

Informe de reconocimiento pasivo de una Aplicación Web – OWASP Juice-Shop

Título del informe: Informe de Reconocimiento Pasivo – OWASP Juice-Shop web app	
Portafolio: Lección 1: Identificar visualmente amenazas y vulnerabilidades en una aplicación web	Entorno: Imagen de OWASP Juice-Shop levantada con Docker Compose.
Clasificación del documento: Estudio de información pública - libre acceso.	Modulo: Análisis de Amenazas y Vulnerabilidades en Aplicaciones Web
Autor(es): Sebastián Hernández Téllez	Fecha de elaboración: 08 de agosto de 2025

1. Resumen Ejecutivo

El presente informe documenta un análisis de reconocimiento pasivo sobre la aplicación vulnerable OWASP Juice Shop, desplegada en un entorno controlado mediante Docker. El objetivo fue identificar amenazas visibles sin recurrir a técnicas de explotación activa.

Se detectaron vulnerabilidades críticas como almacenamiento inseguro de tokens en localStorage, falta de expiración de sesiones, exposición de información sensible en cabeceras HTTP y ausencia de mecanismos de protección ante ataques de fuerza bruta.

Estos hallazgos evidencian que la aplicación presenta una superficie de ataque considerable incluso bajo una revisión pasiva. La adopción de buenas prácticas como gestión segura de sesiones, uso de cookies protegidas, sanitización de entradas y controles de acceso más robustos permitiría reducir de manera significativa el riesgo de accesos indebidos y compromisos de información.

Principales hallazgos:

Amenazas Internas Detectadas

- Exposición de Tokens de Autenticación en localStorage
- Falta de Expiración o Rotación de Tokens
- Información sensible expuesta en cabeceras HTTP (Server, X-Powered-By)

Amenazas Externas Detectadas

- Potencial de XSS mediante acceso a localStorage
 - Exposición a ataques de fuerza bruta en login (no se evidencia límite de intentos)
-

2. Alcance de la Evaluación

Tipo de aplicación:

E-commerce simulado vulnerable (formularios login, comentarios y buscador)

Entorno evaluado:

OWASP Juice-Shop

Restricciones o limitaciones:

- El laboratorio contempla solo una exploración pasiva del sitio con un intento de acceso al formulario login con credenciales utilizadas habitualmente por defecto para observación.
-

3. Metodología

Técnicas aplicadas:

Luego de levantado el entorno de forma local y segura, se realiza una exploración técnica en modo desarrollador.

Se inicia el modo desarrollador del navegador (F12 o clic derecho → “Inspeccionar”) y navega por la aplicación como un usuario regular.

Se observan y documentan:

- Formularios disponibles (login, contacto, búsqueda)
- URLs y parámetros visibles
- Comportamiento ante errores
- Consola y pestaña de red
- Cookies almacenadas
- Códigos de estado HTTP (pestaña "Red")

Luego se realiza un acceso según credenciales por defecto (Email: admin@juice-sh.op Contraseña: admin123 → Resultado: acceso exitoso.) con el fin de analizar respuestas de la aplicación, el intento da como resultando en un acceso no autorizado.

Herramientas utilizadas:

- Docker desktop
 - Visual studio code
 - Mozilla firefox
 - OWASP Juice-Shop
-

4. Hallazgos de Seguridad

1. Exposición de Tokens de Autenticación en localStorage

- **Riesgo:** Almacenar tokens en localStorage los deja expuestos a ataques XSS.
- **Mejora:** Migrar a cookies seguras (HttpOnly y Secure), idealmente en una sesión de backend administrada.
- **Estándares:** OWASP recomienda evitar localStorage para datos sensibles; NIST recomienda controles de sesión seguros.

2. Falta de Expiración o Rotación de Tokens

- **Riesgo:** La ausencia de control de expiración permite el uso prolongado de tokens robados.
- **Mejora:** Implementar tiempos de expiración cortos y rotación periódica. Usar JWT con control de revocación.
- **Estándares:** OWASP Top 10 (A05:2021), NIST 800-63 recomienda expiración adaptativa según contexto de riesgo.

3. Información sensible expuesta en cabeceras HTTP (Server, X-Powered-By)

- **Riesgo:** Revelar información sobre el servidor (por ejemplo, Express o Nginx) facilita ataques dirigidos.
- **Mejora:** Ocultar cabeceras innecesarias y aplicar técnicas de "security through obscurity".
- **Estándares:** ISO/IEC 27001 — Control A.13.1.1 (Protección de servicios de red).

4. Potencial de XSS mediante acceso a localStorage

- **Riesgo:** Si el sitio es vulnerable a inyecciones, un atacante podría extraer tokens desde el navegador.
- **Mejora:** Sanitización de inputs, uso de frameworks que escapen contenido por defecto, Content Security Policy (CSP).
- **Estándares:** OWASP Top 10 A03:2021, NIST SP 800-53 SI-10.

5. Exposición a ataques de fuerza bruta en login (no se evidencia límite de intentos)

- **Riesgo:** Permitir múltiples intentos sin restricción facilita el acceso no autorizado.
- **Mejora:** Implementar rate limiting, captchas, bloqueo por IP.
- **Estándares:** NIST 800-63B recomienda monitoreo activo y respuesta ante intentos fallidos reiterados.

5. Evaluación de Riesgos

Matriz de Riesgo

Nº	Tipo	Descripción	Riesgo Potencial	Medida de Mitigación
1	Vulnerabilidad	Token de autenticación expuesto en localStorage	Robo de sesión vía XSS; acceso no autorizado	Usar cookies HttpOnly y Secure; evitar localStorage para datos sensibles
2	Vulnerabilidad	Tokens sin expiración ni rotación	Uso prolongado de tokens robados; persistencia de accesos indebidos	Implementar expiración corta; rotación de tokens; revocación mediante listas negras
3	Vulnerabilidad	Cabeceras HTTP revelan tecnología del servidor (X-Powered-By, Server)	Focalización de ataques dirigidos (exploit específico según tecnología)	Ocultar o eliminar cabeceras innecesarias; reforzar configuración del servidor

Nº	Tipo	Descripción	Riesgo Potencial	Medida de Mitigación
4	Amenaza externa	Posible ataque XSS aprovechando acceso al DOM y localStorage	Robo de credenciales; manipulación del contenido	Sanitizar entradas; implementar Content Security Policy (CSP); usar frameworks seguros
5	Amenaza externa	Formulario de login sin mecanismos de protección ante fuerza bruta	Acceso indebido por prueba sistemática de contraseñas	Implementar rate limiting, captchas y bloqueo de IP tras múltiples intentos fallidos

6. Buenas practicas sugeridas

1. Gestión de contraseñas

- Usar *hashing seguro* (bcrypt, Argon2) en lugar de contraseñas en texto plano.
- Forzar políticas de contraseñas seguras (longitud mínima, complejidad, expiración).

2. Gestión de sesiones

- Invalidar tokens/cookies al cerrar sesión.
- Habilitar SameSite en cookies para mitigar CSRF.
- Evitar el uso de credenciales por defecto en entornos productivos.

3. Seguridad de transporte

- Implementar **HTTPS/TLS** en todos los entornos (incluso de pruebas).
- Usar **HSTS (HTTP Strict Transport Security)**.

4. Cabeceras de seguridad

- Añadir **Content-Security-Policy (CSP)**.
- Usar **X-Frame-Options**, **X-Content-Type-Options**, **Referrer-Policy**.

5. Control de acceso y autenticación

- Implementar MFA (Multi-Factor Authentication).
- Monitorear y registrar intentos de login fallidos.

6. Pruebas automatizadas

- Integrar escaneos con herramientas como **OWASP ZAP** o **Nikto** para complementar el análisis manual.
- Considerar **CI/CD con análisis estático de seguridad (SAST)**.

7. Protección contra fuerza bruta

- Además de *rate limiting* y captchas, sugerir bloqueo progresivo de cuenta o delays exponenciales en respuestas.

8. Gestión de errores

- No exponer mensajes detallados de error al cliente.
- Centralizar logs y tratarlos con un SIEM para detección temprana.

9. Seguridad de la infraestructura

- Mantener Docker y dependencias actualizadas.
- Usar imágenes oficiales verificadas (evitar riesgos de imágenes maliciosas).
- Aislar el contenedor en una red privada, sin exponer innecesariamente puertos.

7. Conclusiones/reflexión final

La aplicación evaluada muestra un nivel de exposición importante frente a vulnerabilidades comunes, como el almacenamiento inseguro de tokens, la ausencia de controles en el login y la filtración de información en cabeceras HTTP.

Me sorprendió comprobar que estos hallazgos pudieron identificarse únicamente con técnicas pasivas y el inspector del navegador, lo que resalta lo visibles que resultan ciertos errores de configuración.

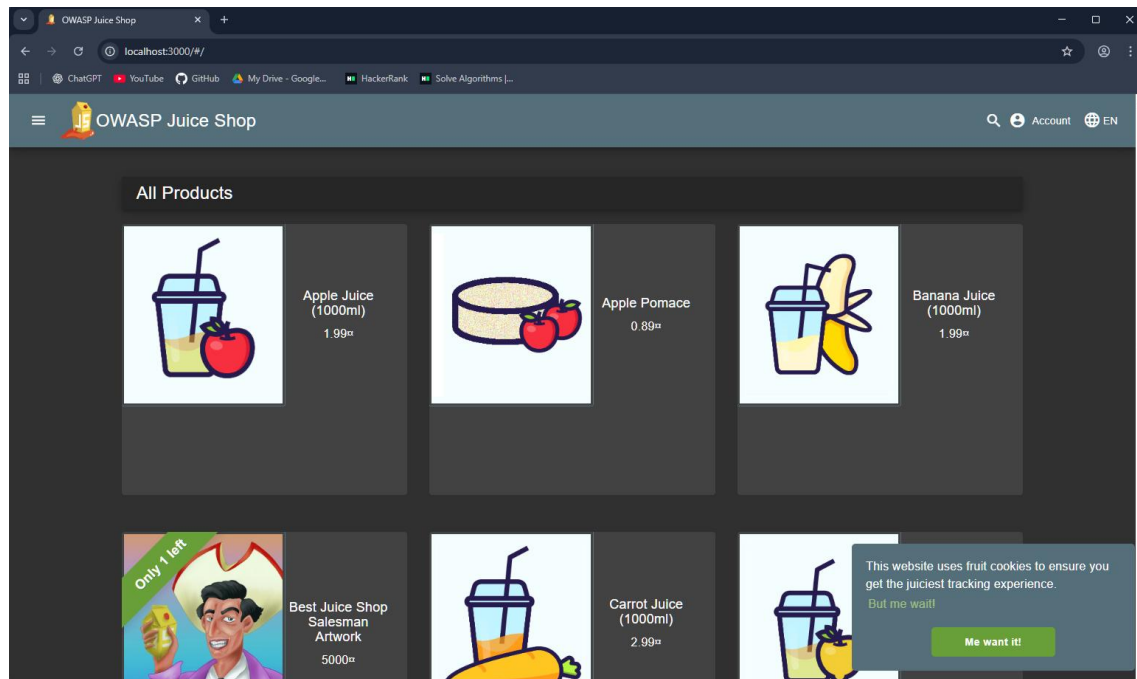
Aplicar las buenas prácticas recomendadas —gestión segura de sesiones, políticas de contraseñas, cabeceras de seguridad y mecanismos contra fuerza bruta— reduciría de forma notable la superficie de ataque.

En conjunto, estas medidas fortalecerían la protección de los datos sensibles, dificultarían los intentos de intrusión y aumentarían la resiliencia de la aplicación frente a amenazas internas y externas.

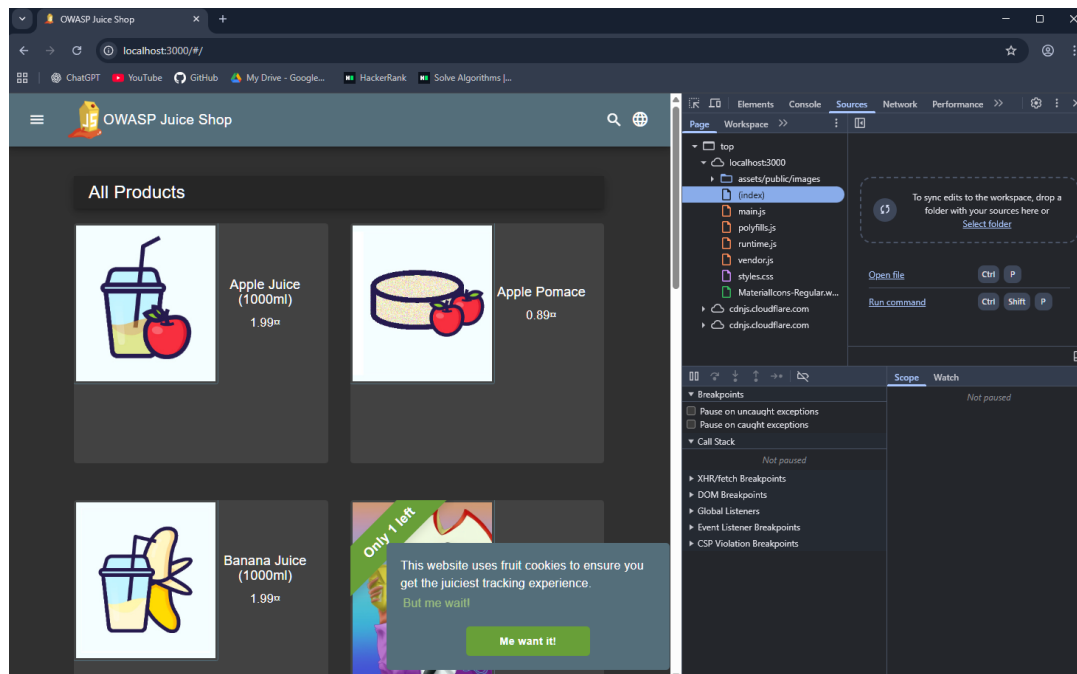
8. Anexos

Anexos – capturas de pantalla utilizadas en análisis

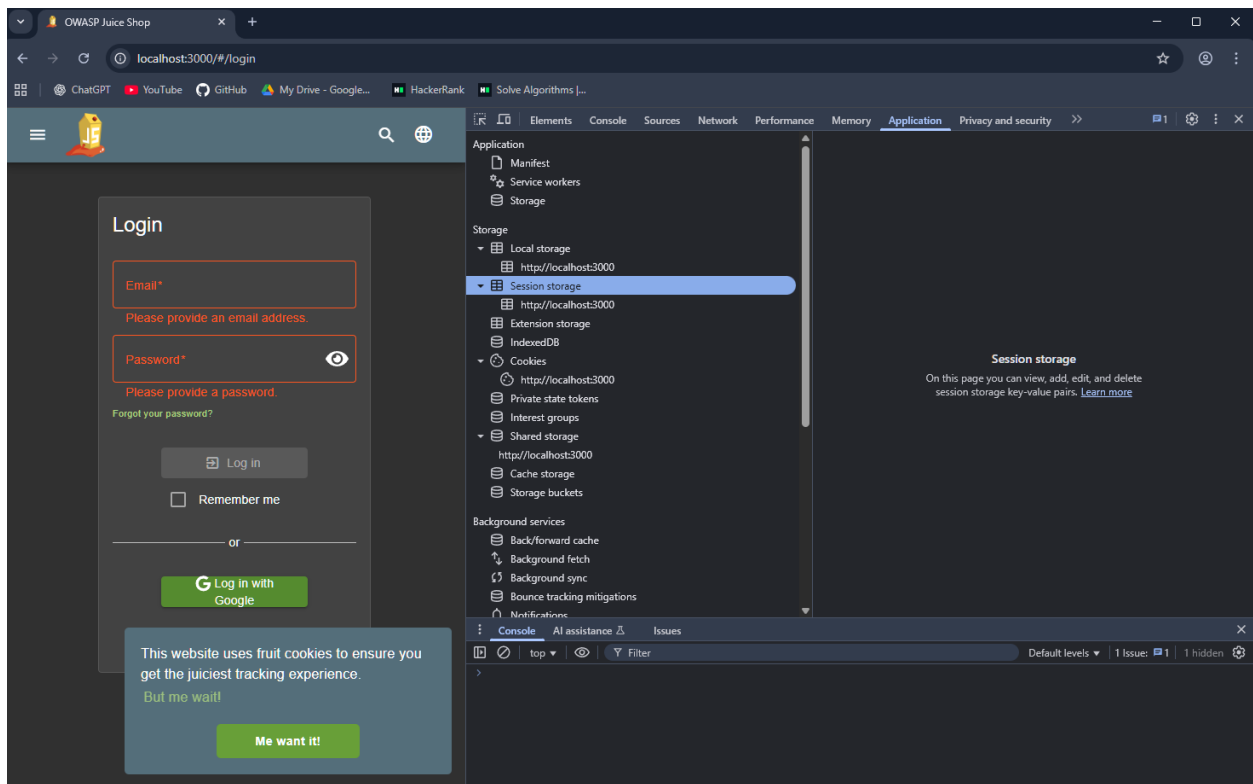
1.-imagen de OWASP Juice-Shop desplegado en el navegador, levantado localmente por Docker Compose a través del puerto 3000.



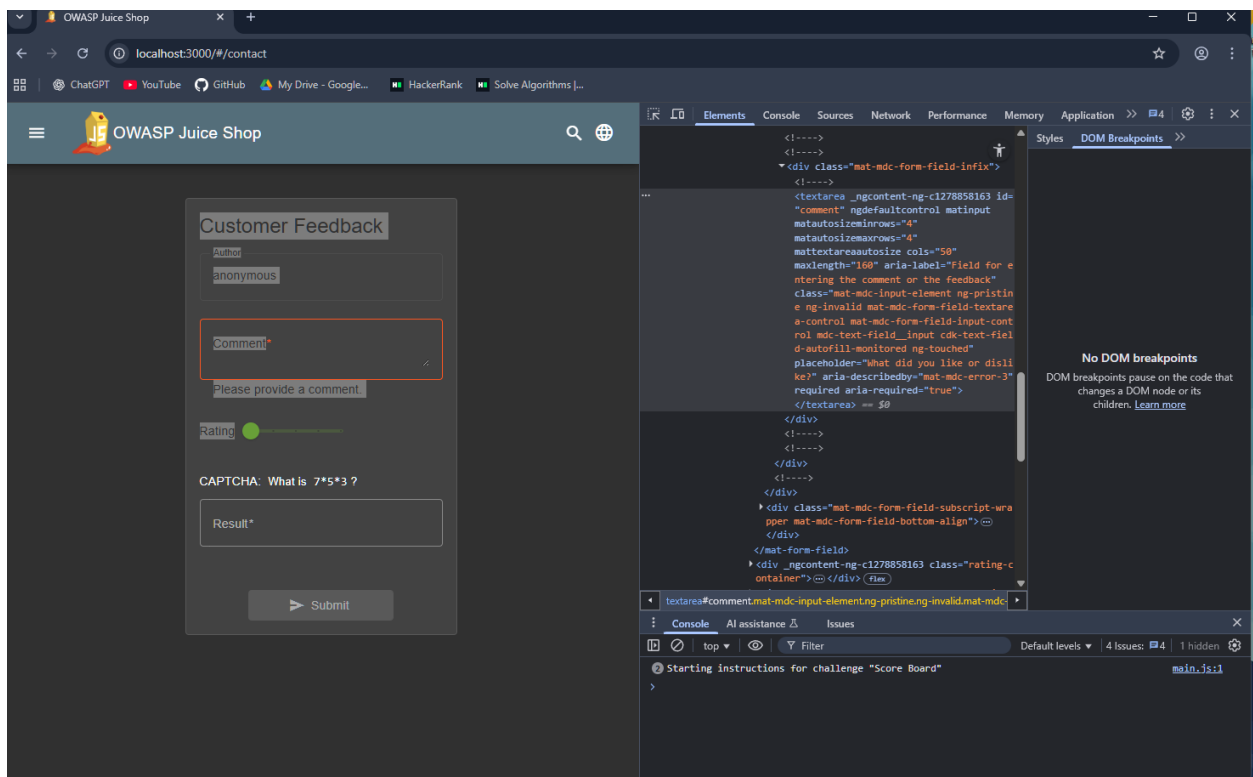
2.- modo inspección:



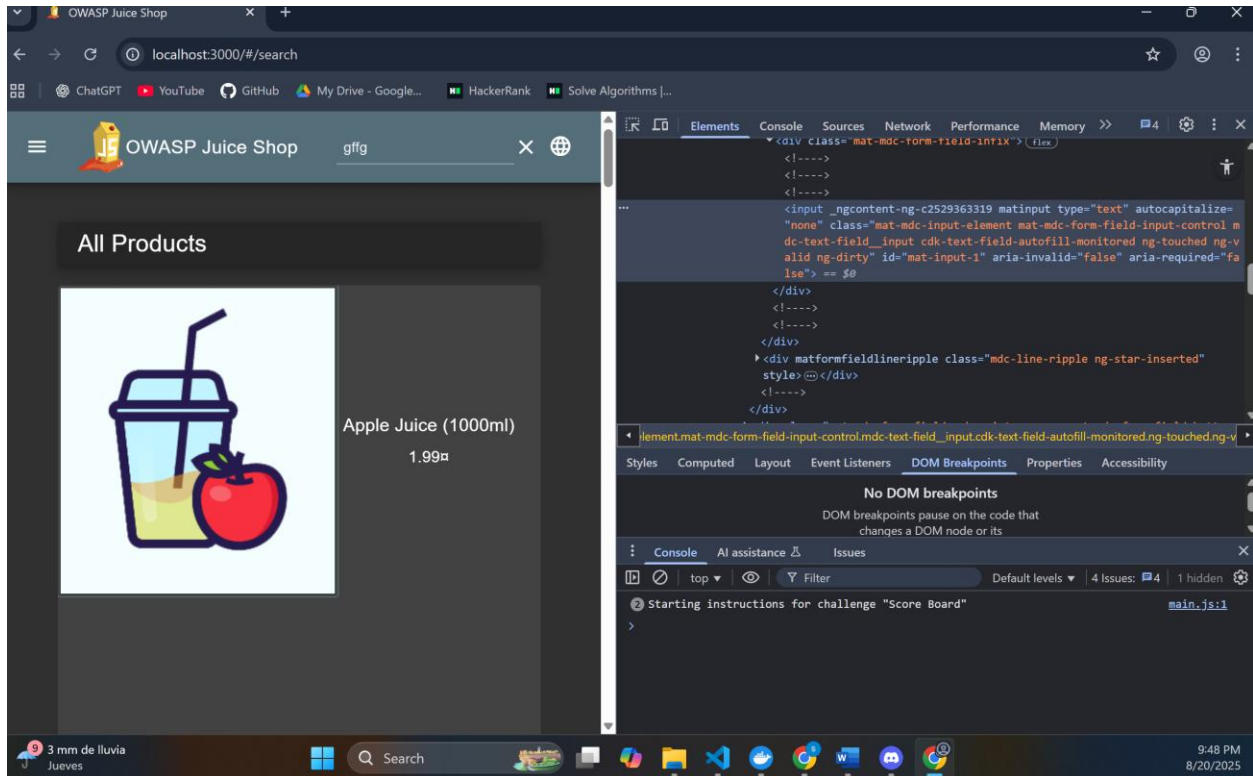
3.- inspección del formulario login:



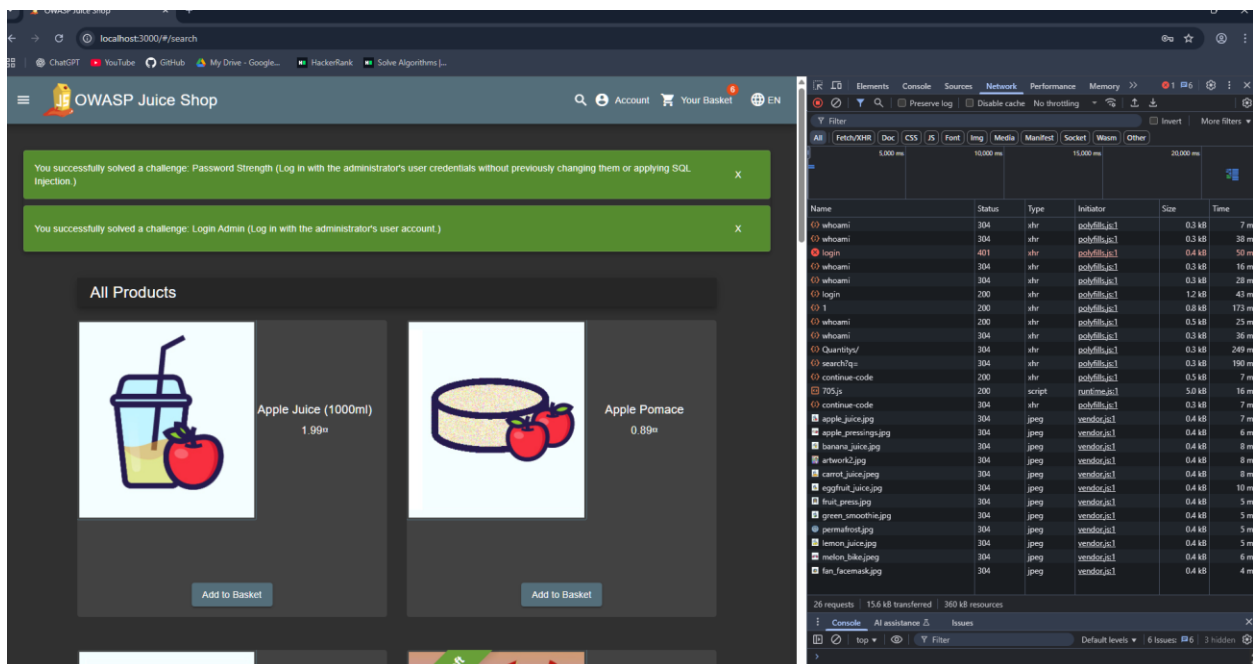
4.- inspección de formulario de comentarios:



5.-Inspección de formulario de búsqueda:



6.- acceso por credenciales por defecto:



7.- Token expuesto en localStorage:

localhost:3000/#/search


ChatGPTYouTubeGitHubMy Drive - Google...HackerRankSolve Algorithms...

OWASP Juice Shop

You successfully solved a challenge: Password Strength (Log in with the administrator's user credentials without previously changing them or applying SQL Injection.)

You successfully solved a challenge: Login Admin (Log in with the administrator's user account.)

All Products



Apple Juice (1000ml)
1.99€

Application

Filter

Only show cookies with an issue

Name	Value	Domain	Path	Expires	Size	HttpO...	Secure	SameS...	Partia...	Cross ...	Priority
continueCode	PrqGPjVSZ79ZmqawQB4M1evkLdcauIQG...	localh...	/	2026-...	72						Medium
cookieconsent_status	dismiss	localh...	/	2026-...	27						Medium
language	en	localh...	/	2026-...	10						Medium
token	eyJ0eXA0UlY1Qm9kaG9Gc0U5USt1NE9ybz...	localh...	/	2025-...	737						Medium

No cookie selected
Select a cookie to preview its value

Console

AI assistance

Issues

top

Filter

Default levels

6 issues

6

3 hidden