Task 4 SQL Injection

Task 1:

1. ran command `mysql -u root -pseedubuntu`, in the `student@web-server`
    - this logged in to the mysql database
2. ran command `use Users;`
    - switch to the Users Database
3. ran command `show tables;`
    - saw that there's 1 table in the Users database which is credential

Task 2:

1. input the SQL code `' OR Name = 'Admin' #`
    - What the `'` do is effectively ending the previous clause (since that check for the EID). I terminated and add my own SQL clause which we're trying to find the record with the name Admin. So I added a `OR` clause (which bypass the EID check) and find a match of `Admin`, then, I added the `#` which is the comment for SQL which commented out the `password` check and I've logged into the Admin account
2. run the command `curl 'http://seedlabsqlinjection.com/unsafe_credential.php?EID=%27+OR+Name%3D%27Admin%27+%23Password='`
    - we essentially replaced the symbols with they encoding. like the document suggested the `'` is mapped to `%27` and `=` is mapped to the `%3D` and `#` is mapped to `%23`
    - But it is practically the same code as Task 2.1
3. input the SQL Code `' OR Name = 'Admin'; UPDATE credential SET PhoneNumber='1234567890' WHERE Name = 'Admin'; #`
    - obtained an output of

      There was an error running the query [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 'UPDATE credential SET PhoneNumber='1234567890' WHERE Name = 'Admin'; #' and Pass' at line 3]\n

- So 2 things here, the command, we used the same approach to “log into” to the Admin account, then we terminated the statement with a `;` and begin our second statement which is an attempt to update a field attribute PhoneNumber for the `Admin` field with the Where clause trying to match the `Name = 'Admin'`, then we end the 2nd statement `;` and follow with the `#` comment trick again
- We noticed that we cannot execute the 2nd query due to a syntax error despite it being a valid SQL syntax. It is due to the SQL code `query()` in PHP can only execute 1 Query.

Task 3:

1. First login to an employee account, which I logged into Alice with the `' OR Name = 'Alice' #`, then in the edit profile page of Alice, instead of putting actual stuff, we put `',salary='999999999` in the Nickname field (can be other field as well since they’re the same syntax). And now Alice’s salary is 999999999 which is so nice to have.
    - The idea behind that is, we terminate the NickName field early with the `'`, then a `,` which allow us to start a new field in which we use the `salary` field and and set it to `'999999999`, the `'` at the end is missing is because there’s already an ending quote in the PHP code so we don’t need another closing one.
2. Continue in Alice account, we will use the profile page again and the NickName field again, we will input the command `',password='a94a8fe5ccb19ba61c4c0873d391e987982fbbd3' WHERE Name = 'Ryan'; #`
    - I first try to dig through what kind of hash is the original hash format which I believe is SHA1
    - Then I compute a new password `test` and its SHA1 Value and pasted into the password field

- the `',` works the same idea as previous Task 3.1, then we update the password field to be the SHA1 value of test and we follow it up with a WHERE clause which we selected Ryan as our target and we terminated the command with `;` and `#` to comment out rest of the SQL query
- This modified Ryan password to `test` and upon logging with the Ryan EID (looked up in the MySQL database) and inputting the password `test`, I was logged into Ryan Account

Task 4:

1. To Fix Login SQL injection

```
$stmt = $conn->prepare("
   SELECT id, name, eid, salary, birth, ssn, phonenumber, ad
   FROM credential
   WHERE eid = ? AND password = ?");
$stmt->bind_param("is", $input_eid, $input_pwd);
$stmt->execute();
$result = $stmt->get_result();

/* convert the select return result into array type */
$return_arr = array();
while($row = $result->fetch_assoc()){
    array_push($return_arr,$row);
}
$stmt->close();
```

- Basically followed the lab but the $result is obtained by using the `get_result()` so the website still function normally with the old code
- It is following the prepared statement mechanism as stated in the lab

2. To Fix the Edit Profile SQL Injection

```
if ($input_pwd != '') {
      $input_pwd = sha1($input_pwd);
```

```
        $stmt = $conn->prepare("
        UPDATE credential
        SET nickname = ?, email = ?, address = ?, Password = ?,
        WHERE ID = ?");
        $stmt->bind_param("sssssi", $input_nickname, $input_ema
    } else {
        $stmt = $conn->prepare("
        UPDATE credential
        SET nickname = ?, email = ?, address = ?, PhoneNumber =
        WHERE ID = ?");
        $stmt->bind_param("ssssi", $input_nickname, $input_emai
    }
    $stmt->execute();
    $stmt->close();
    $conn->close();
```

- Similar concept, we prepared the statement ahead of time and therefore SQL injection no longer works here. If I put `', salary = '9999999` in the nickname field. That will actually be the nickname (I tried it on Ryan and it is a cool nickname I got to say)