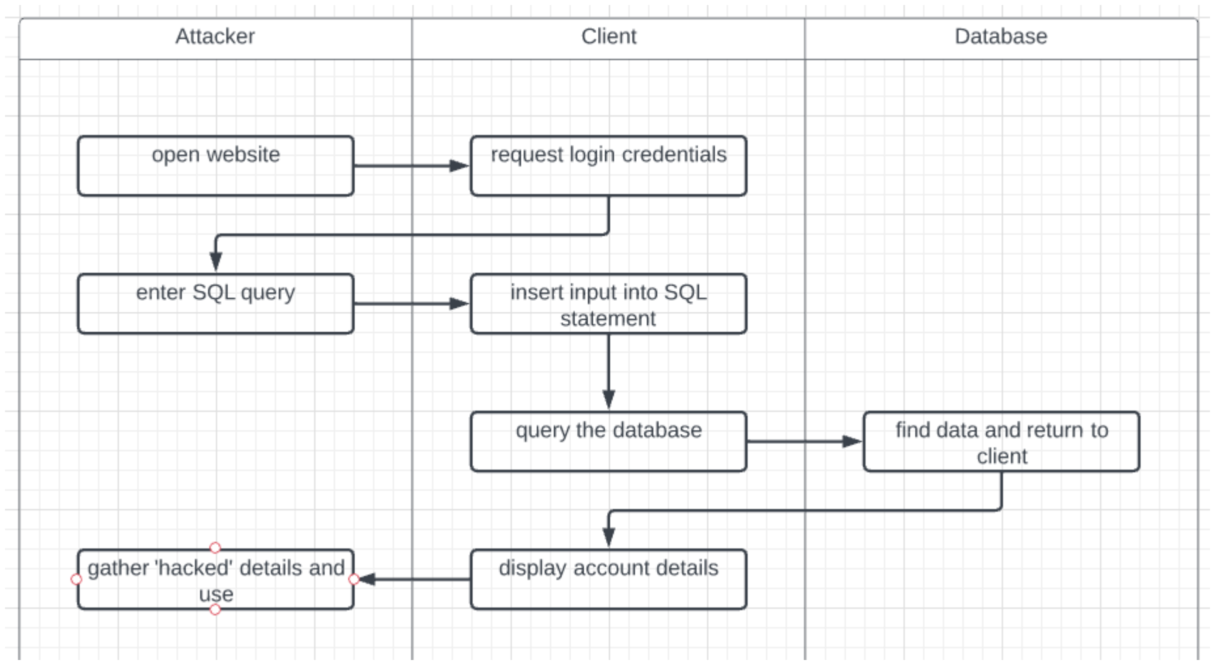
Unit 1 Initial Post

SQL Injection

SQL Injection is a common vulnerability for web-based application that communicates with a database.


Attacker
Client
Database
open website
request login credentials
enter SQL query
insert input into SQL statement
query the database
find data and return to client
gather 'hacked' details and use
display account details

SQL injection occurs when a user inputs data into the program that is then used to dynamically build a SQL query. This can result in unauthorised access to the data in the database, with the potential for the attacker to steal confidential data, or even change authorization for existing users blocking all access to the database.

According to the OWASP foundation, there are a number of ways of preventing SQL Injection. These include:
Use of Prepared Statements (with Parameterized Queries)
Use of Stored Procedures
Allow-list Input Validation
Escaping All User Supplied Input

For the purpose of this post, I am going to look at the first one, using parameterized queries. In the diagram above you can see that the user enters some SQL, the program inserts the SQL into a query and queries the database. Without the use of parameterized queries, the Python code may look like this:

```
username_on_form = request.form.get("username")
password_on_form = request.form.get("password")
cursor = connection.cursor()
cursor.execute(f"SELECT id, username, password, credit_card FROM
    customer WHERE username = '{username_on_form}' and password =
    '{password_on_form}';")
results = cursor.fetchall()
```

SQL Injection

So if the attacker entered `' or ' 1=1 --` in the username input the query that would be executed on the database would be:

```
SELECT id, username, password, credit_card FROM customer WHERE username
      = OR 1=1 -- and password = '{password_on_form}';
```

The -- act to comment out the remaining code, so as 1 is always equal to 1, the attacker would get all of the usernames, passwords and credit cards returned from the database. Editing the code to use parameterized queries means that all of the input is actually treated like a string instead of executable SQL code. The updated code would be something like:

```
cursor = connection.cursor(prepared=True)
query = "SELECT id, username, password, credit_card FROM customer WHERE
    (username, password) VALUES (%s, %s)"
username_on_form = request.form.get("username")
password_on_form = request.form.get("password")

cursor.execute(query, username_on_form, password_on_form)
result = cursor.fetchall()
```

References:
OWASP (N.A.) SQL Injection Prevention Cheat Sheet. Available from: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html [Accessed on 12/03/2022]