Informe de Auditoría de Seguridad (Entrega 3)

Asignatura: Sistemas de Gestión de Seguridad de la Información **Autor:** Equipo Rojo **Fecha:** 7/11/2025

---

1. Introducción a los problemas y Resumen

Este documento es el informe de auditoría que se pide para la Entrega 3. El objetivo principal era analizar la aplicación web que hicimos en la Entrega 1 para ver qué fallos de seguridad tenía y ahora solucionarlos documentando la información.

Para hacerlo, hemos usado la herramienta OWASP ZAP (v2.16.1). Después de configurar el proxy y escanear la web (que corría en `http://localhost:81`), ZAP ha encontrado **12 tipos de alertas** diferentes.

Los problemas más importantes que hemos visto son los de riesgo "Medio", como la **Ausencia de Tokens Anti-CSRF** y problemas con las **cookies (no tenían `HttpOnly` ni `SameSite`)**. También hay un montón de fallos de "Configuración de Seguridad Insuficiente", como cabeceras (headers) que faltan.

Lo bueno es que, aunque hemos lanzado el escaneo activo, **ZAP no ha encontrado ningún fallo de Inyección SQL**.

---

2. Herramientas y Proceso de Auditoría

- **Herramienta:** Hemos usado OWASP ZAP (Zed Attack Proxy), la versión 2.16.1.
- **Objetivo:** La aplicación web de la Entrega 1, ejecutándose en `http://localhost:81`.

El proceso que hemos seguido ha sido este:

1. **Configuración:** Primero tuve que configurar el proxy de Firefox (en `localhost:8080`) y cambiar los ajustes (`about:config`) para que pudiera "ver" el tráfico de `localhost`.
2. **Exploración Manual:** Hemos navegado por toda la web (haciendo login, visitando el perfil, etc.) para que ZAP "aprendiera" todas las páginas y formularios que tiene el sitio.

3. **Escaneo Activo:** Una vez ZAP tenía el mapa del sitio, hemos lanzado el "Active Scan" para que intentara atacar la web y buscar fallos de forma automática.

---

3. Resultados de la Auditoría (Alertas Encontradas)

El escaneo de ZAP ha terminado con 12 tipos de alertas. Las agrupo aquí por el riesgo que ZAP les ha dado:

Fallos de Riesgo Medio (Los más importantes)

ZAP encontró 4 alertas de riesgo medio, casi todas relacionadas con la gestión de la sesión y el control de acceso:

- **Ausencia de Tokens Anti-CSRF:** Una solicitud falsa entre sitios en un ataque que compromete y obliga a una víctima a enviar su solicitud HTTP a un destino objetivo sin su conocimiento o intención para poder realizar una acción como víctima.

  Detectado en :http://localhost:81/register

  Descripción en detalle: Cuando el usuario inicia sesión, el servidor le asigna una sesión para identificarlo. No obstante, en el estado inicial de la aplicación, las acciones realizadas por el usuario podían ser falsificadas, ya que no existía un mecanismo adicional que verificara que cada solicitud provenía realmente del usuario legítimo. Para mitigar este riesgo, se decidió implementar el patrón *Synchronizer Token Pattern*. Este mecanismo consiste en generar un token único asociado a la sesión del usuario y enviarlo junto con cada solicitud que implique el envío de información.

  Solución: Se ha añadido la verificación de dicho token en todas las clases donde se procesan formularios que reciben datos del usuario. De esta forma, el servidor comprueba tanto la sesión como el token antes de ejecutar la acción solicitada, asegurando que la operación ha sido iniciada por el usuario legítimo y reduciendo la posibilidad de ataques de suplantación.

- **Cookie Sin Flag HttpOnly:** Se ha establecido una cookie sin el flag HttpOnly, lo que significa que JavaScript puede acceder a la cookie. Si un script malicioso puede ser ejecutado en esta página, entonces la cookie será accesible y puede ser transmitida a otro sitio para un secuestro de sesión .

  Detectado en: http://localhost:81/sitemap.xml

Solución: modificar el `index.php` para unificar el inicio de la sesión. Se eliminaron las llamadas separadas a `session_set_cookie_params` y se reemplazaron por un único comando `session_start([...])` explícito. Este comando ahora incluye toda la configuración de seguridad directamente en sus parámetros de inicio, destacando la directiva clave:

```
'cookie_httponly' => true
```

Al forzar la configuración de seguridad dentro de la misma llamada que inicia la sesión, nos aseguramos de que es imposible que la sesión se cree sin que la bandera `HttpOnly` esté aplicada, protegiéndola del acceso por JavaScript del lado del cliente

- **Cookie sin el atributo SameSite:** Se ha establecido una cookie sin el atributo SameSite, lo que significa que la cookie puede ser enviada como resultado de una solicitud 'cross-site'. El atributo SameSite es una medida eficaz para contrarrestar la falsificación de peticiones entre sitios, la inclusión de scripts entre sitios y los ataques de sincronización.

  Solución: a la hora de arreglar el problema de la ausencia de tokens Anti-CSRF se eliminó este problema con el. Sospechamos que eran dos maneras diferentes de solucionar el mismo problema.

- **Falta de cabecera Anti-Clickjacking (X-Frame-Options):** La web se puede meter en un `<iframe>` en una página maliciosa, engañando al usuario para que haga clic en cosas que no quiere. La respuesta no protege contra ataques de "ClickJacking". Se debe incluir Content-Security-Policy con la directiva "frame-ancestors" o X-Frame-Options.

  Solución: A la hora de solucionar este problema, vimos que hay dos maneras principales de arreglarlo, o bien mediante legado X-frame-options, o bien mediante cabeceras CSP, viendo que necesitábamos hacer CSP de todas maneras, hemos decidido tomar este camino.

Fallos de Riesgo Bajo (Configuraciones)

Estos son principalmente "Configuraciones de seguridad insuficientes" que hay que arreglar:

- **Cabecera Content Security Policy (CSP) no configurada:** La Política de seguridad de contenido (CSP) es unacomo capa adicional de seguridad que ayuda a detectar y mitigar ciertos tipos de ataques. CSP proporciona un conjunto de encabezados HTTP estándar que permiten a los propietarios de sitios web declarar fuentes de contenido aprobadas que los navegadores deberían poder cargar en esa página.

  Solución: Este ha sido un caso complejo, ya que para solucionar este problema concreto bastaba con definir la directiva principal de CSP, el problema nos ha llegado cuando al definirla, nos decía ZAP que tenía bastantes brechas de seguridad, esto se ha solucionado especificando las fuentes de contenido de la directiva, sobretodo con el fin de limitar el riesgo de inyección.

- **Falta encabezado X-Content-Type-Options:** La cabecera Anti-MIME-Sniffing X-Content-Type-Options no se ha establecido en 'nosniff'. Esto permite que las versiones anteriores de Internet Explorer y Chrome realicen MIME-sniffing en el cuerpo de la respuesta, lo que puede provocar que el cuerpo dé la respuesta se interprete y se muestre como un tipo de contenido distinto del tipo de contenido declarado.

  Solución: **modificar el `Dockerfile`** para corregir ambos problemas en la raíz, antes de que el servidor se inicie:

  Se añadió el comando `RUN a2enmod headers` para activar el módulo de cabeceras de Apache.

  Se añadió un comando (`RUN sed...`) para editar el `apache2.conf` y cambiar la directiva de `AllowOverride None` a `AllowOverride All`.

  Tras reconstruir la imagen de Docker con estos cambios (`docker build --no-cache ...`), el servidor Apache finalmente pudo activar el módulo, leer nuestro `.htaccess` y aplicar la cabecera `X-Content-Type-Options: nosniff` a todos los archivos estáticos, solucionando la vulnerabilidad

- **Petición de Autenticación Identificada:** ZAP simplemente informa de que ha encontrado el formulario de login.

  Detectado en: http://localhost:81/login

Alertas Informativas (Pistas para un atacante)

Estas no son vulnerabilidades directas, pero dan demasiada información:

- **El servidor divulga información: "X-Powered-By"**: El acceso a tal información podría facilitarle a los atacantes la identificación de otros marcos/componentes de los que su aplicación web depende y las vulnerabilidades a las que pueden estar sujetos tales componentes.
- **El servidor filtra información de versión: "Server"** (también dice qué servidor Apache es usado).
- **Divulgación de Información - Información sensible en URL** (ZAP ha visto IDs o datos en la URL).
- **Divulgación de información - Comentarios sospechosos** (hemos dejado comentarios en el HTML/JS que ZAP ha visto).
- **Respuesta de Gestión de Sesión Identificada:** ZAP informa de que ha detectado cómo el servidor gestiona las cookies de sesión, la respuesta dada contiene un token de gestión de sesión.

---

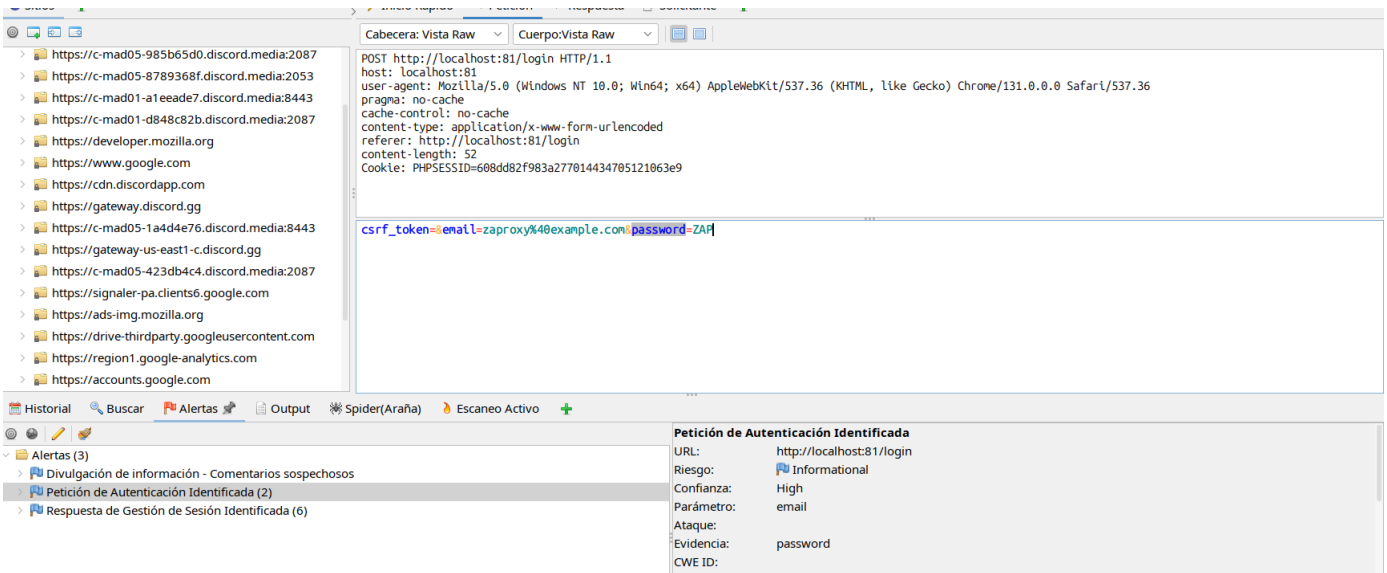
4. Vulnerabilidades NO Detectadas (Inyección SQL)

El documento del trabajo pedía explícitamente buscar vulnerabilidades de **Inyección**. Lancé el "Active Scan" de ZAP contra todas las páginas y formularios, pero la herramienta **no reportó ningún fallo de Inyección SQL**.

Esto es un resultado positivo. Como se pide en el trabajo, la explicación es que el código de la Entrega 1 ya se hizo pensando en esto: **toda la comunicación con la base de datos se hace usando consultas preparadas (prepared statements) de PDO**. Esto evita que la entrada del usuario se pueda ejecutar como código SQL, neutralizando el ataque.

---

5. Segunda revisión de Zap



Como se puede observar en la captura de pantalla adjunta, el resultado de la segunda revisión es exitoso. El panel de "Alertas" ya no reporta ninguna vulnerabilidad de Riesgo Medio (bandera naranja) ni de Riesgo Bajo (bandera amarilla).

Las únicas alertas que permanecen son de carácter "Informacional" (bandera azul), las cuales no representan fallos de seguridad activos.

Esta segunda revisión confirma que los objetivos de remediación se han cumplido y que las vulnerabilidades críticas identificadas han sido solucionadas de manera efectiva.

---

6. Observaciones y puntos de interés

- A la hora de arreglar los fallos de seguridad, hemos ido haciendo revisiones con ZAP no incluidas en la documentación, en muchas de estas revisiones nos hemos percatado que muchas veces los problemas subían o bajaban de rango “medium -> low” despues de hacer cambios de otros problemas que no eran ese.

---

7. Conclusión

La auditoría ha funcionado. ZAP encontró 12 problemas claros, sirviendo como una base excelente para la remediación. Aunque fue positivo no encontrar fallos críticos de Inyección SQL, los problemas de Riesgo Medio como CSRF y la configuración de cookies **eran importantes y han sido solucionados**.

Todos estos hallazgos **fueron** la base para el trabajo de esta Entrega 3, donde **hemos solucionado** cada uno de ellos y, como se demostró en la segunda revisión, **hemos verificado** su corrección.