

WikiInfoSeguridad

Galder García

Entrega 2 - Versión 2 (2022-11-20)

Índice de contenidos

Índice de contenidos	2
Changelog	3
Introducción	4
Página principal	5
Barra de navegación	7
Pie de página	8
Inicio de sesión / Registro	9
Creador de artículos	11
Visor de artículos	12
Pentesting al sistema inicial	13
Fallos criptográficos	14
Inyección	15
Rotura de control de acceso	15
Diseño inseguro	16
Configuración de seguridad insuficiente	16
Componentes vulnerables y obsoletos	17
Fallos de identificación y autenticación	17
Fallos en la integridad de datos y software	17
Fallos en la monitorización de la seguridad	18
Pentesting al sistema final	19

Changelog

Entrega 1 - Versión 1: Documentar cada apartado de la página web.

Entrega 2 - Versión 2: Añadir el informe de la auditoría web.

Introducción

Esta documentación pertenece al proyecto práctico de la asignatura Sistemas de Gestión de Seguridad de la Información.

El objetivo de esta práctica consiste en crear una página web alojada en un servidor, que se comunique con una base de datos alojada en otro servidor, y ejecutar todo con contenedores docker. Esto nos permite que se pueda ejecutar correctamente en cualquier sistema Linux.

La página web que he decidido crear es “WikiInfoSeguridad”, una Wiki en la que tendremos artículos relacionados con la seguridad informática. Cualquier usuario podrá acceder a todos los artículos ya creados. También podrá registrarse y crear sus propios artículos*.

Todo lo referente al proyecto puede encontrarse en el siguiente repositorio público de github:

<https://github.com/galdergcupv/WikiInfoSeguridad>

No he usado plantillas ni copiado código directamente, pero si que he seguido [tutoriales](#), implementando las modificaciones necesarias y ajustándolos a mi caso, para la creación del diseño de la página web.

Auditoría Sistema Web

Una vez creada la web, se nos pide realizar dos auditorías web usando ZAP (herramienta de pentesting) y solucionar las vulnerabilidades del informe OWASP 2021. Primero realizaremos una auditoría de la primera versión de la web, luego actualizaremos la web para solucionar las vulnerabilidades, y volveremos a realizar otra auditoría sobre esta segunda versión de la web.

*De momento no es necesario registrarse para crear artículos.

Página principal

La página principal de la web está compuesta por una **barra de navegación** en la parte superior (ver pág. 6), el **cuerpo**, y un **pie de página** (ver pág. 7). La barra de navegación y el pie de página se tratarán en sus respectivos apartados.

En el cuerpo de la página principal tenemos: A la derecha una imagen, y a la izquierda un texto acompañado con un botón que nos llevará a la pantalla de registro / inicio de sesión.

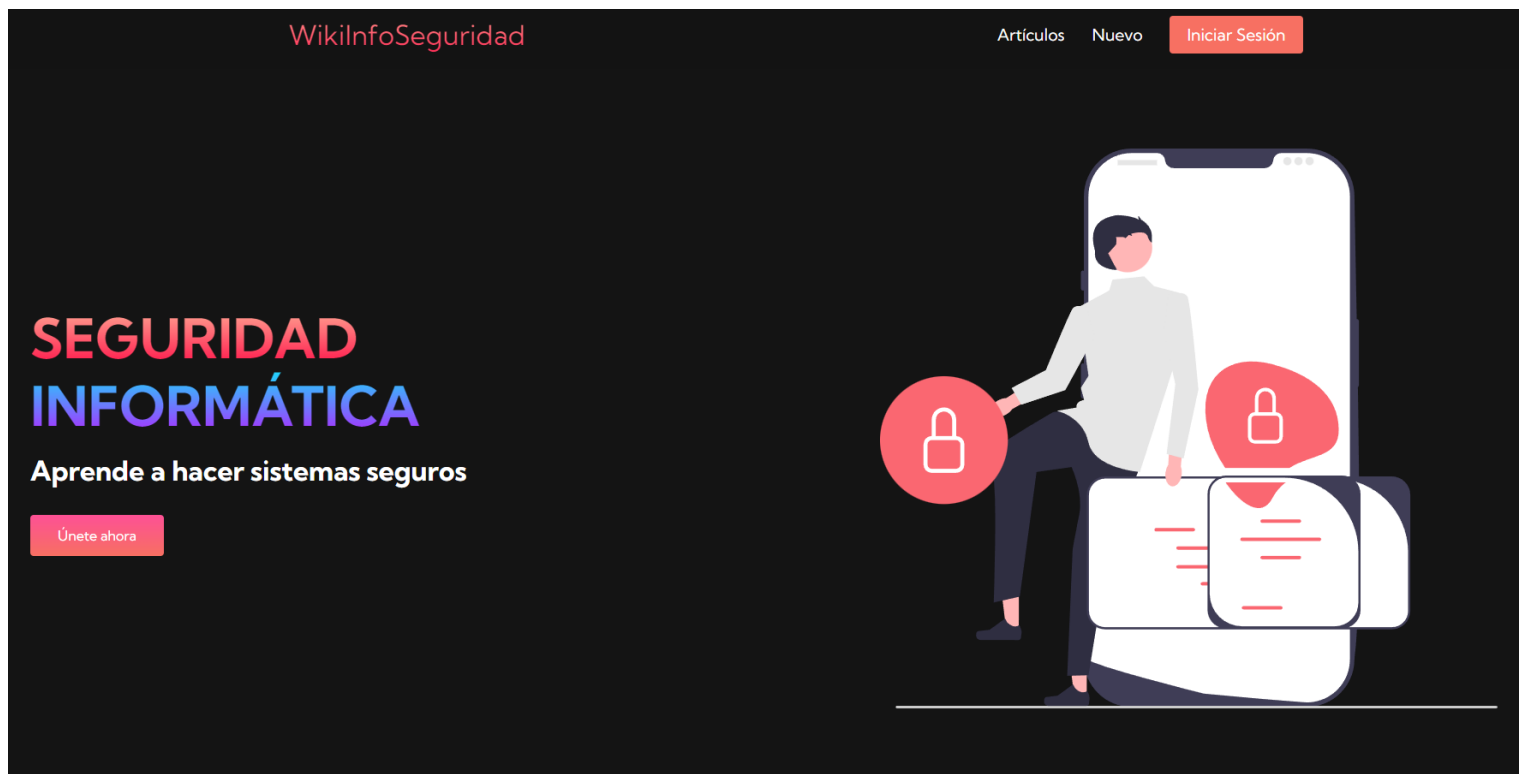


Imagen 1. Página principal (arriba)

Si bajamos en la página principal encontramos dos imágenes interactivas con texto y botones que nos llevan a la página del **visor de artículos** y de **creación de artículos**.



Imagen 2. Página principal (abajo)

Barra de navegación

En la parte superior de todas las páginas se encuentra una barra de navegación que nos permite acceder rápidamente a las pantallas: **Principal** (haciendo click en el nombre de la web a la izquierda), **visor de artículos**, **creador de artículos** e **inicio de sesión / registro**.

WikiInfoSeguridad

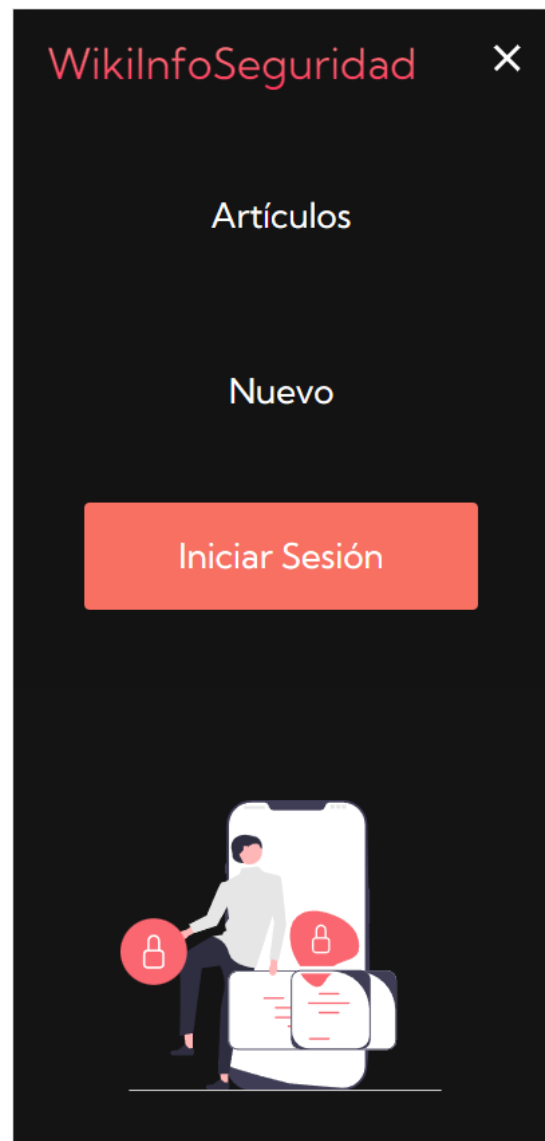
Artículos

Nuevo

Iniciar Sesión

Imagen 3. Barra de navegación (modo normal)

La barra de navegación se ajusta si la ventana es muy estrecha (por ejemplo en un dispositivo móvil), mostrando un boton que al pulsarlo desplegará un menú vertical.



Imágenes 4 y 5. Barra de navegación móvil sin menú (izq.) y con menú (der.)

Pie de página

En la parte inferior de todas las páginas se encuentra un pie de página donde podemos encontrar información adicional. De momento esta sección no es operativa, todos los enlaces redirigen a la página principal.

También encontramos el nombre de la web (al pinchar en él nos redirige a la página principal) y un aviso con el copyright de la web.

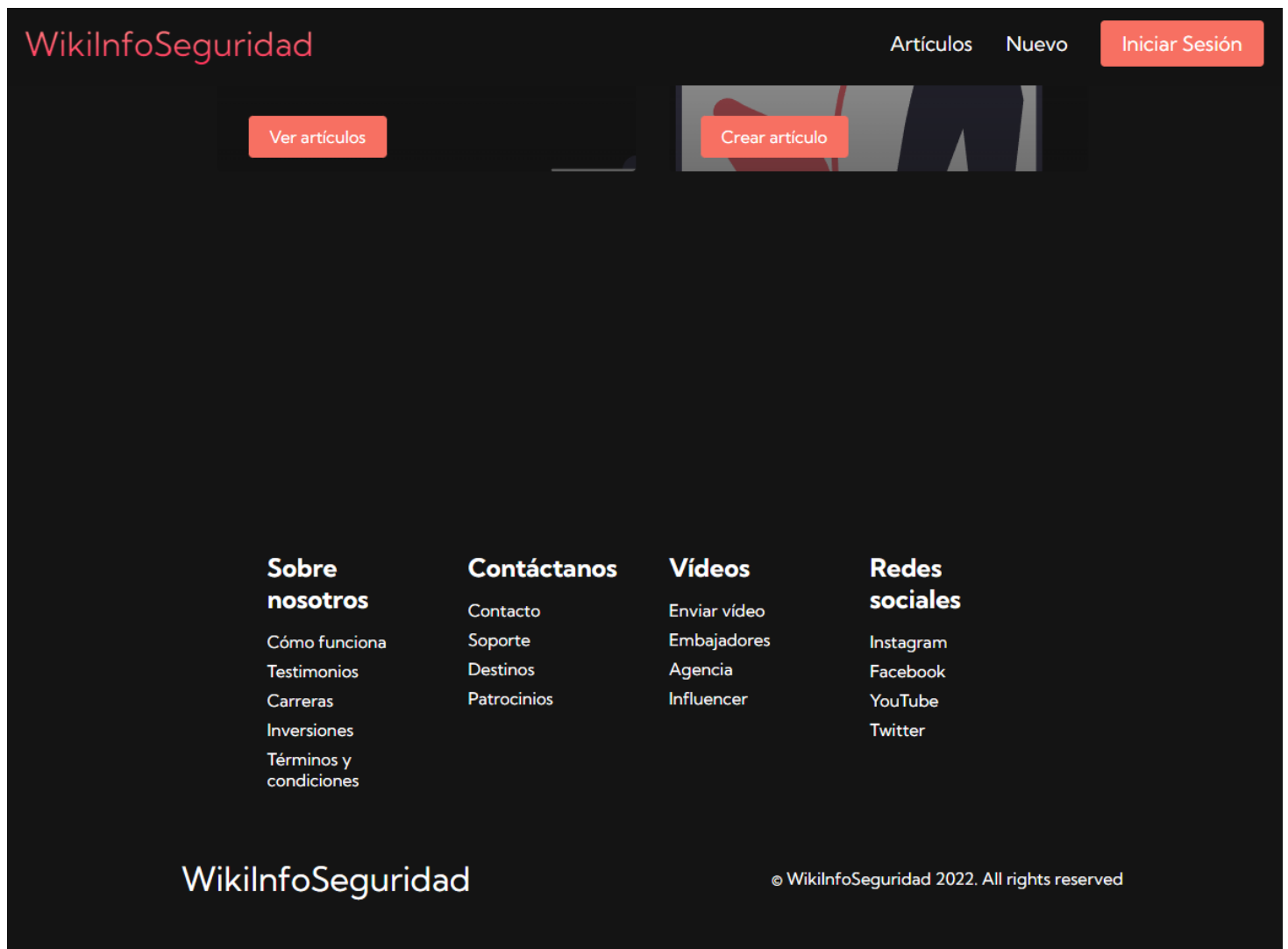
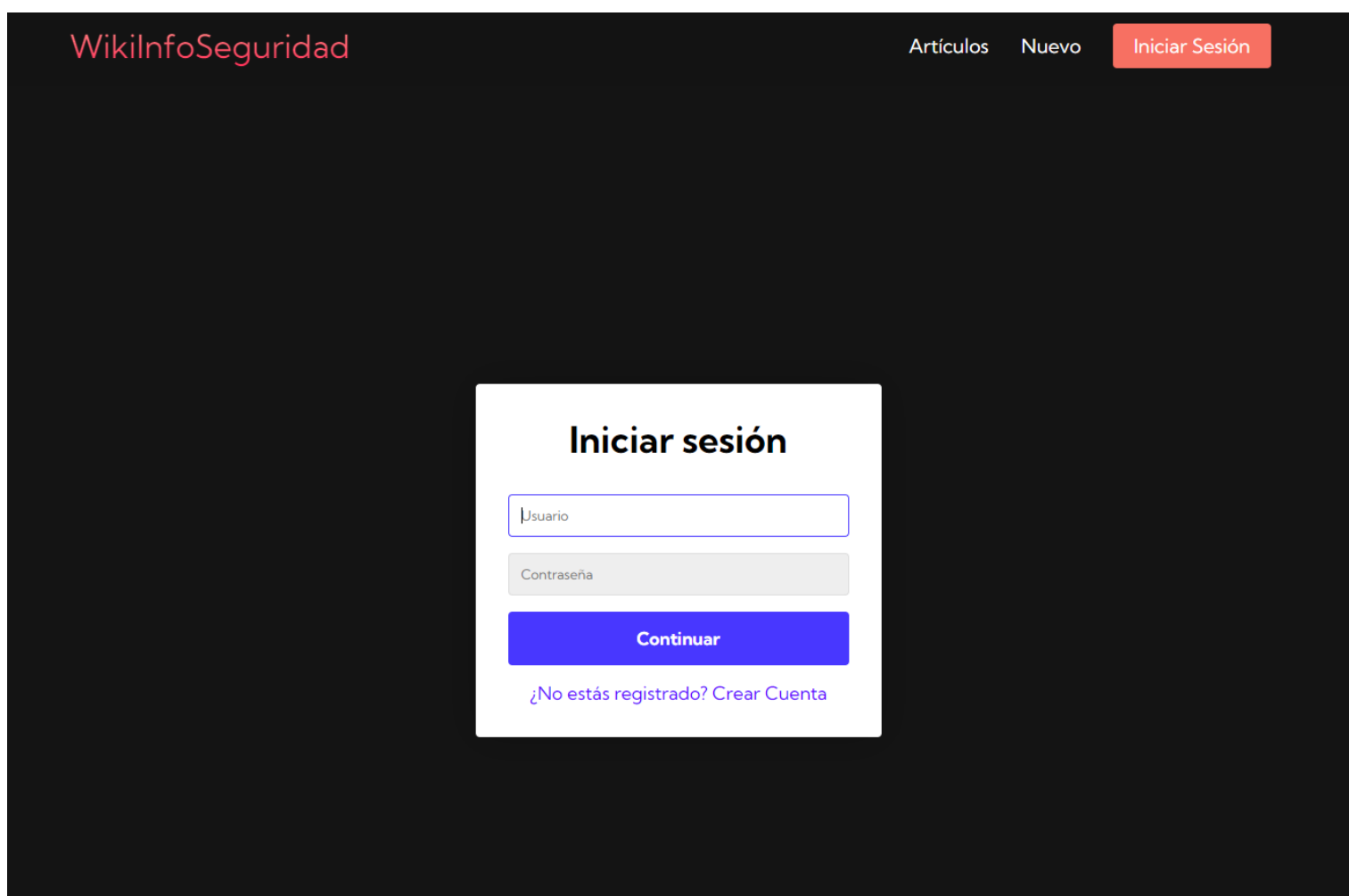


Imagen 6. Pie de página

Inicio de sesión / Registro

En la página de inicio de sesión tenemos un formulario para introducir el usuario y contraseña (**Imagen 7**). También hay un enlace para que si no estamos registrados accedamos al formulario de registro (**Imagen 8**). En el formulario de registro introduciremos nuestros datos y se comprobará si son correctos (El nombre y los apellidos son texto, la fecha es válida, el DNI tiene la letra correspondiente, etc...), si no son correctos aparecerá un texto indicando qué debemos modificar hasta que lo corrijamos.

Una vez registrado podremos iniciar sesión, al hacerlo nos llevará a una página en la que veremos nuestros datos personales (**Imagen 9**).



WikInfoSeguridad

Artículos Nuevo Iniciar Sesión

Iniciar sesión

Usuario

Contraseña

Continuar

[¿No estás registrado? Crear Cuenta](#)

Imagen 7. Inicio de sesión

WikiInfoSeguridad

ArtículosNuevoIniciar Sesión

Crear Cuenta

Nombre

Apellidos

DNI (12345678-A)

Teléfono (9 dígitos)

Fecha de nacimiento (aaaa-mm-dd)

Email (ejemplo@servidor.extensión)

Usuario

Contraseña

Continuar

¿Ya tienes una cuenta? [Iniciar Sesión](#)

Imagen 8. Registro

WikiInfoSeguridad

ArtículosNuevoIniciar Sesión

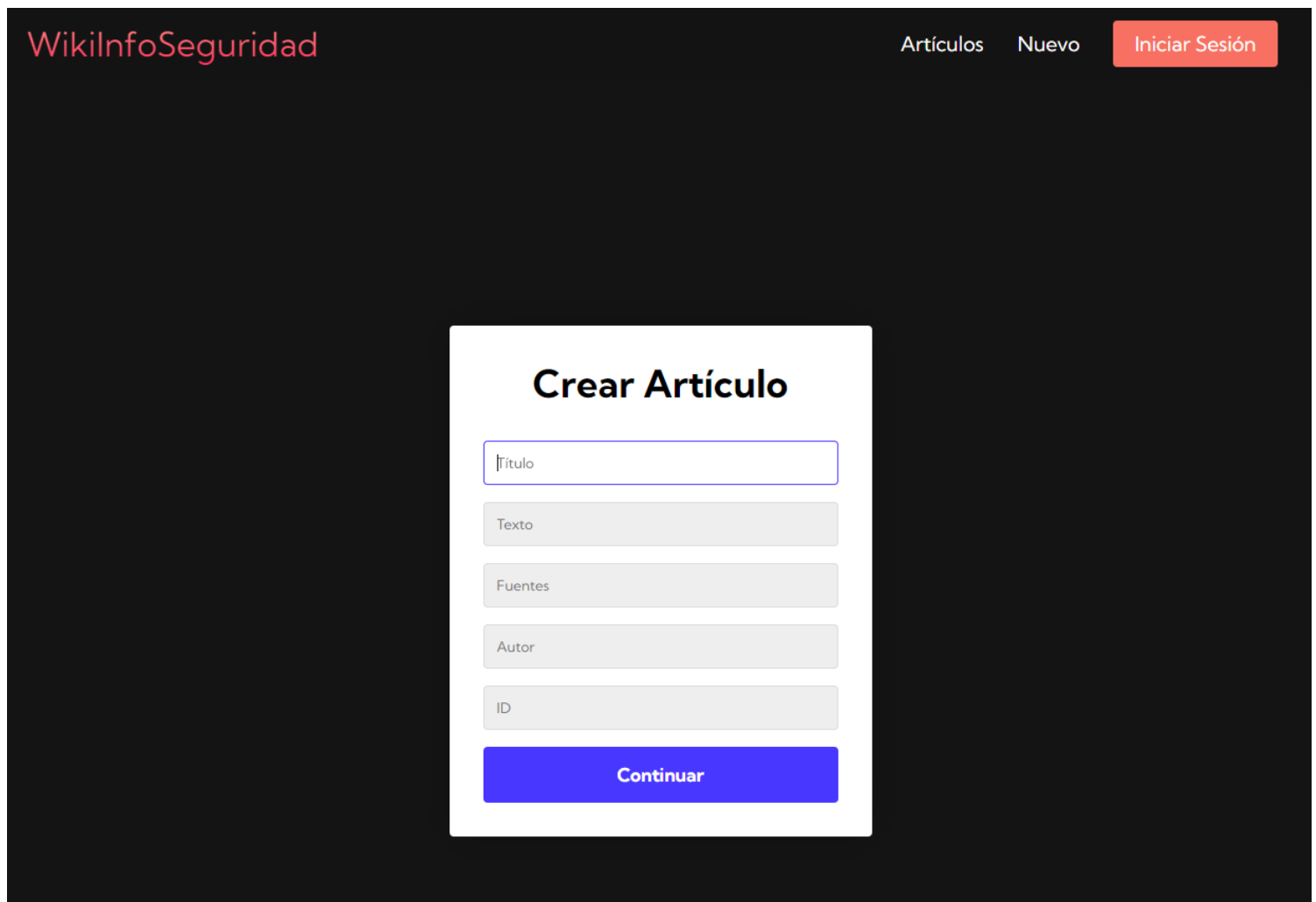
Sesión iniciada correctamente...

Nombre	Apellidos	DNI	Fecha de nacimiento	Telefono	Email	Usuario
Galder	Garcia	79138114-J	2001-01-11	634430400	galdergcupv@gmail.com	Galder

Imagen 9. Datos usuario

Creador de artículos

En la página de creación de artículos tenemos un formulario en el que podemos introducir la información que tendrá el nuevo artículo, al pulsar continuar se creará el artículo en la base de datos y nos redirigirá al **visor de artículos**.



The screenshot shows a web application interface with a dark background. In the top left corner, the logo 'WikilInfoSeguridad' is displayed in red. In the top right corner, there are two links: 'Artículos' and 'Nuevo', followed by a red button labeled 'Iniciar Sesión'. Centered on the screen is a white card titled 'Crear Artículo'. Inside this card, there is a form with five input fields: 'Título' (with a cursor), 'Texto', 'Fuentes', 'Autor', and 'ID'. Below these fields is a large blue button labeled 'Continuar'.

Imagen 10. Creador de artículos

Visor de artículos

En la página del visor de artículos podemos ver todos los artículos que tenemos en nuestro sistema, leerlos y conocer sus fuentes y autor.

WikiInfoSeguridad			Artículos	Nuevo	Iniciar Sesión
ID	Título	Texto	Fuentes	Autor	
1	Algoritmos resumen	Generan un criptograma que representa el contenido original. De tamaño constante, independientemente del contenido original. Representa todo el contenido original. Si el contenido cambia lo más mínimo cambia completamente. Para el mismo contenido, siempre genera el mismo.	https://github.com/mikel-egana-aranguren/EHU-SGSSI-01/tree/main/Cifrado_resumen	Galder	
2	Cifrado simétrico	En el cifrado simétrico se utilizan sistemas de clave privada. Los objetivos son: Convertir el mensaje en ininteligible. Recuperar la información cifrada. Implementación lo más sencilla posible; Se basan en técnicas de criptografía clásica: Transposición (los caracteres originales simplemente cambian de posición). Sustitución (los caracteres originales se sustituyen por otros).	https://github.com/mikel-egana-aranguren/EHU-SGSSI-01/tree/main/Cifrado_simetrico	Galder	
3	Cifrado asimétrico	En el cifrado asimétrico se utilizan sistemas de clave pública. Hay dos claves por usuario: La clave pública que conoce todo el mundo. La clave privada que sólo conoce el usuario; Están relacionadas matemáticamente, lo que una clave cifra sólo lo puede descifrar la otra.	https://github.com/mikel-egana-aranguren/EHU-SGSSI-01/tree/main/Cifrado_asimetrico	Galder	
4	Firma digital	Podemos firmar un documento cifrándolo con nuestra clave privada. Como solo se puede descifrar con nuestra clave pública garantizamos que lo hemos firmado nosotros. Con esto se consigue: Sólo el usuario legítimo puede firmar su documento. Nadie podrá falsificar una firma. Cualquiera puede verificar una firma digital. No se puede reutilizar una firma. No se puede modificar una firma. No se puede negar haber firmado un documento. No se puede alterar un documento después de haberlo firmado; Logramos: Autenticidad, Integridad y No repudio.	https://github.com/mikel-egana-aranguren/EHU-SGSSI-01/tree/main/Cifrado_firma	Galder	
5	Certificados digitales	Una entidad (Autoridad de Certificación) certifica que el usuario/entidad (su clave pública) es quien dice ser (Depende de la confianza en la AC que lo certifica) y almacena las claves públicas por nosotros. La AC emite un certificado digital. En el certificado digital el CA firma mediante su clave privada la clave pública de un usuario/entidad. Esto se encadena y se crea una jerarquía de ACs. Una AC raíz certifica otras de ACs que certifican usuarios/entidades.	https://github.com/mikel-egana-aranguren/EHU-SGSSI-01/tree/main/Cifrado_certificados	Galder	
6	Comunicaciones seguras	Se usa Transport Layer Security (TLS). Comienzo TLS: El cliente le pide al servidor usar TLS. HTTP: cambiar de puerto 80 a 443. Email: comando STARTTLS; TLS hand-shake: El cliente presenta al servidor una lista de algoritmos de cifrado soportados (simétricos, asimétricos, resumen). El servidor elige de esa lista los que soporta. El servidor presenta un certificado al cliente; el cliente valida el certificado (con un CA). El cliente genera una clave de sesión (Cifrado simétrico): El cliente genera un número aleatorio, lo cifra con la clave pública del servidor y se lo envía. En el cliente y el servidor generan una clave compartida a partir de ese número. Usando el algoritmo Diffie-Hellman, se genera una clave secreta compartida; Si el hand-shake ha sido exitoso se establece la conexión propiamente dicha: Los datos transmitidos se cifran con la clave de sesión y su integridad se verifica con los algoritmos resumen consensuados. Es un conxexión que mantiene el estado (stateful).	https://github.com/mikel-egana-aranguren/EHU-SGSSI-01/tree/main/Cifrado_comunicaciones	Galder	
7	Bitcoin	Bitcoin es un sistema de dinero digital basado en una red a la que cualquiera puede unirse a través de un nodo, y no gobernada por bancos ni gobiernos. Protocolo: Bitcoin (con B) Moneda: bitcoin (con b). Símbolo: BTC o XTC. Satoshi: 0.00000001 BTC. Bitcoin asegura: No repudio: no se puede* deshacer una transacción. Integridad: no se puede* modificar la historia del blockchain. Autenticidad. Pseudo-anonimato. *Es computacionalmente y socialmente muy caro e improbable	https://github.com/mikel-egana-aranguren/EHU-SGSSI-01/tree/main/Cifrado_bitcoin	Galder	
Sobre nosotros		Contáctanos	Videos	Redes sociales	
Cómo funciona		Contacto	Enviar vídeo	Instagram	
Testimonios		Soporte	Embajadores	Facebook	
Carreras		Destinos	Agencia	YouTube	
Inversiones		Patrocinios	Influencer	Twitter	
Términos y condiciones					
WikiInfoSeguridad			© WikiInfoSeguridad 2022. All rights reserved		

Imagen 11. Visor de artículos

Pentesting al sistema inicial

Para comprobar la seguridad de la web vamos a realizar un ataque controlado a nuestro sistema usando la herramienta de pentesting **OWASP ZAP**. Para esto necesitamos acceder a la web oficial de la aplicación zapproxy.org y descargar el instalador para Linux. Una vez instalada la aplicación **OWASP ZAP** la ejecutamos, elegimos que no queremos guardar la sesión actual y empezamos. En la pestaña “Quick Start” elegimos “Automated Scan” para hacer un ataque automatico a nuestro sistema. Comprobamos que la web se está ejecutando correctamente (si no la ejecutamos con *docker-compose up* en la carpeta donde tengamos el contenido) y copiamos la dirección URL (en nuestro caso <http://localhost:81/index.html>). Pegamos la dirección URL en **OWASP ZAP** e iniciamos el ataque (**Imagen 12**).

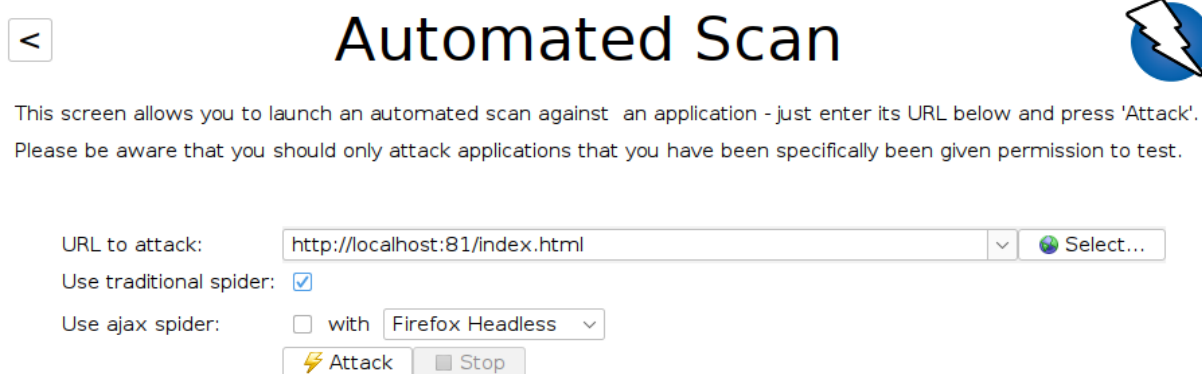


Imagen 12. Iniciar ataque

Esperamos a que finalice el ataque y en la pestaña inferior “Alerts” aparecerán todas las vulnerabilidades que se han encontrado y el nivel de riesgo que tiene cada una. En rojo las de riesgo alto, naranja las de riesgo medio, amarillo las de riesgo bajo y azul la información que ha conseguido sobre nuestro sistema web.

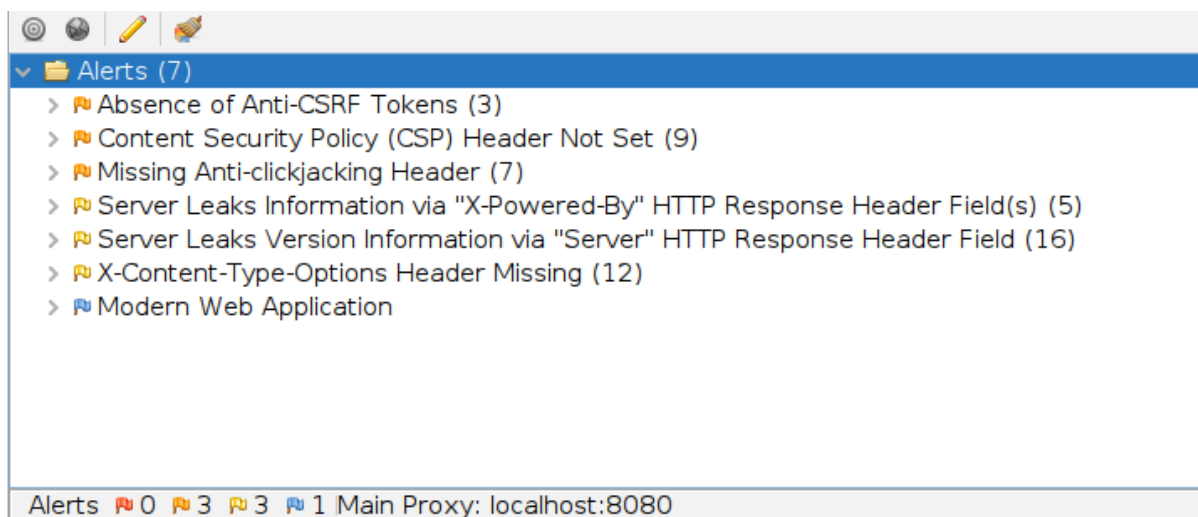


Imagen 13. Resultado del ataque

En este caso ha encontrado 3 vulnerabilidades de riesgo medio, 3 de riesgo bajo, y ha determinado que el sistema es una “aplicación web moderna”. A continuación vamos a solucionar una serie de vulnerabilidades generales y veremos como afecta al analisis de pentesting de **OWASP ZAP**.

Fallos criptográficos

Los **fallos criptográficos** son fallos en los métodos criptográficos que resultan en la exposición de datos confidenciales.

En el caso de nuestra web almacenamos las contraseñas en texto plano, esto hace que sea vulnerable ya que si la base de datos quedase comprometida y se filtraran los datos, los atacantes tendrían acceso a todas las contraseñas de nuestros usuarios.

Para solucionar esto, en vez de guardar las contraseñas en texto plano, guardaremos el hash que produce la contraseña. Cuando el usuario introduzca su contraseña para iniciar sesión compararemos el hash de la contraseña introducida con el hash de la contraseña que tenemos almacenado en la base de datos y, si son iguales, daremos la autenticación por válida.

Con esto solucionamos el que obtengan las contraseñas de los usuarios si acceden a la información de la base de datos, ya que solo obtendrán los hashes de las contraseñas. Pero como para cada contraseña se genera siempre el mismo hash, si varios usuarios usan la misma contraseña también tendrán el mismo hash almacenado. El atacante puede usar esto y comparar el porcentaje de repeticiones de cada hash con los porcentajes de uso de las contraseñas más comunes para “adivinar” qué contraseñas se han usado. También podría prepararse una lista de las contraseñas más comunes hasheadas con la misma función que usamos y buscar en nuestra base de datos hashes que coincidan para obtener las contraseñas.

Por esto, a la hora de crear el hash, añadimos una “sal” (caracteres adicionales) a la contraseña del usuario y aplicamos el hash a todo. En la base de datos guardamos el hash y la sal. Cuando queramos comprobar si la contraseña proporcionada es correcta, también le añadimos la sal que tenemos guardada antes de hacer el hash y, si coincide con el hash guardado, entonces será la misma contraseña.

Para implementar esto en la web hacemos uso las funciones predefinidas en php `password_hash($Password, PASSWORD_BCRYPT)`, y `password_verify($Password, $data['Password'])`. Que, con estos parámetros, usará el algoritmo BCRYPT para crear un hash seguro de la contraseña. También se encargará de generar la sal y guardarla en el hash generado (**Imagen 14**). De esta forma solo necesitamos una única columna para guardar el hash y la sal en la base de datos.

\$2y\$10\$6z7GKa9kpDN7KC3ICW1Hi.f00/to7Y/x36WUKNP0IndHdkdR9Ae3K

— Salt

— Hashed password

— Algorithm options (eg cost)

— Algorithm

Imagen 14. Formato del hash generado

Inyección

Un ataque de **inyección SQL** consiste en aprovechar el formato de las consultas SQL y añadir código en los parámetros de los formularios para que el sistema, al convertirlos en consultas SQL, ejecute este código directamente en la base de datos. Por ejemplo, si el nombre del usuario introducido es: "**Roberto'); DROP TABLE users; --**" al hacer la consulta SQL `insert into users(Nombre) values('nombre');`, tendremos:

`insert into users(Nombre) values('Roberto'); DROP TABLE users; --`". Esto ejecutaría en la base de datos `DROP TABLE users` sin nuestro permiso y borraría la tabla users.

En nuestra web evitamos esto usando consultas parametrizadas al acceder a la base de datos (**Imagen 15**), por lo que no tenemos esta vulnerabilidad.

```
$stmt = $conn->prepare("insert into users(Nombre, Apellidos, DNI, Telefono, Fecha, Email, Usuario, Password) values(?, ?, ?, ?, ?, ?, ?, ?)");  
$stmt->bind_param("sssissss", $Nombre, $Apellidos, $DNI, $Telefono, $Fecha, $Email, $Usuario, $HashedPassword);  
$stmt->execute();
```

Imagen 15. Consulta parametrizada

Rotura de control de acceso

Una **rotura de control de acceso** sucede cuando se viola el principio de “denegación por defecto”, es decir, todos los usuarios tienen acceso a todo por defecto excepto unos pocos roles (pueden ser la mayoría de usuarios aunque sean pocos roles) a los que decidimos quitar el acceso a funciones concretas.

La manera correcta sería denegar el acceso por defecto a todas las funciones no públicas a todos los usuarios, y solo permitir el acceso a las funciones concretas a los roles que las necesitan (por ejemplo usuarios registrados a la información de usuario, administradores a la configuración de la web, etc...).

En nuestro caso esto se cumple ya que la mayoría de funciones son públicas (no es necesario registrarse ni iniciar sesión), y la única función que requiere identificación es la de ver los datos del usuario, que siempre requiere que el usuario en cuestión inicie sesión para poder mostrarle sus datos.

Diseño inseguro

Las vulnerabilidades que se producen en consuencia del diseño del sistema y no necesariamente de una mala implementación es lo que llamamos **diseño inseguro**.

En nuestro caso la web permite realizar un número ilimitado de intentos de inicio de sesión y, aunque el protocolo que usemos para comprobar ese inicio de sesión sea seguro, se podrían crear bots que prueben muchas combinaciones de usuarios y contraseñas comunes hasta que den con una válida y consigan acceder.

Para prevenir esta situación hemos implementado un sistema que tiene en cuenta el número de intentos fallidos. Si se producen 3 o más intentos fallidos seguidos, se bloquea el acceso y el usuario tendrá que esperar 30 segundos antes de volver a intentar iniciar sesión.

Con esto hacemos que sea más difícil el uso de bots que prueben muchas combinaciones en poco tiempo para vulnerar la seguridad de nuestro sistema.

Configuración de seguridad insuficiente

Podemos tener una **configuración de seguridad insuficiente** si usamos permisos inadecuados, tenemos activas funciones que no necesitamos, o si usamos cuentas y contraseñas por defecto.

En el caso de nuestra web no usamos permisos inadecuados ni tenemos funciones innecesarias, pero si usamos el usuario y contraseña por defecto de MySQL (phpmyadmin) "admin" y "test".

Por eso hemos cambiado la contraseña por una generada aleatoriamente, la nueva contraseña es la indicada en las instrucciones de uso en el archivo **README.md** del repositorio de GitHub de esta entrega.

Componentes vulnerables y obsoletos

Son **componentes vulnerables y obsoletos** aquellos componentes que no tienen soporte, se conocen vulnerabilidades no parcheadas, no tienen los últimos parches de seguridad aplicados, o no se conoce qué versión exacta se está usando (siempre se usa “latest”).

En nuestra web todos los componentes que usamos tienen soporte y están actualizados, por lo que no tenemos vulnerabilidades por esta parte. Pero la imagen que usamos en docker de phpmyadmin usa la versión “latest”, esto significa que si saliese una nueva versión de phpmyadmin con nuevas vulnerabilidades también “infectaría” nuestro sistema.

Para solucionar esto cambiamos el docker-compose para que use la versión 5.2.0 (la última) de la imagen phpmyadmin. Ahora sabemos exactamente qué versión se está usando en todo momento.

Si queremos seguir usando la web de forma segura tendremos que realizar escáneres de seguridad y aplicar actualizaciones seguras de forma regular, basandonos en el riesgo.

Fallos de identificación y autenticación

Puede haber **fallos de identificación y autenticación** si no usamos métodos seguros para confirmar la identidad de los usuarios y gestionar las sesiones.

Para prevenir esto: Restringimos el acceso de funciones privadas a usuarios identificados, bloqueamos el inicio de sesión si han ocurrido muchos intentos fallidos seguidos, almacenamos las contraseñas con hashes potentes y sal, almacenamos logs de los intentos de inicio de sesión fallidos, y no hacemos despliegues con credenciales por defecto.

Como se describe en los apartados **rotura de control de acceso**, **diseño inseguro**, **fallos criptográficos**, **fallos en la monitorización de la seguridad**, y **configuración de seguridad insuficiente** de esta documentación.

Fallos en la integridad de datos y software

Los **fallos en la integridad de datos y software** ocurren cuando usamos librerías, plugins, o módulos obtenidos de fuentes sin confianza; cuando se usa un pipeline de integración continua sin comprobaciones de seguridad adecuadas; o cuando se incluyen funciones de auto-actualización sin los debidos tests de integridad.

En nuestro caso solo usamos fuentes fiables para los módulos como phpmyadmin, docker y apache. No usamos pipelines de integración continua ni funciones de auto-actualización, por lo que no estamos expuestos a estas vulnerabilidades.

Fallos en la monitorización de la seguridad

Tendremos **fallos en la monitorización de la seguridad** si no monitorizamos adecuadamente nuestro sistema creando logs con los eventos auditables.

En el caso de nuestra web no guardamos ningún tipo de registro o log de los intentos de inicio de sesión fallidos. Tener un log con los intentos de inicio de sesión fallidos nos ayudará a detectar comportamientos sospechosos al hacer auditorías de seguridad. Si tuviesemos un administrador del sistema podríamos hacer que se le mande un sms/email cuando se produzca una cantidad inusual de intentos fallidos.

Por eso hemos hecho que cada vez que se produzca un intento de inicio de sesión fallido se guarde el usuario introducido, la fecha y la hora en un archivo log.txt. El archivo log.txt se crea en el directorio /tmp del container de docker que estamos utilizando para la web. Si queremos ver el contenido de log.txt podemos copiarlo a nuestro ordenador usando el siguiente comando:

```
docker cp wikiinfoseguridad-entrega_2_web_1:/tmp/log.txt ~/Desktop/log.txt
```

Esto copia el archivo log.txt del container en el archivo log.txt de nuestro escritorio, podemos cambiar la ruta de destino si queremos guardarlo en otra ubicación.

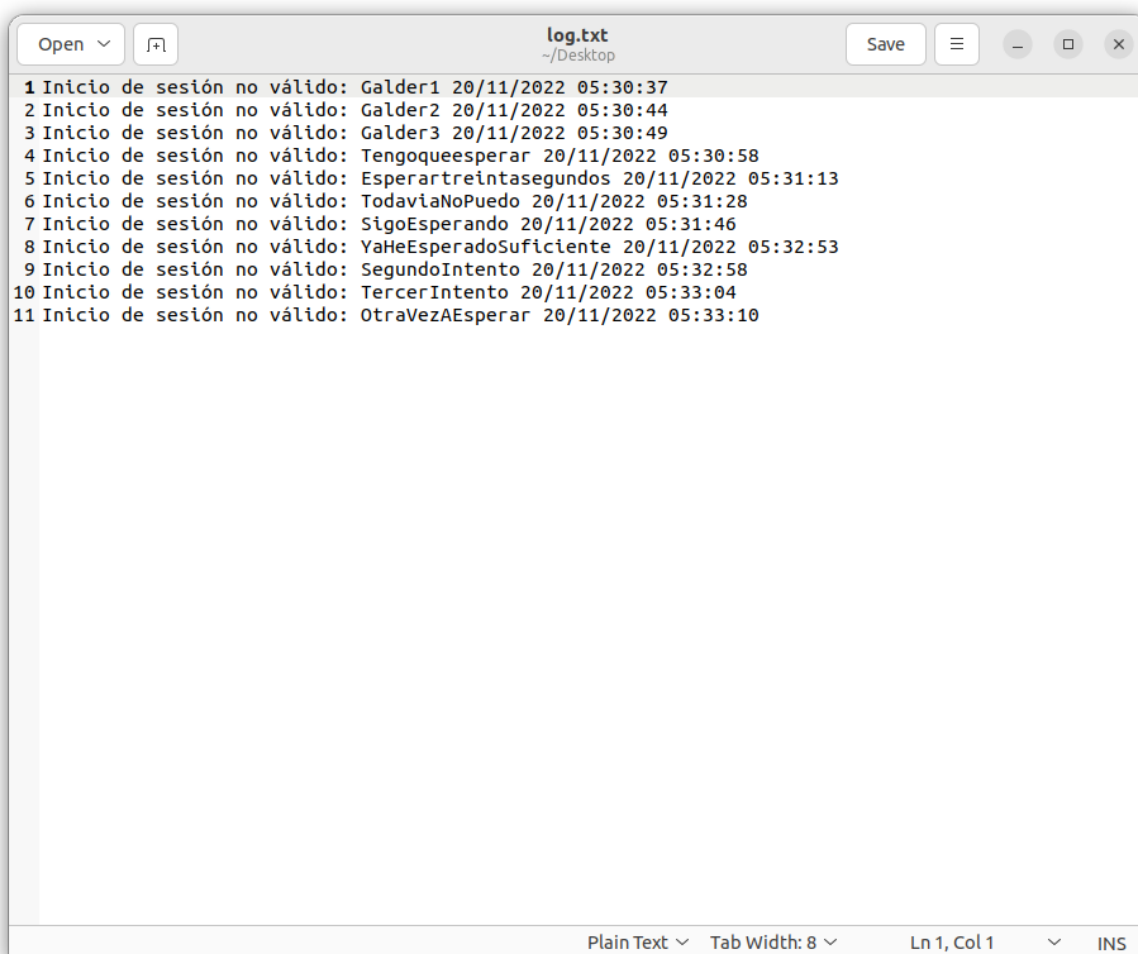


Imagen 16. Ejemplo del contenido de log.txt

Pentesting al sistema final

Una vez solucionadas las vulnerabilidades generales volvemos a hacer un ataque controlado a nuestro sistema usando la herramienta de pentesting *OWASP ZAP* y comparamos los resultados con el ataque inicial.

Al finalizar el ataque al sistema final obtenemos en alertas las vulnerabilidades que se han encontrado (**Imagen 17**). Aunque no se hayan detectado vulnerabilidades de alto riesgo, han aparecido nuevas vulnerabilidades de riesgo bajo. Esto significa que a pesar de haber solucionado vulnerabilidades generales y hacer que nuestro sistema sea más seguro en general, *OWASP ZAP* nos avisa de que podemos hacerlo aún más seguro si solucionamos las vulnerabilidades de riesgo bajo y medio que todavía tenemos.

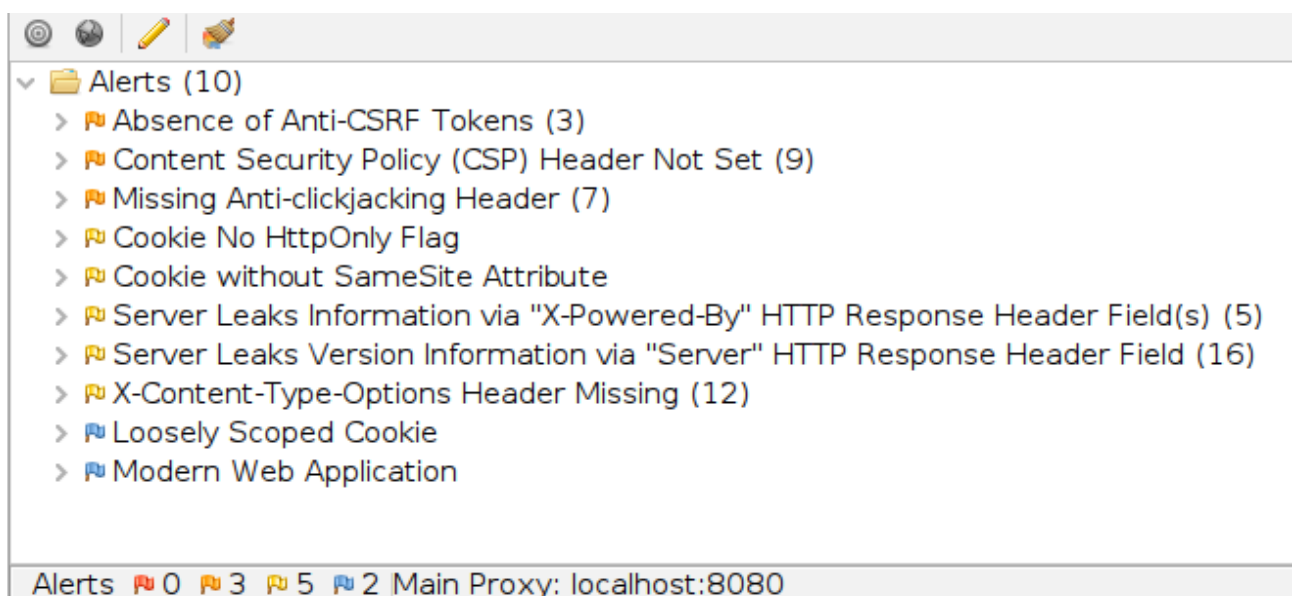


Imagen 17. Resultado Final

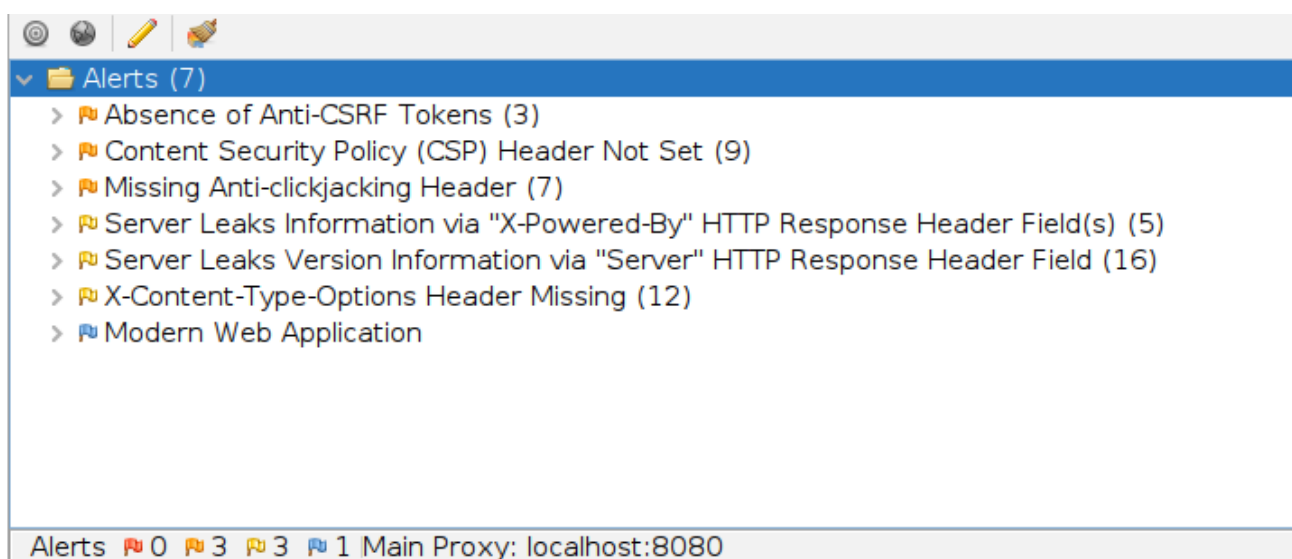


Imagen 18. Resultado Inicial