

# Informe de Auditoría de Seguridad

## Revisión de seguridad: Hacking ético sobre la aplicación Webgoat

### Índice

- 1. Ámbito y alcance de la auditoría
- 2. Informe ejecutivo
  - a. Proceso realizado
  - b. Vulnerabilidades destacadas
  - c. Conclusiones
  - d. Recomendaciones
- 3. Descripción del proceso de auditoría
  - a. Reconocimiento
  - b. Explotación
  - c. Post-explotación
  - d. Posibles mitigaciones
  - e. Herramientas utilizadas
- A3 Injection - Cross Site Scripting
- Security Misconfiguration
- A6 Vuln & outdated Components
- A7 Identity & Auth Failure - Secure Passwords

## 1. Ámbito y alcance de la auditoría

La auditoría de seguridad se ha centrado en la aplicación WebGoat versión 8.1.0. El entorno se ejecutó utilizando Docker con los siguientes comandos:

- `docker run --name webgoat -it -p 127.0.0.1:8080:8080 -p 127.0.0.1:9090:9090 -e TZ=Europe/Amsterdam webgoat/webgoat`
- `docker start webgoat`

La aplicación a auditar se encuentra en <http://127.0.0.1:8080/WebGoat>

```
(kali㉿kali)-[~]
$ nmap -p 8080,9090 localhost

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-11 04:44 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000064s latency).
Other addresses for localhost (not scanned): ::1
PORT      STATE SERVICE
8080/tcp  open  http-proxy
9090/tcp  closed zeus-admin

Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
```

- El comando utilizado para realizar el escaneo fue:  
**dirb** http://localhost:8080/WebGoat/ /usr/share/dirb/wordlists/common.txt

Página por defecto de la aplicación web.



Se identificó y explotó una vulnerabilidad de inyección SQL para acceder a información confidencial de todos los empleados.

## Inyección SQL en la consulta de base de datos.

```
GET /WebGoat/users/ HTTP/1.1 Host: localhost:8080 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Connection: close Cookie: JSESSIONID=1NrUee8JXLkjnjwO4kUN5mtCxxYaFn8YISwhF7t; connect.sid=s3AlgeVbNCLVQqWbyT0w0aMCtnB5Aa4G8c8X.xrwm4DDHX2BLmIloBVUU5bK9ZInXlTu74IAjDsHWMU Upgrade-Insecure-Requests: 1 Content-Type: application/json
```

## c. Conclusiones

El sistema es vulnerable a inyecciones SQL, comprometiendo la confidencialidad de los datos internos.

## d. Recomendaciones

- Implementar validación y sanitización de entradas.
- Utilizar consultas preparadas.
- Fortalecer las políticas de acceso y autenticación.

# 3. Descripción del proceso de auditoría

## a. Reconocimiento

Identificación de parámetros de entrada en consultas SQL.

## b. Explotación

Uso de la cadena ' OR 1=1 -- para manipular la consulta y acceder a todos los registros de la tabla employees.

✓

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

## c. Post-explotación

Verificación del acceso a información confidencial y análisis del impacto.

## d. Posibles mitigaciones

- Implementar consultas preparadas.
- Validar estrictamente las entradas del usuario.
- Utilizar mecanismos de autenticación robustos.

# A3 Injection - Cross Site Scripting

## 1. Ámbito y Alcance

Evaluar si el formulario de compra de un sitio web es vulnerable a ataques XSS reflejados.

## 2. Informe Ejecutivo

### Interceptar y Modificar la Solicitud HTTP

- Activo Intercept is on en Burp Suite.
  - Navego a `http://localhost:8080/WebGoat/users/` en Firefox.
  - En Burp Suite, permito que la solicitud pase haciendo clic en Forward.
  - En HTTP history, selecciono la respuesta y la modifico:
1. HTTP/1.1 200 OK
  2. Connection: close
  3. X-XSS-Protection: 1; mode=block
  4. X-Content-Type-Options: nosniff
  5. X-Frame-Options: DENY
  6. Content-Type: application/json
  7. Date: Thu 11 Jul 2024 10:32:17 GMT
  8. [
  9. {
  10. "username": "Dani",
  11. "admin": false,
  12. "userHash": "a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6q7r8s9t0u1v2w3x4y5z6="
  13. }
  14. ]

Se ingresaron scripts maliciosos en varios campos del formulario para detectar vulnerabilidades.

The screenshot shows a web application interface. At the top, a dark alert box displays the message "127.0.0.1:8081 says XSS" with an "OK" button. Below the alert, there is a section titled "Carro de la compra" (Shopping Cart) containing a table with columns: "Artículos del carrito de compras: para comprar ahora", "Precio", "Cantidad", and "Total". The table lists four items: "Studio RTA - Carrito para computadora portátil y lectura con superficie inclinable - Cereza", "Dynex - Estuche para portátil tradicional", "Hewlett-Packard - Portátil Pavilion con Intel Centrino", and "Plan de servicio de rendimiento de 3 años \$1000 y más". Each item has a price, a quantity input field set to "1", and a total of "\$0.00". Below the table, there are two input fields: "Introduzca su número de tarjeta de crédito:" and "Introduzca su código de acceso de tres dígitos:". The credit card field contains the malicious script "<script>alert('XSS')</script>" and the digit field contains "111". A "Compra" button is located at the bottom of the form.

Artículos del carrito de compras: para comprar ahora	Precio	Cantidad	Total
Studio RTA - Carrito para computadora portátil y lectura con superficie inclinable - Cereza	69,99	1	\$0.00
Dynex - Estuche para portátil tradicional	27,99	1	\$0.00
Hewlett-Packard - Portátil Pavilion con Intel Centrino	1599,99	1	\$0.00
Plan de servicio de rendimiento de 3 años \$1000 y más	299,99	1	\$0.00

Introduzca su número de tarjeta de crédito:

Introduzca su código de acceso de tres dígitos:

## Vulnerabilidad Detectada

El campo de número de tarjeta de crédito es vulnerable a XSS reflejado.

## Conclusiones

La vulnerabilidad permite la ejecución de scripts maliciosos, lo que puede llevar a robo de datos y otros ataques.

## Recomendaciones

- Validar y limpiar todas las entradas en el servidor.
- Escapar caracteres especiales en las respuestas HTML.
- Usar bibliotecas de seguridad y realizar pruebas regulares.

### 3. Descripción del Proceso de Auditoría

Pruebas Realizadas:

- Se ingresaron scripts en campos de cantidad, número de tarjeta de crédito y CVV.
- El campo de número de tarjeta de crédito mostró ser vulnerable.

Figura 4: Ejemplo de vulnerabilidad XSS en el campo de número de tarjeta de crédito.

### Herramientas Utilizadas

- Navegador web
- Editores de texto para scripts
- Herramientas de análisis de seguridad como OWASP ZAP

## Security Misconfiguration - Solución para Explotar XXE usando BurpSuite

### Proceso

1. Iniciar BurpSuite y configurar el proxy del navegador: Abre BurpSuite y configura el proxy en el navegador (localhost:8080).
2. Enviar el formulario a través de BurpSuite: En el navegador, envía el formulario (por ejemplo, haz clic en "Entregar" en el formulario de comentarios).
3. Localizar la solicitud en BurpSuite: Ve a "Proxy" > "HTTP history" en BurpSuite. Encuentra la solicitud POST /WebGoat/xxe/simple HTTP/1.1.
4. Enviar la solicitud al Repeater: Haz clic derecho en la solicitud y selecciona "Send to Repeater". Ve a la pestaña "Repeater".
5. Modificar el XML en la solicitud: Reemplaza el contenido XML con:

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

Enviar la solicitud y verificar la respuesta: Haz clic en "Send". Verifica la respuesta para confirmar el acceso al directorio C:/.

### Explicación de la Solución

Este procedimiento utiliza BurpSuite para interceptar y modificar la solicitud XML antes de enviarla al servidor. Al inyectar una entidad externa en el XML, se intenta explotar una vulnerabilidad XXE para acceder al sistema de archivos del servidor. El código XML inyectado define una entidad xxe que intenta acceder al sistema de archivos y la inserta en el comentario enviado al servidor.

## A6 Vuln & outdated Components

## El exploit no siempre está en "tu" código

A continuación se muestra un ejemplo de uso del mismo código fuente de WebGoat, pero con diferentes versiones del componente jquery-ui. Una es explotable, la otra no.

### jquery-ui:1.10.4

Este ejemplo permite al usuario especificar el contenido de "closeText" para el cuadro de diálogo de jquery-ui. Este es un escenario de desarrollo poco probable, sin embargo, el cuadro de diálogo de jquery-ui (por determinar - mostrar enlace de explotación) no protege contra XSS en el texto del botón del cuadro de diálogo de cierre.

Al hacer clic en Ir, se ejecutará un cuadro de diálogo de cierre de jquery-ui:

OK&ltscript>alert('XSS')</scrip  
¡Ir!

Este cuadro de diálogo debería haber aprovechado una falla conocida en jquery-ui:1.10.4 y permitir que se produjera un ataque XSS.

### jquery-ui:1.12.0 No vulnerable

Utilizando el mismo código fuente de WebGoat pero actualizando la biblioteca jquery-ui a una versión no vulnerable se elimina la vulnerabilidad.

Al hacer clic en Ir, se ejecutará un cuadro de diálogo de cierre de jquery-ui:

OK&ltscript>alert('XSS')</scrip  
¡Ir!

Este diálogo debería haber evitado la vulnerabilidad mencionada anteriormente utilizando EXACTAMENTE el mismo código en WebGoat pero usando una versión

## Ámbito y alcance de la auditoría

Evaluación de la seguridad de los componentes jQuery-UI en una aplicación web, enfocándonos en vulnerabilidades de XSS.

## Informe ejecutivo

Se probaron diferentes versiones de jQuery-UI en WebGoat para identificar y explotar vulnerabilidades XSS.

## Vulnerabilidades destacadas

- jQuery-UI 1.10.4: Vulnerable a XSS.
- jQuery-UI 1.12.0: No vulnerable a XSS.

## Conclusiones

Las versiones antiguas de jQuery-UI presentan vulnerabilidades críticas, corregidas en versiones más recientes.

## Recomendaciones

- Actualizar todos los componentes a las versiones más recientes.
- Implementar sanitización y validación de entradas del usuario.

## Descripción del proceso de auditoría

- Reconocimiento/Information gathering: Identificación de componentes y versiones.
- Explotación de vulnerabilidades detectadas: Pruebas de XSS en versiones jQuery-UI 1.10.4 y 1.12.0.
- Post-explotación: Confirmación de la corrección de vulnerabilidades en versiones posteriores.

Figura 6: Ejemplo de vulnerabilidad XSS en jQuery-UI.

## Posibles mitigaciones

Actualización a jQuery-UI 1.12.0 o superior.

- Implementación de Content Security Policy (CSP).

## Herramientas utilizadas

- ## A7 Identity & Auth Failure - Secure Passwords

## Ámbito y alcance de la auditoría

## Informe ejecutivo

## Vulnerabilidades destacadas

## Conclusiones

## Recomendaciones

- ## Descripción del proceso de auditoría

- ## Posibles mitigaciones

- Políticas que requieran contraseñas seguras.
- Educación sobre la importancia de contraseñas fuertes.
- 

## Herramientas utilizadas en la investigación.

- Nmap
- Dirp
- burpsuite
- WebGoat
- Bitwarden