

# INFORME DE AUDITORÍA WEBGOAT 8.1.0

**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:**

```

Session Acciones Editar Vista Ayuda
> nmap 127.0.0.1
Starting Nmap 7.98 ( https://nmap.org ) at 2026-01-15 19:25 -0300
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000070s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
3000/tcp   open  ppp
8080/tcp   open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds

```

## Análisis:

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

## Sistema Operativo

Comando: docker exec webgoat uname -a

Resultado:

```

[sudo] contraseña para gabo:
Linux 8ffd24309915 6.17.10+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.17.10-1kali1 (2025-12-08) x86_64 x86_
64 x86_64 GNU/Linux

```

Sistema Linux (contenedor Docker)

## Lenguajes de Programación Identificados:

Request	Response
GET /login	HTML Content (Login Form)
POST /login	HTML Content (Login Success or Error)
GET /style.css	stylesheet (cached)
GET /bootstrap.min.css	stylesheet (cached)
GET /favicon.ico	image (cached)
GET /font-awesome.min.css	stylesheet (cached)
GET /animate.css	stylesheet (cached)
GET /	HTML Content (Home Page)

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 inyección SQL a través 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

The screenshot shows a web form with two input fields. The first field is labeled "Employee Name:" and contains the value "Smith". The second field is labeled "Authentication TAN:" and contains the value "TAN". Below these fields is a button labeled "Get department".

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.

The screenshot shows a browser window with a URL of <http://127.0.0.1:8080/WebGoat/start.mvc?username=gabriel#lesson/SqlInjection.lesson/10>. The page content includes sections on Identity & Auth Failure, Software & Data Integrity, Security Logging Failures, and Server-side Request Forgery. A challenge titled "It is your turn!" is displayed, describing a scenario where an employee named John Smith wants to view internal data. The user has entered the exploit query "SELECT \* FROM employees WHERE last\_name = '" + name + "' AND auth\_tan = '" + auth\_tan + "'"; into a text area. The page then displays a success message: "You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!" followed by a table of employee data.

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

## 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.

The screenshot shows a browser window with the URL <http://127.0.0.1:8080/WebGoat/start.mvc?username=gabriele#lesson/CrossSiteScripting.lesson/6>. The left sidebar shows a navigation tree with sections like Introduction, General, A1 Broken Access Control, A2 Cryptographic Failures, A3 Injection, and A5 Security Misconfiguration. The main content area is titled "Try It! Reflected XSS". It contains a brief assignment description and a "Shopping Cart" table with items: Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry (Price: 69.99, Quantity: 1, Total: \$0.00), Dynex - Traditional Notebook Case (Price: 27.99, Quantity: 5, Total: \$0.00), Hewlett-Packard - Pavilion Notebook with Intel Centrino (Price: 1599.99, Quantity: 3, Total: \$0.00), and 3 - Year Performance Service Plan \$1000 and Over (Price: 299.99, Quantity: 3, Total: \$0.00). Below the cart, there are fields for credit card number and access code, both of which contain the payload <script>console.log("holo")</script>. A "Purchase" button is present. At the bottom, a message says "Congratulations, but console logs are not very impressive are they? Let's continue to the next assignment." and "Thank you for shopping at WebGoat. Your support is appreciated".

## A5 Security Misconfiguration - Apartado 4

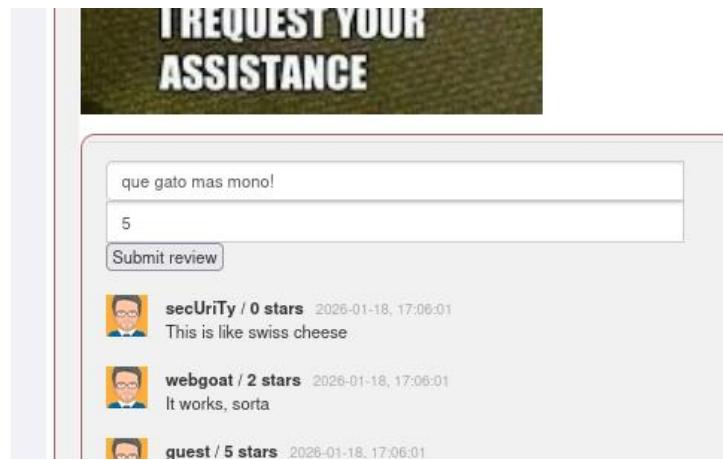
En este apartado se observó que la aplicación presenta una mala configuración de seguridad, ya que confía excesivamente en el estado de la sesión del usuario sin aplicar controles adicionales. Esto demuestra cómo una configuración insegura puede facilitar comportamientos no deseados y representar un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

The screenshot shows a browser window with the URL <http://127.0.0.1:8080/WebGoat/start.mvc?username=gabriele#lesson/CrossSiteRequestForgeries.lesson/1>. The left sidebar shows a navigation tree with sections like Cross Site Scripting (stored), Cross Site Scripting (mitigation), Path Traversal, A5 Security Misconfiguration, A6 Vuln & Outdated Components, A7 Identity & Auth Failure, A8 Software & Data Integrity, A9 Security Logging Failures, and A10 Server-side Request Forgery. The main content area is titled "Cross-Site Request Forgeries". It shows a review form for a poster titled "HUMAN" featuring a cat. The review text is "I REQUEST YOUR ASSISTANCE". Below the form, a list of reviews is shown, including:

- sestoby / 0 stars (2020-01-18, 16:31:27)  
This is like swiss cheese
- webgoat / 2 stars (2020-01-18, 16:31:27)  
I wanna, sorta
- gabriele / 5 stars (2020-01-18, 16:31:27)  
Best. App. Ever.
- gabriele / 1 stars (2020-01-18, 16:31:27)  
This app is so meeee... I didn't even post this review, can you pull that off too?

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.

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	POST	127.0.0.1:8080	review	jquery.min.js:2 (xhr)	json	253 B	8.4...
200	GET	127.0.0.1:8080	CSRF.lesson	jquery.min.js:2 (xhr)	json	811 B.	605 B
200	GET	127.0.0.1:8080	lessonmenu.mvc	jquery.min.js:2 (xhr)	json	8.61 kB	8.4...
200	GET	127.0.0.1:8080	lessonmenu.mvc	jquery.min.js:2 (xhr)	json	8.61 kB	8.4...
200	POST	127.0.0.1:8080	review	jquery.min.js:2 (xhr)	json	415 B	253 B
200	GET	127.0.0.1:8080	CSRF.lesson	jquery.min.js:2 (xhr)	json	811 B.	605 B
200	GET	127.0.0.1:8080	lessonmenu.mvc	jquery.min.js:2 (xhr)	json	8.61 kB	8.4...

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 más mono”**

**stars: “5”**

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

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	POST	127.0.0.1:8080	review	jquery.min.js:2 (xhr)	json	415 B	253 B
200	GET	127.0.0.1:8080	CSRF.lesson	jquery.min.js:2 (xhr)	json	811 B.	605 B
200	GET	127.0.0.1:8080	lessonmenu.mvc	jquery.min.js:2 (xhr)	json	8.61 kB	8.4...

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

What's new?

Burp AI comes to Repeater

Build your own Rep feature, now with AI

```
Promise {<pending>}
> await fetch('http://127.0.0.1:8888/WebGoat/carf/review', {
  'credentials': 'include',
  'headers': {
    'Accept': '*/*',
    'Accept-Language': 'en-US,en;q=0.5',
    'Content-Type': 'application/x-www-form-urlencoded; charset=UTF-8',
    'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0',
    'X-Request-Type': 'RepeaterRequest',
    'X-Repeater-Path': '/',
    'Sec-Fetch-Dest': 'script',
    'Sec-Fetch-Mode': 'cors',
    'Sec-Fetch-Site': 'same-origin',
    'SameSite': 'lax'
  },
  'referrer': 'http://127.0.0.1:8888/WebGoat/carf/review',
  'method': 'POST',
  'mode': 'cors'
})
<Response {type: 'basic', url: 'http://127.0.0.1:8888/WebGoat/carf/review', redirected: false, status: 200, ok: true, ...}>
```

**gabriel / 5 stars** 2026-01-18, 18:41:37  
Keepcoding

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 contraseña muy débil: "zapallo" con longitud 7

← → ⌂ 127.0.0.1:8080/WebGoat/start.mvc?username=gabriel#lesson/SecurePasswords.lesson/3

# Secure Passwords

Introduction >

General >

(A1) Broken Access Control >

(A2) Cryptographic Failures >

(A3) Injection >

(A5) Security Misconfiguration >

(A6) Vuln & Outdated Components >

(A7) Identity & Auth Failure >

Authentication Bypasses

Insecure Login

JWT tokens

Password reset

Secure Passwords 

(A8) Software & Data Integrity >

(A9) Security Logging Failures >

(A10) Server-side Request Forgery >

Client side >

Challenges >

Reset lesson

◀ 1 2 3 4 5 6 ▶

## How long could it take to brute force your password?

In this assignment, you have to type in a password that is strong enough (at least 4/4).

After you finish this assignment we highly recommend you try some passwords below to see why they are not good choices:

- password
- johnsmith
- 2018/10/4
- 1992home
- abcabc
- ffgfgt
- pouiz
- @dmin

zapallo

You have failed! Try to enter a secure password.

Your Password: \*\*\*\*\*

Length: 7

Estimated guesses needed to crack your password: 10000001

Score: 2/4

Estimated cracking time: 0 years 11 days 13 hours 46 minutes 40 seconds

Suggestions:

- Add another word or two. Uncommon words are better.

Score: 2/4

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

A screenshot of a web-based password cracking tool. The input field contains the password "passwordanotheronesupersecurity". A blue "Submit" button is visible. Below the input field, the text "You have succeeded! The password is secure enough." is displayed. Underneath this, there is detailed information about the password: "Your Password: \*\*\*\*\*", "Length: 31", "Estimated guesses needed to crack your password: 996964000000000000", "Score: 4/4", "Estimated cracking time: 31613521 years 20 days 4 hours 26 minutes 40 seconds", and "Score: 4/4".

### 3.c Post-explotación

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

Possible acceso no autorizado a información sensible y exposición de los datos.

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

Una vez explotada la vulnerabilidad, el código injectado 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**
- **Possible 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 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 código de verificación que cambia cada vez**
- **El servidor revisa que el código sea correcto**

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

## **2) Configurar las Cookies bien**

**Las cookies deben tener:**

- **SameSite=Strict (No se envían desde otras páginas)**
- **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.