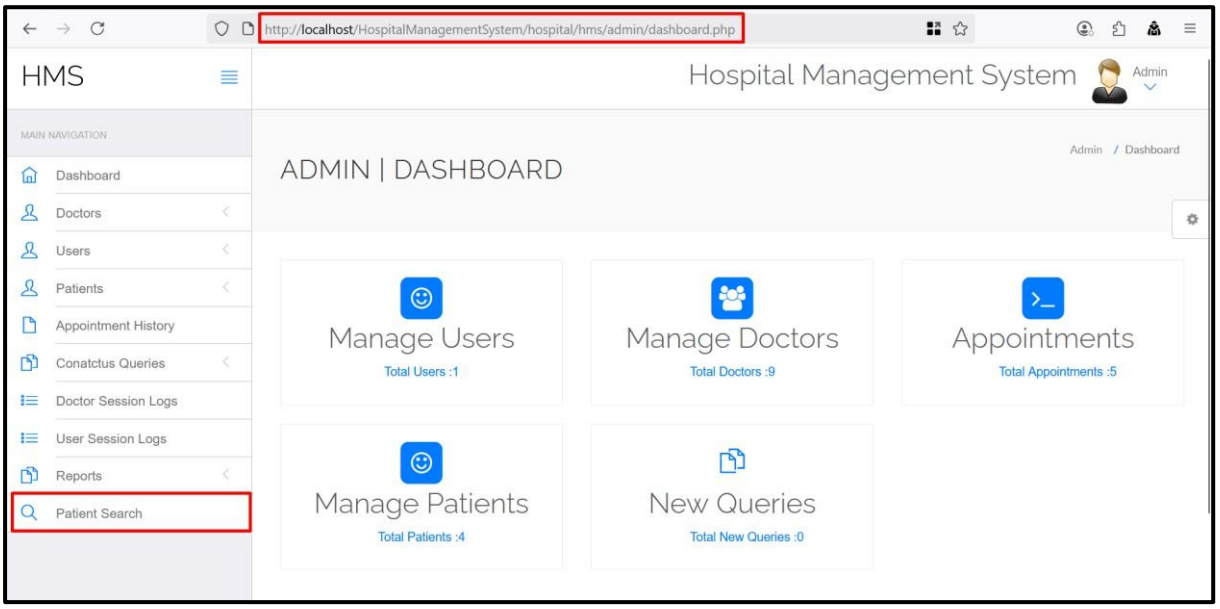
CVE-2025-9753: Patient Search Page XSS

A vulnerability was detected in Campcodes Online Hospital Management System 1.0. The affected element is an unknown function of the file /admin/patient-search.php of the component Patient Search Module. Performing manipulation of the argument Search by Name Mobile No results in cross-site scripting. The attack may be initiated remotely. The exploit is now public and may be used.

Proof of concept (POC):

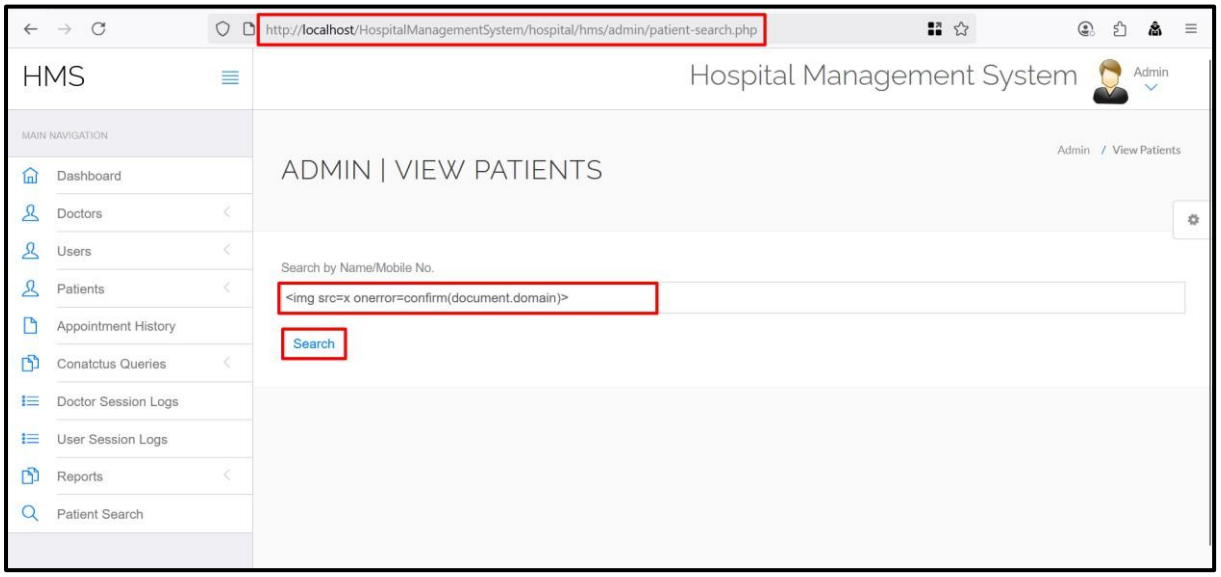
Step 1: Access the URL and log in as an admin, as shown in the screenshot.

Step 2: After successfully logging in, click on "Patient Search", as shown in the screenshot below.



Step 3: Enter the following payload into the "Search by Name/Mobile No" field and hit search 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:

The patient search feature (*/admin/patient-search.php*) in Campcodes Online HMS 1.0 is vulnerable to XSS. NVD reports that the "Search by Name Mobile No" argument can be manipulated to trigger cross-site scripting. In the code, the search term is echoed back into the HTML without escaping, and it is also used unsafely in an SQL query:

The line that outputs *<h4>Result against "$sdata" keyword</h4>* (around line 87) directly injects the user's search term into the page. If an attacker submits a string like *"><script>alert(1)</script>*, it will be embedded and executed in the page. This is a classic reflected XSS. (The SQL query line is also vulnerable to SQL injection, since $sdata is unescaped, but the CVE specifically notes XSS.)

- **Unsanitized echo:** The *echo ... $sdata ...* line injects user input into HTML without htmlentities.
- **Impact:** An attacker's script can run in the admin's browser context when viewing search results.
- **Mitigation:** Encode output: use echo htmlspecialchars($sdata, ENT_QUOTES) or similar. Also fix SQL injection by using prepared statements.

References:

https://nvd.nist.gov/vuln/detail/CVE-2025-9753
https://www.cvedetails.com/cve/CVE-2025-9753/
https://www.tenable.com/cve/CVE-2025-9753
https://www.cve.org/CVERecord?id=CVE-2025-9753
https://vuldb.com/?id.322054