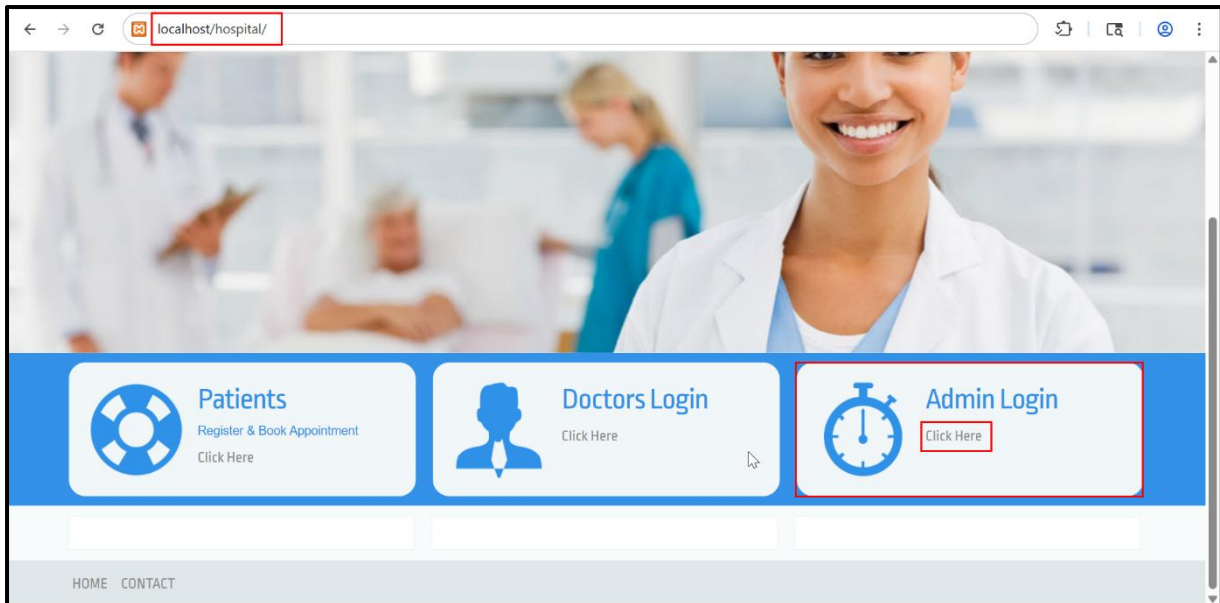


CVE-2025-9770: Admin Dashboard Login SQL Injection

A weakness has been identified in Campcodes Hospital Management System 1.0. Affected by this vulnerability is an unknown functionality of the file /admin/ of the component Admin Dashboard Login. This manipulation of the argument Password causes SQL Injection. It is possible to initiate the attack remotely. The exploit has been made available to the public and could be exploited.

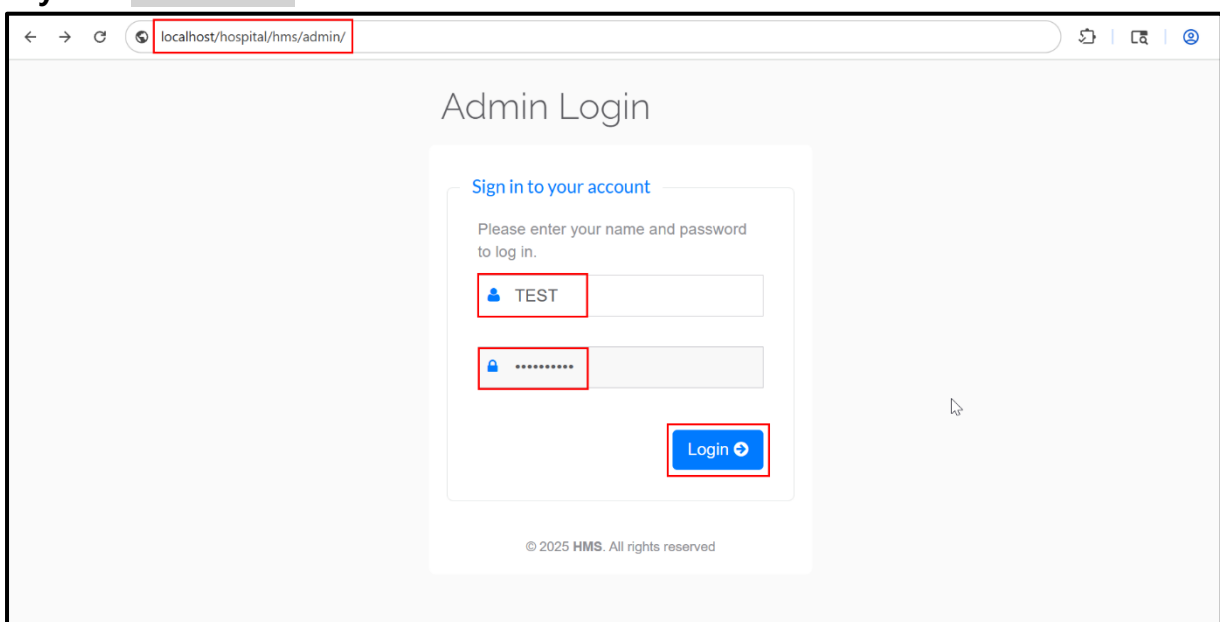
Proof of concept (POC):

Step 1: Access the URL and click on the "Click Here" button under Admin login, as shown in the screenshot.

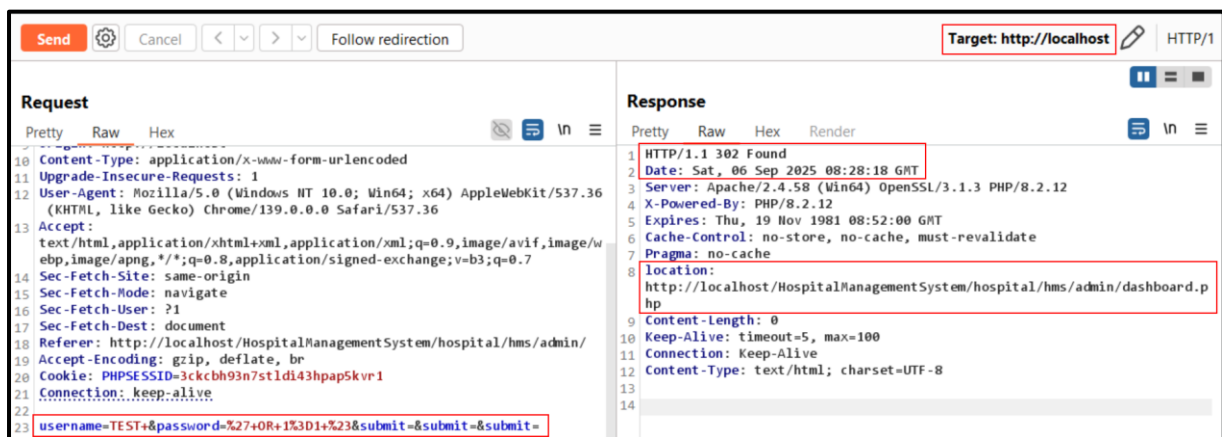
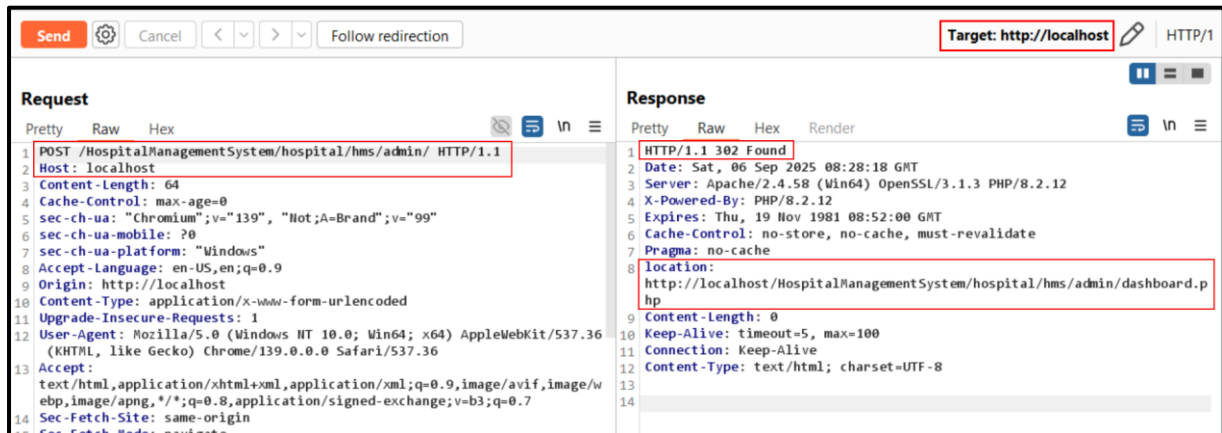


Step 2: Enter any value in the username field, then insert the following payload into the password field and click the login button, as shown in the screenshot below.

Payload: `'+OR+1=1 #`



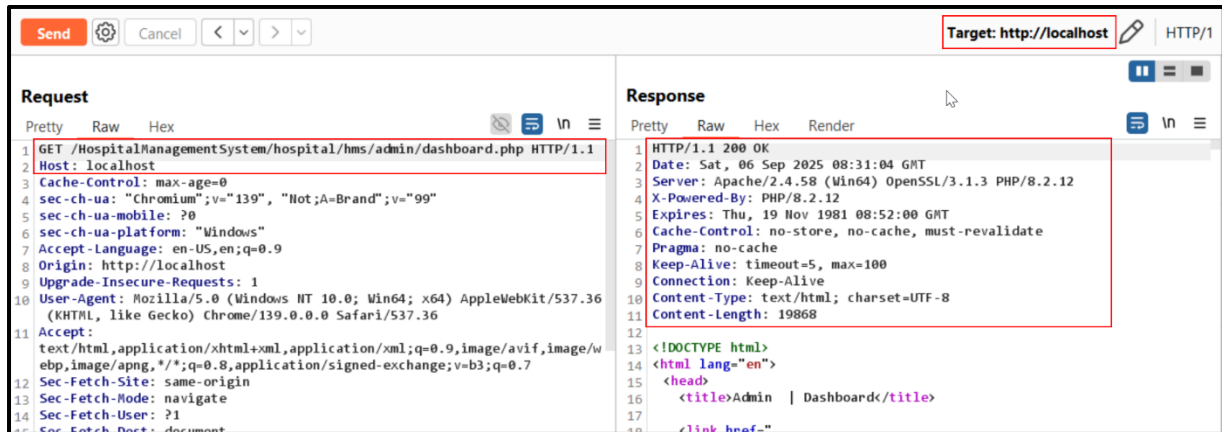
Step 3: Intercept the request using a proxy tool such as Burp Suite and send it to the Repeater. It can be observed that the application responds with a 302 status code along with a Location header that redirects to `/admin/dashboard.php`, as shown in the screenshot below.



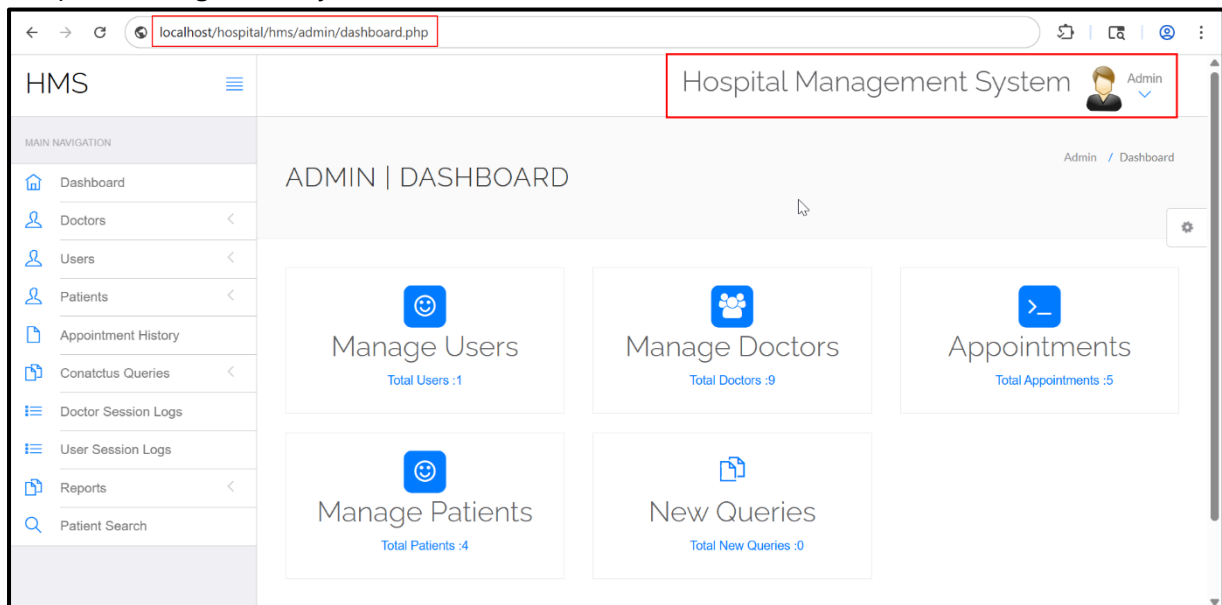
From the screenshot below, we can see the decoded version of the payload.



After following the redirection, the application responds with a 200 OK status.

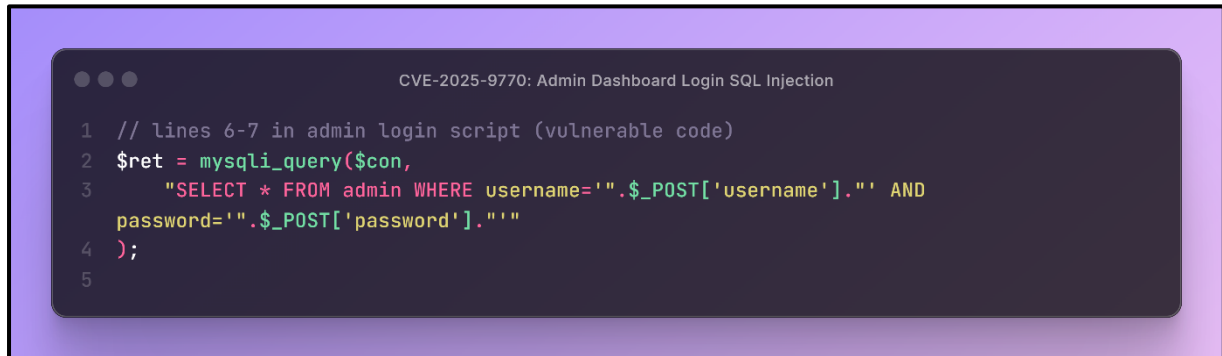


Step 4: As shown in the screenshot below, by using the provided SQL injection payload, the login page was successfully bypassed, and access was gained to the Admin Dashboard of the Hospital Management System.



Observation:

This vulnerability affects Campcodes Hospital Management System (HMS) 1.0's admin login page. The NVD description notes that "manipulation of the argument Password causes SQL Injection". In the PHP code (lines 6–7 of the login script), the SQL query is constructed by directly concatenating the `$_POST['username']` and `$_POST['password']` inputs without any sanitization:

A screenshot of a code editor with a dark background and light-colored text. The title bar of the editor window reads "CVE-2025-9770: Admin Dashboard Login SQL Injection". The code is as follows:

```
1 // lines 6-7 in admin login script (vulnerable code)
2 $ret = mysqli_query($con,
3     "SELECT * FROM admin WHERE username='".$_$_POST['username']."' AND
4     password='".$_$_POST['password']."'";
5 );
```

Because the inputs are not escaped or parameterized, an attacker can inject SQL syntax. For example, submitting a password like `'OR '1'='1` would turn the WHERE clause into a tautology, allowing unauthorized access. The exploit is remote; as the login page is public, an attacker can send crafted credentials to bypass authentication.

- **Vulnerable code:** Lines 6–7 in the login script build the SQL query with raw input (shown above).
- **Consequence:** Successful SQL injection can bypass login and access the admin dashboard, compromising the system.
- **Mitigation:** Use prepared statements or escape inputs. For example, use `mysqli_prepare()` with bound parameters instead of concatenation.

References:

<https://nvd.nist.gov/vuln/detail/CVE-2025-9770>
<https://vuldb.com/?submit.640807>
<https://www.cvedetails.com/cve/CVE-2025-9770/>
<https://www.cve.org/CVERecord?id=CVE-2025-9770>
<https://vulners.com/cve/CVE-2025-9770>
<https://www.tenable.com/cve/CVE-2025-9770>