Stored Cross-Site Scripting (XSS) in Simple Customer Relationship Management System (CRM-PHP)

Hemant Raj Bhati 3 min read · Mar 16, 2026

Introduction

During security testing of the **Simple Customer Relationship Management System (PHP)**, I discovered a **Stored Cross-Site Scripting (XSS)** vulnerability in the **Create Ticket** functionality.

This vulnerability allows an attacker to inject malicious JavaScript code which is stored in the application's database and executed whenever the ticket is viewed.

Stored XSS is particularly dangerous because it affects every user who views the malicious content.

Vulnerability Details

The vulnerability exists in the **Create Ticket** feature.

User input from the following fields is not properly sanitized before being stored and rendered in the application.

Affected fields:

- Ticket Subject
- Ticket Description

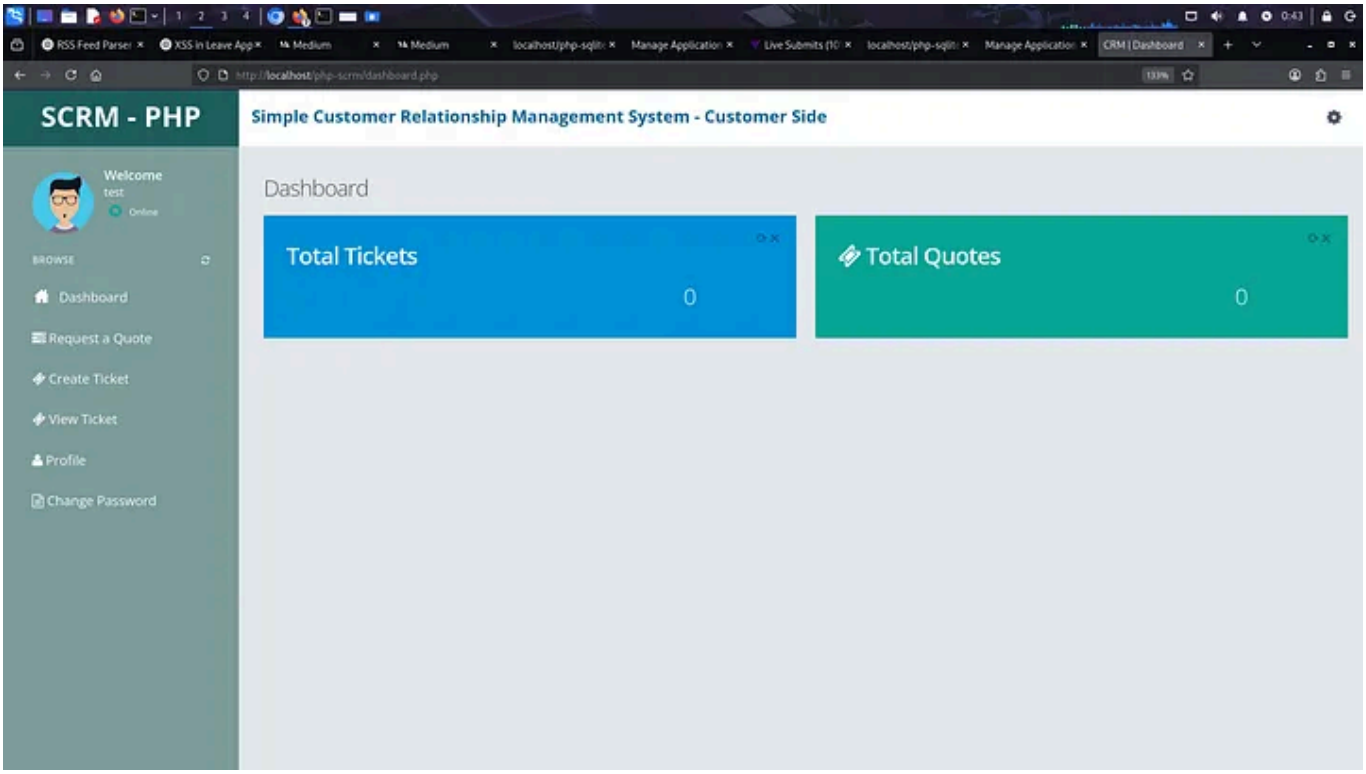
Because the application fails to perform proper output encoding, an attacker can inject malicious JavaScript payloads.

. . .

Steps to Reproduce

1. Login to the CRM application

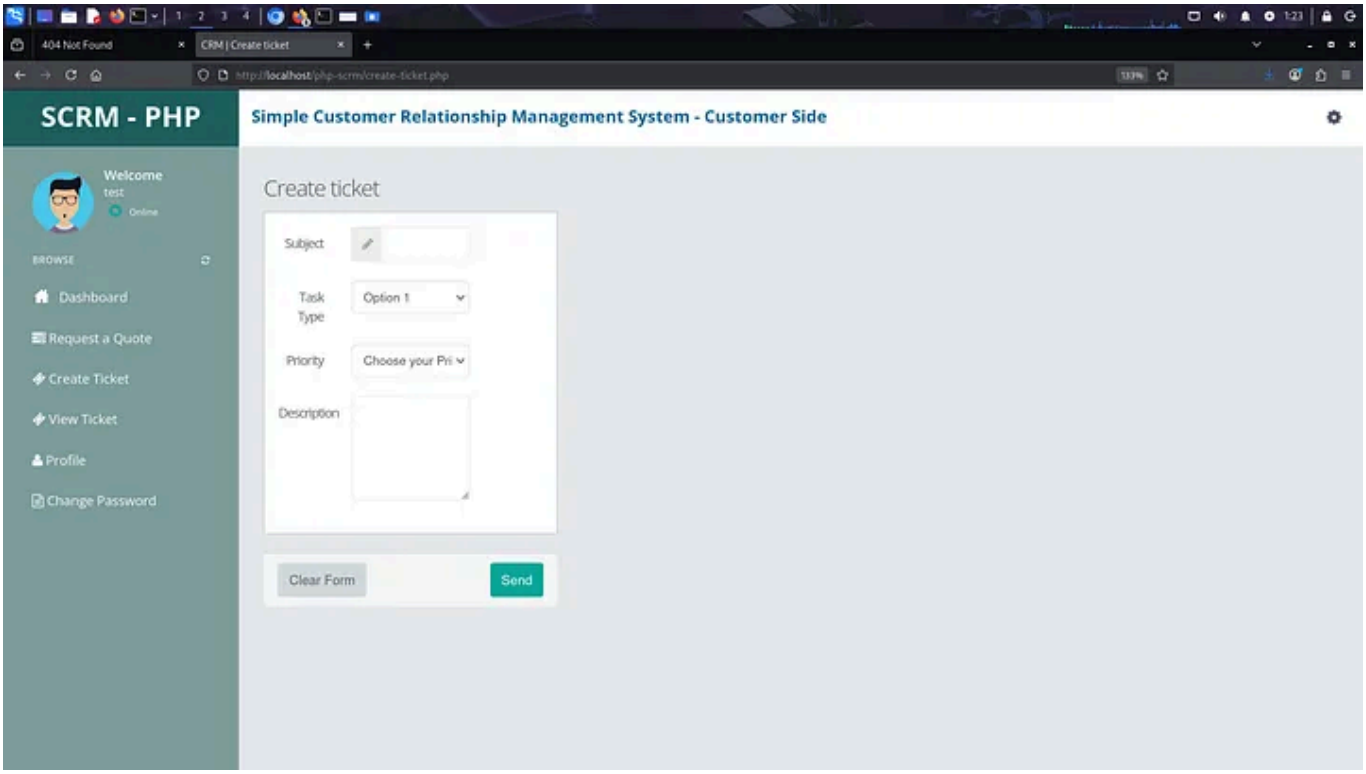
Login with a valid user account.



2. Navigate to the ticket creation page

Go to:

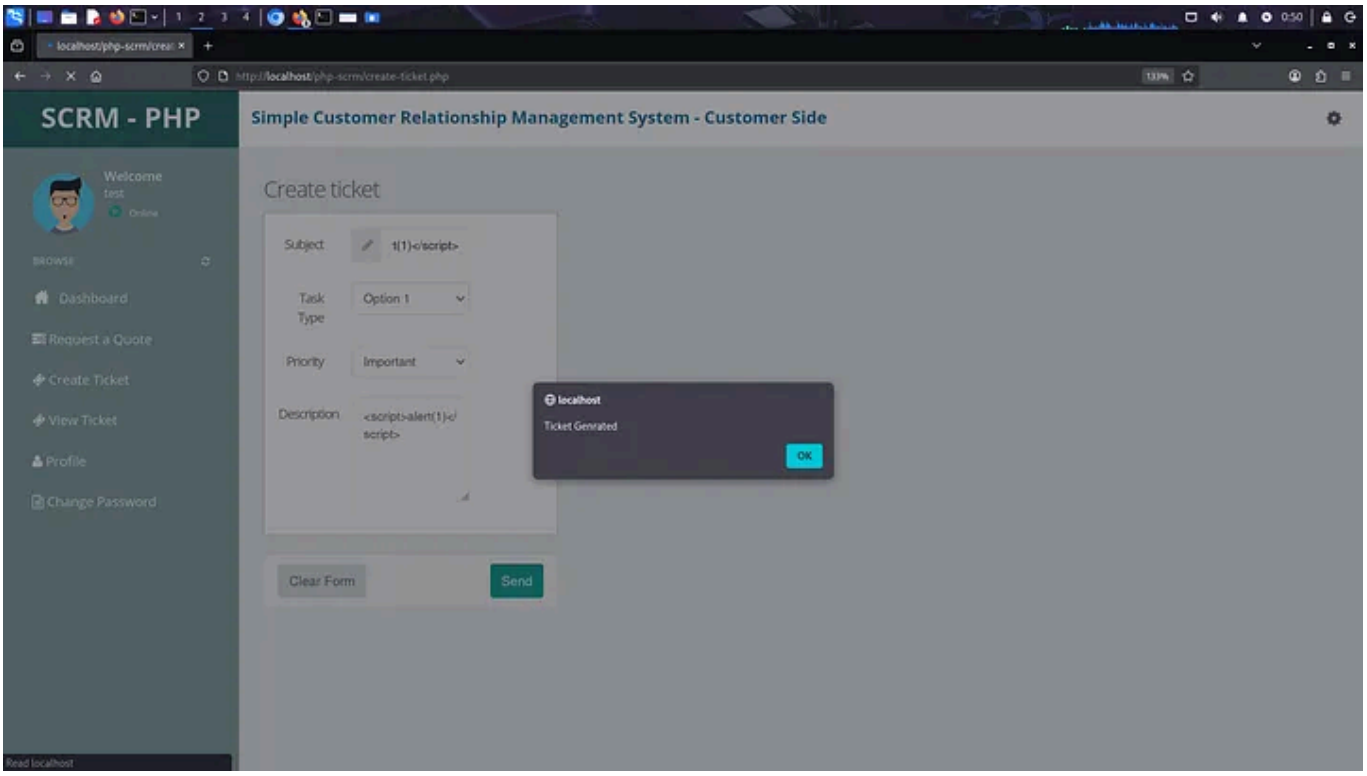
```
Create Ticket
```



3. Inject a malicious payload

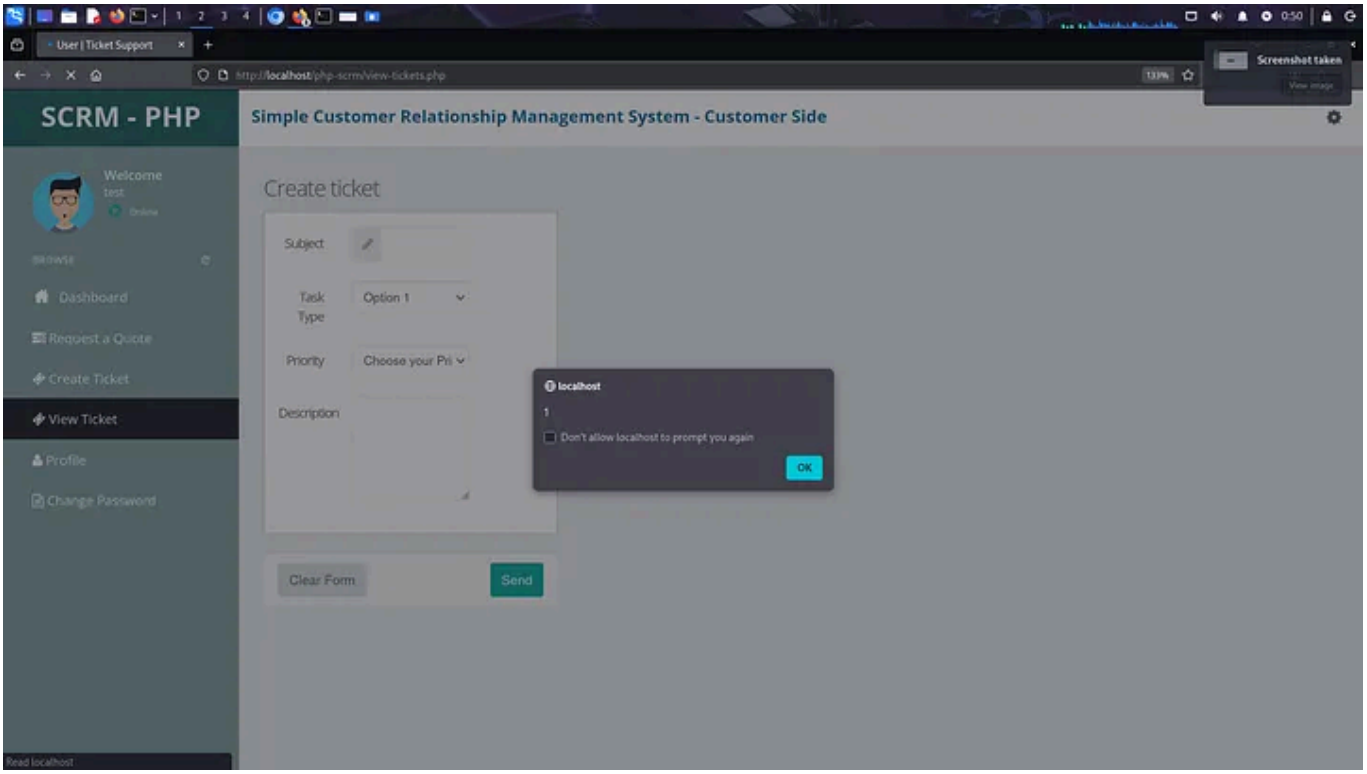
Insert the following payload inside the **Description** field:

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



4. Submit the ticket

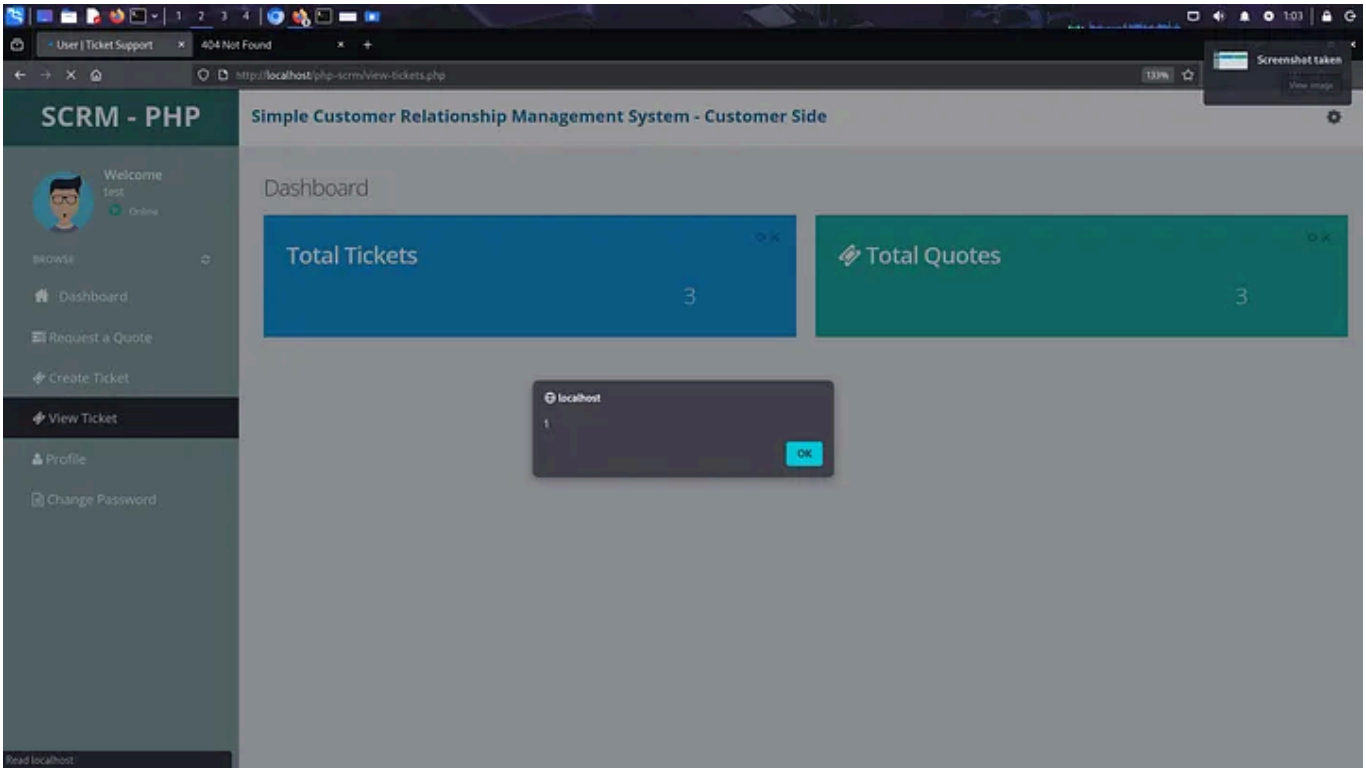
Click **Send** to create the ticket.



5. Navigate to the ticket list

Go to:

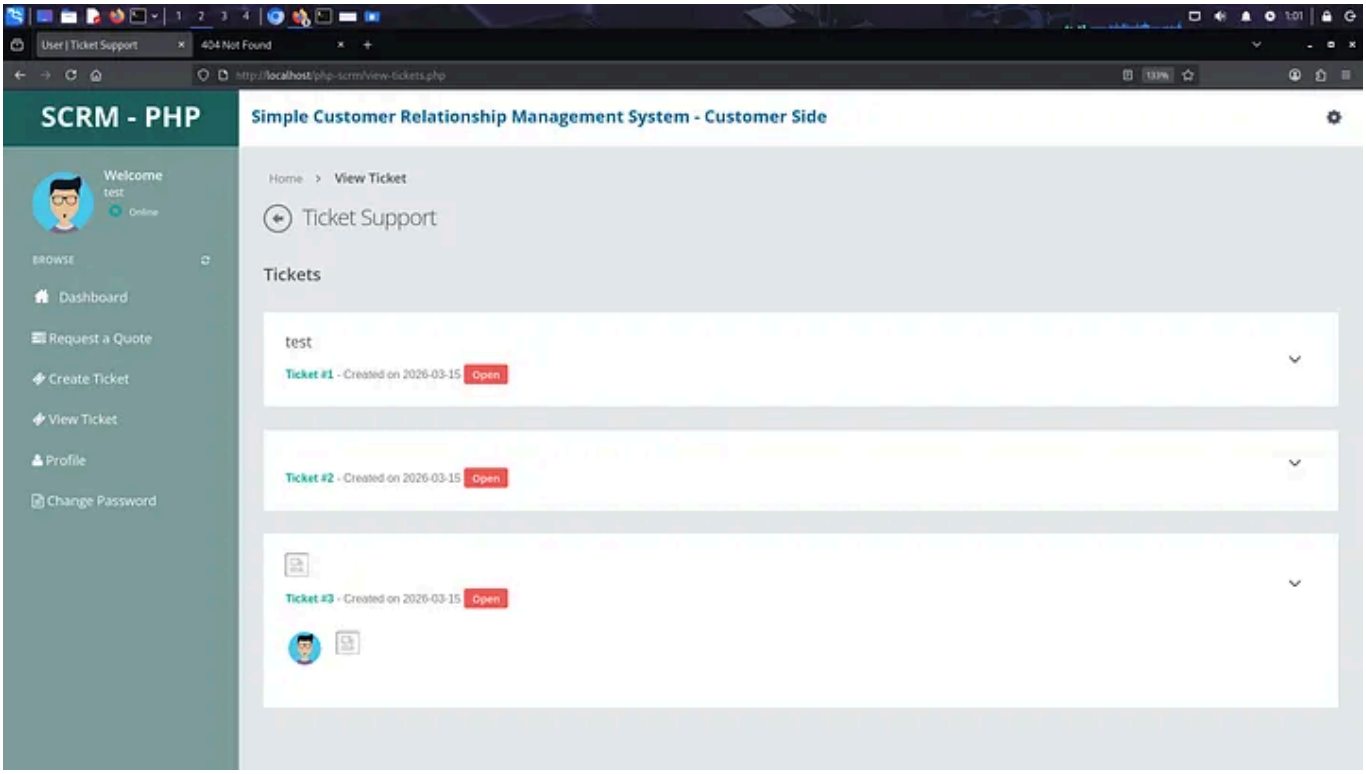
```
View Ticket
```



6. Trigger the vulnerability

When the stored ticket is viewed, the malicious JavaScript payload executes automatically.

The browser displays an alert box.



• • •

Proof of Concept Payload

Example payload used during testing:

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

Another payload that demonstrates access to sensitive information:

```
<script>alert(document.cookie)</script>
```

• • •

Impact

An attacker can exploit this vulnerability to:

- Execute arbitrary JavaScript in the victim's browser
- Steal session cookies
- Perform actions on behalf of the victim
- Conduct phishing attacks
- Deface application content

If an administrator views the malicious ticket, the attacker may gain access to privileged accounts.

. . .

Root Cause

The application does not properly sanitize or encode user-supplied input before displaying it in the browser.

Lack of:

- Output encoding
- Input validation
- Content Security Policy (CSP)

results in the execution of injected scripts.

. . .

Recommended Fix

To mitigate this vulnerability:

1. Sanitize user input

Validate and filter user input before storing it in the database.

2. Encode output

Use proper output encoding when displaying user data.

Example in PHP:

```
htmlspecialchars($input, ENT_QUOTES, 'UTF-8');
```

3. Implement Content Security Policy

Use a strong CSP header to prevent inline script execution.

Example:

```
Content-Security-Policy: default-src 'self';
```

4. Input validation

Reject input containing malicious scripts.

. . .

Conclusion

This Stored XSS vulnerability demonstrates the importance of proper input validation and output encoding in web applications.

By implementing secure coding practices, developers can prevent attackers from injecting malicious scripts and protect users from potential exploitation.

. . .

Author

Security Researcher: Hemant Raj Bhati
Category: Web Application Security
Affected Application: Simple Customer Relationship Management (CRM) System
Affected URL: http://localhost/php-scrm/
Vulnerability Type: Stored Cross-Site Scripting (Stored XSS)
Affected Feature: Create Ticket / View Ticket
Severity: Medium

Cybersecurity | Penetration Testing | Ethical Hacking | Red Team | Web Security

Written by Hemant Raj Bhati

6 followers · 1 following

Cybersecurity Researcher | OSCP Certified | CVE Analysis & Exploitation | Web Application Security | Documenting real-world vulnerabilities and PoCs.

No responses yet