FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

Department of Computer Engineering

1. Course, Subject & Experiment Details

Practical No:	
Title:	**Implementing SQL Injection**
Name of the Student:	**Warren Fernandes**
Roll No:	**8940**
Date of Performance:	**28-03-2022**
Date of Submission:	**09-04-2022**

Evaluation:

Sr. No.	Rubric	Grade
1	On time submission/completion (2)	
2	Preparedness (2)	
3	Skill (4)	
4	Output (2)	

Signature of the Teacher

SQL Injection:

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).

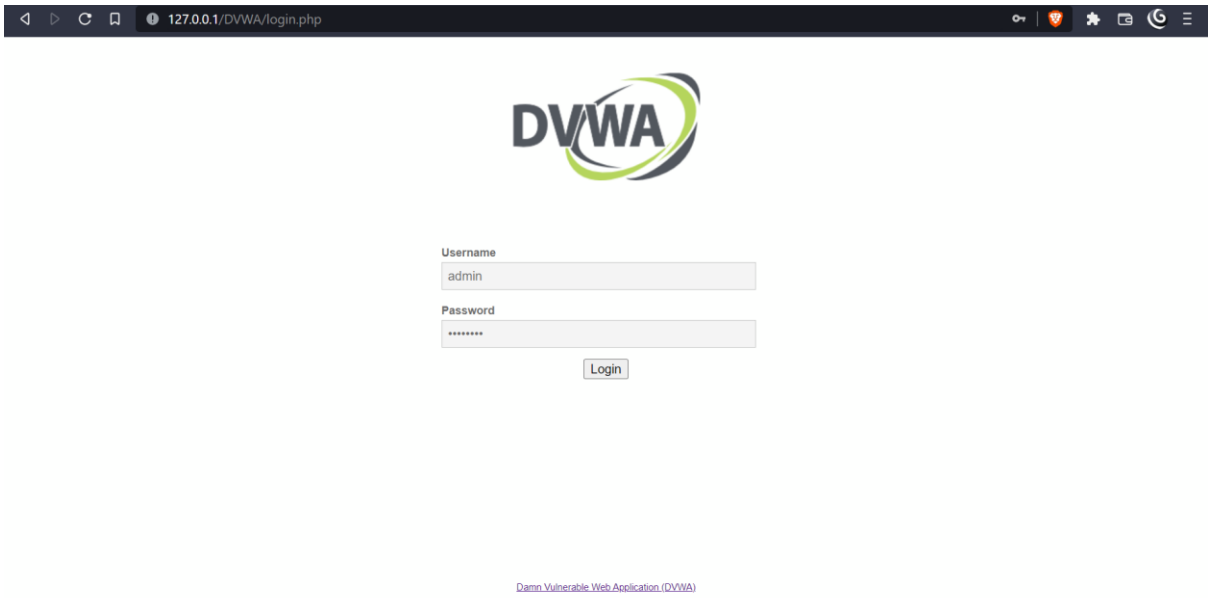
SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server.

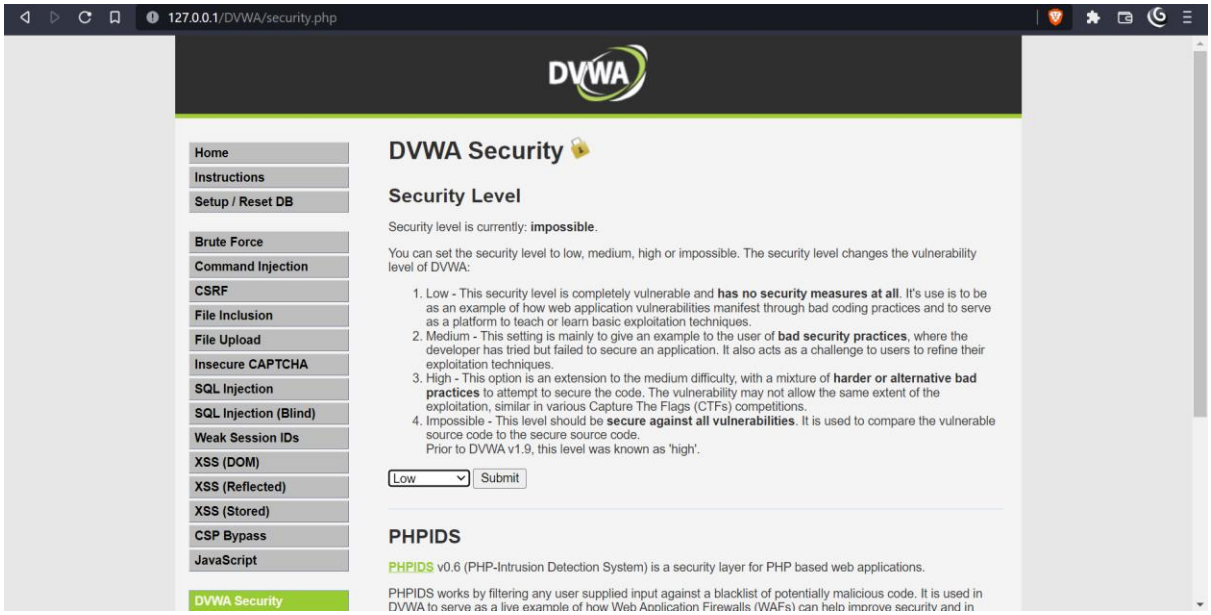
Implementation:

Step 1: Setup DVWA for SQL Injection

After successfully installing DVWA, open your browser and enter the required URL 127.0.0.1/dvwa/login.php Log in using the username "admin" and password as "password". These are the default DVWA login credentials. After a successful login, set the DVWA security to LOW then click on SQL Injection on the left-side menu.

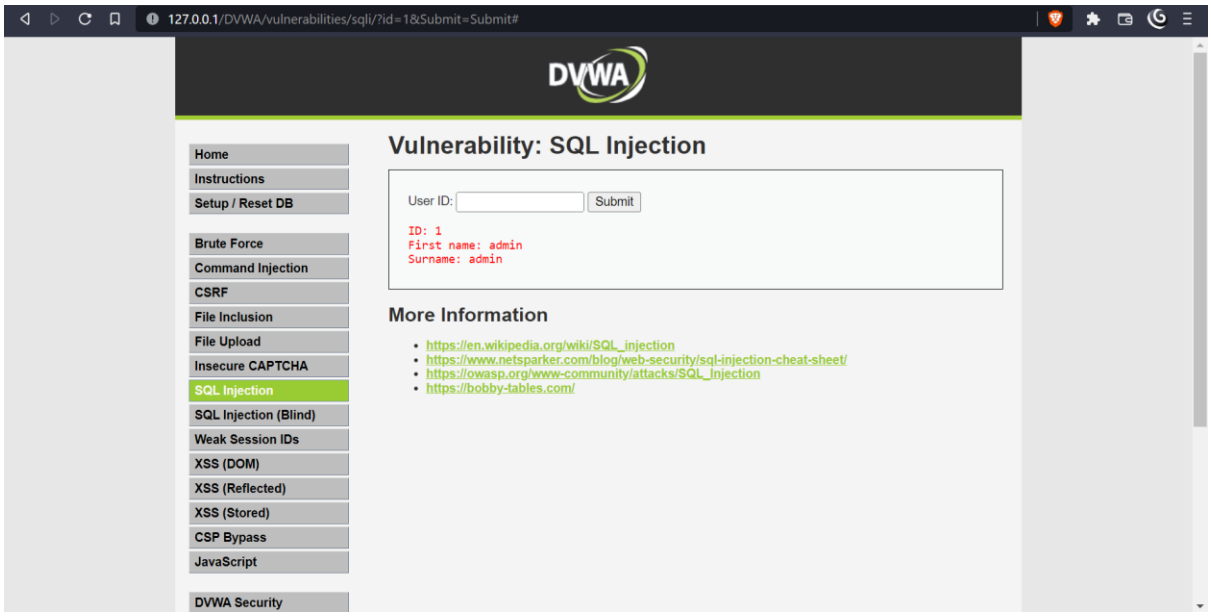


Step 2: — Set your DVWA security to low.



Step 2: Basic Injection

On the User ID field, enter “1” and click Submit. That is supposed to print the ID, First_name, and Surname on the screen as you can see below.



Step 4: Display Database Version

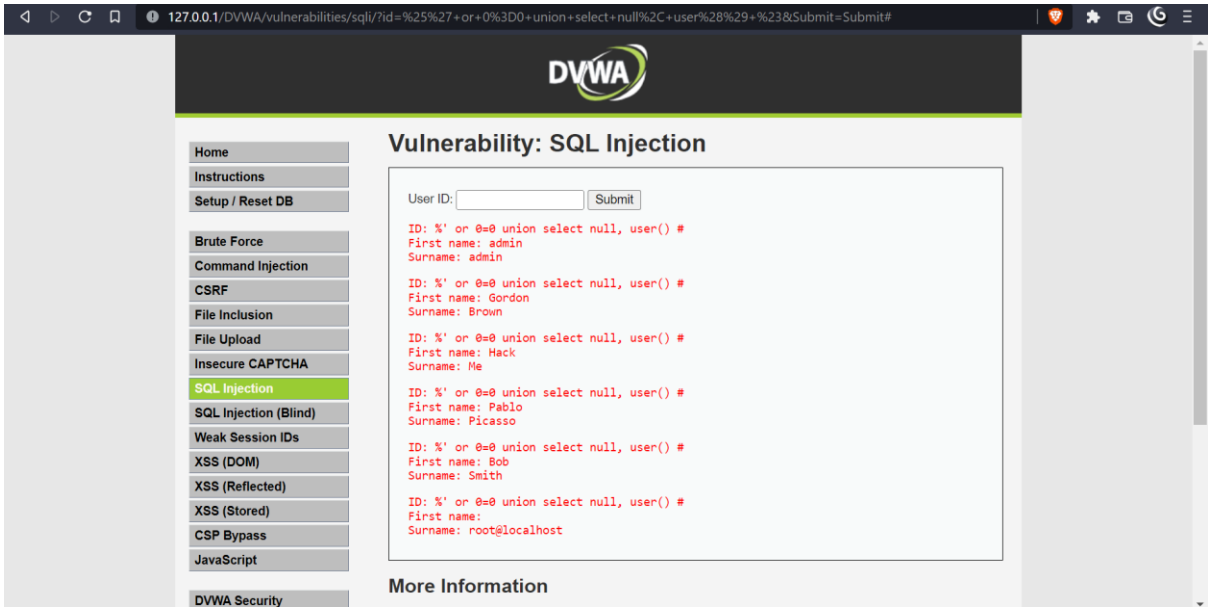
To know the database version the DVWA application is running on, enter the text below in the User ID field.

The database version will be listed under surname in the last line as shown in the image below.

Step 5: Display Database User

To display the Database user who executed the PHP code powering the database, enter the text below in the USER ID field.

The Database user is listed next to the surname field in the last line as in the image below.

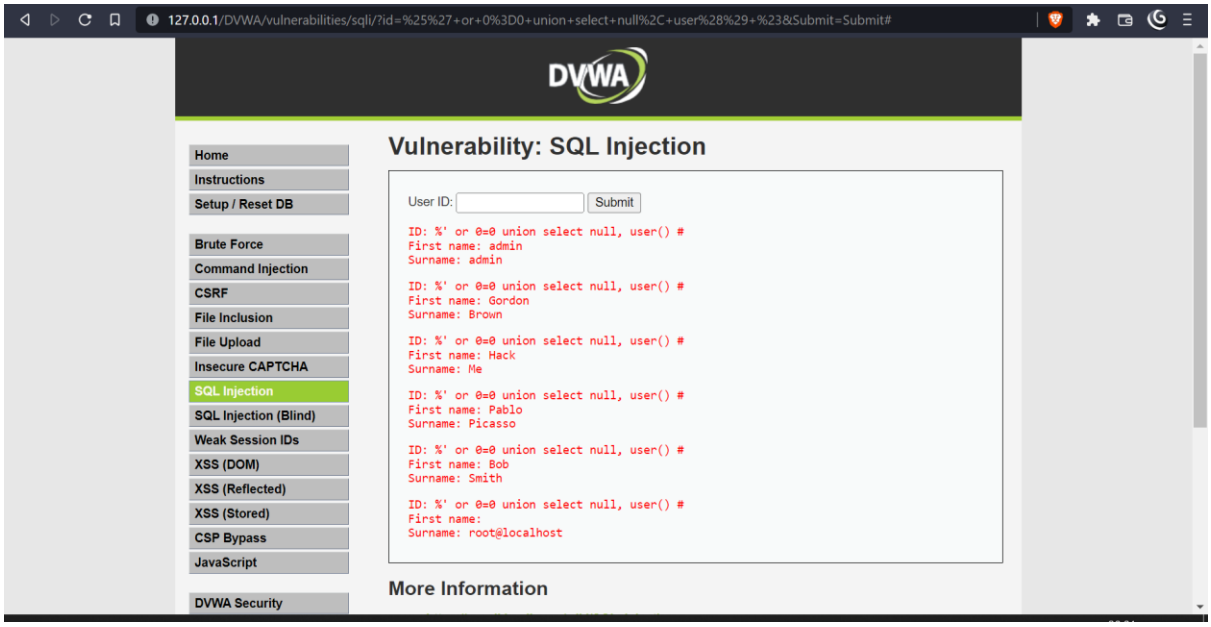


Step 6: Display Database Name

To display the database name, we will inject the SQL code below in the User ID field.

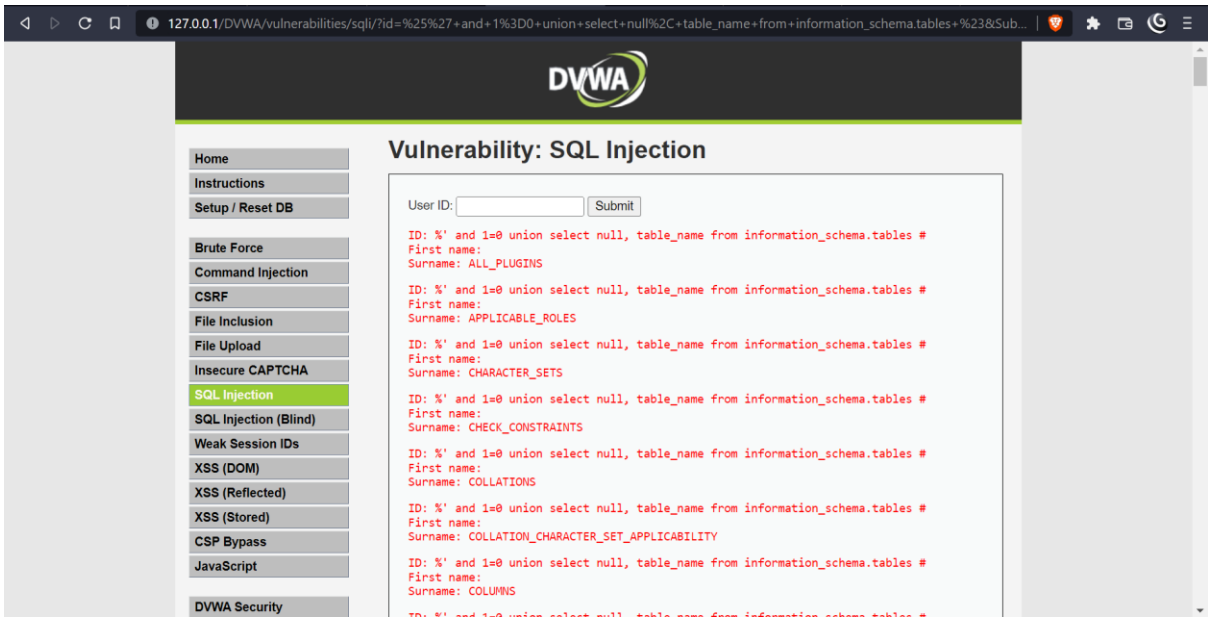
%' or 0=0 union select null, user() #

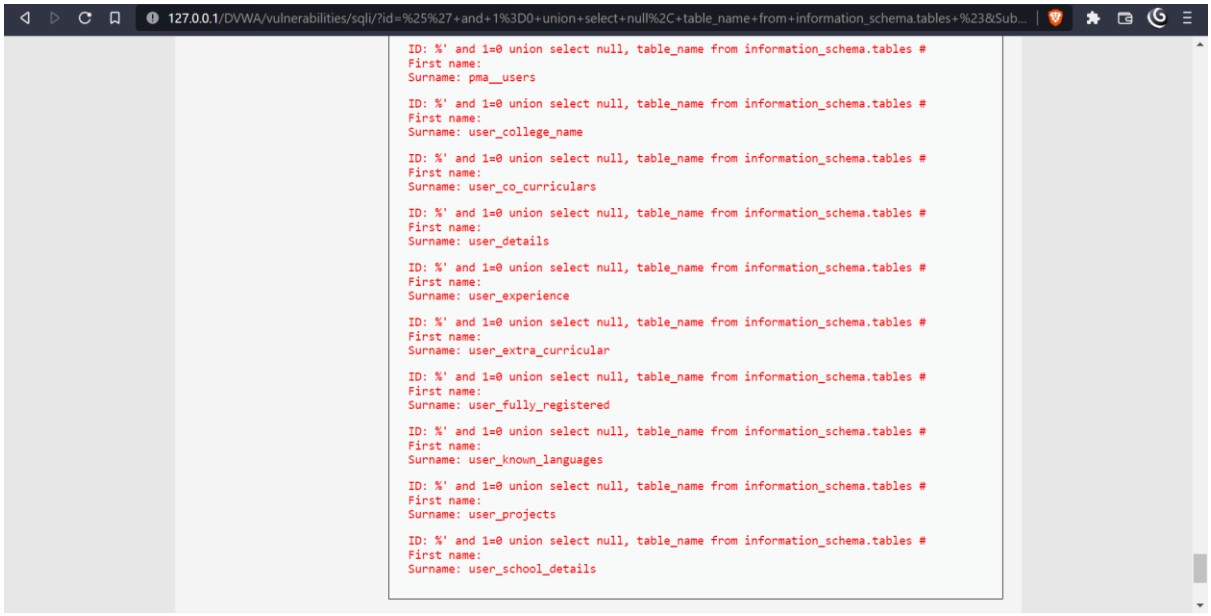
The database name is listed next to the surname field in the last line.



Step 7: Display all tables in information_schema

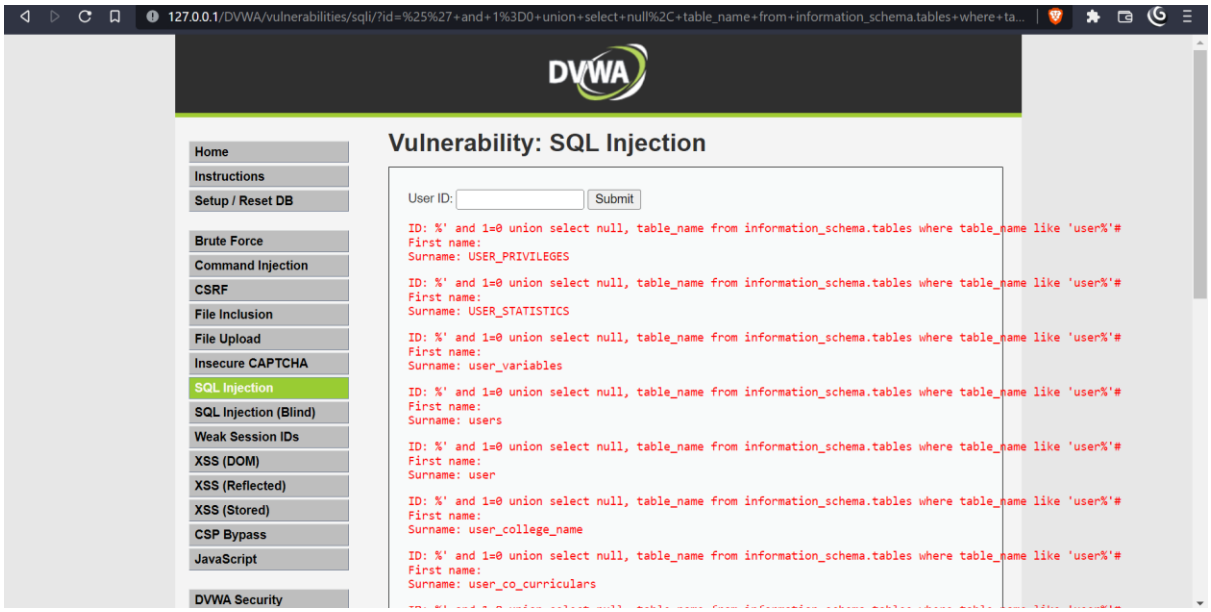
The Information Schema stores information about tables, columns, and all the other databases maintained by MySQL. To display all the tables present in the information_schema, use the text below.





Step 8: Display all the user tables in information_schema

For this step, we will print all the tables that start with the prefix user as stored in the information_schema. Enter the SQL code below in the User ID.

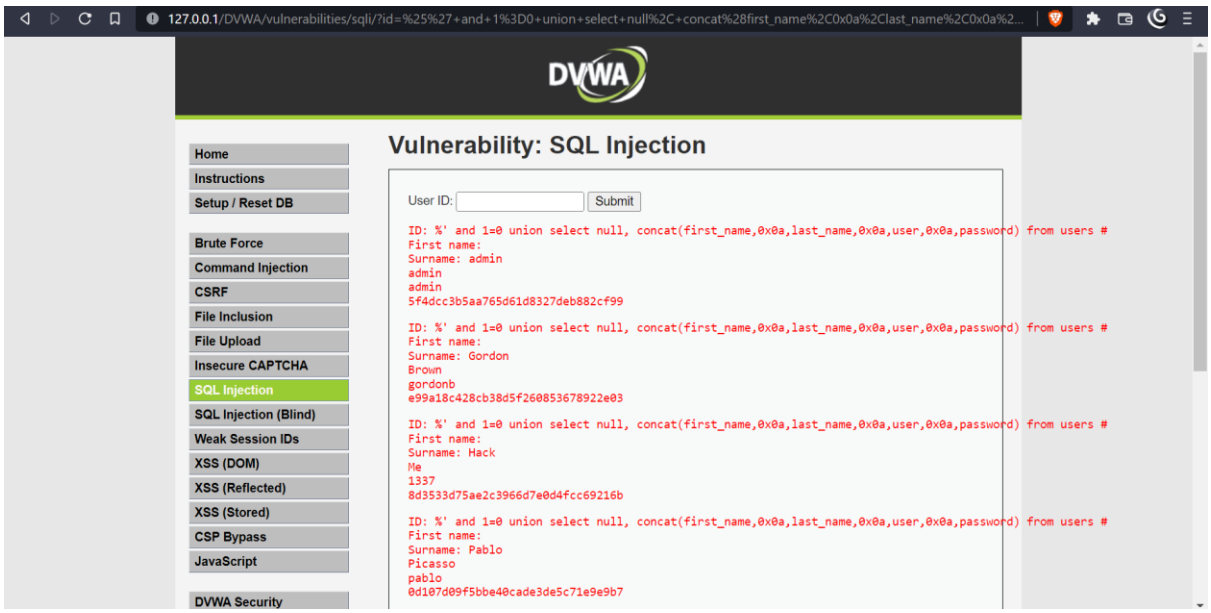


Step 9: Display all the columns fields in the information_schema user table

We will print all the columns present in the users' table. This information will include column information like User_ID, first_name, last_name, user, and password. Enter the input in the User_ID field.

Step 10: Display Column field contents

To display all the necessary authentication information present in the columns as stored in the information_schema, use the SQL syntax below:



Postlab:

1. How to Prevent SQL Injection attacks?
   Preventing SQL Injection vulnerabilities is not easy. Specific prevention techniques depend on the subtype of SQLi vulnerability, on the SQL database engine, and on the programming language. However, there are certain general strategic principles that you should follow to keep your web application safe.

   Step 1: Train and maintain awareness
   To keep your web application safe, everyone involved in building the web application must be aware of the risks associated with SQL Injections. You should provide suitable security training to all your developers, QA staff, DevOps, and SysAdmins. You can start by referring them to this page.

   Step 2: Don't trust any user input - Treat all user input as untrusted. Any user input that is used in an SQL query introduces a risk of an SQL Injection. Treat input from authenticated and/or internal users the same way that you treat public input.

   Step 3: Use whitelists, not blacklists

   Don't filter user input based on blacklists. A clever attacker will almost always find a way to circumvent your blacklist. If possible, verify and filter user input using strict whitelists only.

   Step 4: Adopt the latest technologies

   Older web development technologies don't have SQLi protection. Use the latest version of the development environment and language and the latest technologies associated with that environment/language. For example, in PHP use PDO instead of MySQLi.

   Step 5: Employ verified mechanisms

   Don't try to build SQLi protection from scratch. Most modern development technologies can offer you mechanisms to protect against SQLi. Use such mechanisms instead of trying to reinvent the wheel. For example, use parameterized queries or stored procedures.

   Step 6: Scan regularly (with Acunetix)

   SQL Injections may be introduced by your developers or through external libraries/modules/software. You should regularly scan your web applications using a web vulnerability scanner such as Acunetix. If you use Jenkins, you should install the Acunetix plugin to automatically scan every build.