Final Project

Sudhanshu Tarale
UFID:66177686

1. Exercise

Goal: Get the Key.

Tasks:
1. Brian has made a prototype for the artstailor website. Go to the website. Try to login. Hint: Visualize how a Select query is written in SQL expecially when it is used to get login credentials from the user and check it against the database. How can our input modify the query and get us access.

2. Once inside the website, try to find the key.

3. As an optional step, try to delete a product from the products table.

4. All of this can be done by putting in the appropriate input in the fields provided.

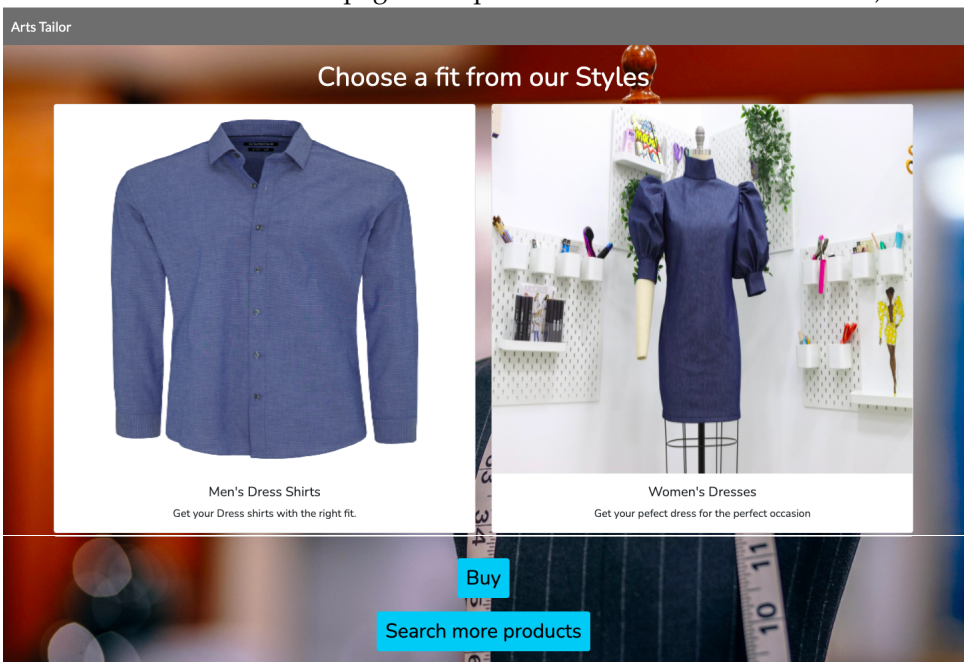
1.1 Requirements for the exercise

There are no such technical requirements for this exercise. Video needed for more understanding of the exercise is: `https://www.youtube.com/watch?v=ciNHn38EyRc&t=616s`
Lecture 33 from Module 0x0d also covers a small part of how SQL injection works.
OVF file in zip is at `https://bit.ly/3dTZArf`
To run the code:
Step 1: Open Xampp on VM and click on start all servers.
Step 2: Navigate to localhost/sql-injection/login.html
The source code for the web application is inside the htdocs folder of the XAMPP application folder. The database is already created.
The tables in the DB are:
customer
Columns are : customer_id,name,email,password
payment_info:
Columns are :payment_id,buyer_name,product_bought,credit_card_number,cvv,comments
products
Columns are :id,product_name,cost,customer_cost

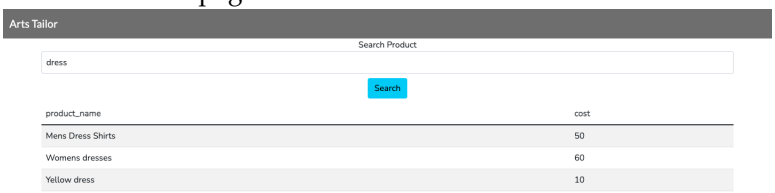
1.2 Exercise Solution

Since we do not know the login credentials, we can visualize the SQL statement to be something like Select .. from ... where email='..' and password='....'. So here we can put $'\ or\ 1 = 1\ \#$ in the username field and any password with this will work. The idea here is to enter any username and then put an or statement after that which for certain return true. So the solution $'admin\ or\ 2 = 2\ \#$ can also be used.

Now once we get access to the home page, we can see that there is a buy and search more products button. When we navigate to search more products page and type dress, we can see some results (we know that we a product with the name dress in it as the home page has 2 products with the name dress in it).



Search Products page

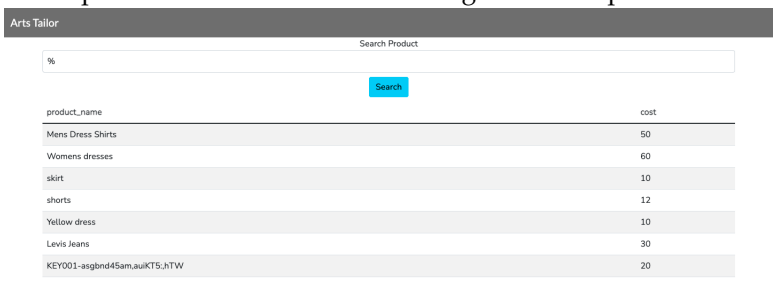


Here we can see that we are getting 3 products with name dress in it. So there isn't any product with the name exactly as "dress". This means that the select statement is not comparing the user input directly but using something where

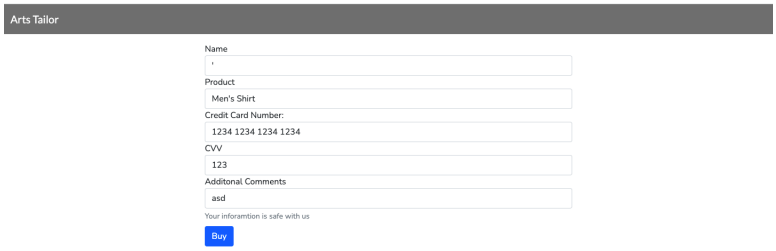
it checks if the product name has dress in it or not. This is the like operator in SQL. In our SQL query, we are using the like operator and comparing it by using the % before and after the user input (this is the general syntax of like operator which can be googled).

```
$sql = "SELECT product_name,cost from products where product_name like '%{$search}%'";
$result = $conn->query($sql);
```

So we put a % in the search text box to get ALL the products



We get the key. Now we go to the buy page. Here we know that the insert query is being used. But we do not know which datatype is used for each field. This is important as we need to know where to put quotes while doing SQL injection. One thing we can do is to deliberately enter a wrong name input to get an error which exposes the SQL Query.
We know that name has to be a string. So we put a ' as we use such quotes while writing a string in SQL. This gives an error.



We can see the query here

ERROR: Could not able to execute INSERT INTO payment_info (buyer_name,product_bought,credit_card_number,cvv,comments) VALUES ('','Mens Shirt','1234 1234 1234 1234','123','asd'). You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Mens Shirt','1234 1234 1234 1234','123','asd')' at line 2

From this we can understand that in the comments textbox we can put something like $somecomment'); delete\ from\ products\ where\ cost\ =\ 30;\ --$. The –

are there to comment out the SQL query after the semi-colon so as to remove any unwanted text in the query. We can go back to product search and verify that the product is deleted.

Thus we have accomplished the goal of the exercise.
Code: `https://github.com/SudhanshuTarale/pen-test-project`
The flow of the webpage is login.html → index.html → search.php or buy.html.

2. Mitigation for SQL Injection

1. Validate all the data being entered by the user. An example of it would be using the bootstrap email input type for email which does not allow single quotes and such after the @ symbol. Of course this is not ideal too as this is client side verification which can be intercepted with tools like burpsuite and modified. Hence, we should also validate on server side before entering the data into database.

2. In case of PHP, use prepared statements for executing the SQL query. More about this can be found at `https://bit.ly/3pRxjaB`

3. Use frameworks. Generally frameworks like NodeJS, Laravel etc have built in protection for SQL injection. Although, it should be checked which versions of the framework are vulnerable. Using latest software for creating web applications is a big plus.