Estudiante: Brayan Gabriel Gutierrez Rebolledo

Fecha: 18 enero 2026

Módulo: Introducción a la Ciberseguridad

1. ÁMBITO Y ALCANCE

Aplicación: WebGoat 8.1.0 URL: http://127.0.0.1:8080/WebGoat

Objetivo: Realizar auditoría básica siguiendo guía práctica

2. INFORME EJECUTIVO

2.1 Resumen

Se realizó una auditoría web básica que incluyó reconocimiento, identificación de tecnologías y explotación de vulnerabilidades guiadas.

2.2 Vulnerabilidades destacadas

- **SQL Injection**
- **Cross-Site Scripting**
- **Security Misconfiguration**
- **Vulnerable Components**
- **Authentication Failures**

2.3 Conclusiones

Esta práctica permitió entender de forma práctica cómo vulnerabilidades comunes en aplicaciones web pueden ser explotadas cuando no existen validaciones ni configuraciones de seguridad adecuadas.
A través de WebGoat se pudo observar que fallos como inyecciones, configuraciones inseguras y contraseñas débiles pueden derivar en accesos no autorizados y exposición de información.
La realización de los ejercicios permitió comprender la importancia de aplicar buenas prácticas de desarrollo y seguridad desde etapas tempranas.

2.4 Recomendaciones

Se recomienda validar correctamente las entradas del usuario, utilizar consultas preparadas, mantener los componentes actualizados y exigir contraseñas seguras, con el fin de reducir los riesgos de seguridad en aplicaciones web.

3. PROCESO AUDITORIA

3.a Reconocimiento

Puertos Abiertos

Comando ejecutado: **nmap 127.0.0.1**

Resultado:

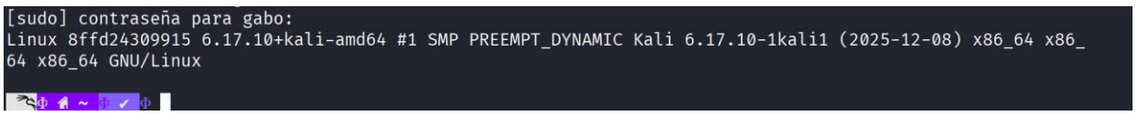
Análisis:

- Puerto 8080: WebGoat (HTTP)
- Puerto 3000: Otra aplicación

Sistema Operativo

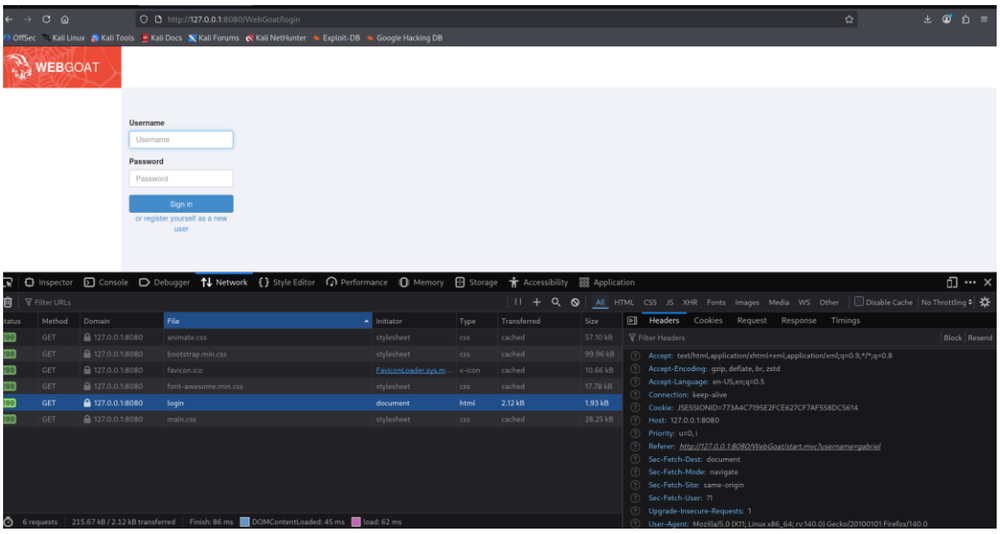
Comando: docker exec webgoat uname -a

Resultado:



Sistema Linux (contenedor Docker)

Lenguajes de Programación Identificados:



En backend se observa que es java, debido al apartado Cookie: JSESSIONID=773A4C7195E2FCE627CF7AF558DC5614

- Backend: Java (por cookie JSESSIONID)
- Frontend: HTML, CSS, JavaScript
- Base de datos: H2 (común en WebGoat)

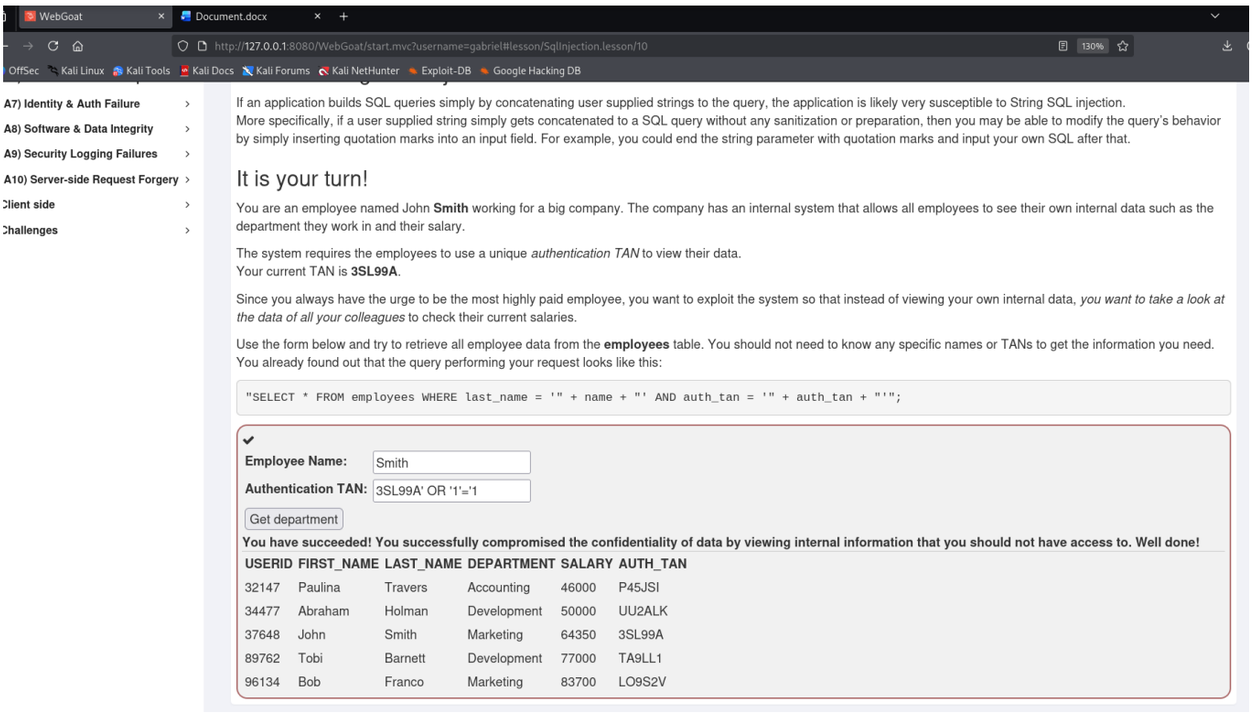
En la etapa de reconocimiento se observó que la aplicación WebGoat funciona como un entorno web de pruebas accesible desde el navegador, en el cual los datos ingresados por el usuario a través de formularios son procesados por el sistema y utilizados para consultar información almacenada en una base de datos de empleados.

3.b Explotación

A3 Injection - SQL Injection (intro) - Apartado 11

Se realiza una inyeccion SQL a traves del TAN del empleado utilizando de referencia su TAN asignado y manipulando el input del mismo debido a sus fallas de interpretación usando comillas que permiten que el input pase a formar parte de la consulta SQL en este caso: auth_tan de esta forma:
Empleado: John Smith
TAN : 3SL99A

Entonces, en la sección *Authentication TAN* se modifica la lógica de autenticación colocando el valor 3SL99A' OR '1'='1, logrando que el filtro de empleados no muestre únicamente los datos del usuario autenticado, sino que permita el acceso a los datos de todos los empleados.
Esto se consigue mediante el uso de comillas para cerrar el string original, provocando que el input del usuario pase a ser interpretado como parte de la consulta SQL.

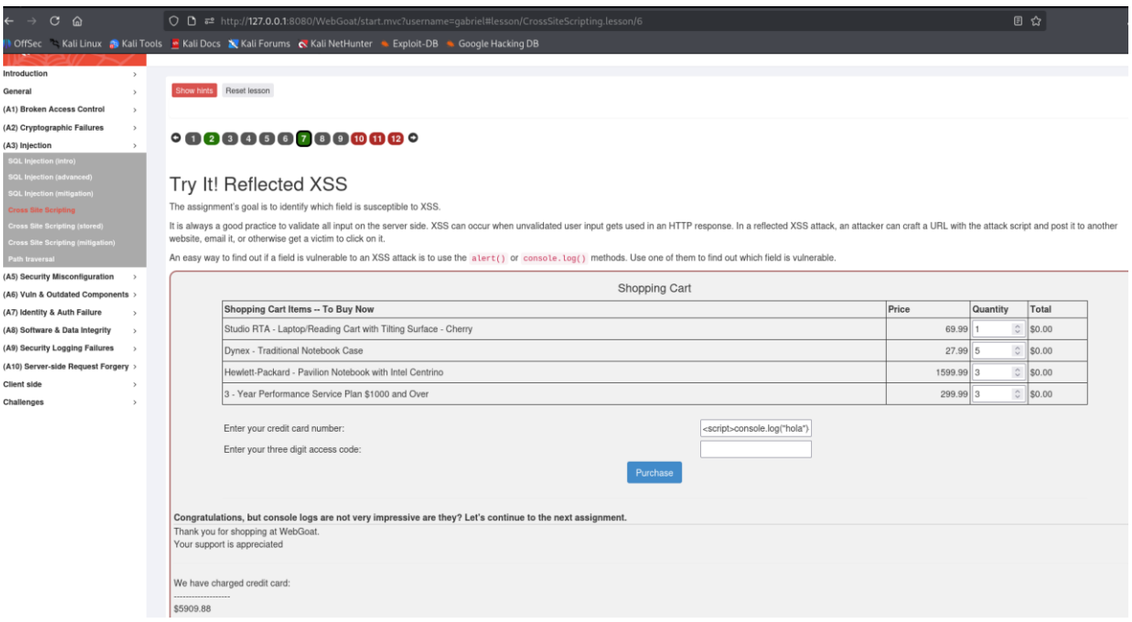


A3 Injection - Cross Site Scripting - Apartado - Apartado 7

Se analizó la vulnerabilidad Cross-Site Scripting (XSS) reflejado con el objetivo de identificar un campo de entrada vulnerable.

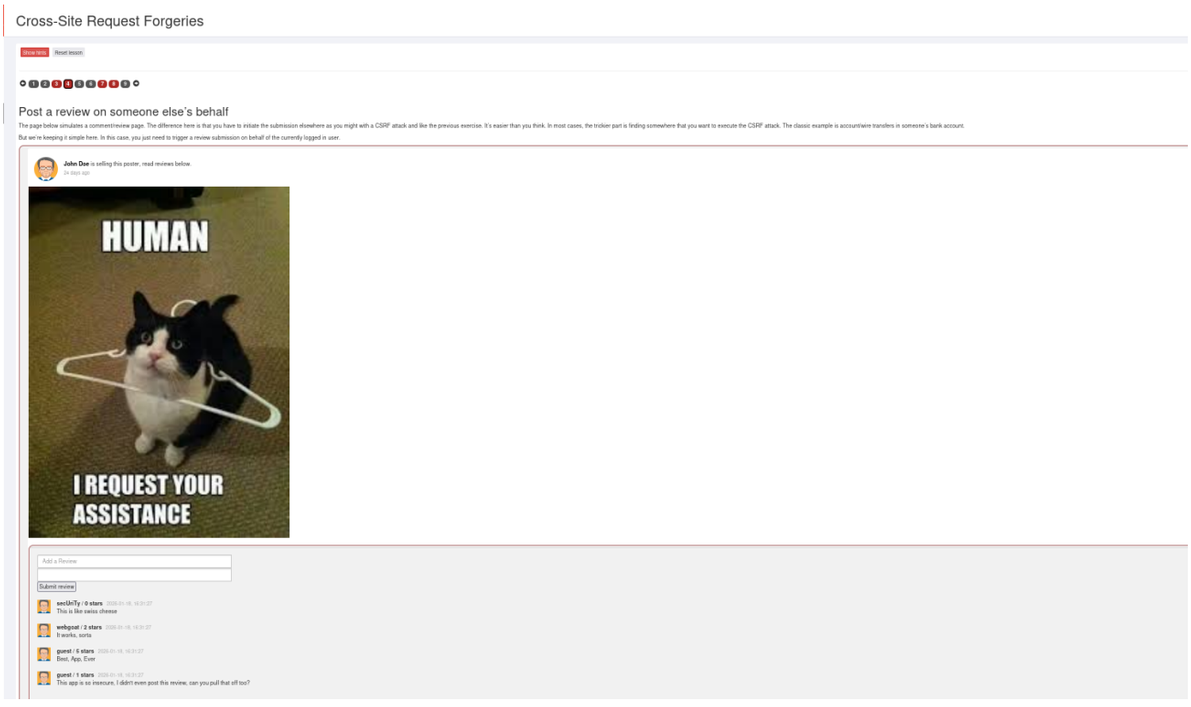
Durante las pruebas se ingresó el payload: <script>alert("XSS Test")</script> en los campos del formulario.

Al enviar el formulario, el navegador ejecutó el código y mostró una alerta, confirmando que la entrada del usuario se refleja sin sanitización y es vulnerable a XSS reflejado.



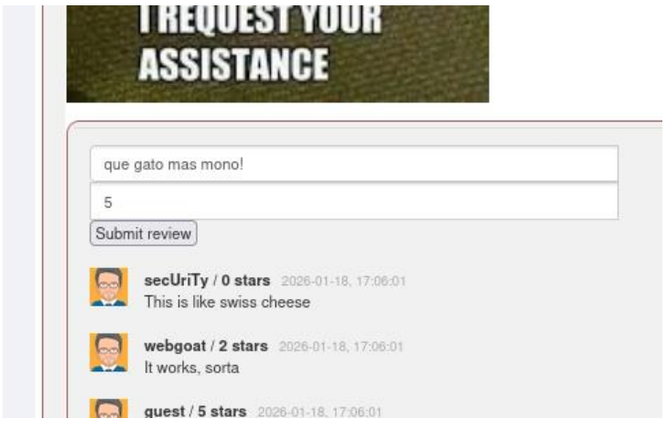
A5 Security Misconfiguration - Apartado 4

En este apartado se observó que la aplicación presenta una mala configuración de seguridad, ya que confía excesivamente en el estado de la sesión del usuario sin aplicar controles adicionales. Esto demuestra cómo una configuración insegura puede facilitar comportamientos no deseados y representar un riesgo, incluso sin necesidad de una explotación avanzada.

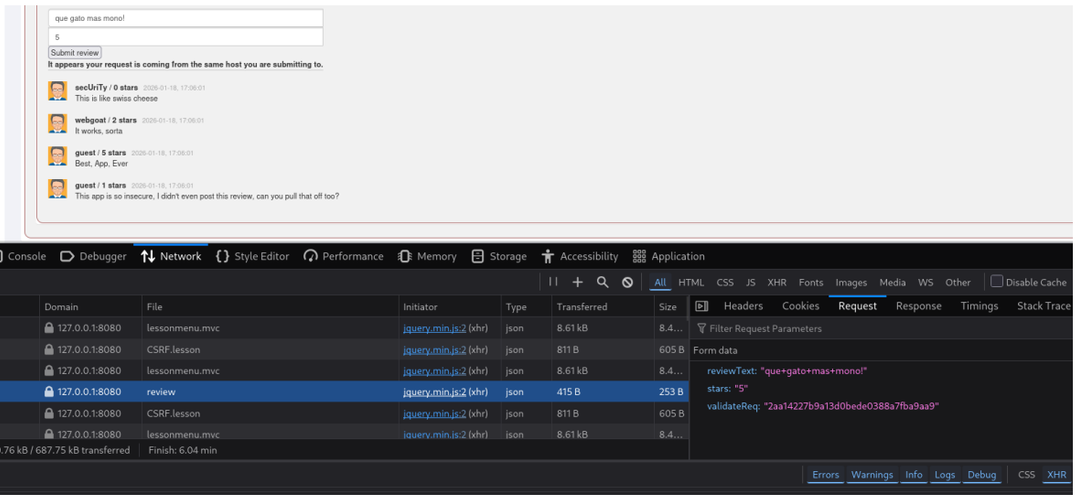


En este ejercicio se solicita enviar una reseña utilizando la sesión del usuario que se encuentra autenticado. Para observar cómo se realiza esta acción, se abrieron las herramientas de desarrollador con la tecla **F12** y se accedió a la pestaña **Network**.

Luego, se escribió el texto **“que gato más mono!”** en el campo de la reseña y se ingresó el valor **5** en el campo correspondiente a las estrellas. Posteriormente, se envió la reseña para analizar la solicitud generada por la aplicación.



Posteriormente, se revisó el contenido de esta petición en el panel derecho, específicamente en **Request → Form Data**, donde se pudo visualizar la información enviada por la aplicación para publicar la reseña.



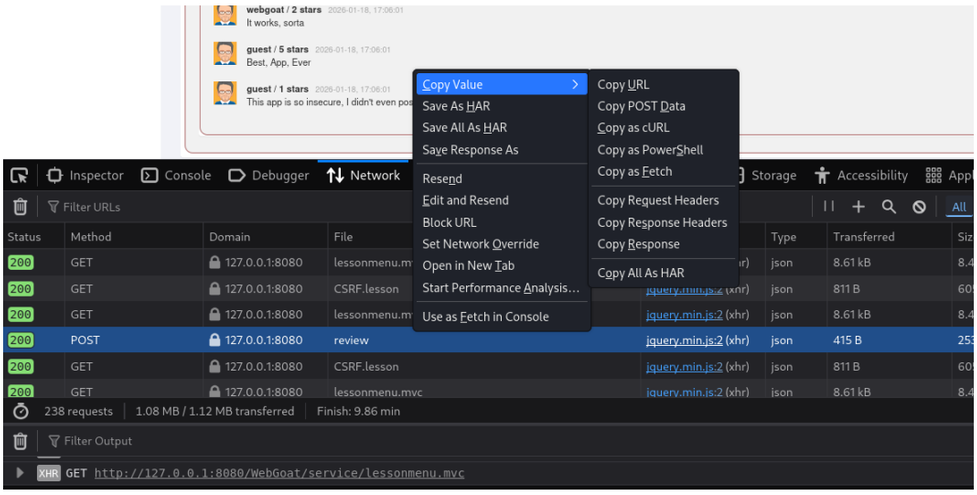
A partir de esta información, se obtuvo el código JavaScript que realiza la acción de enviar la reseña mediante una petición **POST**. En dicho código se identificó el parámetro **body**, el cual contiene los datos enviados, como el texto de la reseña, la cantidad de estrellas y el valor **validateReq**.

Form data

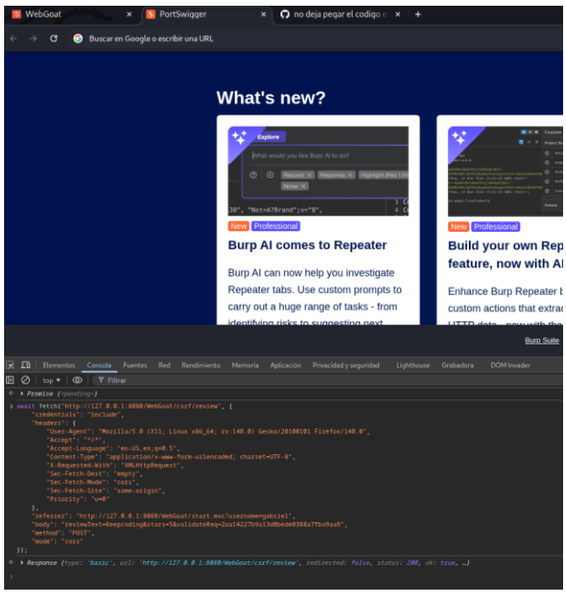
reviewText: “que gato mas mono”

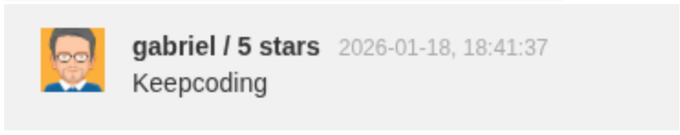
stars: “5”

Ahora, se da click derecho en review →Copy Value → Copy as Fetch



El codigo copiado se coloca en la consola, pero desde una pagina externa que no sea en la que estamos, abrimos nueva pagina y pegamos el codigo:





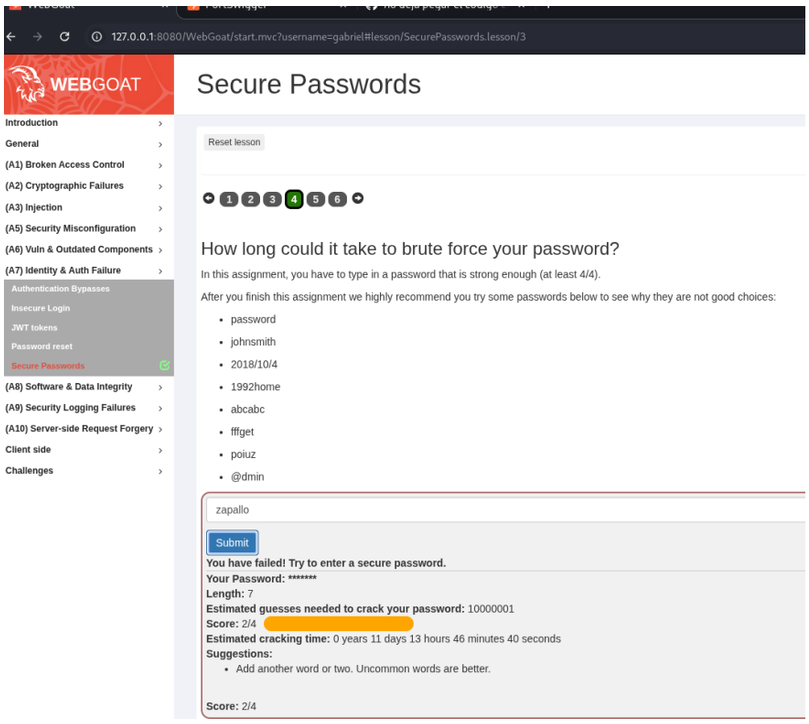
Inicialmente se intentó realizar el ejercicio utilizando Firefox, pero se presentaron errores que impedían su correcta ejecución.
Posteriormente, se cambió al navegador Chromium, donde el ejercicio funcionó correctamente. Se analizó el comportamiento de la aplicación, observando una mala configuración de seguridad al confiar en la sesión del usuario sin validaciones adicionales.

A6 Vuln & outdated Components - Apartado 5

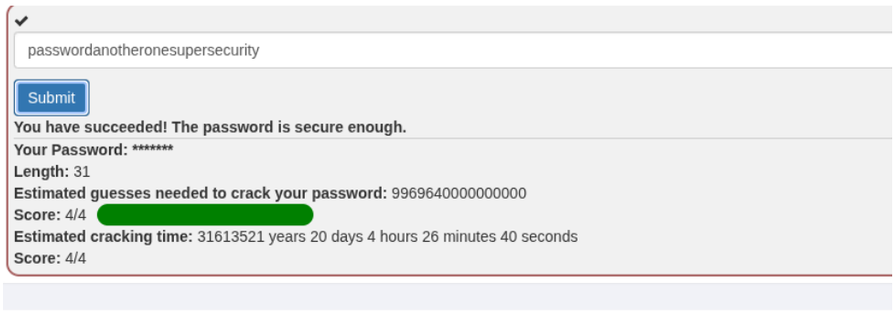
A7 Identity & Auth Failure - Secure Passwords Apartado 4

Este apartado muestra que la seguridad de una contraseña depende en gran parte de su longitud. La aplicación indica cuánto tiempo tomaría crackearla, dejando en evidencia que las contraseñas cortas o simples pueden romperse en muy poco tiempo.

En la primera imagen se muestra una contrasena muy debil: “zapallo” con longitud 7



En la segunda imagen se puede ver como es su seguridad con longitud 31



3.c Post-explotación

- **A3 Injection - SQL Injection (intro) - Apartado 11**
  Posible acceso no autorizado a información sensible y exposición de los datos.

- **A3 Injection - Cross Site Scripting - Apartado - Apartado 7**
  Una vez explotada la vulnerabilidad, el código inyectado se ejecuta con los permisos del navegador del usuario afectado, esto puede permitir el acceso a información sensible y comprometer la confidencialidad e integridad de la sesión del usuario sin que este lo note.
  -Robo de cookies (CRITICO)

- **A5 Security Misconfiguration - Apartado 4**
  - **Publicación de contenido no autorizado**
  - **Posible escalación si la funcionalidad fuera crítica (transferencias, cambios de configuración)**

- **A6 Vuln & outdated Components - Apartado 5**
  El apartado A6 no pudo ser ejecutado correctamente debido a un error en el funcionamiento del ejercicio dentro de la plataforma WebGoat. A pesar de los intentos realizados, el comportamiento esperado no se presentó, lo que impidió obtener evidencias claras para su documentación.
  Debido a la limitación de tiempo y al correcto desarrollo de los demás apartados solicitados, se decidió continuar con el resto de la práctica.

- **A7 Identity & Auth Failure - Secure Passwords Apartado 4**
  Debido al uso de contraseñas débiles y de poca longitud, un atacante podría comprometer una cuenta en poco tiempo mediante técnicas de fuerza bruta o diccionario. Esto permitiría el acceso no autorizado a cuentas de usuario sin necesidad de conocimientos avanzados.

3.d Posibles mitigaciones

- **A3 Injection - SQL Injection (intro) - Apartado 11**
  Aplicar buenas prácticas de desarrollo seguro, validando correctamente las entradas del usuario y evitando la concatenación directa de datos en consultas SQL, mediante el uso de consultas preparadas.

- **A3 Injection - Cross Site Scripting - Apartado - Apartado 7**
  Se debe validar y sanitizar la entrada del usuario y evitar devolver directamente contenido ingresado sin filtrarlo.
  Además, es importante escapar caracteres especiales para prevenir la ejecución de código JavaScript malicioso.

- **A5 Security Misconfiguration - Apartado 4**
  1) Poner un Código Secreto en Cada Formulario
  - **Como un codigo de verificacion que cambia cada vez**
  - **El servidor revisa que el codigo sea correcto**

- **Si no coincide, rechaza la peticion**

2) Configurar las Cookies bien
Las cookies deben tener:

- **SameSIte=Strict (No se envian desde otras paginas)**
- **Secure(Solo por HTTPS)**
- **HttpOnly(No accesible por javascript)**

- **A7 Identity & Auth Failure - Secure Passwords Apartado 4**
  Exigir contraseñas más largas y complejas para reducir el riesgo de compromiso de cuentas.

3.e Herramientas utilizadas
Docker, nmap, navegador web(Firefox, Chromium), terminal.