

# **INFORME**

## **DE**

# **AUDITORIA**

**WEB GOAT** T

# **INDICE**

- 1. INTRODUCCION**
- 2. AMBITO Y ALCANCE DE LA AUDITORIA**
- 3. INFORME EJECUTIVO**
  - 3.A. RESUMEN DEL PROCESO REALIZADO**
  - 3.B. VULNERABILIDADES DESTACADAS**
  - 3.C. CONCLUSIONES**
  - 3.D. RECOMENDACIONES**
- 4. PROCESO DE AUDITORÍA**
  - 4.A. RECONOCIMIENTO/INFORMATION GATHERING**
  - 4.B. EXPLOTACIÓN DE VULNERABILIDADES DETECTADAS**
  - 4.C. POST-EXPLOTACIÓN**
  - 4.D. POSIBLES MITIGACIONES**
  - 4.E. HERRAMIENTAS UTILIZADAS**

## 1. INTRODUCCION

En este informe, detallaremos las vulnerabilidades descubiertas en la aplicación Web Goat, así como los procedimientos y herramientas empleados para detectar dichos fallos.

Nuestro equipo hizo una evaluación exhaustiva de la aplicación, con la meta principal de identificar y documentar cualquier vulnerabilidad existente.

Seguimos un enfoque sistemático que abarcó una serie de procesos y la utilización de diversas herramientas para identificar posibles fallos de seguridad. Además, proporcionaremos recomendaciones destinadas a mejorar la seguridad de la aplicación.

## 2. AMBITO Y ALCANCE DE LA AUDITORIA

El alcance de la auditoría es explotar las siguientes vulnerabilidades de Web Goat:

- A3 Injection - SQL Injection (intro) apartado 10
- A3 Injection -SQL Injection (intro) apartado 11
- A3 Injection- Cross Site Scripting apartado 10
- A5 Security Misconfiguration apartado 4
- A5 Security Misconfiguration apartado 7
- A6 Vuln & outdated Components apartado 5
- A7 Identity & Auth Failure -Secure passwords 4

Nos enfocaremos en abordar cada una de estas vulnerabilidades con el propósito de solucionar los fallos identificados en cada sección. Emplearemos diversas herramientas y códigos para detectar y comprender estas vulnerabilidades, permitiéndonos tomar decisiones informadas sobre cómo podemos fortalecer la seguridad.

La meta principal es abordar y resolver todos los problemas identificados, implementando las medidas de seguridad necesarias para fortificar la aplicación web y prevenir posibles ataques.

### **3. INFORME EJECUTIVO**

#### **3.A RESUMEN DEL PROCESO REALIZADO**

Hemos realizado los siguientes pasos:

1. Entrar en la aplicación web a través de la herramienta Docker.
2. Identificar los puertos abiertos.
3. Identificar el sistema operativo
4. Reconocer el lenguaje de programación
5. Explotacion de vulnerabilidades.

### *3.A.1 ABRIR LA APLICACIÓN WEB A TRAVÉS DE DOCKER*

Hemos ejecutado el entorno a través de 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 ● <http://127.0.0.1:8080/WebGoat>

### 3.A.2 PUERTOS ABIERTOS

A través de la función nmap 127.0.0.1 hemos podido identificar que puertos estaban abiertos.

- 8080/tcp open http-proxy
- 8081/tcp open blackice-icecap
- 9090/tcp open zeus-admin

The terminal window shows the following Nmap output:

```
(kali㉿kali)-[~]
$ docker start webgoat
webgoat
(kali㉿kali)-[~]
$ pwd
/home/kali
(kali㉿kali)-[~]
$ nmap -p- 127.0.0.1
Starting Nmap 7.94SVN ( https://nmap.org ) at 2023-12-07 05:29 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00063s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE
8080/tcp  open  http-proxy
9090/tcp  open  zeus-admin
41153/tcp open  unknown

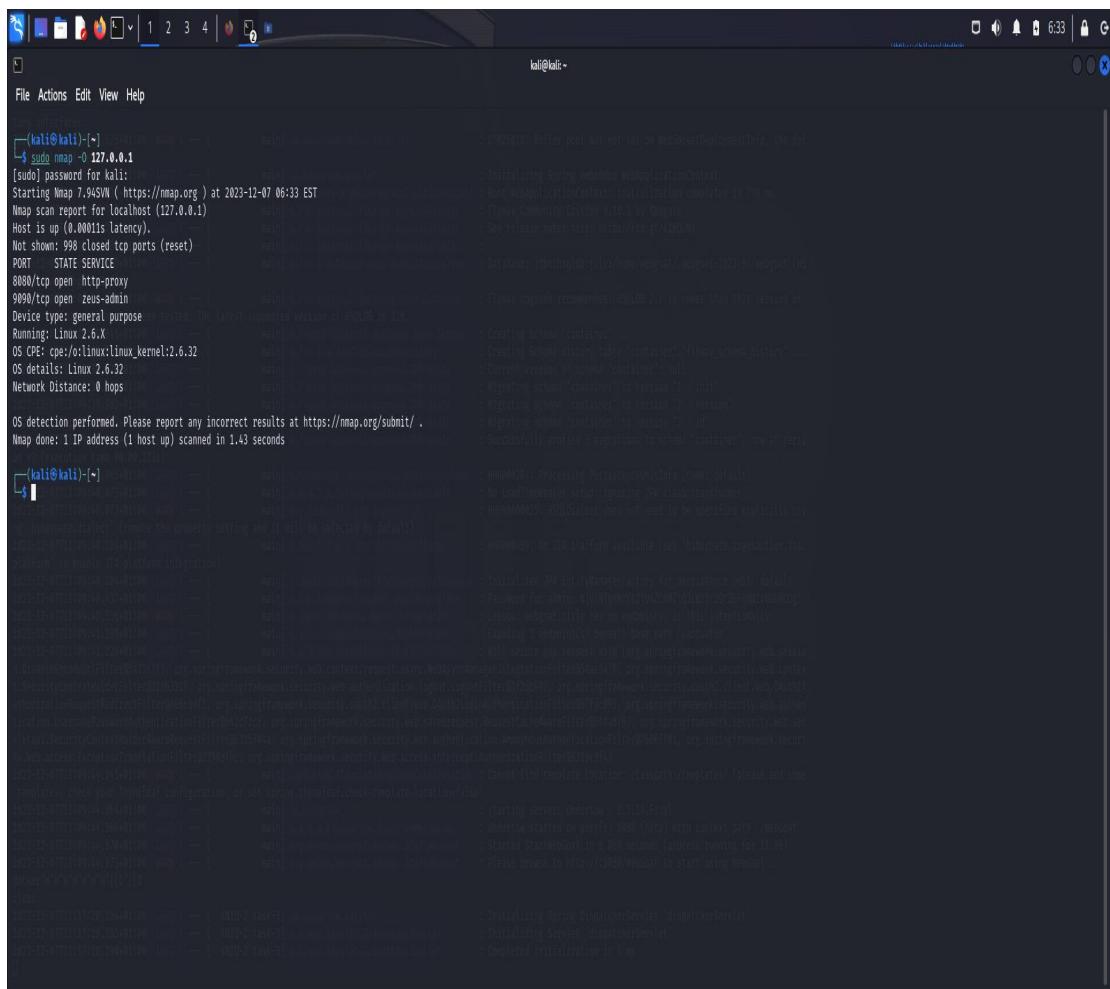
Nmap done: 1 IP address (1 host up) scanned in 1.90 seconds
```

The browser window shows a login page for "WEBGOAT". The URL is <http://127.0.0.1:8080/webgoat/>. The page has fields for "Username" and "Password", and a "Sign In" button. To the right, there is a sidebar with various menu items.

### 3.A.3 SISTEMA OPERATIVO

Introduzco la función nmap -O 127.0.0.1 e identifico el sistema operativo.

- Running: Linux 2.6.X
- OS CPE: cpe:/o:linux:linux\_kernel:2.6.32
- OS details: Linux 2.6



```
(kali㉿kali)-[~] $ sudo nmap -O 127.0.0.1
[sudo] password for kali:
Starting Nmap 7.94SN ( https://nmap.org ) at 2023-12-07 06:33 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0001s latency).
No show: 998 closed tcp ports (reset)
PORT      STATE SERVICE
8080/tcp  open  http-proxy
9990/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 1.43 seconds
```

### ***3.A.4 SISTEMA OPERATIVO***

A través de la aplicación Wappalyzer identificamos el lenguaje de programación de la web.

-JAVA

### 3.A.5 ATAQUES

#### PRIMER ATAQUE

##### **A3 Injection - SQL Injection (intro) - Apartado 10**

Para atacar este apartado hemos utilizado la consulta: *SELECT \* From user\_data WHERE Login\_Count = 0 and userid= 1 or 1=1.*

#### SEGUNDO ATAQUE

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

En este apartado queríamos acceder a la base de datos donde aparecen los salarios de cada trabajador.

Lo hemos conseguido construyendo una consulta SQL:

```
"SELECT * FROM empleados WHERE apellido = '" + nombre + "' AND  
auth_tan = '" + auth_tan + "'";
```

## TERCER ATAQUE

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

En esta sección, empleamos la herramienta "Burp Suite" y observamos cómo el número de la tarjeta de crédito se almacenaba y resultaba fácilmente identifiable.

Para llevar a cabo el ataque, utilizamos el siguiente script:

"<script>alert("1")</script>", con el fin de evaluar la posibilidad de exponer el número de tarjeta, dado que cualquier campo de datos devuelto al cliente tiene el potencial de ser susceptible a inyecciones.

## CUARTO ATAQUE

### A5 Security Misconfiguration - Apartado 4

La herramienta que empleamos para llevar a cabo el ataque es "**Burp Suite**", la cual posibilitó la interceptación del código de solicitud de comentarios. Posteriormente, modificamos este código e incorporamos consultas como "<!DOCTYPE user [<!ENTITY root SYSTEM "file:///> ]>" y "<comment><text>&root;test</text></comment>". Gracias a estas acciones, conseguimos alcanzar nuestro objetivo de interferir en el procesamiento de datos XML de la aplicación web.

## QUINTO ATAQUE

### A5 Security Misconfiguration - Apartado 7

En esta sección, aprovechamos la herramienta "**Burp Suite**" para interceptar los códigos generados por la aplicación web. Posteriormente, implementamos un ataque mediante la adición del siguiente fragmento:

```
<?xml version="1.0"?><!DOCTYPE user [<!ENTITY root SYSTEM  
"file:///> ]><comment><text>&xxe;</text></comment>". Además,  
llevamos a cabo una modificación del "content type" de "JSON" a "XML",  
lo cual nos posibilitó bloquear los comentarios de tipo "JSON".
```

## SEXTO ATAQUE

### A6 Vuln & outdated Components - Apartado 5

En este segmento, notamos la utilización del mismo código fuente en "*Webgoat*" pero con distintas versiones del componente "*jquery-ui*". Una de estas versiones es vulnerable a explotación, mientras que la otra no lo es.

En la versión *query-ui:1.10.4*, al inyectar una consulta como "OK<script>alert('XSS')</script>", se vuelve susceptible a un ataque XXE, tal como se evidencia en la captura de pantalla. En contraste, con la versión actualizada *query-ui:1.12.0* y utilizando el mismo código empleado en la versión anterior, "OK<script>alert('XSS')</script>", observamos que no es vulnerable al ataque, eliminando así el exploit.

## SÉPTIMO ATAQUE

### A7 Identity & Auth Failure - Secure Passwords Apartado 4

En este segmento, evaluamos la robustez de una contraseña y verificamos si cumple con los requisitos de seguridad establecidos. La herramienta proporciona información sobre el tiempo estimado necesario para descifrar la contraseña creada, su longitud y la puntuación alcanzada.

No obstante, al probar las contraseñas sugeridas, observamos que estas son fácilmente descifrables. En este caso, utilizamos la palabra "password", que figura entre las 10 contraseñas más comunes y, por ende, podría ser descubierta con facilidad.

### **3.B VULNERABILIDADES DESTACADAS**

Destacamos que "Web Goat" presenta una notable vulnerabilidad, manifestándose a través de diversas deficiencias significativas:

- 1. Permite la inserción de instrucciones SQL maliciosas, comprometiendo las medidas de seguridad y facilitando el acceso a información protegida.**
- 2. La manipulación o eliminación de datos es fácilmente realizada en la plataforma.**
- 3. Acceso y modificación directa del código de la aplicación.**
- 4. Existe la posibilidad de sustraer credenciales de usuarios registrados en la aplicación.**
- 5. Se puede llevar a cabo la deshabilitación del sistema de manera sencilla.**
- 5. Vulnerabilidad a ataques de falsificación de solicitudes del lado del servidor.**
- 7. Susceptibilidad a ataques XSS (Cross-Site Scripting), un método en el cual actores malintencionados inyectan scripts maliciosos en la web con el objetivo de robar datos personales, cookies de sesión y emplear tácticas de ingeniería social, entre otros.**
- 8. Presencia de componentes desactualizados y vulnerables, lo cual ocurre cuando un componente de software no está en consonancia, está desactualizado o es vulnerable a exploits conocidos.**

### 3.C. CONCLUSIONES

WebGoat representa una aplicación de insegura, construida en el lenguaje de programación Java y de código abierto. Durante nuestras pruebas, llevamos a cabo diversos ataques para evaluar sus vulnerabilidades.

Logramos sortear las medidas de seguridad, acceder a datos confidenciales y intervenir en el procesamiento de datos XML, incluso interceptando y modificando porciones específicas del código.

Es evidente que la plataforma es susceptible a inyecciones SQL, lo que implica la posibilidad de modificar o eliminar datos, planteando así un riesgo significativo para la seguridad.

Además, verificamos que la aplicación utiliza programas desactualizados, dejándola expuesta a posibles "exploits".

En consecuencia, la aplicación presenta una notable fragilidad ante diversos tipos de ataques, haciendo que la base de datos asociada esté desprotegida y accesible para cualquier individuo con conocimientos básicos. En este sentido, se recomienda implementar mejoras sustanciales en el servidor para convertirla en una aplicación más segura.

### **3.D RECOMENDACIONES**

Aquí te dejo una redacción alternativa:

Para elevar el nivel de seguridad de la aplicación, proponemos las siguientes acciones:

#### **1. Mantenimiento de Software:**

- Mantener actualizados los programas y software utilizados para garantizar la incorporación de las últimas correcciones de seguridad.

#### **2. Optimización del Código:**

- Optimizar el código fuente de la aplicación de código abierto para mejorar su eficiencia y seguridad.

#### **3. Configuración de Analizadores XML:**

- Configurar analizadores XML para rechazar definiciones de documentos personalizados (DTD) y fortalecer la seguridad contra ataques.

#### **4. Reconstrucción de Cargas Útiles:**

- Reconstruir completamente las cargas útiles para identificar y prevenir ataques de expansión de entidades XML (XXE).

## **5. Uso de Procedimientos Almacenados:**

- Implementar procedimientos almacenados para prevenir ataques de inyección SQL y asegurar los campos de entrada contra accesos no autorizados a la base de datos.

## **6. Consulta Parametrizada:**

- Utilizar consultas parametrizadas que empleen variables en lugar de constantes en las cadenas de consulta SQL para fortalecer la seguridad.

## **7. Encriptación de Datos:**

- Implementar la encriptación de datos para restringir el acceso a la información solo a personas autorizadas mediante autenticación adecuada.

## **8. Contraseñas Seguras y Cambio Regular:**

- Establecer contraseñas robustas y cambiarlas de forma regular para mantener la integridad del sistema.

## **9. Control y Registro de Acciones:**

- Monitorear y registrar todas las acciones y movimientos en la base de datos para identificar posibles manipulaciones de información y mantener un registro de auditoría.

## **10. Eliminación de Elementos Innecesarios:**

- Eliminar funcionalidades, componentes, archivos y documentación que no sean esenciales para reducir la superficie de ataque.

## **11. Revisión de Componentes sin Parches de Seguridad:**

- Evaluar los componentes que no reciben actualizaciones de seguridad para versiones anteriores y, si es necesario, implementar soluciones como parches virtuales para monitorear, detectar o protegerse contra problemas identificados.

## 4. PROCESO DE AUDITORIA

### 4. A. RECONOCIMIENTO/INFORMATION GATHERING

#### A3 Injection - SQL Injection (intro) - Apartado 10

En esta sección, identificamos que la tabla de datos que almacena información sobre los empleados presenta vulnerabilidades frente a inyecciones SQL, las cuales son instrucciones maliciosas. Estas inyecciones tienen como objetivo eludir las medidas de seguridad establecidas para acceder a datos protegidos.

Tabla:

Employees Table

userid	first_name	last_name	department	salary	auth_tan
32147	Paulina	Travers	Accounting	\$46.000	P45JSI
89762	Tobi	Barnett	Development	\$77.000	TA9LL1
96134	Bob	Franco	Marketing	\$83.700	LO9S2V
34477	Abraham	Holman	Development	\$50.000	UU2ALK
37648	John	Smith	Marketing	\$64.350	3SL99A

Con esta información podemos recuperar o acceder a datos simplemente con el nombre del usuario a través de consultas SQL.

Ejemplo:

```
SELECT * FROM user_data WHERE login_count = " + Login_Count + " AND userid = " + User_ID;
```

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

En esta sección, al igual que en la anterior, mediante la tabla que contiene detalles de los usuarios, tenemos la posibilidad de obtener información comprometida de distintos empleados mediante el uso de consultas SQL.

Ejemplo:

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

## A3 Injection - Cross Site Scripting - Apartado 7

En este apartado hemos averiguado que es posible realizar ataques XXS por lo que se puede crear una URL con el script de ataque y publicarla en otro sitio web.

Utilizando este Scripting: "" podemos dejar expuesta los nº de tarjeta.

Shopping Cart			
Shopping Cart Items .. To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tiling Surface - Cherry	69.99	70	\$0.00
Dynex - Traditional Notebook Case	27.99	38	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1600	\$0.00
3 - Year Performance Service Plan \$1000 and Over	299.99	300	\$0.00

Enter your credit card number:

Enter your three digit access code:

## A5 Security Misconfiguration - Apartado 4

En esta sección, observamos que la aplicación es vulnerable a ataques de expansión de entidades XML (XXE), lo que posibilita la manipulación y la interferencia en el procesamiento de datos XML mediante la siguiente consulta.

`<!DOCTYPE user [<!ENTITY root SYSTEM "file:///>]>" y  
<coment><text>&root:test</text></comment>"`

The screenshot shows the Burp Suite Community Edition interface. On the left, a browser window displays a challenge from the WebGoat application. The URL is `localhost:8080/WebGoat/start.mvc?username=kikeee#lesson/XXE.lesson/3`. The challenge involves adding a comment to a photo uploaded by John Doe. The photo is a black and white cat with the caption "HUMAN" and the text "I REQUEST YOUR ASSISTANCE". Below the photo is a comment input field with placeholder "Add a comment" and a "Submit" button. There are three previous comments from "kikeee": "esto es un gato" (2023-12-10, 09:29:35), "esto es un gato" (2023-12-10, 09:23:23), and another identical entry (2023-12-10, 09:23:23). On the right side of the Burp interface, there is a sidebar with various tabs like Collaborator, Sequencer, Decoder, Compare, Logger, Organizer, Extensions, Learn, and Settings. A status message at the bottom right says "Intercept is off".

## A5 Security Misconfiguration - Apartado 7

En este apartado hemos averiguado que se puede interceptar los códigos de la aplicación web y poder añadir el siguiente ataque:

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

y modificar “content type” de “JSON” a “XML”

> endpoints being vulnerable to XXE attacks.  
> Again same exercise but try to perform the same XML injection as we did in first assignment.  
>

Add a comment Submit

**ttmpruebas** 2019-01-23, 21:57:11  
.cache 0 bin boot dev etc home initrd.img initrd.img.old lib lib32 lib64 libx32 lost+found media mint opt proc root run sbin srv sys tmp usr var vmlinuz vmlinuz.old Este gato

**ttmpruebas** 2019-01-23, 21:42:40  
.cache 0 bin boot dev etc home initrd.img initrd.img.old lib lib32 lib64 libx32 lost+found media mint opt proc root run sbin srv sys tmp usr var vmlinuz vmlinuz.old

Modern REST framework

In modern REST frameworks the server might do not think about. So this might result in XXE attacks.

Again same exercise but try to perform the same XML injection as we did in first assignment.

**John Doe uploaded a photo**  
24 days ago

**HUMAN**  
I REQUEST YOUR ASSISTANCE

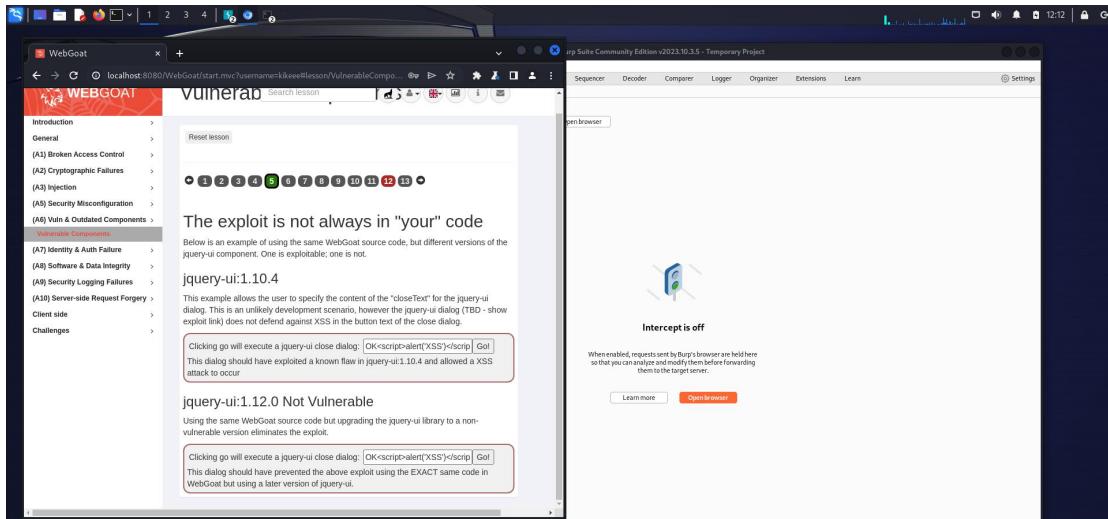
estoy en un gato

**webgoat** 2023-12-10, 08:43:05  
Silly cat...

**guest** 2023-12-10, 08:43:05  
i think I will use this picture in o

**guest** 2023-12-10, 08:43:05  
Lol!! :-)

## A6 Vuln & outdated Components - Apartado 5

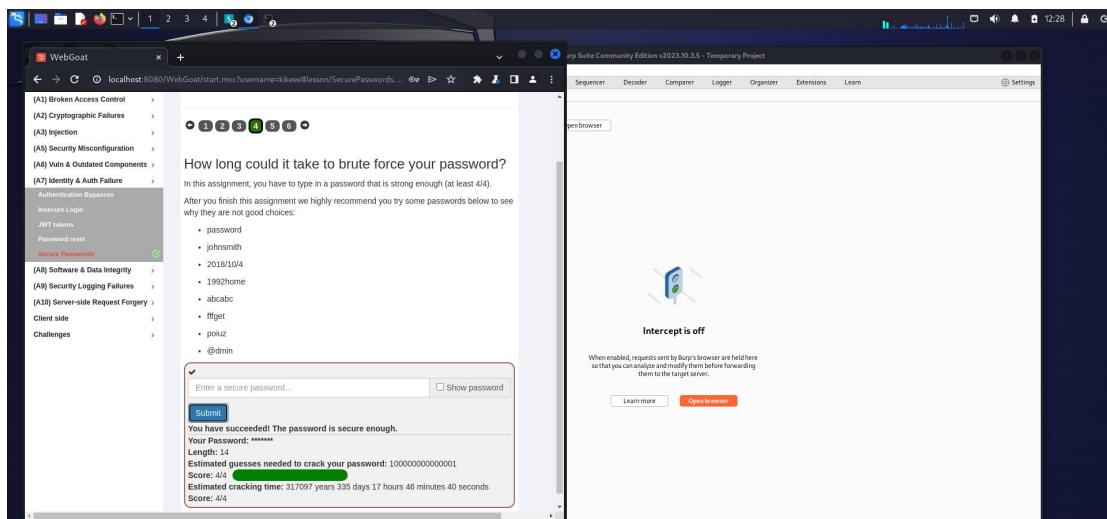


En esta investigación, hemos examinado la presencia de componentes desactualizados o susceptibles a posibles "exploits". Observamos que el código fuente utilizado en "WebGoat" es el mismo, pero existen diferentes versiones del componente "jquery-ui". Una de ellas es explorable, mientras que la otra no.

Especificamente, la versión "jquery-ui:1.10.4" es propensa a un ataque de expansión de entidades XML (XXE). Sin embargo, en la versión actualizada "jquery-ui:1.12.0", no existe vulnerabilidad frente a este tipo de ataque, lo que implica la eliminación del posible "exploit".

## A7 Identity & Auth Failure - Secure Passwords Apartado 4

En este apartado hemos podido recolectar información sobre el tipo de contraseñas que deberían usar los usuarios, hemos podido detectar como hay contraseñas muy inseguras y por lo tanto fáciles de encontrar.



## 4.B. EXPLOTACIÓN DE LAS VULNERABILIDADES DETECTADAS.

### A3 Injection - SQL Injection (intro) - Apartado 10

Para llevar a cabo el ataque en esta sección, empleamos la siguiente consulta: `SELECT \* FROM user\_data WHERE Login\_Count = 0 AND userid = 1 OR 1=1`. La instrucción SELECT nos permite recuperar registros de la base de datos.

The screenshot shows the Burp Suite interface with the following details:

- Left Panel (Menu):** Cross Site Scripting (Unencoded), Cross Site Scripting (Encoded), Path traversal, (A) Security Misconfiguration, (A) Vuln & Outdated Components, (A7) Identity & Auth Failure, (A8) Software & Data Integrity, (A9) Security Logging Failures, (A10) Server-side Request Forgery, Client side, Challenges.
- Middle Panel (Request/Response):**
  - Request:** "SELECT \* FROM user\_data WHERE login\_count = " + Login\_Count + " AND user\_id = 1 OR 1=1".
  - Response:** A form with fields 'Login\_Count' and 'User\_Id'. Below it, a success message: "You have succeeded: USERID, FIRST\_NAME, LAST\_NAME, CC\_NUMBER, CC\_TYPE, COOKIE, LOGIN\_COUNT, 101, Joe, Snow, 987654321, VISA, , 0, 101, Joe, Snow, 2234200065411, MC, , 0, 102, John, Smith, 2435600002222, MC, , 0, 102, John, Smith, 33330000000000000000000000000000, AMEX, , 0, 103, Jane, Plane, 123456789, MC, , 0, 103, Jane, Plane, 333498765333, AMEX, , 0, 10312, Jolly, Hershey, 17696789, MC, , 0, 10312, Jolly, Hershey, 333300003333, AMEX, , 0, 10323, Grumpy, youaretheweakestking, 678334489, MC, , 0, 10323, Grumpy, youaretheweakestking, 33413003333, AMEX, , 0, 15603, Peter, Parker, 33330000000000000000000000000000, MC, , 0, 15603, Peter, Parker, 33330000000000000000000000000000, AMEX, , 0, 15812, Jeosp, Something, 3384545533, AMEX, , 0, 15837, Chaos, Monkey, 3264936533, CM, , 0, 19204, Mr, Goat, 3312953533, VISA, , 0, Your query was: SELECT \* From user\_data WHERE Login\_Count = 0 and userid= 0 or 1=1".
- Right Panel (Tools):** Intercept is off, Learn more, Open browser.

proporcionamos una condición que sea "true" (verdadera), toda la consulta se evalúa como "true". Esto nos permite anular la condición "WHERE" y obtener todos los resultados de la base de datos.

Finalmente, la condición '1'='1 siempre se cumple, ya que el número 1 es siempre igual a 1.

Al llevar a cabo este ataque, observamos cómo pudimos acceder a una base de datos que contiene información sobre las tarjetas de crédito de los empleados.

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

En esta sección, nuestro objetivo era acceder a la base de datos que contiene los salarios de cada empleado. Logramos este acceso construyendo una consulta SQL: "SELECT \* FROM empleados WHERE apellido = "" + nombre + "" AND auth\_tan = "" + auth\_tan + """.

The screenshot shows a browser window for the WebGoat application at localhost:8080/WebGoat/start.mvc?username=lkeeee#lesson/SqlInjection.lesson. The page displays a challenge message: "It is your turn! You are an employee named John Smith working for a big company. The company has an internal system that allows all employees to see their own internal data such as the department they work in and their salary. 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 =
```

The form fields are:

- Employee Name:
- Authentication TAN:
- Get department

The response message says: "You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!" Below this, a table of employee data is shown:

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN
32147	Paulina	Travers	Accounting	46000	P45JSI
34477	Abraham	Holman	Development	50000	UU24LK
37648	John	Smith	Marketing	64350	3SL99A
89762	Tobi	Barnett	Development	77000	TA9LL1
96134	Bob	Franco	Marketing	83700	LC952V

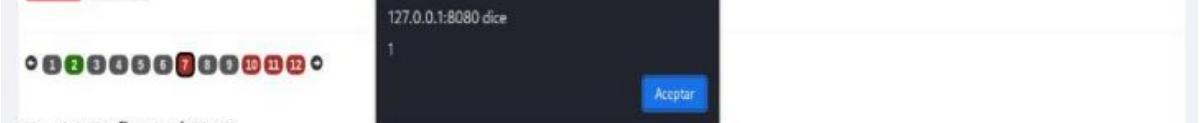
The Burp Suite interface shows the status bar: "Intercept is off". A tooltip for "Intercept is off" explains: "When enabled, requests sent by Burp's browser are held here so that you can analyze and modify them before forwarding them to the target server." There are "Learn more" and "Open browser" buttons.

Empleamos el operador "OR" con el propósito de transformar la consulta en verdadera, anulando la función "WHERE" mediante la condición "1=1", la cual siempre es verdadera. De esta manera, conseguimos obtener los datos de la tabla deseada.

## A3 Injection - Cross Site Scripting - Apartado 7

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. The 'Intercept' button is enabled. The 'HTTP history' section displays a list of requests and responses. A specific request from port 8080 is highlighted, showing a GET request to '/WebGoat/CrossSiteScripting/attack5a?QTY1=1&QTY2=1&QTY3=1&QTY4=1&field2=111 HTTP/1.1'. The response body contains JSON data related to a shopping cart item and an alert message. The 'Request' and 'Response' panes show the raw and pretty-printed versions of the captured message. The 'Inspector' pane on the right shows various request attributes like 'Request attributes', 'Request query parameters', and 'Request cookies'.

podemos ver como el numero de la tarjeta de crédito se queda guardado y es fácilmente detectable con "Burp Suite".



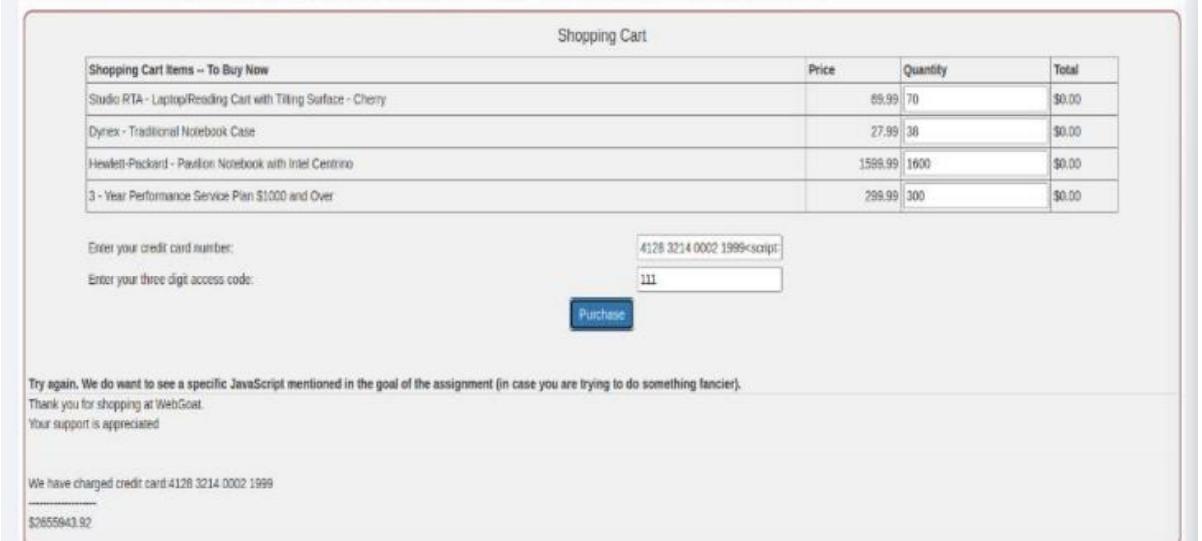
The screenshot shows a dice rolling application at 127.0.0.1:8080/dice. The interface includes a row of numbered buttons from 1 to 12, with the number 7 highlighted in red. Below the buttons is a large black box containing the number '1'. A blue button labeled 'Aceptar' (Accept) is located in the bottom right corner of the black box.

**Try It! Reflected XSS**

The assignment's goal is to identify which field is susceptible to XSS.

It is always a good practice to validate all input on the server side. XSS can occur when unvalidated user input gets used in an HTTP response. In a reflected XSS attack, an attacker can craft a URL with the attack script and post it to another website, email it, or otherwise get a victim to click on it.

An easy way to find out if a field is vulnerable to an XSS attack is to use the `alert()` or `console.log()` methods. Use one of them to find out which field is vulnerable.



The screenshot shows a shopping cart application titled "Shopping Cart". The table lists four items:

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	\$9.99	70	\$0.00
Dynex - Traditional Notebook Case	\$27.99	38	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	\$159.99	1600	\$0.00
3 - Year Performance Service Plan \$1000 and Over	\$299.99	300	\$0.00

Below the table, there are two input fields:

- "Enter your credit card number:" with value: 4128 3214 0002 1999<script>
- "Enter your three digit access code:" with value: 123

A blue "Purchase" button is located below the input fields.

Text at the bottom of the page:

Try again. We do want to see a specific JavaScript mentioned in the goal of the assignment (in case you are trying to do something fancier).  
 Thank you for shopping at WebGoat.  
 Your support is appreciated.

We have charged credit card 4128 3214 0002 1999  
 -----  
 \$2655643.92

Aquí podemos observar cómo atacando con este scripting:

“<script>alert(“1”)</script>” podemos dejar expuesta el nº de tarjeta ya que cualquier campo de datos devuelto al cliente es potencialmente inyectable.

## A5 security misconfiguration - apartado 4

Utilizamos la herramienta "Burp Suite" para llevar a cabo nuestros ataques. Esta herramienta nos facilitó la capacidad de interceptar el código de solicitud relacionado con los comentarios. Posteriormente, pudimos modificarlo al añadir consultas como "<!DOCTYPE user [<!ENTITY root SYSTEM "file:///">>]" y "<comment><text>&root;test</text></comment>". Gracias a estas modificaciones, logramos con éxito el objetivo de interferir en el procesamiento de datos XML de la aplicación web.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A request to `http://localhost:8080/` is being viewed. The 'Raw' tab of the request editor contains the following XML payload:

```
1 GET /WebGoat/service/lessonmenu.mvc HTTP/1.1
2 Host: localhost:8080
3 sec-ch-ua: "Chromium";v="119", "Not?A Brand";v="24"
4 Accept: application/json, text/javascript, */*; q=0.01
5 X-Requested-With: XMLHttpRequest
6 sec-ch-ua-mobile: 70
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36
8 sec-ch-ua-platform: "Linux"
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://localhost:8080/WebGoat/start.mvc?username=kikeeee
13 Accept-Encoding: gzip, deflate, br
14 Accept-Language: en-US,en;q=0.9
15 Cookie: JSESSIONID=251Nxe405GvlpjVN79z70tK2tUH3_XBBHgWw
16 Connection: close
17
18
```

The 'Inspector' panel on the right shows various request details such as attributes, query parameters, body parameters, cookies, and headers. The 'Notes' section is empty.

## A5 Security Misconfiguration - Apartado 7

En esta sección, empleamos la herramienta "Burp Suite" para interceptar los códigos de la aplicación web. Luego, implementamos el siguiente ataque: "<?xml version="1.0"?><!DOCTYPE user [<!ENTITY root SYSTEM "file:///">> ]><comment><text>&xxe;</text></comment>".

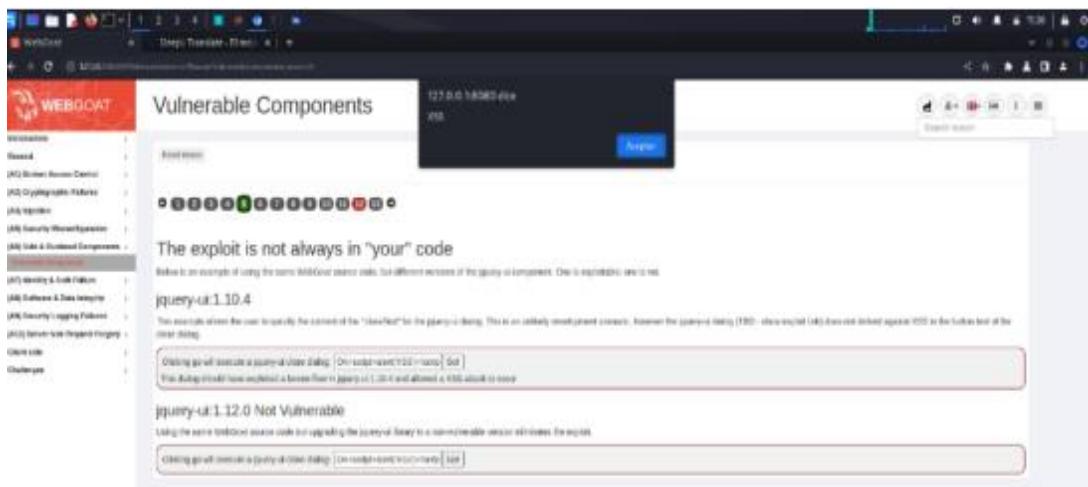
Además, realizamos la modificación del "content type" de "JSON" a "XML", lo que nos posibilitó bloquear los comentarios de tipo "JSON".

The screenshot shows the Burp Suite interface with the following details:

- Burp Suite Community Edition v2023.10.3.5 - Temporary Project**
- Request**: Request to `http://localhost:8080 [127.0.0.1]` (Intercept is on)
- Raw Request** (Visible in the Request pane):

```
POST /WebGoat/xxe/content-type HTTP/1.1
Host: localhost:8080
Content-Length: 26
sec-ch-ua: "Chromium";v="119", "Not?A_Brand";v="24"
Accept: */*
Content-Type: application/json
X-Requested-With: XMLHttpRequest
sec-ch-ua-mobile: 70
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost:8080
Sec-Fetch-Site: same-origin
Sec-Fetch-Dest: empty
Referrer: http://localhost:8080/WebGoat/start.mvc?username=kikeee
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: SESSIONID=vCYATZ0NshEMBrq8LTrSfo91gjybdDa4q;H-s
Connection: close
{
  "text": "esto es un gato"
}
```
- Inspector** pane showing Request attributes, Request query parameters, Request cookies, and Request headers.
- Browser Window** (Left side): A sidebar for "Modern REST framework" with sections: (A6) Vuln & Outdated Components, (A7) Identity & Auth Failure, (A8) Software & Data Integrity, (A9) Security Logging Failures, (A10) Server-side Request Forgery, Client side, and Challenges. It also displays a photo of John Doe uploaded by him 24 days ago.
- Browser Window** (Right side): A chat interface showing messages from "webgoat" and "guest".

## A6 Vuln & outdated Components - Apartado 5



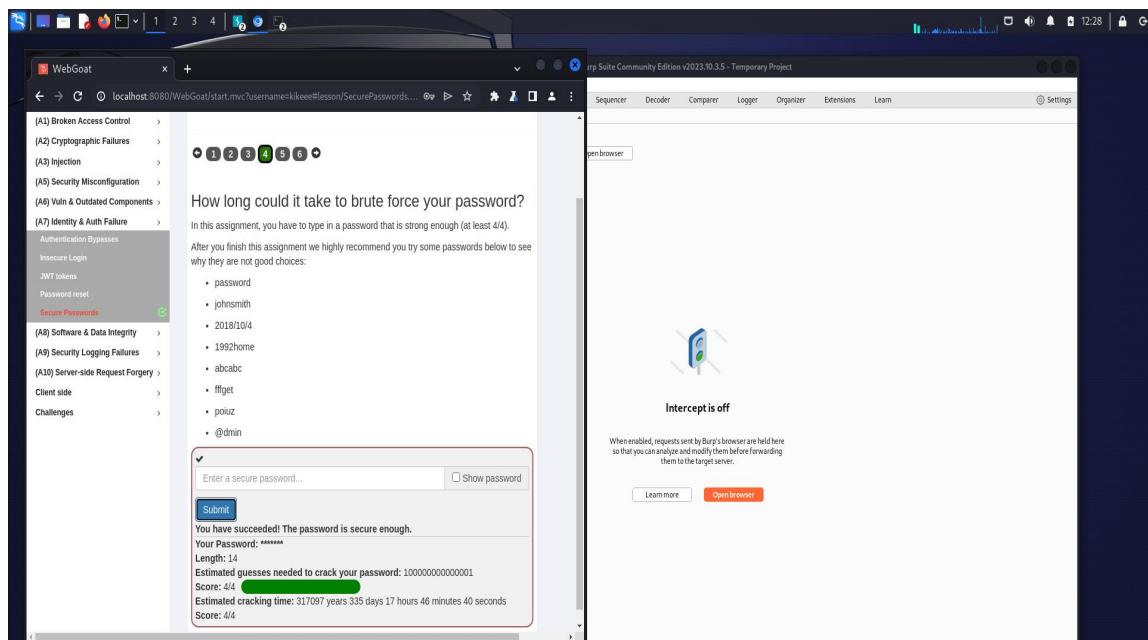
En esta sección, analizamos el uso del mismo código fuente en "WebGoat" pero con diferentes versiones del componente "jquery-ui". Una de estas versiones presenta vulnerabilidades, mientras que la otra no.

En la versión "query-ui:1.10.4", al inyectar la consulta "OK<script>alert('XSS')</script>", observamos que es susceptible a un ataque XXE, como evidencia la captura de pantalla proporcionada.

En contraste, en la versión actualizada "query-ui:1.12.0", utilizando el mismo código empleado en la versión anterior ("OK<script>alert('XSS')</script>"), notamos que la aplicación no es vulnerable al ataque, eliminando así la posibilidad de explotación.

## A7 Identity & Auth Failure - Secure Passwords Apartado 4

En este contexto, la herramienta nos brinda la capacidad de verificar la fortaleza de una contraseña y asegurar que cumple con determinados requisitos de seguridad. Además de indicarnos la longitud de la contraseña, nos ofrece información sobre el tiempo estimado para descifrarla y la puntuación obtenida en términos de seguridad.



hemos probado las contraseñas que nos dan como ejemplo y vemos que son fácilmente descifrables. En este escenario, tomamos como ejemplo la palabra "password", que figura entre las 10 contraseñas más comunes y, por ende, podría ser descubierta con facilidad.

## 4.C. POST EXPLOTACIÓN

Después de aprovechar las vulnerabilidades de la aplicación web, podemos afirmar que mediante SQL Injection logramos acceder a la base de datos de los empleados, donde encontramos información como sus nombres, apellidos, puestos de trabajo, salarios y contraseñas. Además, pudimos acceder a sus números de tarjetas de crédito.

Con los ataques XXE, interferimos en el procesamiento de datos XML y conseguimos acceder a la propia aplicación. Interceptamos códigos de solicitud de comentarios y los modificamos. También alteramos el "content type" de "JSON" a "XML", lo que nos permitió bloquear los comentarios de tipo "JSON".

Observamos que una versión desactualizada del programa es susceptible a ataques XXE, pero en cambio, la versión actualizada no presenta vulnerabilidad, eliminando así la posibilidad de explotación.

Exploramos diversos tipos de contraseñas para determinar cuáles podrían ser más seguras. Los resultados indican que las contraseñas deben tener al menos ocho caracteres, admitir todos los caracteres de Unicode y evitar el uso de palabras de un "diccionario". No se recomienda utilizar preguntas de seguridad, ya que este método está obsoleto. Asimismo, es importante no proporcionar pistas sobre la contraseña a nadie y evitar cambiarla innecesariamente.

#### **4.D. POSIBLES MITIGACIONES**

1. Mantener la seguridad de la red mediante la actualización continua de sistemas operativos y software.
2. Establecer un plan de seguridad que permita llevar a cabo procesos adecuados, asegurando un control eficiente de las revisiones necesarias.
3. Evaluar de manera regular los elementos que se añaden o eliminan de la red, así como realizar análisis periódicos de vulnerabilidades.
4. Monitorizar la actividad web para identificar posibles amenazas y riesgos.
5. Restringir el acceso a la información únicamente al personal autorizado, garantizando una gestión segura de los datos.
6. Implementar contraseñas robustas y seguras.
7. Realizar copias de seguridad periódicas para preservar la integridad de los datos.
8. Abordar activamente las vulnerabilidades mediante estrategias de ataque y resolución.
9. Utilizar dispositivos de hardware y software seguros para fortalecer la protección.
10. Concientizar y capacitar a los empleados en cuestiones de ciberseguridad para fomentar una cultura de seguridad informática en la organización.

## 4.E. HERRAMIENTAS UTILIZADAS

CONSOLA KALI LINUX

BURPSUITE

WAPPALYZER

CONSULTAS MANUALES EN WEBGOAT:

**SQL:**

- "SELECT \* From user\_data WHERE Login\_Count = 0 and userid= 1 or 1=1."
- "SELECT \* FROM empleados WHERE apellido = "" + nombre + "" AND auth\_tan = "" + auth\_tan + """;

**XXE:**

- "" • "]>" y "&root;test"
- "]>&xxe;" • "OK"