**Case Study 7**

**Aim -** Data Validation Experiment.

**Theory -**

**What is SQL Injection?**

SQL Injection (SQLi) is a cyberattack where hackers exploit weak points in a website or app to mess with the underlying database. They can access, change, or steal data they're not supposed to, which can be really bad for businesses and users. It's a common and serious problem on the internet.

**How and Why is an SQL Injection Attack performed?**

SQL Injection is a type of cyberattack where attackers exploit weak spots in a website or app to manipulate a database. They do this by injecting harmful commands into user inputs. Here's why and how:

Why:

* Stealing User Data: Attackers can steal sensitive information like usernames and passwords from the database.
* Accessing Everything: They can gain complete access to the entire database, which could contain a lot of valuable data.
* Tampering with Data: Attackers can change or delete data, like altering account balances or deleting records.
* Disrupting Services: They can disrupt services by deleting data or even crashing the entire database.
* Potentially Accessing Systems: In some cases, attackers can use this to get into the underlying system behind the website.

How:

* Finding Vulnerabilities: Attackers look for weak spots in a website where user input isn't properly checked.
* Injecting Malicious Commands: They insert harmful commands into the input fields, which can then run on the database.

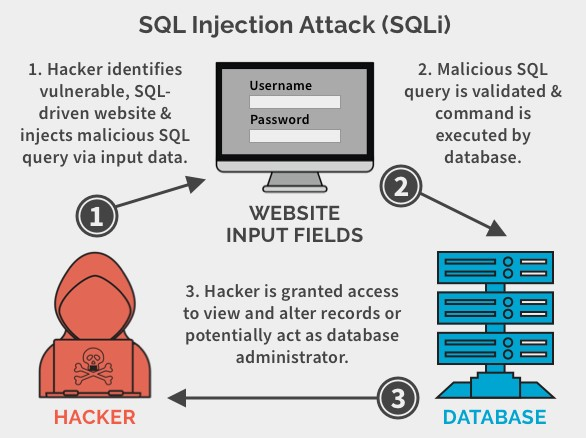
There are different types of SQL Injection attacks, but they all involve exploiting these weaknesses to mess with the database.

**How to prevent an SQL Injection?**

To prevent SQL Injection:

* Train Your Team: Educate everyone involved about the risks.
* Validate User Input: Always consider user input as untrusted.
* Use Whitelists, Not Blacklists: Don't just filter out bad input, allow only approved input.
* Stay Updated: Use the latest technologies and security features in your programming language.
* Leverage Verified Mechanisms: Don't create security from scratch; use built-in protections like parameterized queries.
* Regularly Scan for Vulnerabilities: Use tools like Acunetix to scan your application for potential issues.

**SQL Injection Process**

****

**SQL Injection Exploitation using DVWA**

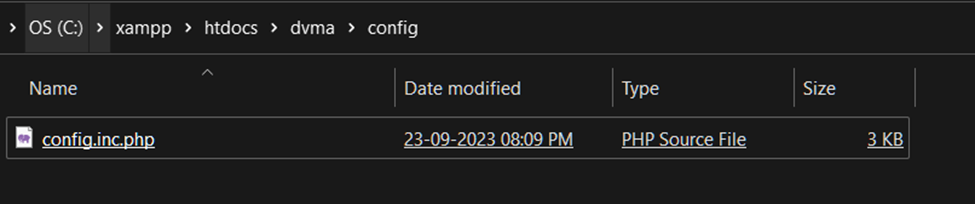
Here, we will use the Damn Vulnerable Web Application (DVWA). It’s a web app developed in PHP and MySQL and intentionally made to be vulnerable.

Step 1: Now Download DVWA (Damn Vulnerable Web Application) from Github for Pen Testing purpose. [digininja/DVWA - Damn Vulnerable Web Application](https://github.com/digininja/DVWA)

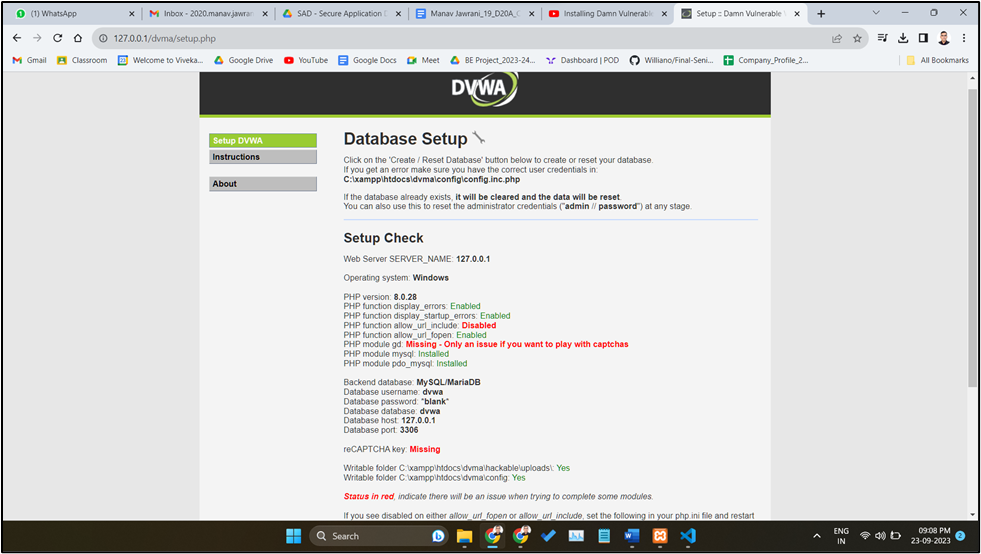
Step 2: Download Xampp and install. Reference for installation of Burp Suite, Xampp and DVWA. [Installing Damn Vulnerable Web Application (DVWA) on Windows 10](https://www.youtube.com/watch?v=cak2lQvBRAo)

Step 3: Move the DVWA folder to htdocs folder under Xampp:

* Goto config -> Change the **config.inc.php.dist** file name to **config.inc.php**



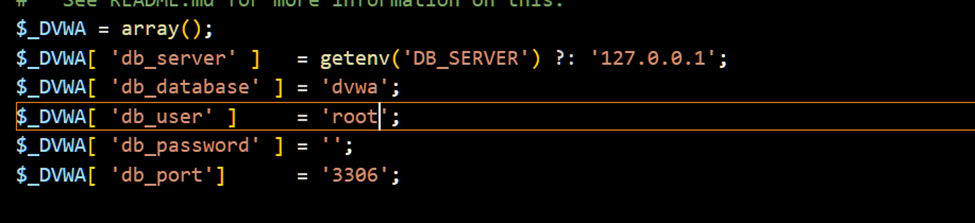
* Browse to <http://127.0.0.1/DVWA-master>

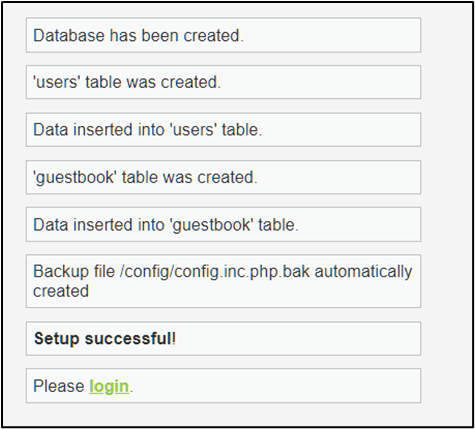


* If it gives Sql Error , Open Config.inc.php file and change the

$\_DVWA[ 'db\_user' ] = 'root';

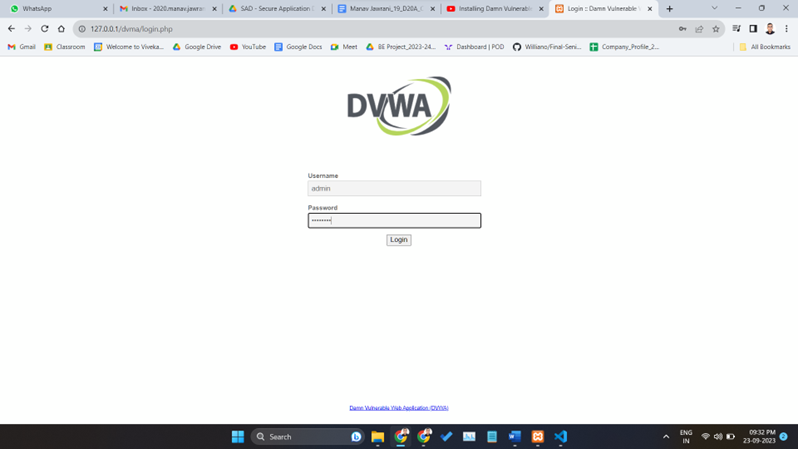
$\_DVWA[ 'db\_password' ] = ' ';



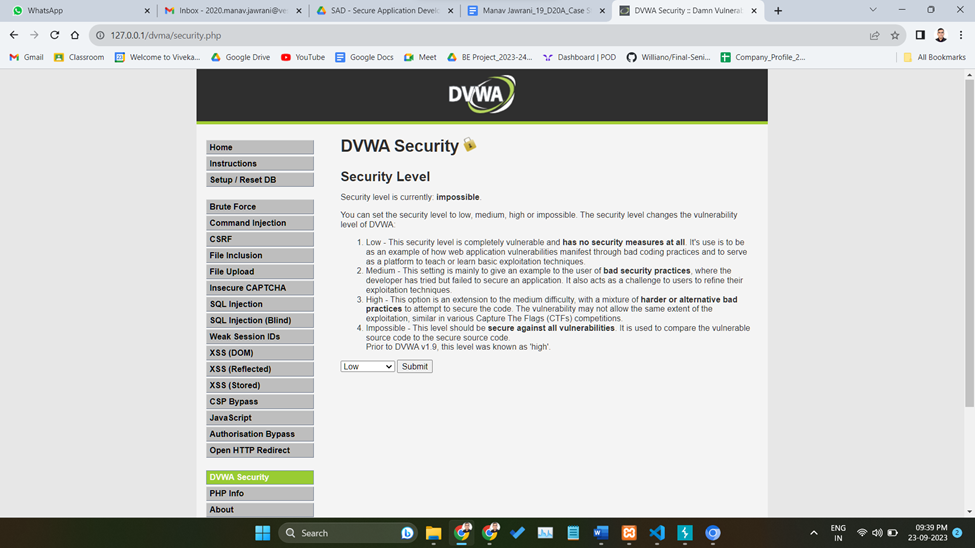


2. Default Credentials

* Default username = admin
* Default password = password



* After a successful login, you will see the DVWA main page. First, click on the DVWA Security on the bottom left, set security to Low, and click Submit.



Step 5: On the left section of the page, you will see the various vulnerable pages to exploit. Click SQL Injection. You should see a page similar to this below.



Step 6: On the SQL injection page, click the View Source button at the bottom right. That will open a page with the SQL Injection source code written in PHP. When you go through the code, you will see a line like:

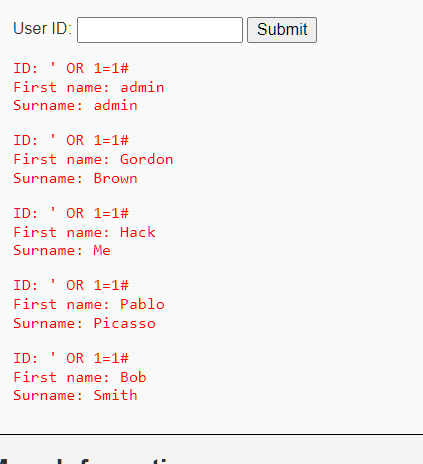
**$query = "SELECT first\_name, last\_name FROM users WHERE user\_id = '$id'";**

That is the vulnerable line of code. At the end of the line, you can see the user input is concatenated to the SQL query without being validated. That allows us to pass arbitrary commands into the database. Let’s get started.

On the SQL Injection page, we have a USER ID field. When we enter number 1, the application returns the Firstname and Surname of the user with ID 1.



Step 7: We looked at this when talking about How an SQL Injection attack works. Enter an input like test**' OR 1=1#** and hit Enter. That will return the username and surname of all users in the database.



Step 8: List all tables in the information schema. The Information Schema is a record that holds information about all other databases maintained by MySQL RDBMS. Enter the query below in the USER ID field.

**test' and 1=0 union select null, table\_name from information\_schema.tables where table\_name like 'user%'#**



Step 9: Display all the column contents in the information schema users table. This is much more interesting. We will display all the authentication information of all users in the database. That includes password hashes. Enter the query below.

**test' and 1=0 union select null, concat(first\_name,0x0a,last\_name,0x0a,user,0x0a,password) from users #**



From the output above, you can see the hashed password. We can go ahead and crack the hash to reveal the actual password. Some of the password cracking tools that come in handy include John the Ripper and Medusa. There are also websites where you can paste the password hash to reveal the actual password.

**Conclusion:**

Thus we have studied how to validate the data and how to perform SQL injection attacks.