json';

// Verifica si la ruta del archivo no está vacía antes de intentarlo
if (!empty($logFile)) {
    $archivo = fopen($logFile, 'a');

    // Verifica si la apertura del archivo fue exitosa antes de escribir en él
    if ($archivo) {
        // Obtiene la fecha y hora actual
        $date = date('Y-m-d H:i:s');

        // Elimina el último elemento del array $_POST
        array_pop($_POST);

        // Obtiene la dirección IP del cliente
        // $ip = $_SESSION['ip'];

        // Crea un array con la fecha, hora y datos del $_POST
        $logData = [
            'date' => $date,
            // 'ip' => $ip,
            'postData' => $_POST,
        ];

        // Convierte el array a formato JSON de manera legible
        $mensaje = json_encode($logData, JSON_PRETTY_PRINT);
        fwrite($archivo, $mensaje . "\n");
        fclose($archivo);
    } else {
        echo "Error al abrir el archivo de registro.";
    }
} else {
    echo "La ruta del archivo de registro está vacía.";
}
?>
```

Figura 3.32: Logs del lado del servidor

Segunda auditoria

Tras arreglar los problemas de la primera auditoria, hemos vuelto a realizar una auditoria de seguridad para comprobar que no se han introducido nuevos problemas de seguridad.

4.1. ZAP

Para llevar la misma estructura que en la primera auditoria, hemos usado la misma herramienta para realizar la segunda auditoria.

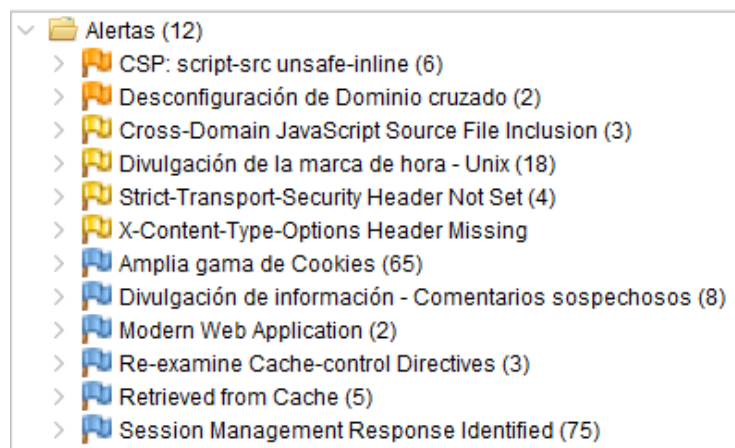


Figura 4.1: Listado de errores de ZAP de la segunda auditoria

Aunque veamos que todavia hay errores estos en realidad son dados porque ZAP no puede comprobar algunas de las librerias que usamos en nuestro proyecto como son jQuery y reCAPTCHA.

Aun asi, ambas librerias hacen que nuestra web sea mas segura por lo que no las vamos a eliminar de nuestro proyecto solo para que ZAP no de errores.

4.2. sqlmap

4.3. nmap

4.4. Metaexploit

Conclusiones

Bibliografia

- OWASP. (2021). Informe de Vulnerabilidades. OWASP. <https://owasp.org/www-project-top-ten/>
- GPT-3.5. (2023). Respuestas a preguntas varias. OpenAI. <https://www.openai.com/>
- GitHub Copilot. (2022). Autocompletado. GitHub. <https://github.com/features/copilot>
- PHP. (2021). Manual de PHP. PHP. <https://www.php.net/manual/es/>
- reCAPTCHA. (2018). Documentacion de reCAPTCHA. Google. <https://developers.google.com/recaptcha/docs/v3>