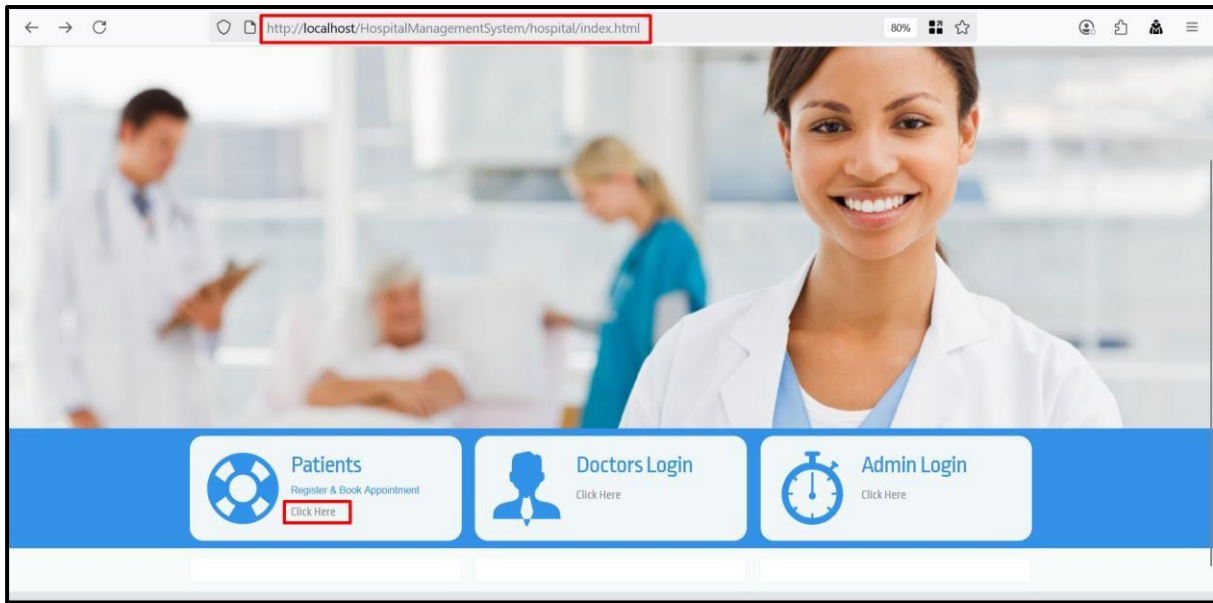


## CVE-2025-9754: Edit Profile Page XSS

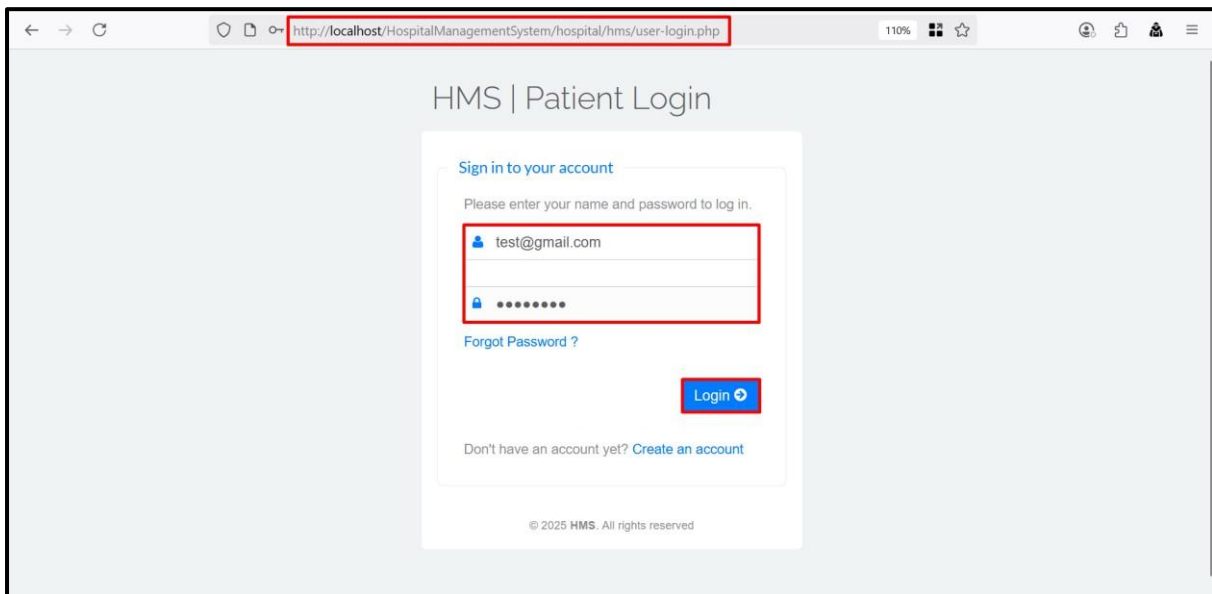
A flaw has been found in Campcodes Online Hospital Management System 1.0. The impacted element is an unknown function of the file /edit-profile.php of the component Edit Profile Page. Executing manipulation of the argument Username can lead to cross site scripting. The attack may be launched remotely. The exploit has been published and may be used.

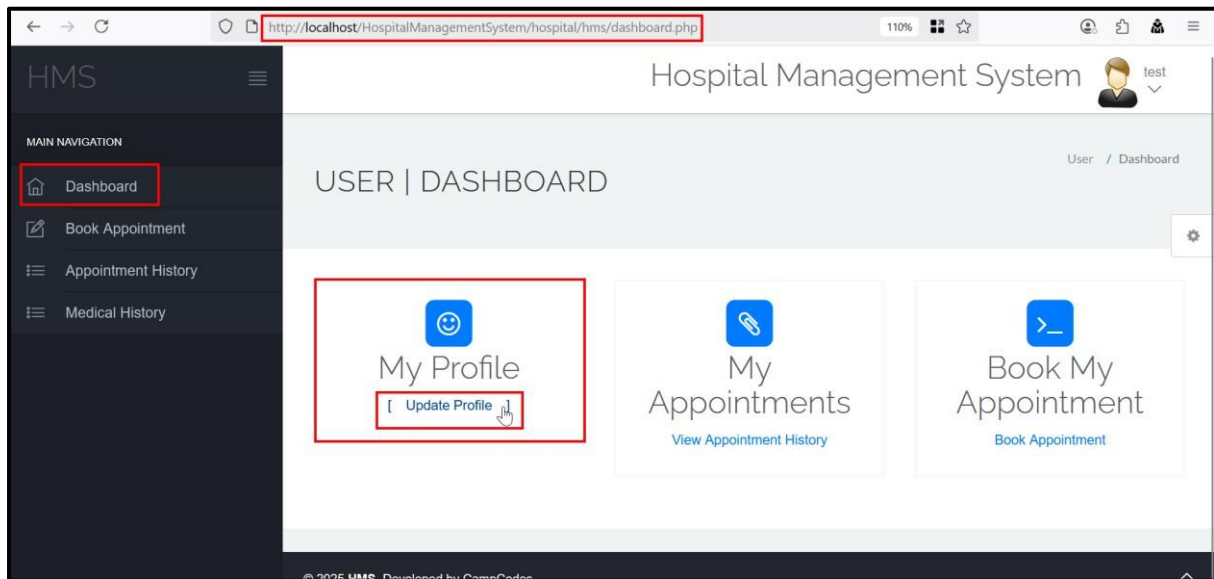
### Proof of concept (POC):

**Step 1:** Access the URL and log in as a patient user, as shown in the screenshot.



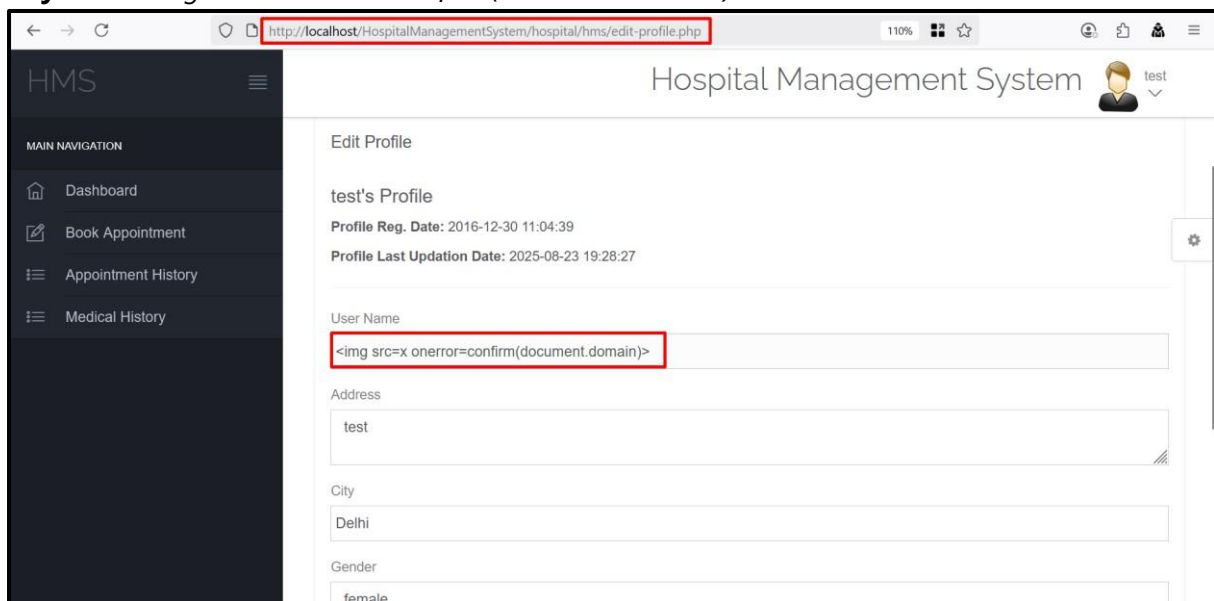
**Step 2:** After successfully logging in, click on "Update Profile" Under the Dashboard Module, as shown in the screenshot below.

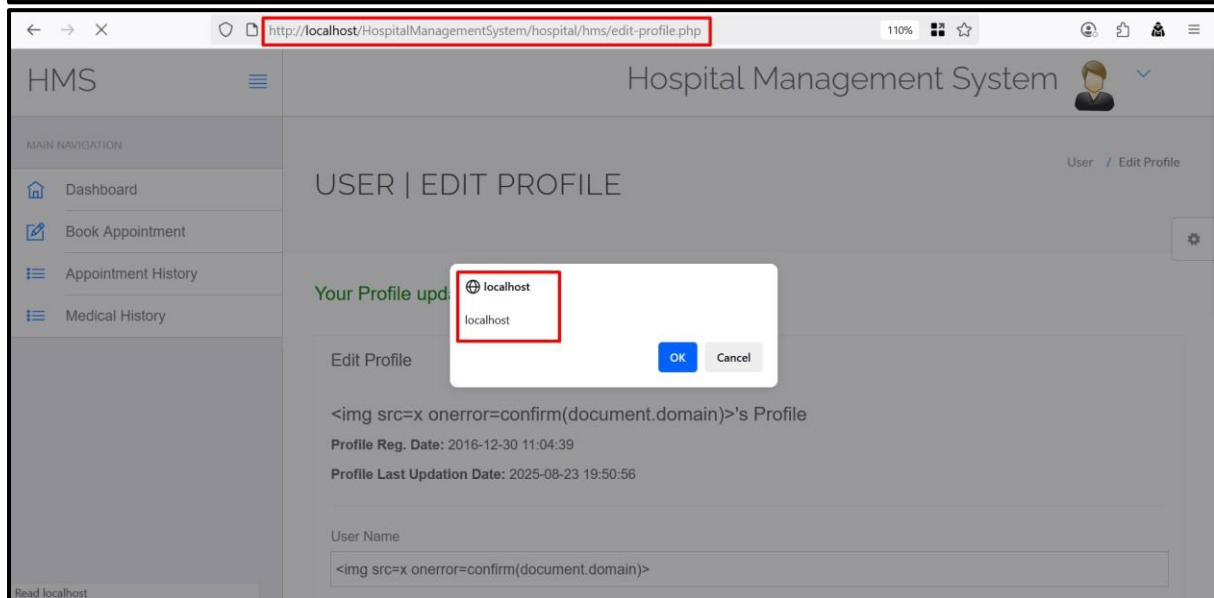
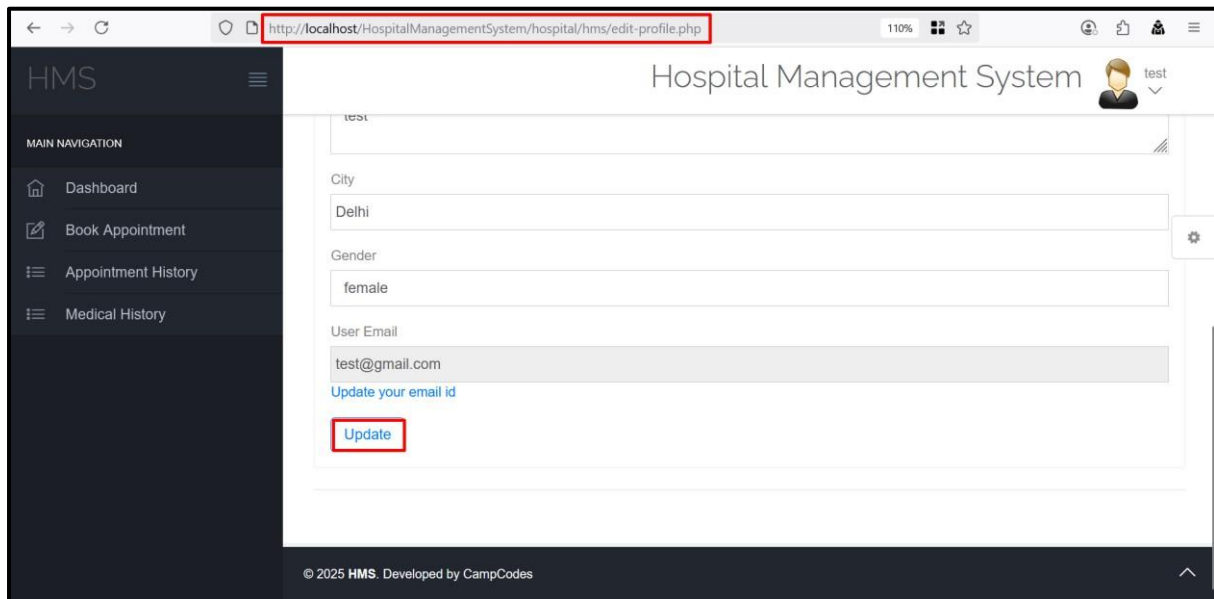




**Step 3:** Enter the following payload into the “User Name” field and click on the update button, as shown in the screenshot below. It can be observed that the payload executes successfully, indicating that the application does not properly sanitize user input.

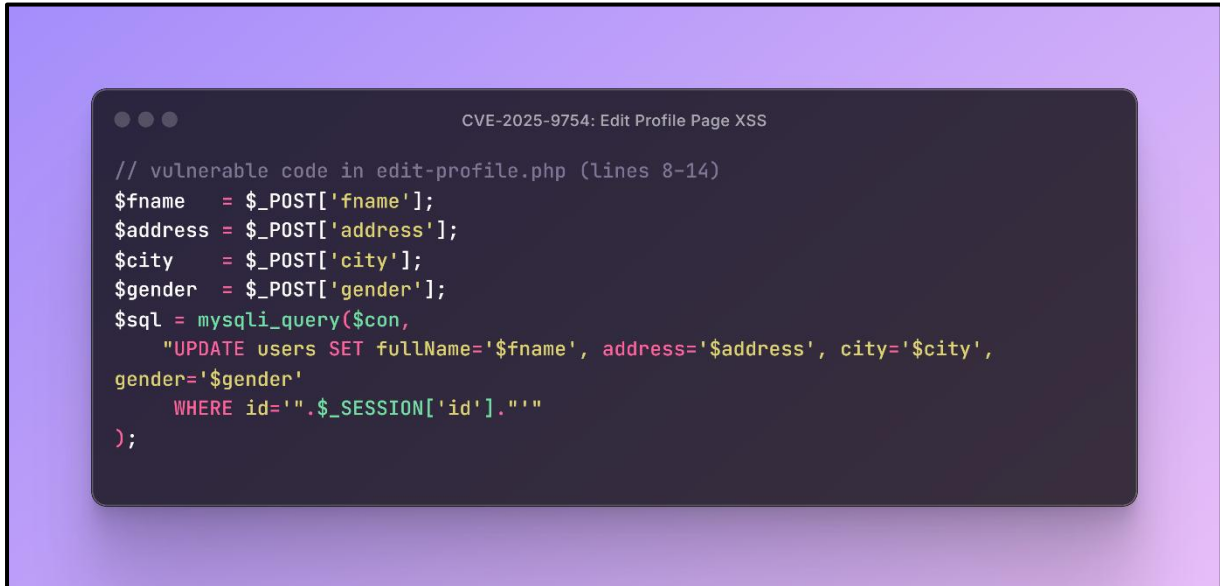
**Payload:** `<img src=x onerror=confirm(document.domain)>`





## Observation:

In Campcodes Online Hospital Management System 1.0, the edit-profile page (*edit-profile.php*) has a stored XSS vulnerability. NVD warns that manipulating the "Username" argument leads to cross-site scripting. In the PHP code, the form inputs are taken from `$_POST` and used directly in an SQL UPDATE without validation:

A screenshot of a code editor window titled "CVE-2025-9754: Edit Profile Page XSS". The code is in PHP and shows the vulnerable section of the edit-profile.php file (lines 8-14). It assigns values from \$\_POST to variables \$fname, \$address, \$city, and \$gender, and then uses these variables directly in an SQL UPDATE statement without any sanitization or escaping.

```
// vulnerable code in edit-profile.php (lines 8-14)
$fname = $_POST['fname'];
$address = $_POST['address'];
$city = $_POST['city'];
$gender = $_POST['gender'];
$sql = mysqli_query($con,
    "UPDATE users SET fullName='$fname', address='$address', city='$city',
    gender='$gender'
    WHERE id='".$_SESSION['id']."'
");
```

Here, `$fname` (the "User Name" field) is inserted into the database verbatim. If an attacker includes HTML/JS in that field (e.g. `<script>alert(1)</script>`), it becomes stored in the profile. The page does use `htmlspecialchars()` when redisplaying these values (which would normally neutralize scripts), but the CVE report indicates the flaw. It's likely the malicious value may appear unescaped elsewhere or that the input is not being sanitized at all.

- **Unsanitized input:** The code above uses raw `$_POST` values in SQL. There is no check or escaping of `$fname` before storing.
- **Potential XSS:** Stored malicious content in `fullName` could execute when output without encoding. (Any page that prints this field without `htmlspecialchars` would trigger it.).
- **Mitigation:** Sanitize or validate inputs before storing. Always use parameterized queries (e.g. with `mysqli_prepare()`) and apply escaping like `htmlspecialchars()` on output.

## References:

<https://nvd.nist.gov/vuln/detail/CVE-2025-9754>  
<https://www.cve.org/CVERecord?id=CVE-2025-9754>  
<https://www.cvedetails.com/cve/CVE-2025-9754/>  
<https://www.tenable.com/cve/CVE-2025-9754>  
<https://vuldb.com/?id.322055>