

INFORME DE AUDITORIA WEBGOAT



WEBGOAT

ÍNDICE

1. Introducción

2. Ámbito y alcance de la auditoría

3. Informe ejecutivo

- 3.A. Breve resumen del proceso realizado
- 3.B. Vulnerabilidades destacadas
- 3.C. Conclusiones
- 3.D. Recomendaciones

4. Descripción del 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.Introducción

En este informe vamos a explicar las vulnerabilidades encontradas en la aplicación Web Goat, así como todos los procesos que hemos hecho y herramientas utilizadas para poder detectar los fallos. Nuestro equipo se embarcó en una evaluación integral de la aplicación, con el objetivo principal de identificar y documentar cualquier vulnerabilidad presente. Seguimos un enfoque sistemático que involucró una serie de procesos y la utilización de varias herramientas para detectar posibles fallos de seguridad y brindaremos recomendaciones para mejorar la seguridad de la aplicación.

2. Ámbito y alcance de la auditoría

El alcance de esta auditoria 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 7
- A5 Security Misconfiguration apartado 4
- A5 Security Misconfiguration apartado 7
- A6 Vuln & outdated Components apartado 5
- A7 Identity & Auth Failure - Secure Passwords 4

Atacaremos cada una de ellas para poder resolver los fallos que hemos encontrado en cada apartado, utilizaremos distintas herramientas y códigos para poder detectarlos y así tomar una decisión de cómo podemos mejorar las vulnerabilidades.

El objetivo principal es resolver todos los problemas que vayamos encontrando y mejorar la web con las medidas de seguridad necesarias para que no sea vulnerable a posibles ataques.

3. Informe ejecutivo

3.A. Resumen del proceso realizado

Los pasos que hemos realizado son:

1. abrir la aplicación web a través de la herramienta Docker.
2. Ver los puertos abiertos.
3. Identificar el sistema operativo
4. Reconocer el lenguaje de programación
5. Atacar los distintos apartados de la web.

3.A.1. abrir la aplicación web a través Docker

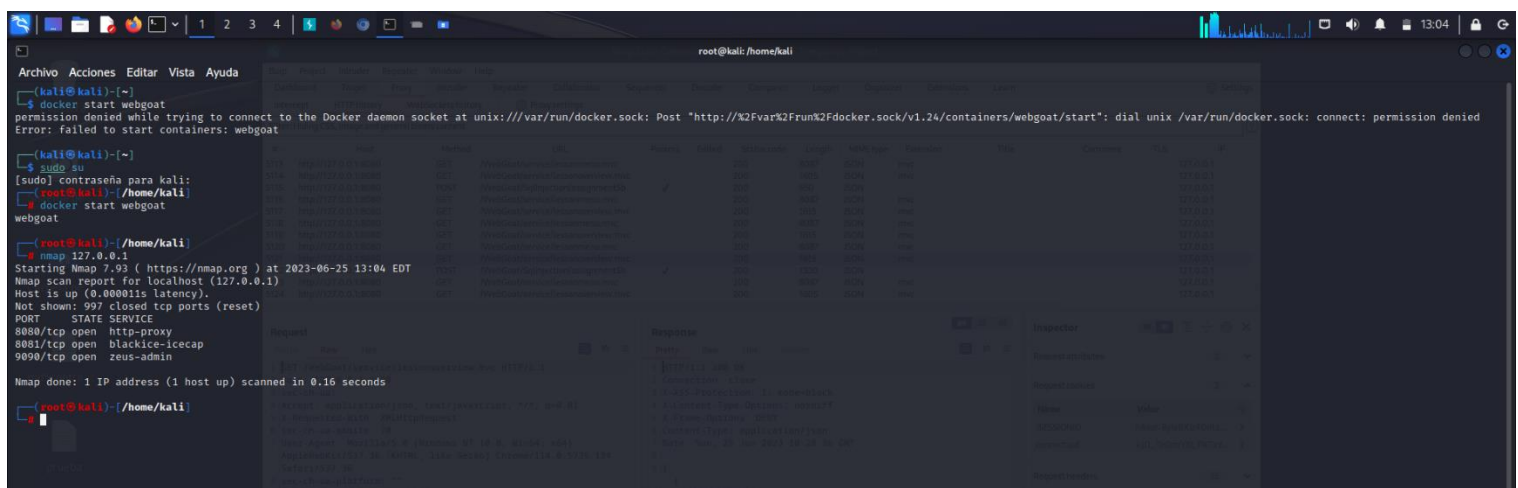
Hemos ejecutado el entorno a través de los siguientes comandos:

- `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

Localizados a través de la función nmap 127.0.0.1 hemos podido comprobar que puertos estaban abiertos.

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



```
root@kali: /home/kali
(kali@kali)~$ docker start webgoat
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/webgoat/start": dial unix /var/run/docker.sock: connect: permission denied
Error: failed to start containers: webgoat

(kali@kali)~$ sudo su
[sudo] contraseña para kali:
root@kali: /home/kali
root@kali:~# docker start webgoat
webgoat

root@kali:~# nmap 127.0.0.1
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-25 13:04 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000011s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
8080/tcp  open  http-proxy
8081/tcp  open  blackice-icecap
9090/tcp  open  zeus-admin

Nmap done: 1 IP address (1 host up) scanned in 0.16 seconds

root@kali:~#
```

3.A.3. Sistema operativo

A través de la función nmap -O 127.0.0.1 identificamos el sistema operativo.

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

```
root@kali: /home/kali
Archivo Acciones Editar Vista Ayuda
(kali@kali)~]
$ nmap -O 127.0.0.1
TCP/IP fingerprinting (for OS scan) requires root privileges.
QUITTING!
(kali@kali)~]
$ sudo su
[sudo] contraseña para kali:
(root@kali)~/home/kali]
$ nmap -O 127.0.0.1
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-25 13:16 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00053s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
8080/tcp  open  http-proxy
8081/tcp  open  blackice-icecap
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 1.67 seconds
(root@kali)~/home/kali]
```

3.A.4. Lenguaje de programación

Identificado a través de la aplicación Wappalyzer pudimos ver que lenguaje de programación utiliza la web.

- Java

3.A.5 Ataques

1º 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.`

2º 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 + ''";`

3º Ataque

A3 Injection - Cross Site Scripting - Apartado - Apartado 7

En este apartado utilizamos la herramienta “Burp Suite” y pudimos ver como el numero de la tarjeta de crédito se queda guardado y era fácilmente detectable.

Para atacar utilizamos este scripting:

`"<script>alert("1")</script>"` para ver como podemos dejar expuesta el nº de tarjeta ya que cualquier campo de datos devuelto al cliente es potencialmente inyectable.

4º Ataque

A5 Security Misconfiguration - Apartado 4

Las herramientas que hemos utilizado para atacar, es la herramienta “Burp Suit” que nos ha permitido interceptar el código de solicitud de los comentarios para luego poder modificarlo y añadir las consultas “<!DOCTYPE user [<!ENTITY root SYSTEM "file:/// ">]>” y “<comment><text>&root;test</text></comment>” con lo que hemos podido lograr el objetivo de interferir el procesamiento de datos XML de la aplicación web.

5º Ataque

A5 Security Misconfiguration - Apartado 7

Para este apartado hemos utilizado la herramienta “Burp Suite” para poder interceptar los códigos de la aplicación web y poder añadir el siguiente ataque: “<?xml version="1.0"?><!DOCTYPE user [<!ENTITY root SYSTEM "file:/// ">]><comment><text>&xxe;</text></comment>” y modificar “content type” de “JSON” a “XML” lo cual nos ha podido permitir poder bloquear los comentarios de tipo “JSON”.

6º Ataque

A6 Vuln & outdated Components - Apartado 5

En este apartado pudimos observar como se usa el mismo código fuente en “Webgoat” pero diferentes versiones del componente “jquery-ui”. Una es explotable y la otra no.

En la versión query-ui:1.10.4 podemos observar como inyectando una consulta “OK<script>alert('XSS')</script>” es susceptible de un ataque XXE como podemos observar en la captura de pantalla.

En la versión actualizada query-ui:1.12.0 utilizando el mismo código que en la versión anterior “OK<script>alert('XSS')</script>” podemos ver que no es vulnerable al ataque y por lo tanto elimina el exploit.

7º Ataque

A7 Identity & Auth Failure - Secure Passwords Apartado 4

En este apartado comprobamos si una contraseña es lo bastante segura y cumple con ciertos requisitos. Nos proporciona una información de cuánto tiempo tardarían en descifrar la contraseña que he creado, la longitud y la puntuación que ha obtenido.

Sin embargo, hemos probado las contraseñas que nos dan como opciones y podemos ver que son contraseñas fácilmente descifrables, en este caso hemos utilizado la palabra “password” que es una de las 10 más comunes por lo que fácilmente podría ser descubierta.

3.B. Vulnerabilidades destacadas

Podemos destacar que “Web Goat” es una página fácilmente vulnerable, de los cuales sus principales fallos son:

1. Se pueden inyectar instrucciones **SQL** de forma maliciosa con el fin de quebrantar las medidas de seguridad y acceder a datos protegidos.
2. Fácilmente se pueden modificar o eliminar datos.
3. Se puede acceder y modificar códigos.
4. Se pueden robar los usuarios y contraseñas registrados en la aplicación.
5. Se puede inhabilitar el uso del sistema.
6. Falsificación de solicitudes del lado del servidor.
7. Se pueden enviar ataques **XXS** que es un ataque en el cual usuarios maliciosos inyectan un script malicioso en la web para ser procesado y ejecutado, teniendo como objetivo: robar datos personales del usuario, cookies de sesión, implementar técnicas de ingeniería social, entre otras.
8. Tiene **componentes vulnerables y desactualizados**. Ocurre cuando un componente de software no es compatible, está desactualizado o es vulnerable a un “exploit” conocido.

3.C. Conclusiones

WebGoat es una aplicación bastante insegura basada en el lenguaje de programación Java y código abierto. Hemos podido realizar varios ataques y comprobar todas las vulnerabilidades que tiene. Hemos quebrantado las medidas de seguridad y acceder a datos protegidos, pudimos interferir con el procesamiento de datos XML e interceptar y modificar ciertos códigos. La web puede ser atacada a través de inyecciones SQL por lo que es posible, modificar o eliminar datos por lo que puede ser un gran problema de seguridad.

Comprobamos también que la aplicación tiene programas desactualizados por lo que es fácilmente vulnerables a “exploits”. Por lo tanto, es una app bastante susceptible a cualquier ataque por lo que la base de datos que tenemos aquí está desprotegida y cualquier persona que controle un poco puede acceder a ella.

Por lo que es recomendable hacer bastantes mejoras en el servidor para poder convertirla en aplicación segura.

3.D. Recomendaciones

Las sugerencias que hacemos para mejorar la aplicación son mantener actualizado los software, optimizar los códigos de código abierto, configurar analizadores XML para rechazar definiciones de documentos personalizados (DTD). Reconstruir completamente las cargas útiles para detectar los ataques XXE.

Utilizar procedimientos almacenados para prevenir ataques de inyección SQL, así se puede impedir que cualquier usuario pueda acceder libremente al servidor de la base de datos a través de campos de entrada de código como "nombre de usuario" o "contraseña".

Realizar consultas parametrizadas, que utilicen variables en lugar de constantes en la cadena de consulta.

Implementar la encriptación de datos. Al hacerlo, solo las personas que autorice la empresa y con la autenticación correspondiente podrán acceder a los datos.

Establecer contraseñas seguras y cambiarlas de manera regular.

Controlar y registrar las acciones y movimientos sobre la base de datos, así permitirá saber quién ha manipulado la información.

Eliminar funcionalidades, componentes, archivos y documentación que no sea necesaria.

Revisar los componentes que no generen parches de seguridad para versiones anteriores. Si la aplicación de parches no es posible, se puede implementar un parche virtual para monitorear, detectar o protegerse contra el problema descubierto.

4. Descripción del proceso de auditoría

4.A. Reconocimiento/Information gathering

A3 Injection - SQL Injection (intro) - Apartado 10

En este apartado hemos podido detectar que la tabla de datos que contiene información sobre los empleados es susceptible de inyecciones SQL que son instrucciones maliciosas. Con el fin de quebrantar las medidas de seguridad y acceder a datos protegidos.

Esta es la tabla que tenemos con información de los empleados:

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 este apartado igual que en el anterior también a través de la tabla con información de los usuarios podemos acceder a datos comprometidos de los diferentes empleados utilizando consultas SQL.

Ejemplo:

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

A3 Injection - Cross Site Scripting - Apartado - 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.

Shopping Cart

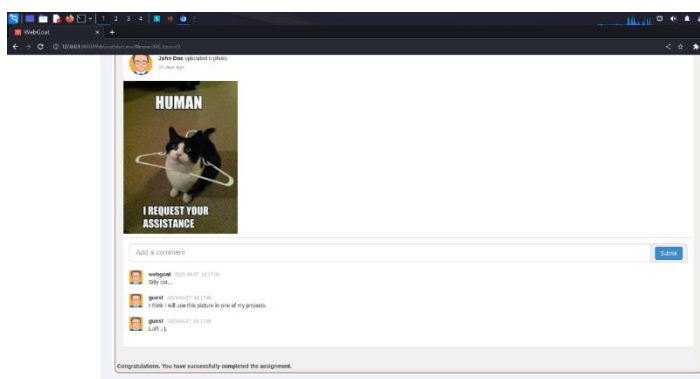
Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	70	\$0.00
Dynex - Traditional Notebook Case	27.99	38	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1600	\$0.00
3 - Year Performance Service Plan \$1000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

Enter your credit card number:

Enter your three digit access code:

Utilizando este Scripting: “<script>alert("1")</script>” podemos dejar expuesta los nº de tarjeta.

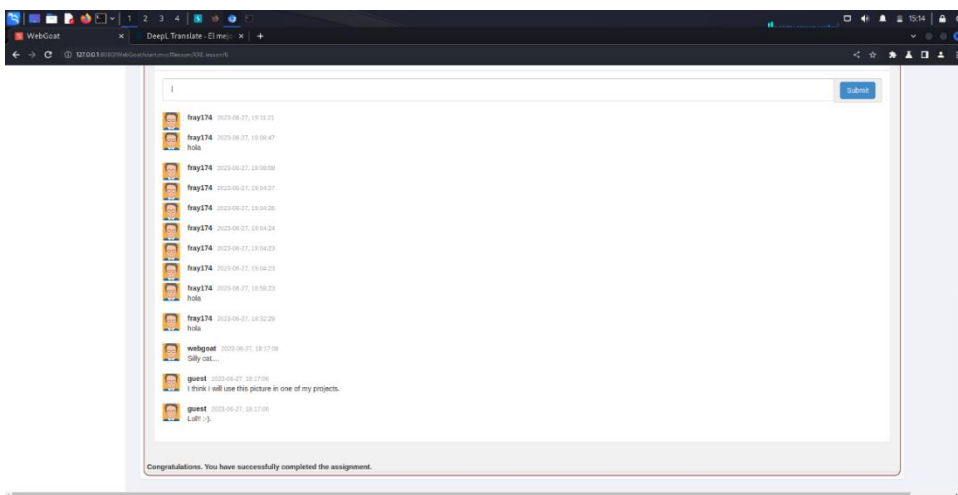
A5 Security Misconfiguration - Apartado 4



En este apartado vemos que es susceptible de ataques XXE que nos permite atacar e interferir con el procesamiento de datos XML de la aplicación a través de la siguiente consulta:

“<!DOCTYPE user [<!ENTITY root SYSTEM "file:///"]>]>” y
“<comment><text>&root;test</text></comment>”

A5 Security Misconfiguration - Apartado 7

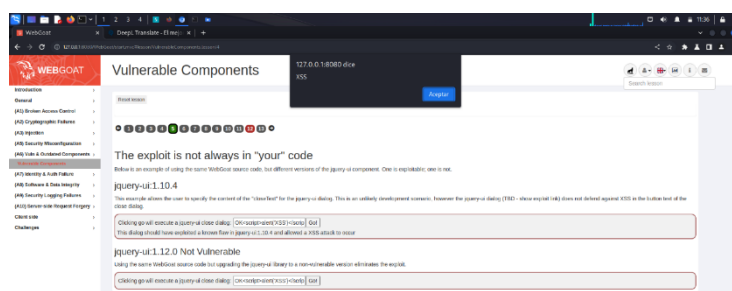


En este apartado hemos averiguado que se puede interceptar los códigos de la aplicación web y poder añadir el siguiente ataque:

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

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

A6 Vuln & outdated Components - Apartado 5



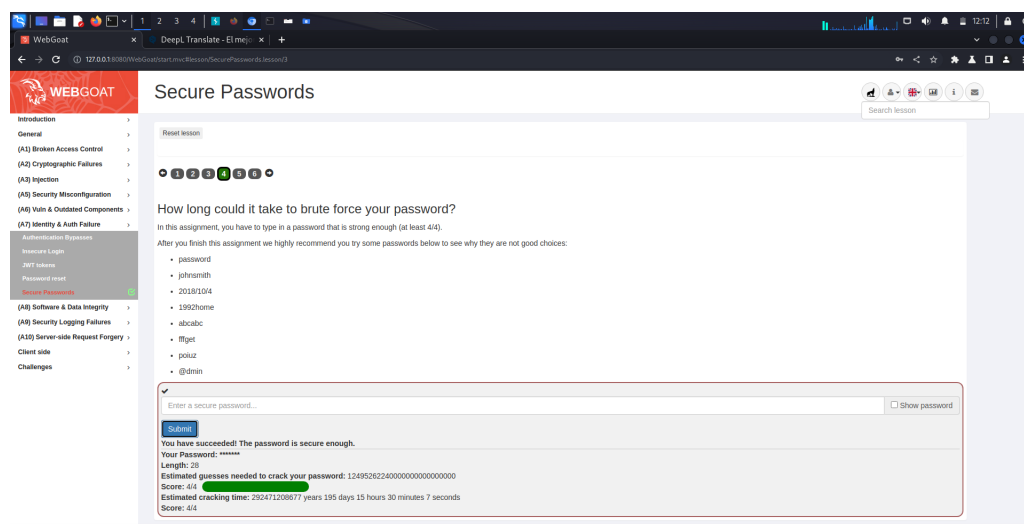
Aquí hemos investigado como hay componentes que están desactualizado o son vulnerable a un “exploit”

Podemos ver como se usa el mismo código fuente en “Webgoat” pero diferentes versiones del componente “jquery-ui”. Una es explotable y la otra no.

En la versión query-ui:1.10.4 es susceptible de un ataque XXE.

En la versión actualizada query-ui:1.12.0 no es vulnerable al ataque y por lo tanto puede eliminar el “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

WebGoat

DeepL Translate - El mejor

127.0.0.1:8080/WebGoat/start.mvc?lesson=SqlInjection.lesson/9

General

(A1) Broken Access Control

(A2) Cryptographic Failures

(A3) Injection

SQL Injection (intro)

SQL Injection (advanced)

SQL Injection (mitigation)

Path traversal

Cross Site Scripting

(A5) Security Misconfiguration

(A6) Vuln & Outdated Components

(A7) Identity & Auth Failure

(A8) Software & Data Integrity

(A9) Security Logging Failures

(A10) Server-side Request Forgery

Client side

Challenges

Show hints

Reset lesson

Try It! Numeric SQL injection

The query in the code builds a dynamic query as seen in the previous example. The query builds a dynamic query by concatenating a number making it susceptible to Numeric SQL injection:

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

Using the two input Fields below, try to retrieve all the data from the users table.

Warning: Only one of these fields is susceptible to SQL injection. You need to find out which, to successfully retrieve all the data.

✓

Login_Count:

User_Id:

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, 4352209902222, AMEX, , 0,
103, Jane, Plane, 123456789, MC, , 0,
103, Jane, Plane, 333498703333, AMEX, , 0,
10312, Jolly, Hershey, 176896789, MC, , 0,
10312, Jolly, Hershey, 333300003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
15603, Peter, Sand, 123609789, MC, , 0,
15603, Peter, Sand, 338893453333, AMEX, , 0,
15613, Joesph, Something, 33843453533, AMEX, , 0,
15837, Chaos, Monkey, 32849386533, CM, , 0,
19204, Mr, Goat, 33812953533, VISA, , 0,
```

Your query was: SELECT * From user_data WHERE Login_Count = 0 and userid= 1 or 1=1

Para atacar este apartado hemos utilizado la consulta: `SELECT * From user_data WHERE Login_Count = 0 and userid= 1 or 1=1`.

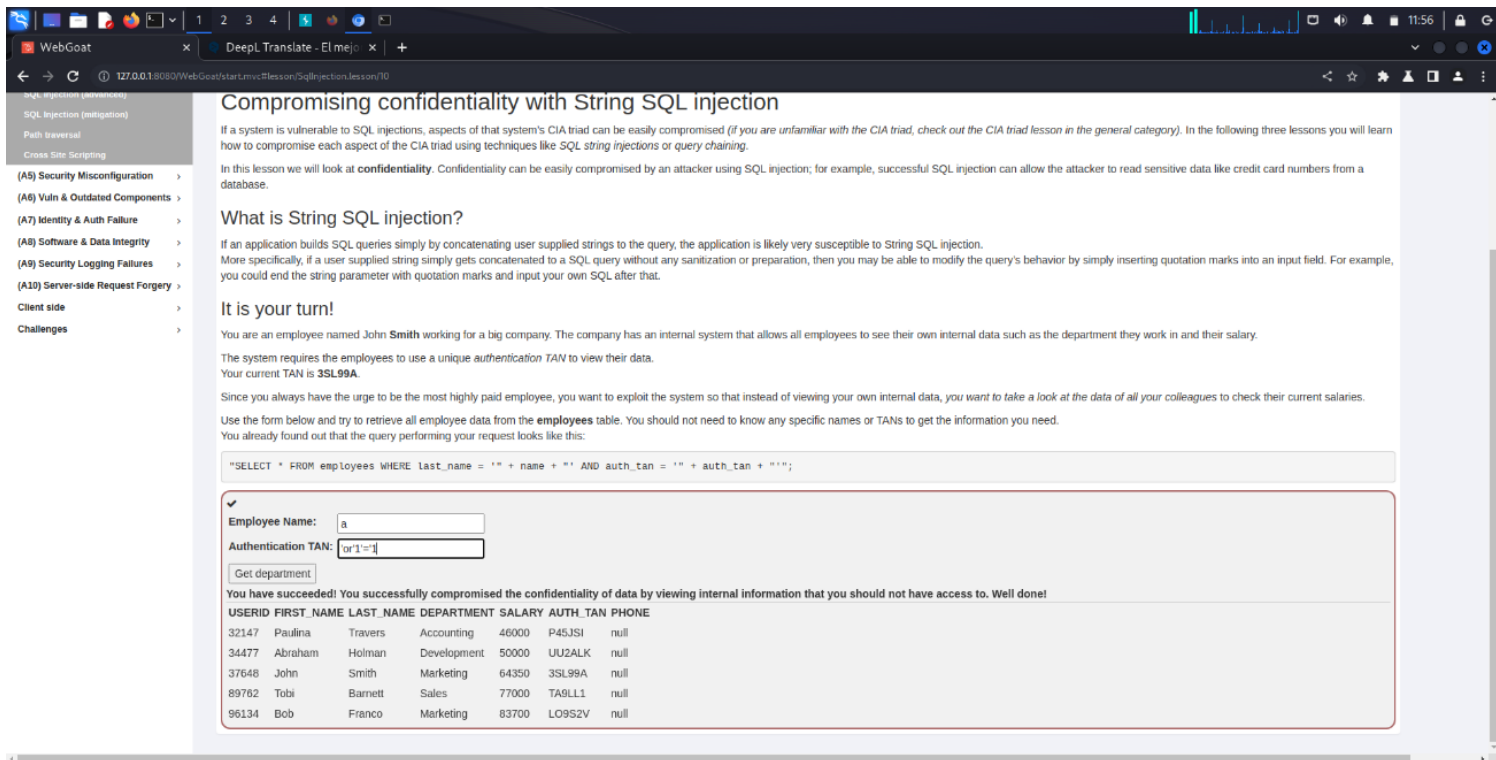
La instrucción `SELECT` permite consultar registros de la base de datos.

El operador `"OR"` indica a la instrucción que si colocamos algo después que sea `"true"` (verdadero), toda la consulta se convierte en `true` (verdadero). Con esto podemos anular la condición `"WHERE"` y obtener todos los resultados de la base de datos.

Y finalmente la condición de `'1'='1` que siempre se cumple, ya que 1 siempre es igual a 1.

Al realizar este ataque hemos visto como hemos podido acceder a una base datos con información de las tarjetas de créditos de los empleados.

A3 Injection - SQL Injection (intro) - Apartado 11



Compromising confidentiality with String SQL injection

If a system is vulnerable to SQL injections, aspects of that system's CIA triad can be easily compromised (if you are unfamiliar with the CIA triad, check out the CIA triad lesson in the general category). In the following three lessons you will learn how to compromise each aspect of the CIA triad using techniques like *SQL string injections* or *query chaining*.

In this lesson we will look at **confidentiality**. Confidentiality can be easily compromised by an attacker using SQL injection; for example, successful SQL injection can allow the attacker to read sensitive data like credit card numbers from a database.

What is String SQL injection?

If an application builds SQL queries simply by concatenating user supplied strings to the query, the application is likely very susceptible to String SQL injection. More specifically, if a user supplied string simply gets concatenated to a SQL query without any sanitization or preparation, then you may be able to modify the query's behavior by simply inserting quotation marks into an input field. For example, you could end the string parameter with quotation marks and input your own SQL after that.

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 = '' + auth_tan + ''";
```

✓

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	PHONE
32147	Paulina	Travers	Accounting	46000	P45J5I	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
37648	John	Smith	Marketing	64350	3SL99A	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
96134	Bob	Franco	Marketing	83700	LO952V	null

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 + ''";

Utilizando el operador "OR" para poder convertir la consulta en verdadero y anular la función "WHERE" con la condición "1=1" ya que siempre es verdadero y así poder obtener los datos de la tabla.

A3 Injection - Cross Site Scripting - Apartado - Apartado 7

4

Burp Suite Community Edition v2023.5.4 - Temporary Project

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings

Intercept HTTP history WebSockets history Proxy settings

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Comment	TLS	IP
2510	http://127.0.0.1:8080	GET	/WebGoat/service/lessonoverview.mvc			200	1216	JSON	mvc				127.0.0.1
2511	http://127.0.0.1:8080	GET	/WebGoat/service/lessonmenu.mvc			200	8087	JSON	mvc				127.0.0.1
2512	http://127.0.0.1:8080	GET	/WebGoat/service/lessonoverview.mvc			200	1216	JSON	mvc				127.0.0.1
2513	http://127.0.0.1:8080	GET	/WebGoat/service/lessonmenu.mvc			200	8087	JSON	mvc				127.0.0.1
2514	http://127.0.0.1:8080	GET	/WebGoat/service/lessonoverview.mvc			200	1216	JSON	mvc				127.0.0.1
2515	http://127.0.0.1:8080	GET	/WebGoat/service/lessonmenu.mvc			200	8087	JSON	mvc				127.0.0.1
2516	http://127.0.0.1:8080	GET	/WebGoat/service/lessonoverview.mvc			200	1216	JSON	mvc				127.0.0.1
2517	http://127.0.0.1:8080	GET	/WebGoat/service/lessonmenu.mvc			200	8087	JSON	mvc				127.0.0.1
2518	http://127.0.0.1:8080	GET	/WebGoat/service/lessonoverview.mvc			200	1216	JSON	mvc				127.0.0.1
2519	http://127.0.0.1:8080	GET	/WebGoat/CrossSiteScripting/attack5a?QTY1=70&QTY2=38&QTY3=1600&QTY4=300&field1=4128%203214%200002%201999&field2=111	✓		200	712	JSON					127.0.0.1
2520	http://127.0.0.1:8080	GET	/WebGoat/service/lessonmenu.mvc			200	8087	JSON	mvc				127.0.0.1
2521	http://127.0.0.1:8080	GET	/WebGoat/service/lessonoverview.mvc			200	1216	JSON	mvc				127.0.0.1

Request

Pretty Raw Hex

```
1 GET /WebGoat/CrossSiteScripting/attack5a?QTY1=70&QTY2=38&QTY3=1600&QTY4=300&field1=4128%203214%200002%201999&field2=111 HTTP/1.1
2 Host: 127.0.0.1:8080
3 sec-ch-ua:
4 Accept: */*
5 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
6 X-Requested-With: XMLHttpRequest
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.134 Safari/537.36
9 sec-ch-ua-platform: ""
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://127.0.0.1:8080/WebGoat/start.mvc
14 Accept-Encoding: gzip, deflate
15 Accept-Language: es-ES,es;q=0.9
16 Cookie: JSESSIONID=hbk0hP050KEj-Zzm_tSc19U-HMPpijCoUNMtMpV2; s%3AjD_TeQdnYJ2_PXTVzZ-P-V0j_BYGzb0u.wDeggDJSwxF9m2h5x7hz60CDg2ZDfKeEtEscfUGzzGc
17 Connection: close
18
19
```

Response

Pretty Raw Hex Render

```
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: Mon, 26 Jun 2023 17:11:21 GMT
8
9 {
10   "lessonCompleted": false,
11   "feedback":
12     "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).",
13   "output":
14     "Thank you for shopping at WebGoat. <br \\\>Your support is a ppreciated<br \\\><p>We have charged credit card:4128 3214 00 02 1999<br \\\>-----<br \\\>                $2655943.92",
15   "assignment": "CrossSiteScriptingLesson5a",
16   "attemptWasMade": true
17 }
```

Inspector

Selection 37 (0x25)

Selected text

4128%203214%200002%201999&field2=111

Decoded from: URL encoding

4128 3214 0002 1999&field2=111

Request attributes 2

Request query parameters 6

Request cookies 2

Request headers 16

Response headers 6

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

127.0.0.1:8080 dice
1

Acceptar

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.

Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilling 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:

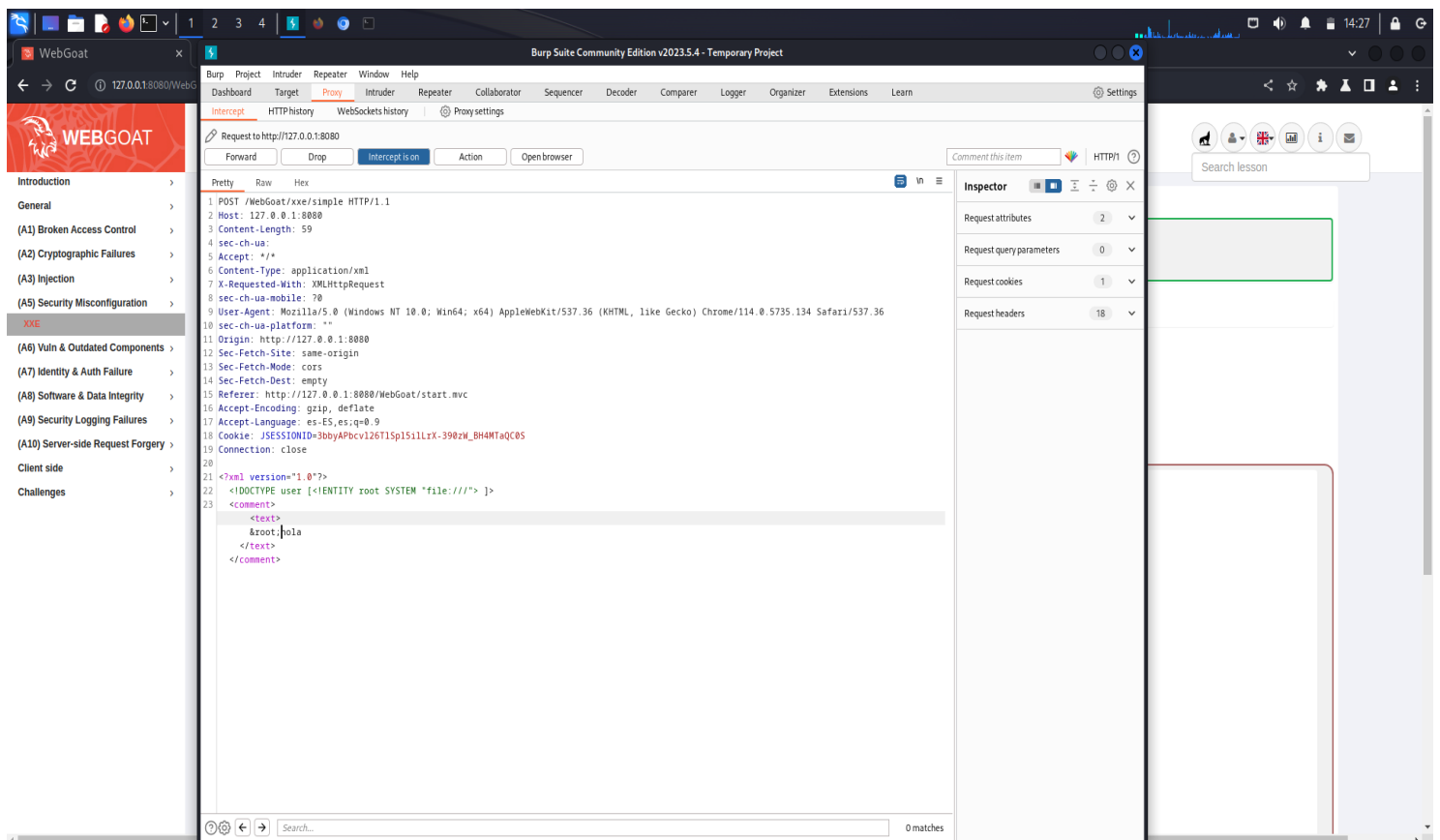
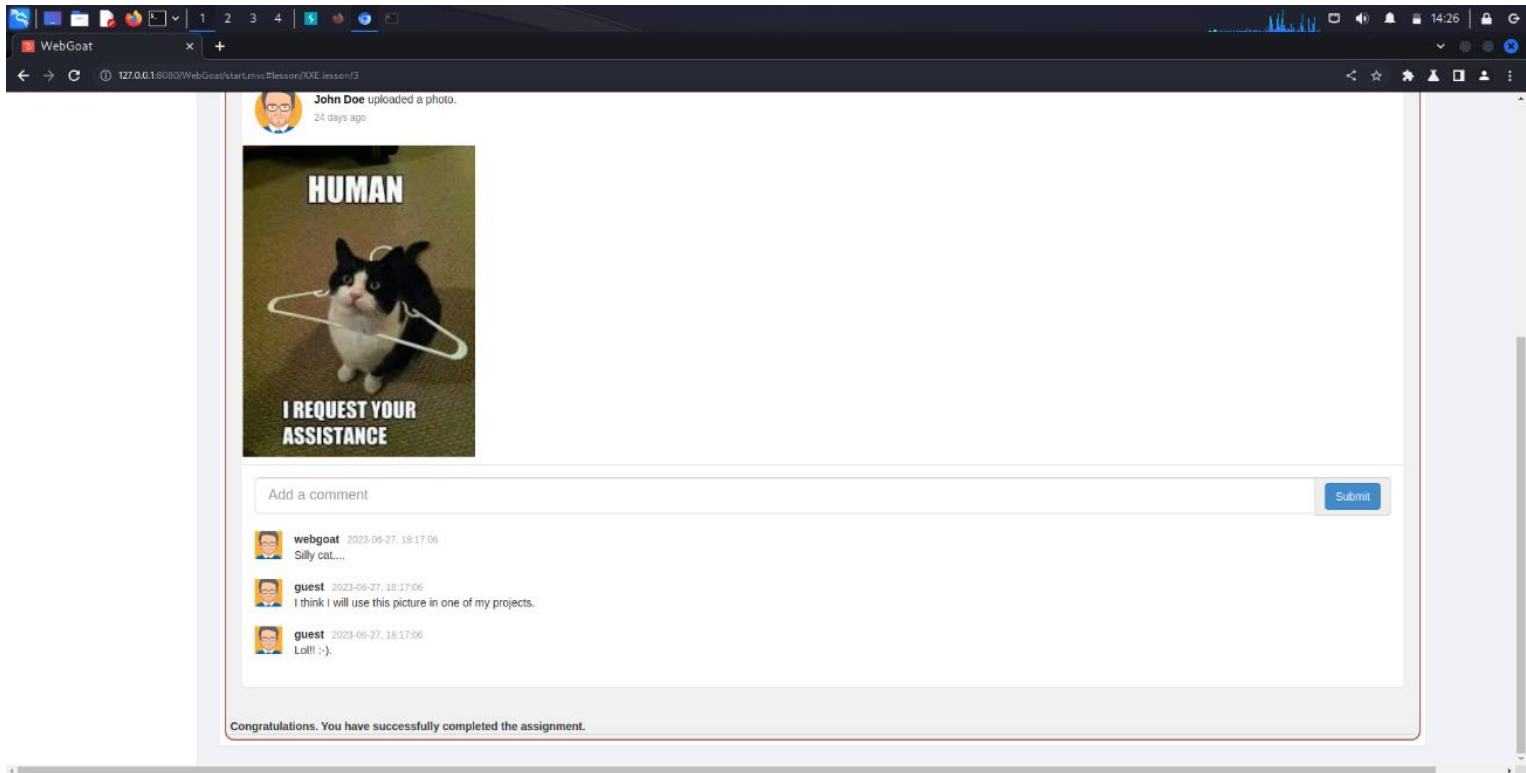
Try again. We do want to see a specific JavaScript mentioned in the goal of the assignment (in case you are trying to do something fancier).
Thank you for shopping at WebGoat.
Your support is appreciated

We have charged credit card:4128 3214 0002 1999

\$2655943.92

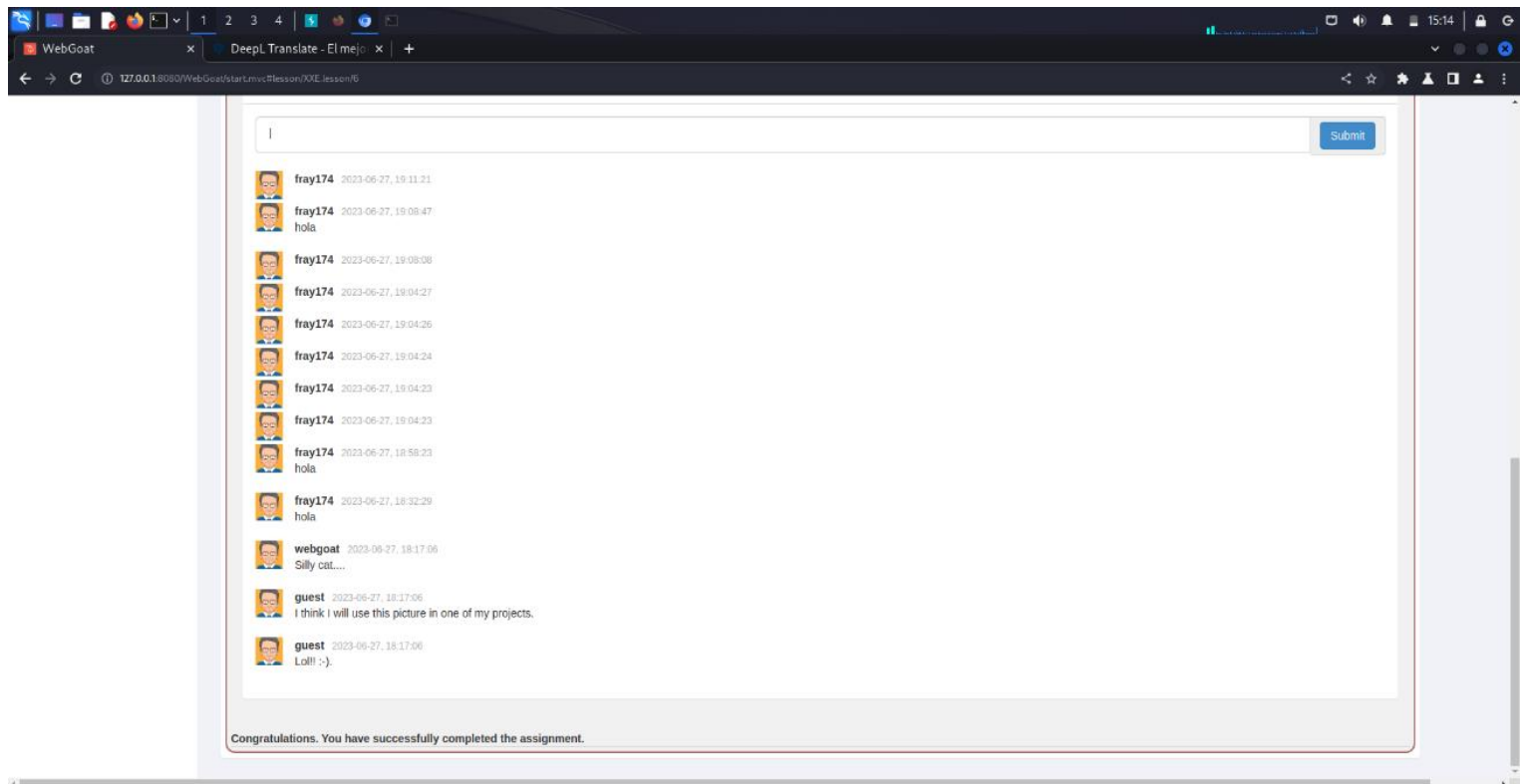
Aquí podemos observar cómo atacando con este scripting:
“`<script>alert("1")</script>`” podemos dejar expuesta el nº de tarjeta ya que cualquier campo de datos devuelto al cliente es potencialmente inyectable.

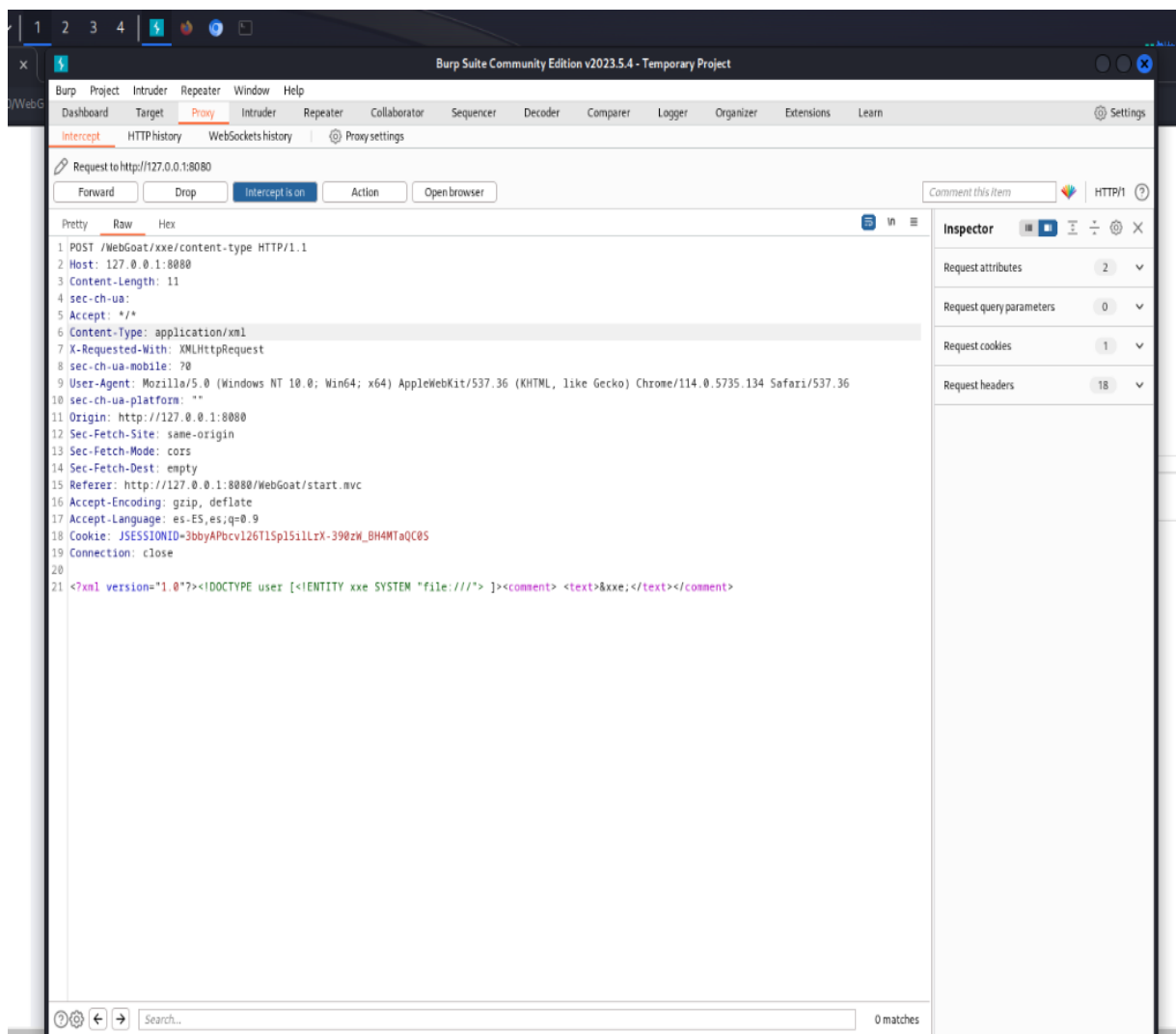
A5 Security Misconfiguration - Apartado 4



Las herramientas que hemos utilizado para atacar, es la herramienta “Burp Suit” que nos ha permitido interceptar el código de solicitud de los comentarios para luego poder modificarlo y añadir las consultas “<!DOCTYPE user [<!ENTITY root SYSTEM "file:///"/>]>” y “<comment><text>&root;test</text></comment>” con lo que hemos podido lograr el objetivo de interferir el procesamiento de datos XML de la aplicación web.

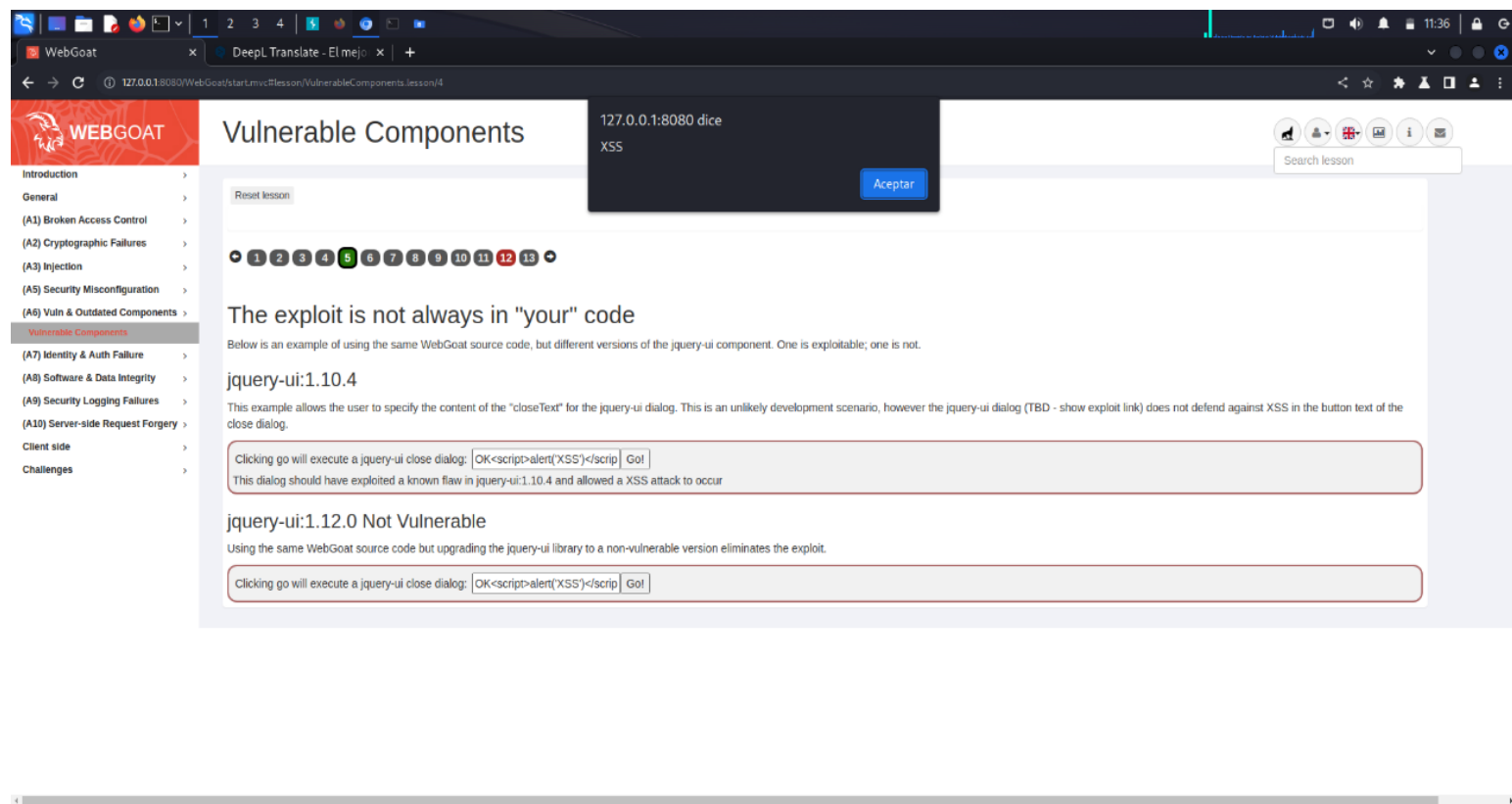
A5 Security Misconfiguration - Apartado 7





Para este apartado hemos utilizado la herramienta “Burp Suite” para poder interceptar los códigos de la aplicación web y poder añadir el siguiente ataque: “<?xml version=“1.0”?><!DOCTYPE user [<!ENTITY root SYSTEM “file:///”>]><comment>
<text>&xxe;</text></comment>” y modificar “content type” de “JSON” a “XML” lo cual nos ha podido permitir poder bloquear los comentarios de tipo “JSON”.

A6 Vuln & outdated Components - Apartado 5



En este apartado podemos ver como se usa el mismo código fuente en “Webgoat” pero diferentes versiones del componente “jquery-ui”. Una es explotable y la otra no.

En la versión query-ui:1.10.4 podemos observar como inyectando una consulta “OK<script>alert('XSS')</script>” es susceptible de un ataque XXE como podemos observar en la captura de pantalla.

En la versión actualizada query-ui:1.12.0 utilizando el mismo código que en la versión anterior “OK<script>alert('XSS')</script>” podemos ver que no es vulnerable al ataque y por lo tanto elimina el exploit.

A7 Identity & Auth Failure - Secure Passwords Apartado 4

The screenshot shows the WebGoat application running in a browser. The browser's address bar displays the URL `127.0.0.1:8080/WebGoat/start.mvc#lesson/SecurePasswords.lesson/3`. The WebGoat interface features a sidebar on the left with a navigation menu. The 'Secure Passwords' lesson is selected and highlighted in red. The main content area is titled 'Secure Passwords' and includes a 'Reset lesson' button. Below this, there is a progress indicator with six numbered steps, where the fourth step is currently active. The lesson content asks, 'How long could it take to brute force your password?' and provides instructions on password strength requirements. A list of weak passwords is shown, including 'password', 'johnsmith', '2018/10/4', '1992home', 'abccabc', 'ffiget', 'poiuz', and '@dmin'. A password input field is present with a 'Show password' checkbox. Below the input field, a 'Submit' button is visible. The feedback section indicates success: 'You have succeeded! The password is secure enough.' It also displays the user's password as '*****', its length as 28, and the estimated guesses needed to crack it as 12495262240000000000000000. The score is 4/4, and the estimated cracking time is 292471208677 years 195 days 15 hours 30 minutes 7 seconds.

WebGoat

Secure Passwords

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
- abccabc
- ffiget
- poiuz
- @dmin

Enter a secure password... ☐ Show password

Submit

You have succeeded! The password is secure enough.

Your Password: *****

Length: 28

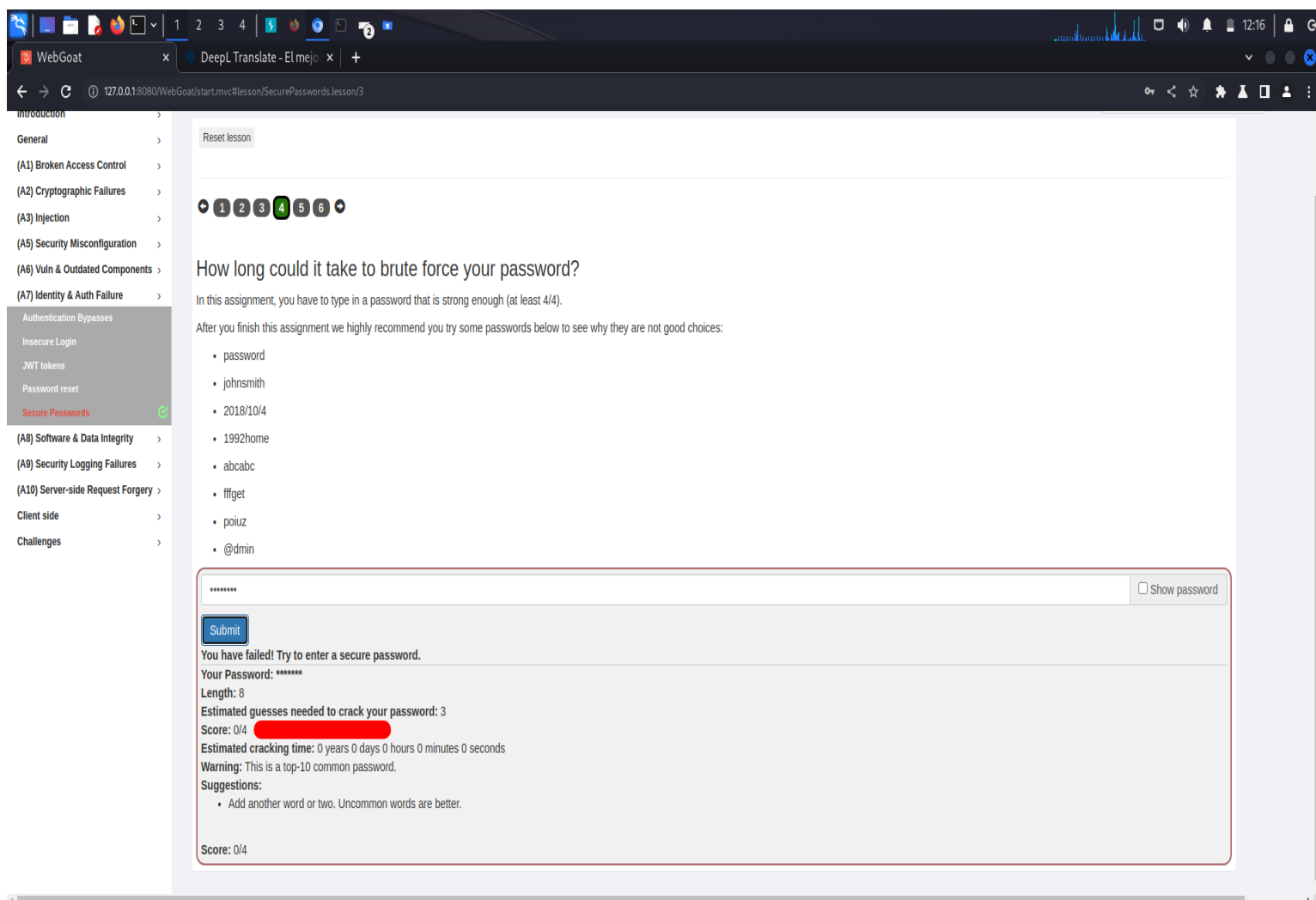
Estimated guesses needed to crack your password: 12495262240000000000000000

Score: 4/4

Estimated cracking time: 292471208677 years 195 days 15 hours 30 minutes 7 seconds

Score: 4/4

Aquí nos permite comprobar si una contraseña es lo bastante segura y cumple con ciertos requisitos. Nos proporciona información de cuánto tiempo tardarían en descifrar la contraseña que he creado, la longitud y la puntuación que ha obtenido.



Sin embargo, hemos probado las contraseñas que nos dan como opciones y podemos ver que son contraseñas fácilmente descifrables, en este caso hemos utilizado la palabra “password” que es una de las 10 más comunes por lo que fácilmente podría ser descubierta.

4.C. Post explotación

Después de haber explotado las vulnerabilidades de la aplicación web podemos indicar que con SQL Injection hemos podido acceder a la base de datos de los empleados, donde aparecen sus nombres, apellidos, puestos de trabajo, salarios y contraseñas. Hemos podido acceder también a sus nº de tarjeta de créditos.

Con los ataques XXE se ha interferido en los procesamientos de datos XML y hemos podido acceder a la propia aplicación para interceptar códigos de solicitud de comentarios y luego poder modificarlos. Se ha modificado “content type” de “JSON” a “XML” lo cual nos ha podido permitir bloquear los comentarios de tipo “JSON”.

Hemos observado también que con una versión desactualizada el programa es susceptible de ataques XXE pero en cambio con la versión actualizada no era vulnerable al ataque por lo que eliminaba el “exploit”.

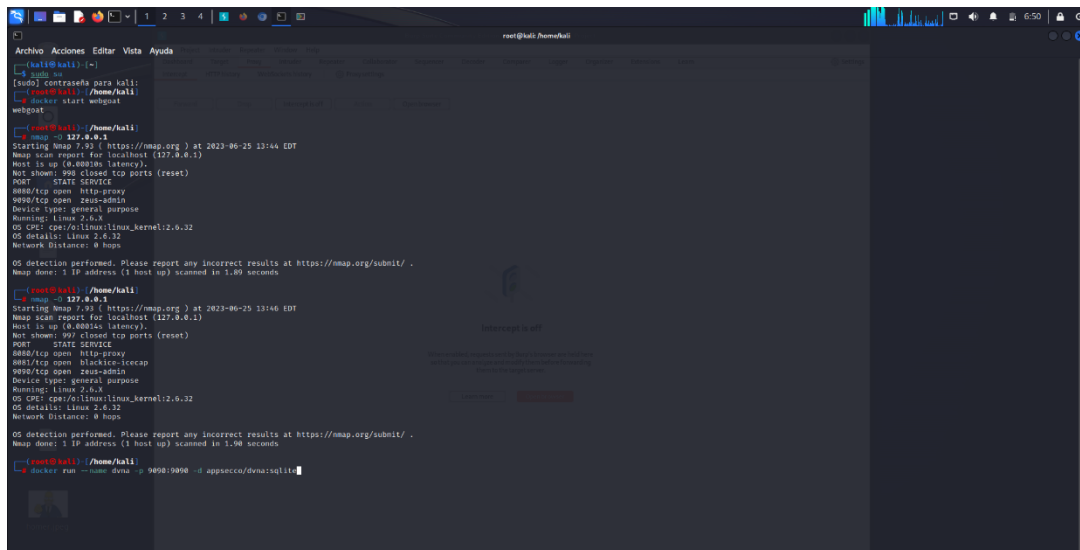
Hemos explotado varios tipos de contraseñas para ver cuáles pueden ser las más seguras y los resultados fueron que las contraseñas deben tener al menos ocho caracteres, admitir todos los caracteres de Unicode, no utilizar las llamadas palabras “diccionario”. No es recomendable utilizar pregunta de seguridad ya que es un método que está obsoleto.

No hay que dar pistas sobre la contraseña a nadie y no hay que cambiar la contraseña innecesariamente.

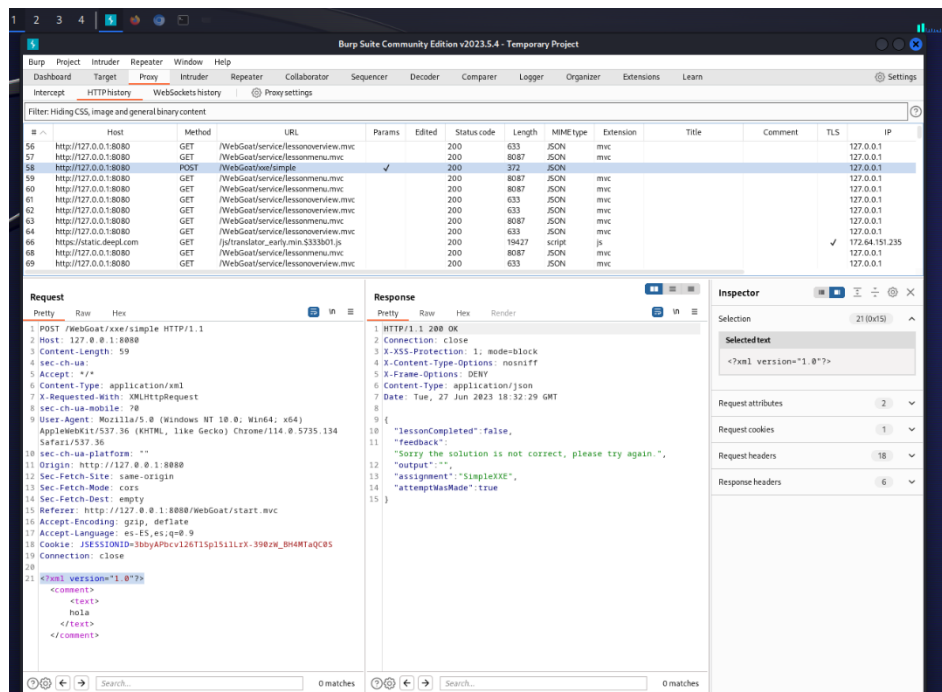
4.D. Posibles mitigaciones

1. Proteger la red actualizando todos los sistemas operativos y software.
2. Tener un plan de seguridad para realizar procesos adecuados con el fin de mantener controladas las revisiones necesarias.
3. Verificar los elementos que se agregan o se quitan a la red y evaluar con regularidad las vulnerabilidades.
4. Monitorizar la actividad de la web para detectar las posibles amenazas.
5. Limitar el acceso a la información a la que solo puedan acceder el personal autorizado.
6. Utilizar contraseñas seguras.
7. Hacer copias de seguridad.
8. Atacar a las vulnerabilidades para poder resolverlas.
9. Utilizar dispositivos seguros.
10. Concienciar a los trabajadores en materia de ciberseguridad.

4.E. Herramientas utilizadas



Consola Kali Linux



Burp suite



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:

- "<script>alert('1')</script>"
- "<!DOCTYPE user [<!ENTITY root SYSTEM 'file:///'">]>" y
"<comment><text>&root;test</text></comment>"
- "<?xml version='1.0'?><!DOCTYPE user [<!ENTITY root SYSTEM 'file:///'">]><comment> <text>&xxe;</text></comment>"
- "OK<script>alert('XSS')</script>"