PROYECTO : SISTEMA WEB

Auditoría de Seguridad

Sistemas de gestión de Seguridad de Sistemas Informáticos

SudoMotors

June Castro

Urko Horas

Aimar Larriba

Daniel Talmaci

Eneko Rodriguez

30/10/2025

Ingeniería Informática de Gestión y Sistemas de Información

ÍNDICE

1. INTRODUCCIÓN

Esta parte del proyecto tiene como finalidad asegurar la protección y resistencia de la página web que desarrollamos previamente.

En primer lugar, para detectar cuáles son los puntos débiles de la aplicación, hemos utilizado herramientas como ZAP, que permiten analizar posibles fallos de seguridad, intentos de inyección y otras técnicas de ataque dirigidas a los puntos de entrada del sistema.

Identificamos las vulnerabilidades, para después proceder a aplicar las medidas necesarias para corregirlas y así garantizar la seguridad y fiabilidad del sitio web. De esta manera asegurando que reconocemos todas las brechas de seguridad que existen en nuestro sistema web.

Este proyecto no solo nos ayuda a fortalecer nuestra aplicación, sino que también nos permite ampliar de forma práctica nuestros conocimientos en materia de ciberseguridad. Gracias a esta experiencia, adquirimos habilidades que serán útiles para enfrentarnos a situaciones reales dentro del ámbito de la seguridad informática.

2. AUDITORÍA DE SEGURIDAD

2.1 ZAP

Para comenzar con esta auditoría de seguridad hemos utilizado el proxy ZAP. Mediante esta herramienta de código abierto hemos conseguido detectar diferentes vulnerabilidades en nuestro sistema web.

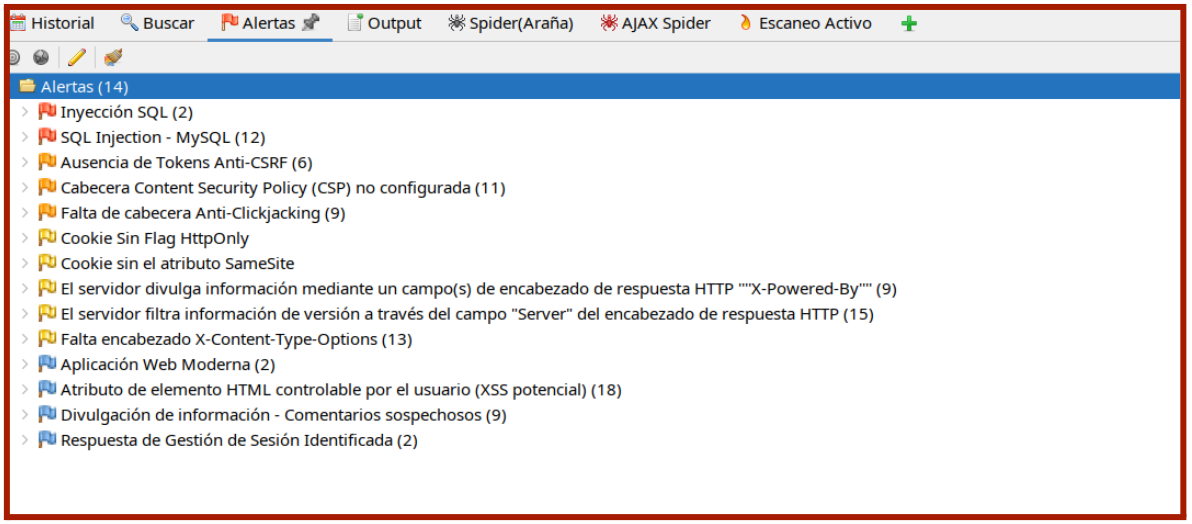

Figura 1

En la Figura 1 podemos ver el listado de errores y fallos en nuestra web, ordenados de mahyor a menor riesgo.
Como podemos comprobar el más grave se refiere en cuanto a inyecciones SQL y por general falta de cookies y cabeceras,

2.2 SQLmap

También hemos utilizado SQLmap, otra herramienta de pentesting de código abierto, la cual detecta fallos de seguridad en cuanto a bases de datos.

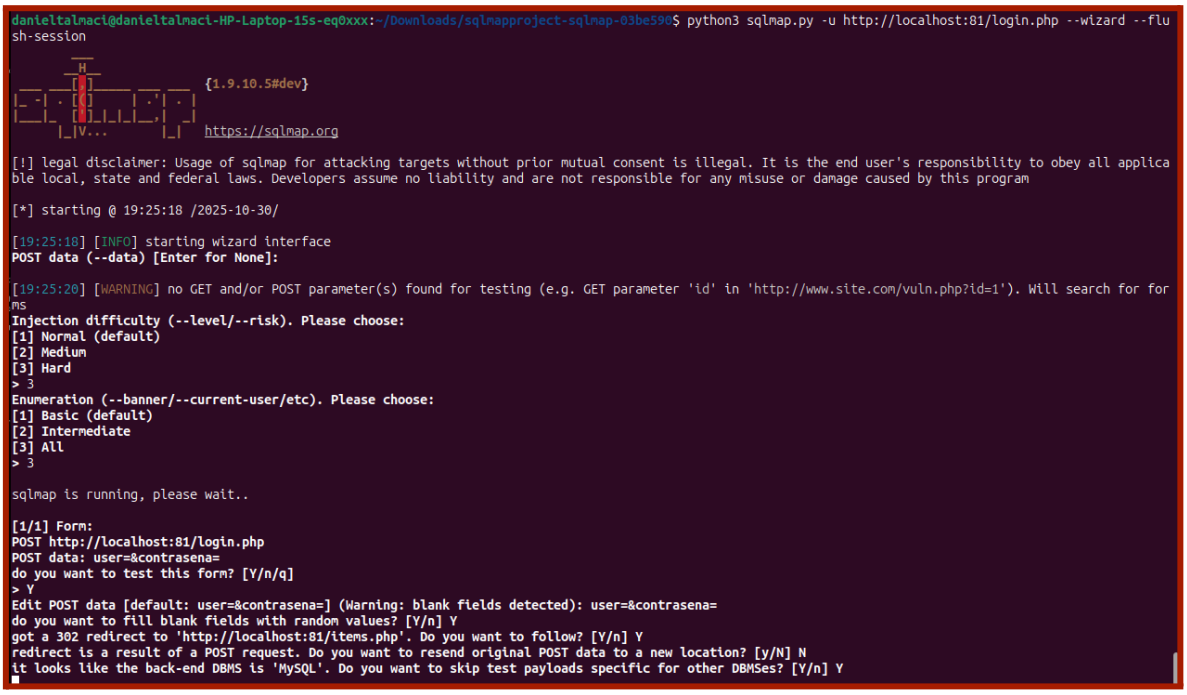

Figura 2

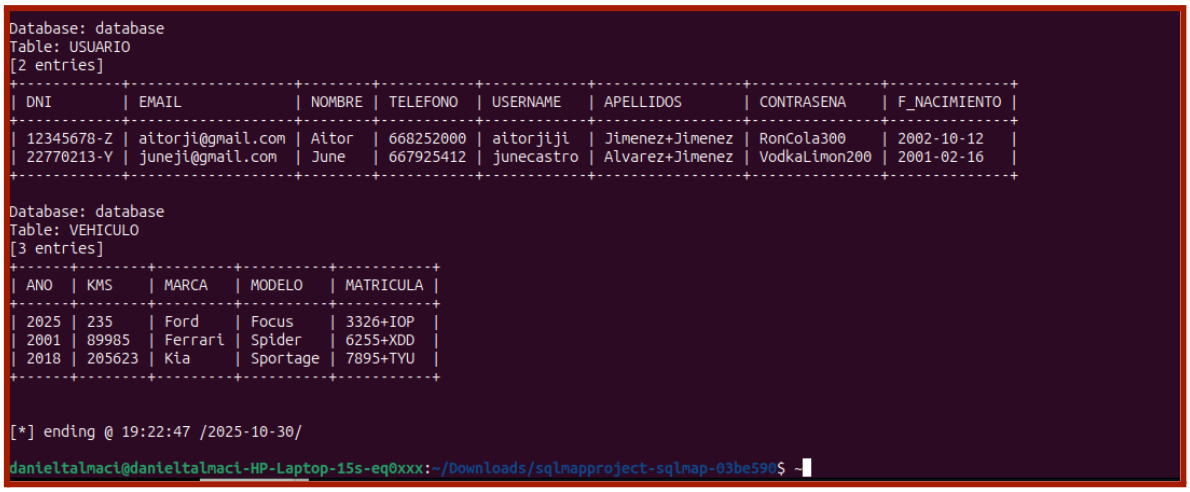

Figura 3

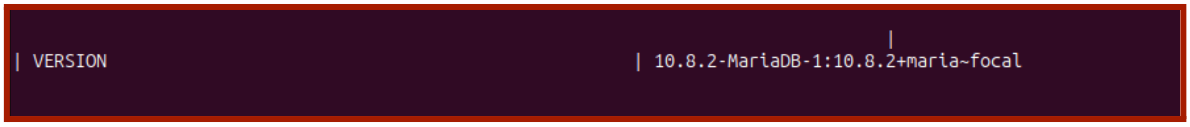

Figura 4

En la primera figura desplegamos la herramienta de SQLmap. En la Figura 2 podemos observar como la herramienta ha podido obtener toda la información de nuestra base de datos con las contraseñas incluidas. Además de mostrar también las versiones de los diferentes servicios que utiliza la página web

3. ANÁLISIS DE VULNERABILIDADES

1. Rotura de control de acceso

Una gestión de acceso ineficiente o, directamente, carecer de ella como en nuestro caso hace que cualquier usuario del sistema, aunque no debería de tener permisos para ello, pueda acceder a secciones del sistema destinadas a la administración o ciertos usuarios, como es el caso de las modificación de datos personales. Esta carencia de mecanismos que controlen los permisos de los usuarios representa una fragilidad en la seguridad del sistema, la cual es el fallo más común del OWASP Top 10.

1.1 Acceso mediante URL

Así pues, se pueden modificar o robar datos de otros usuarios, existe riesgo de incumplir el principio de mínimo privilegio y de violar la confidencialidad del sistema. además de acceder a APIs sin control para los métodos HTTP POST, PUT, y DELETE.

De esta manera, si se quieren evitar estas vulnerabilidades se deberá negar el acceso a todos los recursos privados, centralizar el control de acceso y validar el acceso para cada recurso solicitado, todo ello aplicando el principio de denegación por defecto. Además, es recomendable registrar el intento de múltiples intentos de acceso fallidos, por si se trata de un ataque.

2. Fallos criptográficos

Los fallos en los métodos criptográficos pueden provocar una protección insuficiente o nula de los datos sensibles, tanto en tránsito como en reposo. Estas vulnerabilidades suelen deberse a algoritmos débiles, claves inseguras o configuraciones erróneas.

2.1 Mala gestión de claves

En nuestro sistema, no existe ningún tipo de cifrado mediante claves y su correspondiente gestión. Así pues, la confidencialidad, integridad y disponibilidad de la información pueden quedar seriamente comprometidas en caso de ataque.

Primero que todo, la ausencia de algoritmos robustos de generación de claves y de conexiones seguras puede dar lugar a ataques Man in the middle o robo de información. Además, cuando se implementen claves en el sistema, es necesario asegurar que estas tienen una rotación y revocación de claves correcta, ya que, en caso contrario, puede dar lugar a que estas sean comprometidas.

En la situación actual del sistema, los datos del sistema no poseen seguridad ninguna, poniendo así en duda la fiabilidad del sistema. Por consiguiente, es necesario adoptar distintas medidas que garanticen un uso y gestión de claves como conexiones cifradas seguras y confiables.

De esta forma, es imprescindible la implementación de claves generadas con una alta entropía, para evitar usar claves débiles, y de conexiones seguras al sistema web mediante SSL/TLS (https). Asimismo, es indispensable asegurar que solo el personal autorizado tiene acceso a las claves del sistema. En cuanto a la rotación de las mismas, es

recomendable no alargar su uso durante un largo tiempo y evitar el uso de claves caducadas. Por último, a la hora de revocarlas, no deben ser eliminadas completamente, si no almacenarse de forma segura por si es necesario su uso en el futuro.

2.2 Almacenamiento y transmisión de datos en texto plano

Guardar los diferentes datos del sistema, especialmente las contraseñas de los usuarios, en texto plano o sin las debidas medidas de protección puede derivar en una pérdida masiva de información o en su posible modificación por terceros no autorizados para ello.

Esta falla en la seguridad afecta a todo el sistema web desarrollado, dado que actualmente no se cifran de ninguna forma los datos. Asimismo, en cuanto a las contraseñas, esta ausencia de cifrado permite a un atacante obtener las credenciales de otro usuario y acceder a sus cuentas, poniendo así en riesgo su confidencialidad y vulnerando la normativa de protección de datos.

Con el objetivo de mitigar al máximo lo mencionado anteriormente, es necesario clasificar los datos según su nivel sensibilidad, cifrar toda la información mediante protocolos seguros, como TLS con Forward Secrecy, además de reducir el almacenaje de datos sensibles todo lo posible. En cuanto a la gestión de contraseñas se recomienda almacenarlas utilizando funciones hash seguras combinadas con salts de la mayor entropía posible.

3. Inyección

Si no se comprueban los datos de entrada, existe la posibilidad de que estos sean interpretados como código por el sistema y, de esta forma, que un atacante modifique contenido no accesible o ejecute scripts maliciosos en el navegador de otros usuarios, conocido como Cross-Site Scripting.

Actualmente en el sistema, los datos introducidos por el usuario no se filtran o sanean de ninguna forma antes de ser utilizados en el servidor. En consecuencia, existe una vulnerabilidad importante ante ataques mediante inyección.

De esta forma, en todas las partes de nuestro sistema que se reciben datos externos, como pueden ser formularios, cabeceras etc. existe este peligro. Como consecuencia, la integridad y disponibilidad de la información del sistema, el comportamiento y la privacidad de las cuentas puede quedar comprometida. Esto se debe a que las consultas que se realizan en el sistema son consultas dinámicas, es decir, no están parametrizadas.

Con el fin de evitar este tipo de ataques, es indispensable validar y sanear todos los inputs del usuario; esto es, eliminar todos los caracteres especiales posibles. Además, usar controles como LIMIT para evitar la pérdida masiva de datos y APIs seguras o, en su defecto, llamadas parametrizadas u ORM para evitar que los datos se inserten directamente en sentencias SQL.

4. Diseño inseguro

Durante el análisis se identificó que el sistema no fue desarrollado bajo un enfoque de seguridad desde el diseño (Security by Design). Varias funciones críticas carecen de controles básicos, lo que permite que las vulnerabilidades se propaguen a otras partes de la aplicación.

En el análisis ZAP se detectó falta de validación en el lado del servidor, reflejada en alertas de inyección SQL y XSS. Además, la ausencia de tokens Anti-CSRF, la falta de cabeceras como Content-Security-Policy, X-Frame-Options y X-Content-Type-Options, así como cookies sin atributos HttpOnly ni SameSite, confirman deficiencias en la configuración de seguridad.

También se observó divulgación de información a través de los encabezados X-Powered-By y Server, lo que evidencia un diseño inseguro y la ausencia de medidas preventivas básicas. Estas vulnerabilidades muestran que la aplicación prioriza la funcionalidad sobre la seguridad, comprometiendo la confidencialidad e integridad de los datos.

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.

5.1 Cabecera Anti-Clickjacking (X-Frame-Options)

El análisis de ZAP muestra la ausencia de la cabecera X-Frame-Options. Esta cabecera es fundamental para prevenir ataques de tipo clickjacking, en los que un atacante incrusta el sitio web dentro de un iframe malicioso con el fin de engañar al usuario para que realice acciones no deseadas.

En nuestra página web se manifiesta observando que al seleccionar una petición (por ejemplo, index.php), en el apartado Response Headers, no aparece la cabecera X-Frame-Options.

5.2 Cabecera Content-Security-Policy (CSP)

La cabecera Content-Security-Policy (CSP) no está configurada. Esta política limita los recursos que el navegador puede cargar, y su ausencia deja el sistema vulnerable a XSS e inyecciones de código.

En la consola del navegador (pestaña Network → Headers), al inspeccionar cualquier respuesta HTTP, no aparece la cabecera Content-Security-Policy. Esto significa que cualquier script externo puede ejecutarse sin restricciones.

5.3 Cabecera X-Content-Type-Options

Esta cabecera previene que el navegador interprete archivos con un tipo MIME distinto al declarado, lo que evita ejecución de contenido inesperado (por ejemplo, un script con extensión .jpg). Su ausencia puede derivar en ejecución de código malicioso.

Al inspeccionar las cabeceras HTTP de cualquier recurso (por ejemplo, index.php o login.php), no aparece X-Content-Type-Options. Esto se observa en el panel “Headers” de las herramientas del navegador o dentro del reporte de ZAP.

5.4 Tokens Anti-CSRF

No existen tokens de verificación CSRF en formularios sensibles (login, alta o modificación de datos). Esto permite ataques de Cross-Site Request Forgery, donde un atacante puede enviar peticiones no autorizadas en nombre del usuario autenticado.

Formularios como login.php, editar.php o insertar.php no incluyen ningún campo oculto que contenga un token CSRF. En el código fuente HTML de dichos formularios no aparece ningún valor dinámico o hash vinculado a la sesión. ZAP genera la alerta “Ausencia de Tokens Anti-CSRF”.

5.5 Cookies sin atributos de seguridad (HttpOnly / SameSite)

Las cookies utilizadas para mantener la sesión no incluyen los atributos de seguridad HttpOnly ni SameSite. Esto facilita ataques de robo de sesión mediante JavaScript o peticiones cruzadas.

ZAP reporta las alertas: “Cookie sin flag HttpOnly” y “Cookie sin el atributo SameSite”.

5.6 Divulgación de información en cabeceras HTTP (Server / X-Powered-By)

El servidor revela información sobre la tecnología utilizada, como PHP, Apache o su versión exacta, a través de cabeceras HTTP (Server, X-Powered-By). Esto facilita ataques dirigidos contra versiones concretas del software.

En el reporte de ZAP aparecen las alertas “El servidor divulga información mediante el campo X-Powered-By” y “El servidor filtra información de versión a través del campo Server”.

6. Componentes vulnerables y obsoletos

El uso de componentes vulnerables u obsoletos es una de las causas más comunes de fallos de seguridad en aplicaciones web y sistemas informáticos. Este tipo de vulnerabilidad ocurre cuando un sitio o aplicación utiliza librerías, frameworks, módulos o dependencias de software que contienen errores conocidos, versiones sin soporte o configuraciones inseguras. Es decir, causan vulnerabilidades debido a su antigüedad o a la falta de actualizaciones.

6.1 Versiones de los distintos componentes utilizados en el sistema

Es necesario mantener actualizados los sistemas operativos y los componentes utilizados por el sistema web, de esta manera asegurándonos de contener las últimas actualizaciones de seguridad y parches.

Nuestra página web utiliza servicios, tales como PHP o MariaDB, algo anticuados, debido a que el proyecto está desarrollado a partir de una base proporcionada. Esto puede llevar a no tener todas las funcionalidades de seguridad disponibles tal y como se ha comentado anteriormente.

7. Fallos de identificación y autenticación

Los fallos de identificación y autenticación son vulnerabilidades que afectan a los mecanismos encargados de reconocer y verificar la identidad de los usuarios dentro de un sistema. Estos mecanismos tienen como objetivo garantizar que solo las personas autorizadas puedan acceder a determinados recursos o información.

7.1 Administración insegura

En este caso de proyecto, al realizar nuestro sistema web en base a un proyecto ya iniciado, la contraseña del administrador para identificarse en PhPMyAdmin es muy insegura. A parte de ser conocida por las demás personas las cuales han utilizado la misma base para su proyecto.

También existe la posibilidad de que cualquier persona elimine los vehículos del sistema desde PhPMyAdmin. Además nuestra web no tiene implementada la funcionalidad de Administrador como tal, ya que todos los usuarios tienen los mismos permisos para modificar o eliminar datos del sistema. Esto supone que cualquier usuario del sistema pueda hacer lo que quiera con los datos del sistema, cosa que no conviene en un sistema.

7.2 Gestión deficiente de sesiones

Las sesiones son algo importante a tener en cuenta a la hora identificación y autenticación; ya que si esta queda abierta se pueden llegar a producir ataques de tipo de secuestro de sesión en las cuales un atacante toma el control de una sesión de usuario activa.

En nuestro sistema, a pesar de estar ya implementada desde el principio la opción para cerrar la sesión, no se asegura que si un usuario se olvida de cerrar la sesión, esta finaliza.
Para ello, existen mecanismos que aseguran que después de un tiempo especificado la sesión finaliza.

El hecho de que la sesión quede abierta supone que pueda haber acceso no autorizado por terceros, que se robe el identificador de sesión o que los datos sensibles de un usuario queden expuestos.

8. Fallos en la integridad de datos y software

Durante el análisis se identificó que el sistema no garantiza la integridad de los datos ni del software. No existen controles que aseguren que la información almacenada o modificada por el usuario sea válida y no haya sido alterada de forma maliciosa.

Se detectó falta de validación de datos, una gestión de permisos insuficiente y ausencia de comprobaciones de integridad en operaciones críticas, lo que permite que los usuarios puedan modificar información sin restricciones ni verificaciones adecuadas.

Además, algunos procesos de manipulación de datos no registran cambios ni realizan controles de coherencia, lo que aumenta el riesgo de corrupción o alteración no autorizada del contenido.

Estas debilidades exponen al sistema a ataques como inyección SQL, Cross-Site Scripting (XSS), y al uso de cookies sin atributos seguros (HttpOnly y SameSite), evidenciando un nivel de riesgo alto para la integridad y confidencialidad de la información.

Los atacantes podrían aprovechar los datos expuestos por la aplicación para ejecutar ataques, ya sea manipulando información desde el propio sistema o introduciendo datos maliciosos que comprometan su seguridad.

9. Fallos en la monitorización 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.

9.1. Monitorización de registros

En nuestro sistema no existe una monitorización adecuada de los registros (logs). Actualmente solo se comprueba si un usuario introduce correctamente sus datos para iniciar sesión, pero no se registran ni se analizan los intentos de acceso.

Esto implica que podríamos sufrir un ataque de fuerza bruta sin darnos cuenta, ya que un atacante podría probar un gran número de contraseñas hasta encontrar una válida.

La solución a este problema sería hacer un implementar un sistema de registro que haga un seguimiento de los intentos que hace un único usuario para iniciar sesión y además proporcionaremos un número limitado de intentos. Si se detecta una actividad sospechosa (demasiados intentos de acceso fallido) permite activar alertas y bloquear ataques antes de que causen daño, además de bloquear temporalmente la cuenta o la dirección IP.

4. CONCLUSIONES

A lo largo de esta fase del proyecto hemos podido identificar y analizar las distintas debilidades relacionadas con la seguridad en nuestra página web. Hemos visto que existían varios problemas graves de seguridad que hacían que la seguridad y los datos de nuestra página web se viesen comprometidos, así como los usuarios y su respectiva información.

Este proceso ha resultado muy valioso, ya que nos ha permitido darnos cuenta de que, en muchas ocasiones, la seguridad no depende únicamente de grandes cambios, sino de pequeños ajustes que marcan una gran diferencia. Por ejemplo, la incorporación de un sistema de monitorización de logs o la obligación de utilizar contraseñas más seguras hacen que la posibilidad de que un atacante pueda comprometer nuestra infraestructura se reduzca significativamente.
Estos detalles que en un principio pueden parecer secundarios, son en realidad fundamentales para proteger correctamente cualquier sistema.

Además, gracias a este trabajo, hemos tomado conciencia de la importancia de adoptar una mentalidad preventiva en el ámbito de la ciberseguridad. La seguridad no es algo que se añade al final, sino un aspecto que debe estar presente desde el diseño y mantenerse durante toda la vida del proyecto. También hemos incorporado nuevas herramientas y métodos, como el uso de ZAP para analizar vulnerabilidades, lo cual nos ha permitido aplicar los conocimientos de forma práctica y entender mejor cómo se producen y se corrigen los fallos.

El aprendizaje obtenido nos será de gran utilidad en el futuro, especialmente cuando tengamos que enfrentarnos a situaciones reales en las que la seguridad sea un aspecto clave del desarrollo.

5. BIBLIOGRAFIA

- OpenAI (ChatGPT-5). https://chatgpt.com/
- Manual PHP https://www.php.net/manual/en/index.php
- OWASP. (2021). Informe de Vulnerabilidades. OWASP. https://owasp.org/www-project-top-ten/
- ZAP. Documentación de ZAP. https://www.zaproxy.org/getting-started/
- SQLmap. Documentación de SQLmap. https://sqlmap.org/