CSE565 Lab 2

**Name**: Sri Charan Reddy Teegala

**Email**: teegala@buffalo.edu

**UBID**: teegala

**UB Number**: 50681752

**Academic Integrity Statement:**

I, **Sri Charan Reddy Teegala** have read and understood the course academic integrity policy.

(Your report will not be graded without filling your name in the above AI statement)

# Task 1: Get Familiar with SQL Statements

**Steps Performed:**

* Loaded the *sqllab\_users* database
* Used *show\_tables* command to print out all the tables of the database *sqllab\_users*.
* In the credential table, executed a command to get details of the employee “Alice”

**Observations:**

* After executing the command, all the details of Alice stored in credential table like Name, EID, Salary, birth, SSN, PhoneNumber, Address, Email, NickName, Password as shown in Fig (1).

**Code and Explanation:**

select \* from credential where Name="Alice";

This SQL statement retrieves all columns (\*) from every row in the credential table whose Name column exactly matches the string, Alice.

**Screenshot:**

A screenshot of a computer

AI-generated content may be incorrect.

Fig (1)

# Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage.

**Steps Performed:**

* Used docker commands to build and start the containers.
* Mapped local IP host to seed-server.com by adding it to /etc/hosts file.
* Opened the webpage at <https://www.seed-server.com>
* Entered the input of username as Admin’—and leaving password empty.
* Clicked login button to bypass the authentication and gained accessed the Admin’s account.

A screenshot of a computer

AI-generated content may be incorrect.

Fig (2.1.1)

**Observations:**

* Successfully bypassed the authentication by using comments in the input field to skip password verification in SQL query
* Logged into Admin account and able to access data of all the users in credential table in db.

A screenshot of a computer

AI-generated content may be incorrect.

Fig (2.1.2)

Task 2.2: SQL Injection Attack from command line.

**Steps Performed:**

* To perform the SQL Injection attack from the command line we can use curl command to directly call the endpoint.
* Without encoded payload:

curl ‘http://www.seed-server.com/unsafe\_home.php?username=Admin--&Password=’

* By encoding the payload

curl ‘http://www.seed-server.com/unsafe\_home.php?username=Admin%27--+&Password=’

* Executed both these commands from CLI.

**Observations:**

* It fetched data for the encoded payload i.e., we encoded ‘ as %27 and space is encoded as ‘+’
* For curl to work we need to provide params like how browsers handle them so encoding them is the way to go.

**Screenshots:**

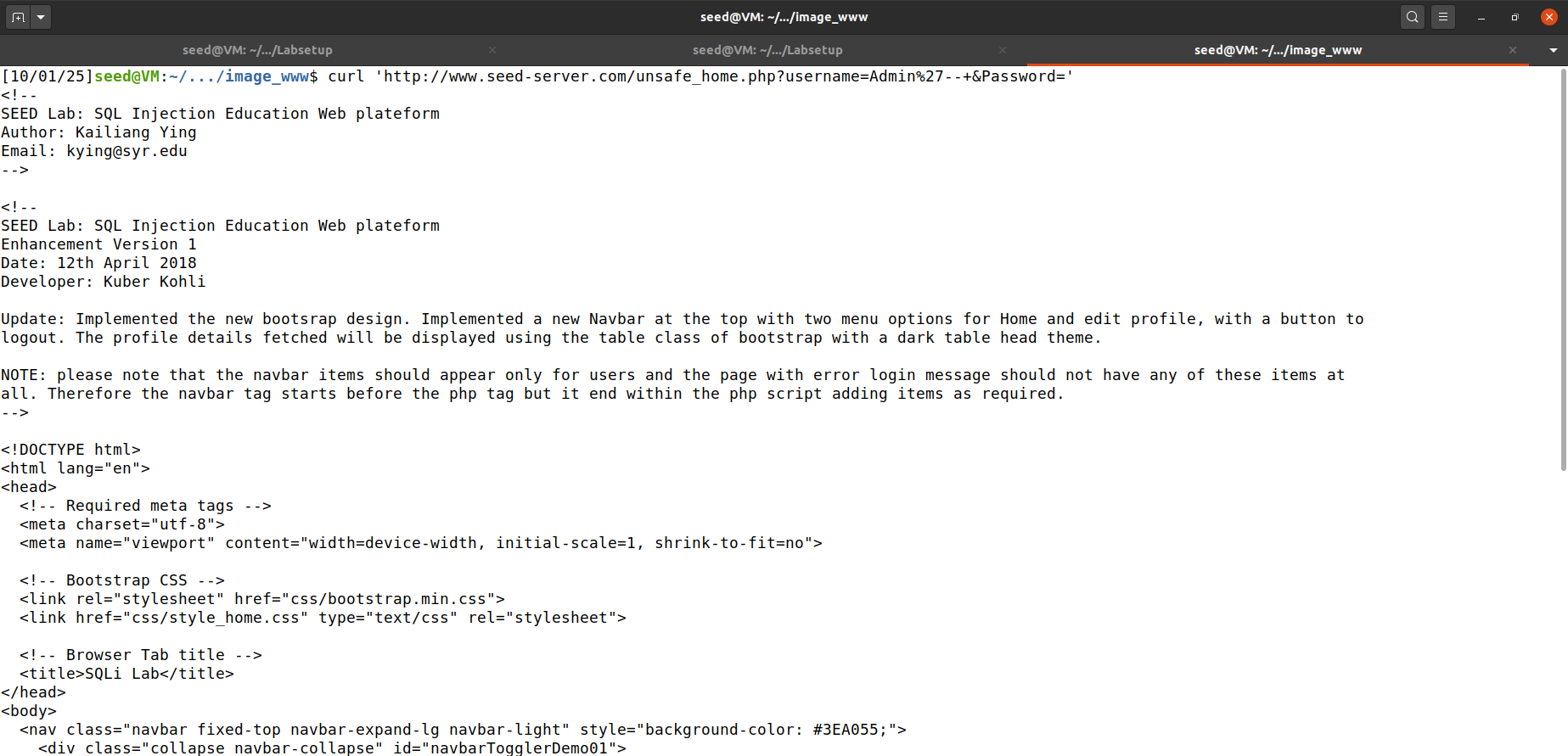


Fig (2.2.1)

A screenshot of a computer

AI-generated content may be incorrect.

Fig (2.2.2)

Task 2.3: Append a new SQL statement.

**Steps Performed:**

* Appended a new SQL UPDATE statement into the param sent for username

Admin’; UPDATE credential set Salary=20001 where Name=”Alice”; --

**Observations:**

* Attempt to execute multiple SQL commands at once failed because query() function can only run one query at a time
* Therefore, the server responds with an error.

**Screenshots:**

**A screenshot of a computer

AI-generated content may be incorrect.**

Fig (2.3)

# Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary.

**Steps Performed:**

* Open the website [www.seed-server.com](http://www.seed-server.com) and log in using user Alice Credentials
* Click on Edit profile route on the nav bar.
* Injected the following payload into an input field which adds the required update columns and comments the rest.

Ali’, salary=45000 where name=‘Alice’ --

* Clicked on the update button to execute the query.

**A screenshot of a computer

AI-generated content may be incorrect.**

Fig (3.1.1)

**Observations:**

* Successfully updated the salary field using the payload mentioned above.
* Commenting out the rest of the query also adding nickname as ‘Ali’ checking if the current field works.

A screenshot of a computer

AI-generated content may be incorrect.

Fig (3.1.2)

Task 3.2: Modify other people’ salary.

**Steps Performed:**

* Open the website [www.seed-server.com](http://www.seed-server.com) and log in using user Alice Credentials
* Click on Edit profile route on the nav bar.
* Injected the following payload into an input field which adds the required update columns and comments the rest.

’, salary=1 where name=‘Boby’ –

* Submitted the form and updated the salary of Boby in the database.

A screenshot of a computer

AI-generated content may be incorrect.

Fig (3.2.1)

**Observations:**

* After submitting the form, Bob’s salary has been updated.
* By using SELECT command on the database we can see that the salary was changed to $1.

**Explanation:**

* As the code executes the input fields directly on the db without checking for authentication, any user can access data by SQL injections.

A screenshot of a computer

AI-generated content may be incorrect.

Fig (3.2.2)

Task 3.3: Modify other people’s password.

**Steps Performed:**

* Open the website [www.seed-server.com](http://www.seed-server.com) and log in using user Alice Credentials
* Click on Edit profile route on the nav bar.
* Injected the following payload into an input field such that the password I changed to 1234 (hashed with sha1)

’, password=sha1(1234) where name=‘Boby’ --

* Submitted the form and updated the password of Boby in the database.

A screenshot of a computer

AI-generated content may be incorrect.

Fig (3.3.1)

**Observations:**

* Boby’s password in the database has been updated and verified by checking the new hash value in the db.
* New password has been verified by authenticating using the new password and fetching boby’s information.

A screenshot of a computer

AI-generated content may be incorrect.

Fig (3.3.2)

A screenshot of a login screen

AI-generated content may be incorrect.

Fig (3.3.3)

A screenshot of a computer

AI-generated content may be incorrect.

Fig (3.3.4)

# Task 4: Countermeasure — Prepared Statement

**Steps performed:**

* In this task, the goal is to modify the unsafe.php file such that it prevents SQL Injection attacks by using prepared statements.
* We will replace the vulnerable SQL query using $conn->query() with a prepared statement which is $conn->prepare().

**Code and Explanation:**

$stmt = $conn->prepare(“SELECT id, name, eid, salary, ssn

FROM credential

WHERE name= ? and Password= ?”);

$stmt->bind\_param("ss", $input\_uname, $hashed\_pwd);

$stmt->execute();

$stmt->bind\_result($bind\_id, $bind\_name, $bind\_eid, $bind\_salary, $bind\_ssn) ;

$stmt->fetch();

sid = $bind id;

$name = $bind\_name;

$eid = $bind\_eid;

$salary = $bind\_salary;

$ssn = $bind\_ssn;

* We use ‘?’ as placeholders in the SQL query inside the prepare statement and when we receive user input we bind them using bind\_param() method.
* Now the results will be fetched securely using bind\_result() and fetch methods.
* We initialize each variable now with the output values fetched.

**A screenshot of a computer

AI-generated content may be incorrect.**

Fig (4.1)

**Observations:**

* Checked if the users are able to authenticate which proves that the new code works.

A screenshot of a computer

AI-generated content may be incorrect.

Fig (4.2)

A screenshot of a computer

AI-generated content may be incorrect.

Fig (4.3)

* Even with crafted payloads like admin’--, the code will treat them as data not sql queries i.e., no data is been fetched.

A screenshot of a computer

AI-generated content may be incorrect.

Fig (4.4)

A screenshot of a computer

AI-generated content may be incorrect.

Fig (4.5)

**References:**

**[Week-6\_Class-1\_Database\_Security.pptx](https://piazza.com/class_profile/get_resource/mertuuq13wzyr/mg61zpjiubz2li" \t "_blank)**

[**https://www.php.net/manual/en/**](https://www.php.net/manual/en/)

[**https://www.handsonsecurity.net**](https://www.handsonsecurity.net)**.**