

Informe de Auditoría de Ciberseguridad

Evaluación de Seguridad de “WebGoat”



Nombre del Auditor: Kacper Mariusz Koper Mielczarek

Fecha: 08/12/2024

Índice

Informe de Auditoría de Ciberseguridad.....	1
1. Introducción.....	4
2. Ámbito y Alcance de la Auditoría.....	4
3. Informe Ejecutivo	5
3.1 Breve Resumen del Proceso Realizado.....	5
3.2 Principales Hallazgos.....	5
3.3 Conclusiones y Recomendaciones Clave	6
3.3.1. Conclusiones:	6
3.3.2. Recomendaciones Clave:.....	7
4. Descripción del Proceso de Auditoría	8
4.1. Reconocimiento/Information Gathering.....	8
4.2. Explotación de Vulnerabilidades Detectadas	9
4.2.1 SQL Injection:.....	9
4.2.2.Cross-Site Scripting (XSS):.....	9
4.2.3.CVE-2013-7285:.....	9
4.3. Post-Explotación.....	9
4.4. Posibles Mitigaciones.....	9
4.4.1. Sanitización de Entradas:	9
4.4.2. Configuración Segura de Componentes:	9
4.4.3. Cifrado de Datos Sensibles:.....	9
4.4.4. Pruebas de Ciberseguridad:	9
5. Herramientas Utilizadas	10
5.1. Práctica de Introducción a la ciberseguridad.....	10
5.2. Nmap en localhost.....	12
5.3. Docker ps.....	12
5.4. Comprobar puertos	12
5.5. Ver sistema operativo.....	13
5.6. Lenguajes utilizados en la página web	13
5.7. Inyección SQL.....	14
5.8. Script base de datos vulnerables.....	14
5.9. Vulnerabilidad XXE	15
5.10. Archivo XML Malicioso	16
5.11. Contraseña Alta Seguridad.....	16

5.12. Contraseña Baja Seguridad..... 16

1. Introducción

En esta auditoría de ciberseguridad se analizará la página web alojada en el entorno local (localhost) de la dirección “<http://localhost:8080/WebGoat/login>”, previamente configurada a través de la terminal en Kali Linux.

El objetivo principal es realizar pruebas de seguridad siguiendo las pautas establecidas en el documento “[Práctica de Introducción a la Ciberseguridad](#)”. Estas pruebas buscan identificar vulnerabilidades y evaluar los riesgos asociados, aplicando los conocimientos adquiridos durante la primera semana del curso.

Cabe destacar que la aplicación web ha sido diseñada intencionalmente con errores de seguridad para facilitar el aprendizaje práctico y el fortalecimiento de habilidades técnicas en el análisis y mitigación de vulnerabilidades.

2. Ámbito y Alcance de la Auditoría

El ámbito de esta auditoría se centra exclusivamente en el análisis de seguridad de una aplicación web desarrollada internamente, la cual se encuentra desplegada en un entorno local (localhost) a través de la terminal en Kali Linux. Este análisis tiene como objetivo principal identificar posibles vulnerabilidades de seguridad en la aplicación web, así como en los componentes de infraestructura asociados, como servicios, puertos y bases de datos.

*Para llevar a cabo la auditoría, se utilizarán herramientas de análisis de seguridad ampliamente reconocidas, como **nmap** para la identificación y exploración de puertos abiertos y servicios activos. Adicionalmente, se investigará el tipo de base de datos utilizada por la aplicación; en caso de tratarse de una base de datos SQL, se intentará acceder a ella mediante técnicas de pruebas de seguridad, como la inyección SQL, para evaluar la exposición de los datos almacenados.*

El alcance de la auditoría incluye:

- *Exploración de puertos abiertos.*
- *Identificación del sistema gestor de bases de datos (SGBD) en uso y sus correspondientes configuraciones.*
- *Ejecución de pruebas de inyección SQL para determinar posibles vulnerabilidades relacionadas con el acceso no autorizado a los datos.*
- *Verificación de la seguridad en el diseño de la aplicación web y su resistencia a ataques comunes (Como hemos visto en el último día del curso a través de DVNA).*

Se excluyen del alcance la evaluación de aspectos de seguridad relacionados con el entorno operativo de Kali Linux más allá de la configuración local requerida para esta auditoría, así como las pruebas que impliquen acceso a redes externas.

El análisis se realizará únicamente en un entorno controlado y con fines educativos, ya que esta página web fue diseñada específicamente para ello.

3. Informe Ejecutivo

La puesta a practica se realizará siguiendo el guion proporcionado por el profesor titulado del curso: [“Práctica de Introducción a la Ciberseguridad”](#).

3.1 Breve Resumen del Proceso Realizado

Primero se ha escaneado el localhost con el comando “[nmap](#)” para verificar que puerto o puertos está utilizando la página web <http://localhost:8080/WebGoat/login>, y después he verificado por Docker con el comando “[Docker ps](#)” los puertos en uso por la web. Por último, se utilizado el comando “[nc -zv <IP> <PUERTO>](#)” para verificar si los puertos que utiliza la web están abiertos.

Adicionalmente, se realizó una búsqueda para identificar el sistema operativo de la página web. Tras ejecutar el comando “[nmap -O <ip><puerto>](#)”, se determinó que el sistema operativo corresponde a un kernel de Linux en su versión 2.6.32.

Por último, mediante el uso de una extensión de la herramienta “[Wappalyzer](#)”, se analizaron las tecnologías utilizadas en la página web, descubriendo que emplea Java como uno de sus lenguajes principales.

3.2 Principales Hallazgos

En la primera parte se pide que se descubra el salario de los compañeros de trabajo, donde nos ofrecen como ayuda un buscador donde podemos introducir nuestro nombre y nuestro código para saber cuanto cobramos, etc. Para saber cuál es el salario de nuestros compañeros de trabajo hay que poner el siguiente código: ['OR 1=1](#)

Este código SQL lo que hace es:

- 1. ' : Cierra la cadena de texto abierta para el last_name.*
- 2. OR 1=1: Introduce una condición que siempre será verdadera, haciendo que la consulta devuelva todos los registros.*
- 3. --: Comenta el resto de la consulta SQL para que no cause errores.*

Por lo tanto, con esta consulta hemos podido sacar no solo el salario de los compañeros de trabajo, sino que también sus datos personales como sus nombres y apellidos, departamento de trabajo, TAN y su ID.

El segundo ejercicio tenemos que inyectar un script para saber si la base de datos es vulnerable. En este caso hemos inyectado el script `<script>console.log('XSS')</script>` pero si queremos ser mas agresivos podemos inyectar `<script>alert('XSS')</script>`.

En el tercer ejercicio hemos observado que la página web tiene una [vulnerabilidad al enviar comentarios](#). Por lo tanto, se iniciaron las aplicaciones WebGoat y WebWolf herramientas utilizadas para la práctica de seguridad que recomendaban en el apartado anterior. A continuación, se creó un archivo DTD malicioso que apuntaba a un archivo sensible en el sistema y se cargó en WebWolf, obteniendo el enlace correspondiente. En WebGoat, se publicó un comentario y se interceptó la solicitud POST mediante una herramienta de análisis que en este caso es ZAP. Posteriormente, se modificó la solicitud para incluir el enlace al archivo subido en WebWolf. Al publicar el comentario, WebGoat procesó el archivo DTD malicioso, revelando el contenido del archivo secreto y explotando una vulnerabilidad de tipo XXE (XML External Entity).

En la cuarta parte, nos hemos encontrado con CVE-2013-7285 que es un problema de seguridad en XStream, una herramienta que convierte datos entre objetos de Java y XML. Este problema permite que un atacante cree un [archivo XML](#) especial que engaña a XStream para que ejecute comandos en un sistema donde se está usando. Esto puede ocurrir si el XML contiene instrucciones maliciosas que XStream interpreta como si fueran seguras. Por lo tanto, es un fallo que podría ser usado para tomar el control de un sistema si no se tiene cuidado al usar esta herramienta.

En el último ejercicio, nos encontramos con una página de validación de contraseñas de seguridad, donde sacamos la conclusión de que las [contraseñas más inseguras](#) son las que son simples y contienen pocos caracteres como "12345" o "abcde". Por lo contrario, observamos que las [contraseñas mas seguras](#) deben contener más de 9 caracteres y deben llevar la combinación entre letras (tanto mayúsculas como minúsculas), números y caracteres especiales como "cOnmiedoSe1\$". El orden de los factores sí que influye en la seguridad de la contraseña.

3.3 Conclusiones y Recomendaciones Clave

3.3.1. Conclusiones:

La auditoría realizada a la aplicación WebGoat en un entorno controlado permitió identificar múltiples vulnerabilidades críticas diseñadas con fines educativos. A continuación, se destacan los principales puntos de aprendizaje y hallazgos clave:

3.3.1.1. Vulnerabilidades SQL Injection (SQLI): La aplicación permite realizar inyecciones SQL, exponiendo datos sensibles como salarios y datos personales de empleados. Esto evidencia la ausencia de medidas para filtrar y validar entradas de usuario y valida la importancia de usar consultas parametrizadas.

3.3.1.2. Cross-Site Scripting (XSS): La inyección de scripts maliciosos en campos vulnerables permite ejecutar código arbitrario en el navegador del usuario. Esto podría ser utilizado por atacantes para robar cookies, redirigir a sitios maliciosos o realizar otras más acciones que no se hayan autorizado.

3.3.1.3. Exposición a Vulnerabilidades de Software (CVE-2013-7285): La implementación de XStream presenta riesgos asociados a la ejecución de comandos maliciosos mediante XML manipulados. Esto expone la necesidad de mantener el software siempre actualizado y limitar el uso de herramientas que no validen adecuadamente las entradas. Por lo tanto, hay que actualizar el código base de la web para limitar la interacción del usuario en la propia web.

3.3.1.4. Gestión de Contraseñas Inadecuada: Las pruebas realizadas a la funcionalidad de validación de contraseñas demostraron que la aplicación no impone políticas estrictas para la creación de contraseñas robustas. Contraseñas simples y de baja complejidad son aceptadas, lo que aumenta el riesgo de ataques de fuerza bruta, donde estas contraseñas con poca envergadura sean fáciles de acceder en un tiempo bastante corto.

3.3.2. Recomendaciones Clave:

3.3.2.1. Implementar Medidas de Prevención de Inyecciones SQL:

Utilizar consultas preparadas y procedimientos almacenados en lugar de concatenar entradas de usuario en consultas SQL.

Validar y filtrar todas las entradas de usuario, asegurándose de que solo se acepten datos esperados.

3.3.2.2. Proteger contra XSS:

Aplicar un filtrado adecuado de las entradas y codificar las salidas antes de mostrarlas en el navegador.

Implementar políticas de seguridad de contenido (Content Security Policy, CSP) para limitar el impacto de scripts maliciosos.

3.3.2.3. Actualizar y Parchear Vulnerabilidades en Componentes de Software:

Revisar y actualizar regularmente las dependencias utilizadas en la aplicación, como XStream, a versiones seguras.

Adoptar herramientas para el análisis de dependencias y su estado de seguridad.

Todo esto debe realizarse periódicamente para disminuir la posibilidad de fallos.

3.3.2.4. Establecer Políticas de Contraseñas Fuertes:

Obligar a que las contraseñas tengan un mínimo de 12 caracteres y contengan combinaciones de letras mayúsculas, minúsculas, números y caracteres especiales.

Implementar mecanismos de bloqueo ante intentos de fuerza bruta y habilitar autenticación multifactorial.

3.3.2.5. Revisar Regularmente la Seguridad de la Aplicación:

Realizar auditorías periódicas y pruebas de penetración en entornos de desarrollo y producción.

3.3.2.6 Vulnerabilidad XXE:

Para solucionar la vulnerabilidad XXE, que permite que un atacante lea archivos secretos del servidor, lo primero es desactivar las entidades externas en el procesador XML. Esto significa que el sistema no podrá cargar archivos o hacer peticiones externas maliciosas. Para lograrlo, se debe configurar el sistema o las herramientas que procesan los datos XML, de manera que no acepten este tipo de solicitudes peligrosas. Esto es una medida sencilla pero muy eficaz para evitar ataques de este tipo. Además, es importante validar los datos que los usuarios envían, especialmente en formularios de comentarios, para asegurarse de que no contengan código malicioso que el sistema pueda ejecutar.

Otra forma de mejorar la seguridad es asegurándose de que el sistema solo tenga acceso a los archivos que necesita para funcionar, y limitar los privilegios de los programas para evitar que un atacante acceda a archivos sensibles. Además, se deben utilizar librerías de XML seguras y hacer pruebas de seguridad constantes en la aplicación, para detectar posibles fallos antes de que sean explotados. Con estas acciones, se puede prevenir que un atacante obtenga información secreta o comprometa el sistema a través de este tipo de vulnerabilidad.

4. Descripción del Proceso de Auditoria

4.1. Reconocimiento/Information Gathering

Durante esta fase, se recopilaron detalles básicos de la aplicación y su infraestructura subyacente:

- Se utilizó nmap para identificar puertos abiertos y servicios activos asociados a la aplicación.
- Con herramientas como Wappalyzer, se determinaron las tecnologías empleadas, confirmando que se trata de una aplicación construida en Java sobre un servidor web que opera en Linux.

4.2. Explotación de Vulnerabilidades Detectadas

4.2.1 SQL Injection:

Se identificó una vulnerabilidad en un campo de búsqueda, donde se pudo realizar una inyección utilizando ' OR 1=1 --, obteniendo acceso no autorizado a datos confidenciales.

4.2.2. Cross-Site Scripting (XSS):

Se confirmó la vulnerabilidad mediante la ejecución de un script como <script>alert('XSS')</script>, demostrando la falta de validación en entradas de usuario.

4.2.3. CVE-2013-7285:

Se comprobó que la vulnerabilidad en XStream podría explotarse mediante un archivo XML manipulado.

4.3. Post-Explotación

Tras la explotación de las vulnerabilidades, se evaluaron las posibles implicaciones para un atacante:

- *Acceso a datos sensibles.*
- *Ejecución de comandos remotos en el sistema, gracias a la vulnerabilidad en XStream.*
- *Robo de cookies o redirección maliciosa por los ciberdelincuentes, en el caso de XSS.*

4.4. Posibles Mitigaciones

Se proponen los siguientes puntos para mitigar las vulnerabilidades:

4.4.1. Sanitización de Entradas: *Garantizar que todas las entradas sean validadas para evitar SQLi y XSS.*

4.4.2. Configuración Segura de Componentes: *Implementar versiones actualizadas y configurar adecuadamente las dependencias utilizadas.*

4.4.3. Cifrado de Datos Sensibles: *Asegurar que datos críticos como contraseñas o información personal estén encriptados.*

4.4.4. Pruebas de Ciberseguridad: *Adoptar un ciclo de pruebas de seguridad continuas para detectar nuevas vulnerabilidades antes de que puedan ser explotadas.*

5. Herramientas Utilizadas

5.1. Práctica de Introducción a la ciberseguridad



Módulo introducción a la ciberseguridad

Objetivo: Realización de un proyecto de auditoría web básica

Contenido:

- Introducción a la ciberseguridad
- Introducción a la auditoría web
- Introducción a la auditoría web: herramientas y metodologías
- Introducción a la auditoría web: casos de estudio

PRÁCTICA

En esta práctica aplicarás diversas herramientas y conocimientos aprendidos durante este módulo. De cara a la realización de la práctica y su cumplimiento se deben seguir los siguientes pasos:

Primera parte

Esta práctica se centra en la auditoría, solución y escritura de un informe sobre una aplicación web vulnerable. La aplicación web que se utilizará durante la práctica será WebGoat versión 8.1.0, <https://github.com/WebGoat/WebGoat>.

En primer lugar se debe ejecutar este entorno la recomendación es hacerlo con Docker:

- `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`

Segunda parte reconocimiento

- Reconocimiento de la aplicación web vulnerable
- Reconocimiento de la aplicación web vulnerable: herramientas y metodologías
- Reconocimiento de la aplicación web vulnerable: casos de estudio



Tercera parte detección y explotación de vulnerabilidades

El objetivo de esta parte es identificar y explotar vulnerabilidades en un sistema. Para ello, se utilizarán herramientas como Metasploit y Burp Suite, así como técnicas de ingeniería inversa y análisis de código fuente.

- Identificación de vulnerabilidades en un sistema.
- Explotación de vulnerabilidades en un sistema.
- Análisis de código fuente para identificar vulnerabilidades.
- Análisis de tráfico de red para identificar vulnerabilidades.
- Análisis de logs para identificar vulnerabilidades.
- Análisis de configuración para identificar vulnerabilidades.
- Análisis de permisos para identificar vulnerabilidades.
- Análisis de credenciales para identificar vulnerabilidades.

Después de haber completado los ejercicios, se debe realizar un informe de auditoría indicando todas las pruebas y procesos realizados durante la práctica.

Cuarta parte informe de auditoría

Tras haber completado los ejercicios se debe realizar un informe de auditoría indicando todas las pruebas y procesos realizados durante la práctica.

Este informe de auditoría ha de contar con las siguientes secciones como mínimo:

1. Ámbito y alcance de la auditoría
2. Informe ejecutivo
 - a. Breve resumen del proceso realizado
 - b. Vulnerabilidades destacadas
 - c. Conclusiones
 - d. Recomendaciones
3. Descripción del proceso de auditoría
 - a. Reconocimiento/Information gathering
 - b. Explotación de vulnerabilidades detectadas
 - c. Post-explotación
 - d. Posibles mitigaciones
 - e. Herramientas utilizadas

EVALUACIÓN

El informe de auditoría es obligatorio y debe ser entregado en formato PDF. El informe debe incluir todas las secciones mencionadas anteriormente y debe ser evaluado por el profesor. El informe debe ser entregado en el tercer día de la práctica.

5.2. Nmap en localhost

```
(kk@kk)-[~]
$ nmap localhost
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-03 18:42 CET
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000020s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
8080/tcp  open  http-proxy
9090/tcp  open  zeus-admin

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

5.3. Docker ps

```
(kk@kk)-[~]
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
b5a3e4c12dd   webgoat/webgoat "java -Duser.home=/h..." 4 minutes ago  Up 4 minutes  127.0.0.1:8080->8080/tcp, 127.0.0.1:9090->9090/tcp  focused_liskov
```

5.4. Comprobar puertos

```
(kk@kk)-[~]
$ nc -zv localhost 9090
localhost [127.0.0.1] 9090 (?) open
```

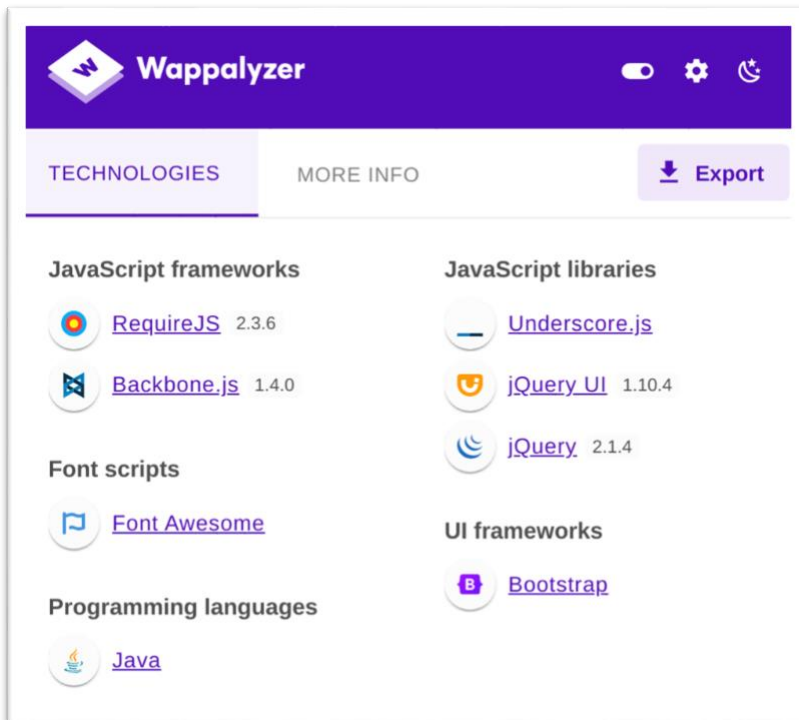
```
(kk@kk)-[~]
$ nc -zv localhost 8080
localhost [127.0.0.1] 8080 (http-alt) open
```

5.5. Ver sistema operativo

```
└─$ nmap -O 127.0.0.1
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-05 18:24 CET
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000090s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
8080/tcp   open  http-proxy
9090/tcp   open  zeus-admin
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 1.45 seconds
```

5.6. Lenguajes utilizados en la página web



The screenshot shows the Wappalyzer interface with a purple header. Below the header, there are tabs for 'TECHNOLOGIES' and 'MORE INFO', and an 'Export' button. The 'TECHNOLOGIES' tab is active, displaying a list of detected technologies categorized into JavaScript frameworks, JavaScript libraries, Font scripts, Programming languages, and UI frameworks.

Category	Technology	Version
JavaScript frameworks	RequireJS	2.3.6
	Backbone.js	1.4.0
JavaScript libraries	Underscore.js	
	jQuery UI	1.10.4
	jQuery	2.1.4
Font scripts	Font Awesome	
Programming languages	Java	
UI frameworks	Bootstrap	

5.7. Inyección SQL

✓

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

5.8. Script base de datos vulnerables

Carro de la compra

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	<input type="text" value="1"/>	\$0.00
Dynex - Estuche tradicional para portátiles	27,99	<input type="text" value="1"/>	\$0.00
Hewlett-Packard - Notebook Pavilion con Intel Centrino	1599,99	<input type="text" value="1"/>	\$0.00
Plan de servicio de rendimiento de 3 años \$1000 y más	299,99	<input type="text" value="1"/>	\$0.00

Introduzca su número de tarjeta de crédito:

Introduzca su código de acceso de tres dígitos:

Felicitaciones, pero los registros de la consola no son muy impresionantes, ¿no? Continuemos con la siguiente tarea.

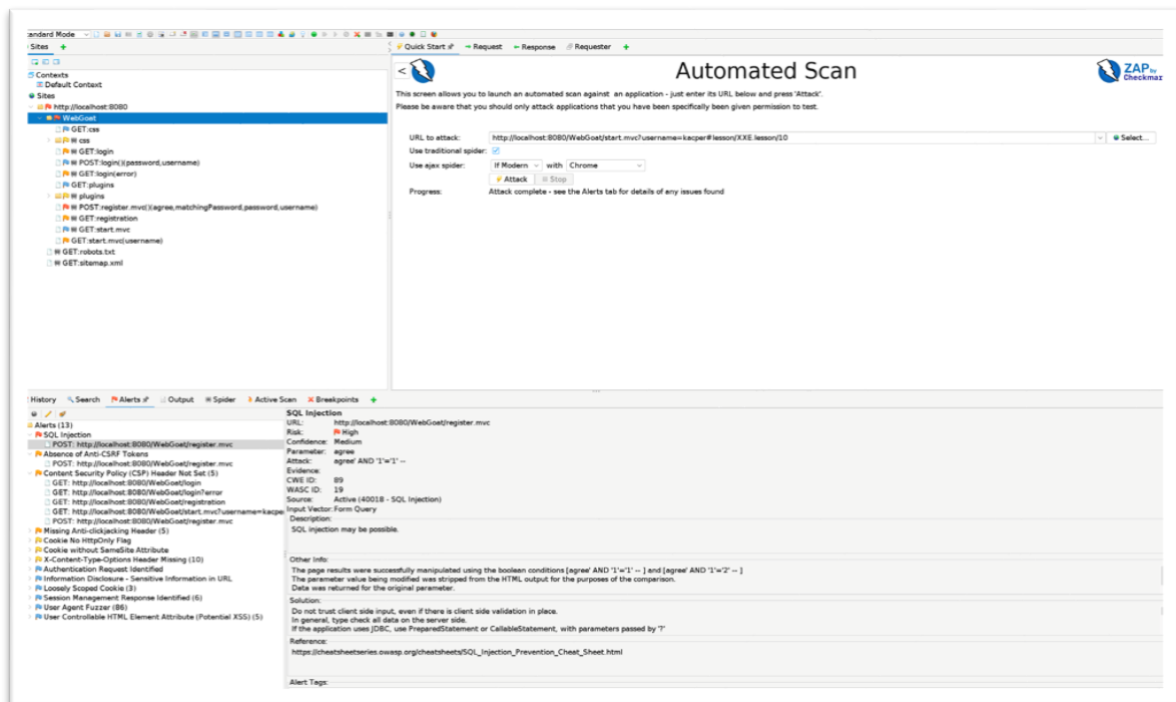
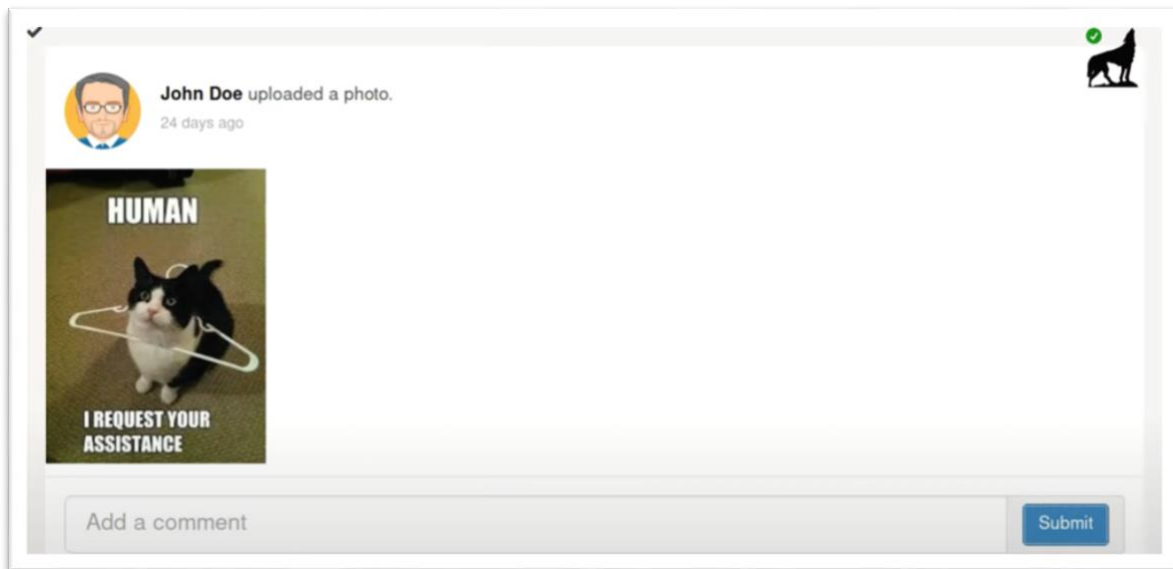
Gracias por comprar en WebGoat.

Agradecemos su apoyo.

Hemos cargado la tarjeta de crédito:

\$1997.96

5.9. Vulnerabilidad XXE



5.10. Archivo XML Malicioso

✓

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>calc.exe</string>
    </command>
  </target>
  <action>start</action>
</handler>
</contact>
```

Go!

You successfully tried to exploit the CVE-2013-7285 vulnerability
java.io.IOException: Cannot run program "calc.exe": error=2, No such file or directory

5.11. Contraseña Alta Seguridad

Enter a secure password...

Show password

Submit

You have succeeded! The password is secure enough.

Your Password: *****

Length: 12

Estimated guesses needed to crack your password: 936100000000

Score: 4/4

Estimated cracking time: 2968 years 129 days 1 hours 46 minutes 40 seconds

Score: 4/4

5.12. Contraseña Baja Seguridad

12345

Show password

Submit

You have failed! Try to enter a secure password.

Your Password: *****

Length: 5

Estimated guesses needed to crack your password: 7

Score: 0/4

Estimated cracking time: 0 years 0 days 0 hours 0 minutes 0 seconds

Warning: This is a top-10 common password.

Suggestions:

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

Score: 0/4