Práctica introducción a la ciberseguridad

Análisis de vulnerabilidades en la aplicación WebGoat

Informe destinado a Keepcoding
25/04/2021

Realizado por Yesurún González Román
Contacto: yesurn_g@hotmail.com

ÍNDICE

1.Introducción

a) Tratamiento del documento

Este Informe tiene como propósito aportar a Keepcoding, documentación sobre fallos de seguridad en la aplicación WebGoat.
La información que se presenta a continuación es confidencial y no debe ser modificada o distribuida a entidades o personas ajenas a la empresa Keepcoding.

b) Información del documento

En esta seccíon se detalla a las partes involucradas del proyecto, y las fases de desarrollo del mismo:

Cliete: Keepcoding

Autor: Yesurún González Román

Proyecto: Exposición de fallos de seguridad

- Realización de pruebas de seguridad sobre la aplicación WebGoat: 20/04/2021
- Redacción de este documento: 24/04/2021
- Envío del documento al cliente: 25/05/2021

2. Ámbito y alcance

a) Breve resumen

Para poder analizar la aplicación se realizó el siguiente comando en la terminal local:

- docker run -p 8080:8080 -p 9090:9090 -e TZ=Europe/Amsterdam webgoat/goatandwolf

De este modo la comunicación se hace posible a través del contenedor docker por los puertos tcp.8080 y tcp.9090 en servidor local.
WebGoat utiliza el lenguaje de programación Java en su versión 11.0.1.

Todas las vulnerabilidades que se mencionan a continuación residen en el siguiente dominio:

- http://127.0.0.1:8080/WebGoat

Para poder realizar las pruebas me he inscrito como usuario "admin1" con contraseña "admin1".

b) Metodología utilizada

El navegador empleado a lo largo del proceso ha sido Mozilla Firefox versión 78.7.0

Una vez permitido mi acceso con usuario y contraseña me he ido desplazando por las diferentes secciones con sus respectivos apartados de forma ascendente.

c) Vulnerabilidades destacadas

Las vulnerabilidades presentes en WebGoat tienen la siguiente denominación:

- **Inyecciones SQL:** Uso de comandos que modifican o permiten la obtencion de información de la base de datos, por parte de usuarios no autenticados o sin privilegios.

- **Referencias de objeto directo inseguras:** Un usuario tiene acceso a información concerniente a otros usuarios de su mismo nivel de privilegios

- **Pérdida de control de acceso según nivel de funciones:** Un usuario puede adquirir información de cualquier tipo con privilegios de administrador de forma ilegítima.

- **Secuencia de comandos en sitios cruzados:** Un usuario puede inyectar código en en la parte de código frontend que se ejecuta en el dispositivo de otro usuario.

d) Conclusión

La aplicación WebGoat alberga varios fallos de seguridad que comprometen la confidencialidad e integridad de los usuarios que tiene registrados.

e) Recomendaciones

Estos fallos de seguridad pueden ser corregidos mediante la implementación de las siguientes medidas:

- **Inyecciones SQL**: Importar e incorporar librerías al código fuente que aplican filtrado a las consultas impidiendo el uso de comandos.

- **Referencias de objeto directo inseguras:** Deshabilitar accesos de ruta privadas a usuarios con diferente número de identificación. Uso de encriptación de las rutas privadas para los usuarios con rol de administrador.

- **Pérdida de control de acceso según nivel de funciones:** Medidas que comprueben la autentificación en peticiones sobre información sensible.

- **Secuencia de comandos en sitios cruzados:** Importar e incorporar librerías al código fuente que aplican filtrado a las consultas impidiendo el uso de comandos.

f) Nivel de riesgo

De no aplicarse medidas de seguridad para estos fallos, WebGoat permanecería susceptible a ataques de ciberdelincuentes. Estos ataques pueden suponer la pérdida o filtración de información sensible, dañando la confianza de los clientes y pudiendo implicar incluso (en el peor de los casos) sanciones económicas.

3. Descripción del proceso

a) Reconocimiento

Los fallos de seguridad que detallaré en el siguiente epígrafe, los he descubierto en las siguientes secciones con sus respectivos apartados:

- Sección SQL injection (A1). Apartados 2, 3, 4, 5, 9, 10, 11, 12, 13.
- Sección broken access control IDOR (A5). Apartados 3, 4, 5.
- Sección broken access control MFLAC (A5). Apartados 2, 3.
- Sección cross-site scripting (A7). Apartado 7.

b) Explotación

A1

2. Utilizando el comando **select department from employees where userid= 96134** averiguo que Bob Franco trabaja en el departamento de marketing.

3. Utilizando el comando **update employees set department = 'Sales' where userid= 89762** cambio el departamento en el que trabaja Tobi Barnett en la base de datos.

4. Utilizando el comando **alter table employees add phone varchar(20)** añado columna gracias a que tengo privilegios garantizados.

5. Utilizando el comando **grant alter table to UnauthorizedUser** adhiero a cualquier usuario la capacidad de alterar tablas.

9. Haciendo una petición en la que pongo **Smith' or '1'='1** obtengo información de todos los usuarios por forzar "true" ya que el comando predefinido está estructurado así: SELECT * FROM user_data WHERE first_name = 'John' and last_name = '**Smith' or '1' = '1**'.

10. Rellenando el campo User_Id con **1 or 1=1** y Login_Count con **1** obtengo información de todos los usuarios por forzar "true" ya que el comando predefinido está estructurado así: SELECT * From user_data WHERE Login_Count = **1** and userid= **1 or 1=1**.

11. Rellenando el campo Employee Name con **Smith** y Authentication TAN con **3SL99A or '1' = '1** obtengo información de todos los usuarios por forzar "true" ya que el comando predefinido está estructurado así: "SELECT * FROM employees WHERE last_name = '"**Smith**"' AND auth_tan = '"**3SL99A or '1' = '1**"'

12. Rellenando los campos employees name con **Smith** y TAN con **3SL99A'; update employees set salary =100000 where last_name = 'Smith** consigo que el usuario Smith quede registrado en la base de datos con un salario de 100.000

13. Rellenando el campo de búsqueda de logs con el comando **Smith'; drop table access_log--** consigo eliminar la tabla de logs para quitar rastros.

A5 IDOR

3. Al utilizar el proxy burp, veo que se realiza la petición GET /WebGoat/IDOR/profile luego al poner **127.0.0.1:8080/WebGoat/IDOR/profile** se descubren dos nuevos atributos no mostrados al usuario; role y userId

4. Poniendo la URL **127.0.0.1:8080/WebGoat/IDOR/profile/2342384** se obtiene acceso a información de usuario sin autenticar y de manera directa

5. Incrementando el número userId uno a uno encuentro información del usario Buffalo Bill en 127.0.0.1:8080/WebGoat/IDOR/profile/**2342388**

(A5) MFLAC

2. En el codigo html se encuentra información accesible al usuario sobre rutas terminadas en **/users** y **/config**

3. Mandando una petición GET desde el repeater de Burp a localHost:8080/WebGoat**/users** y modificando accept: a accept: **application/json** y añadiendo **Content-Type: application/json;charset=UTF-8** se obtiene los usuarios con sus hash de contraseña

(A7)

1. El campo donde se pone el número de tajeta de crédito es susceptible de ataque XSS. Un ejemplo de comando sería **<script>alert('xxx')</script>**

4. Herramientas Utilizadas

Para poder explotar ciertas vulneravilidades es necesario el uso de un proxy para obtener información o modificar las peticiones que se lancen a WebGoat.

En mi caso he utilizado Burp operando con el puerto 8081.