

## Lab 04 – SQL Injection Attack Lab

<Yibiao Liao sgw091>

IS 4463-CY1 – Summer 2019

<8/08/2021>

### INTRODUCTION

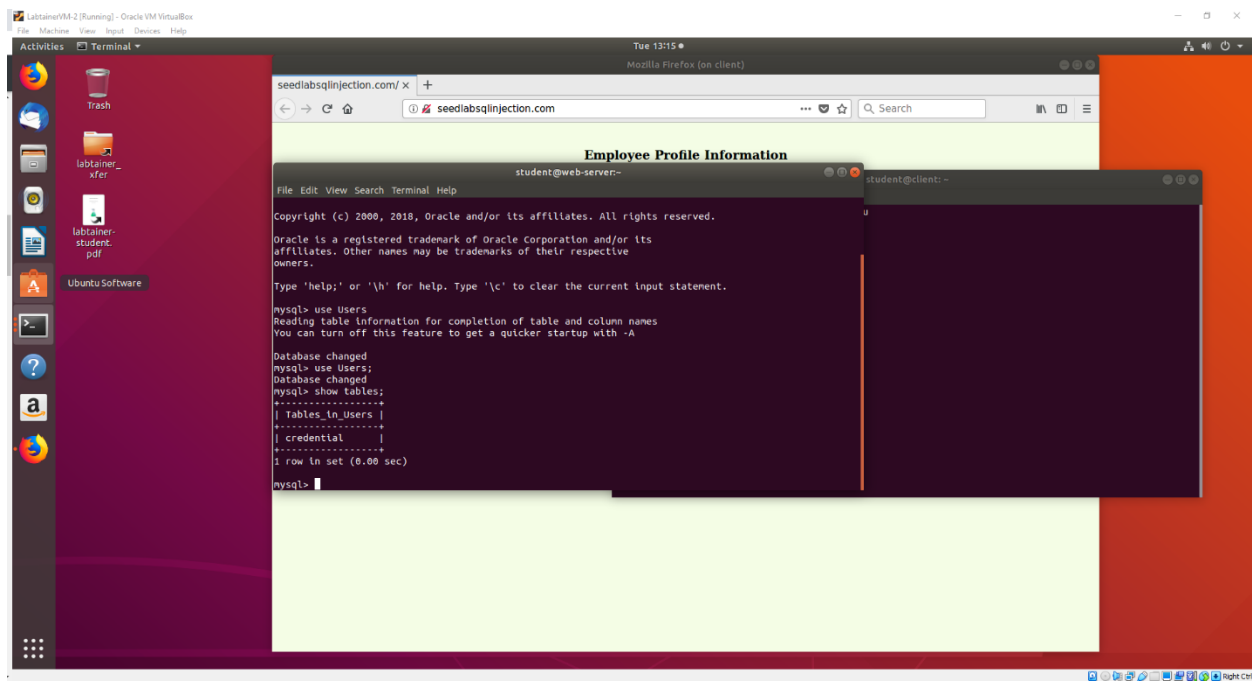
SQL injection attack is an attack by inserting a malicious SQL query or add statement into the input parameters of the application and then parsing and executing it on the backend SQL server, it is currently one of the most common means of hacker attacks on databases.

The main threats posed by SQL injection are as follows

- Guessing the backend database, which is the most exploited way to steal sensitive information from websites.
- Bypassing authentication, such as bypassing authentication to log into the backend of a website.
- Injection can be done with the help of stored procedures in the database for operations such as lifting rights

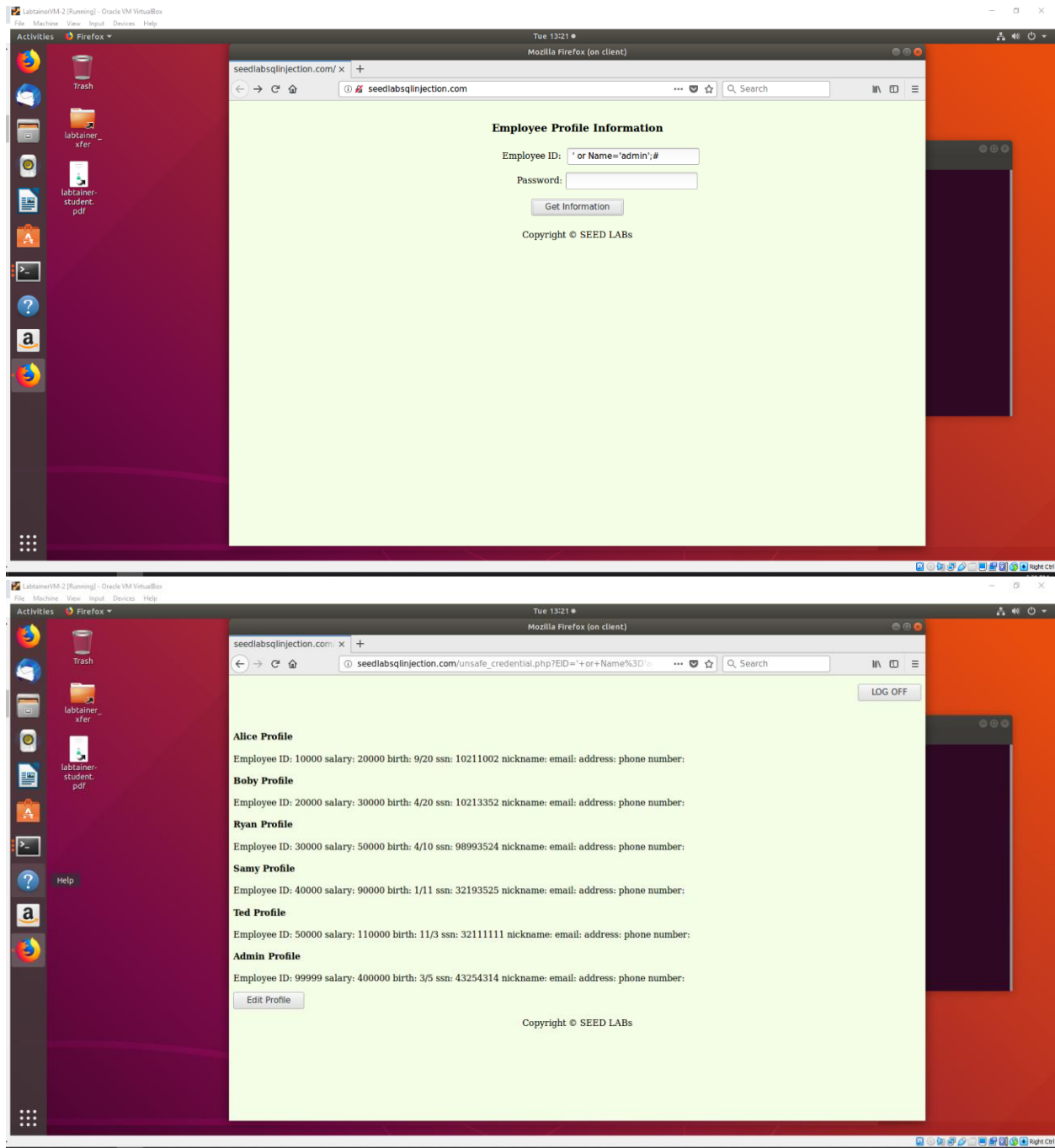
### PROCEDURE

#### Task 1: MySQL Console



## Task 2: SQL Injection Attack on SELECT Statement

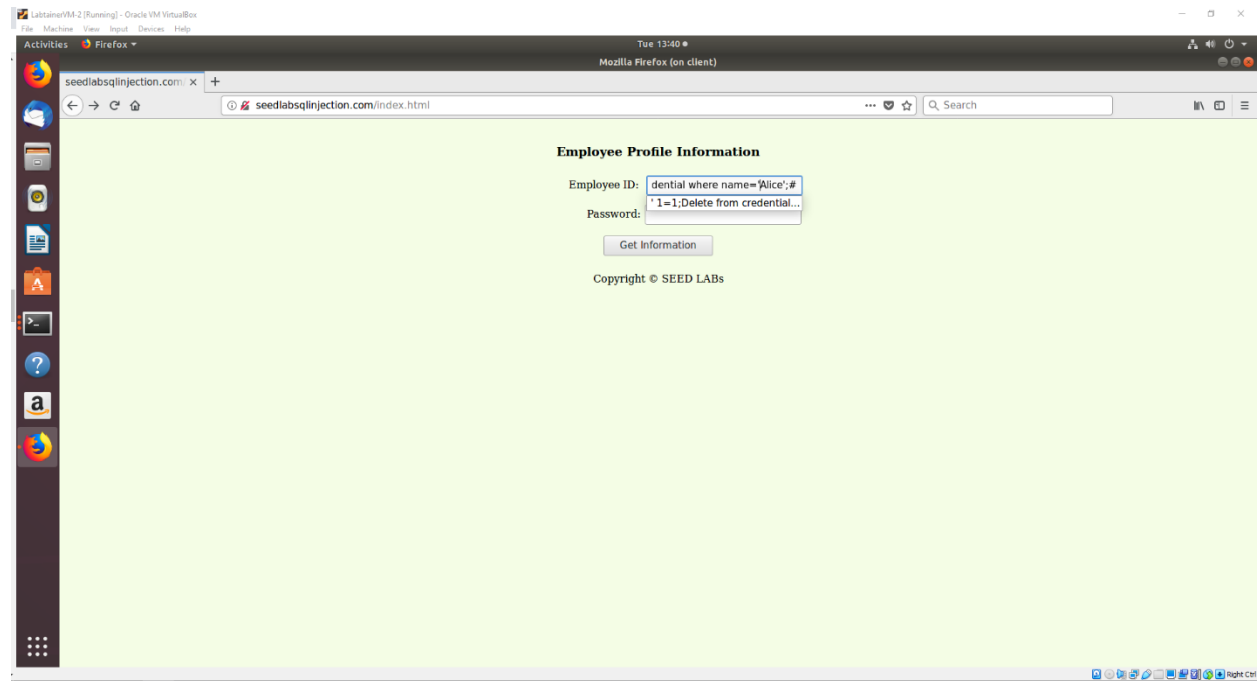
### 2.1: SQL Injection Attack from webpage

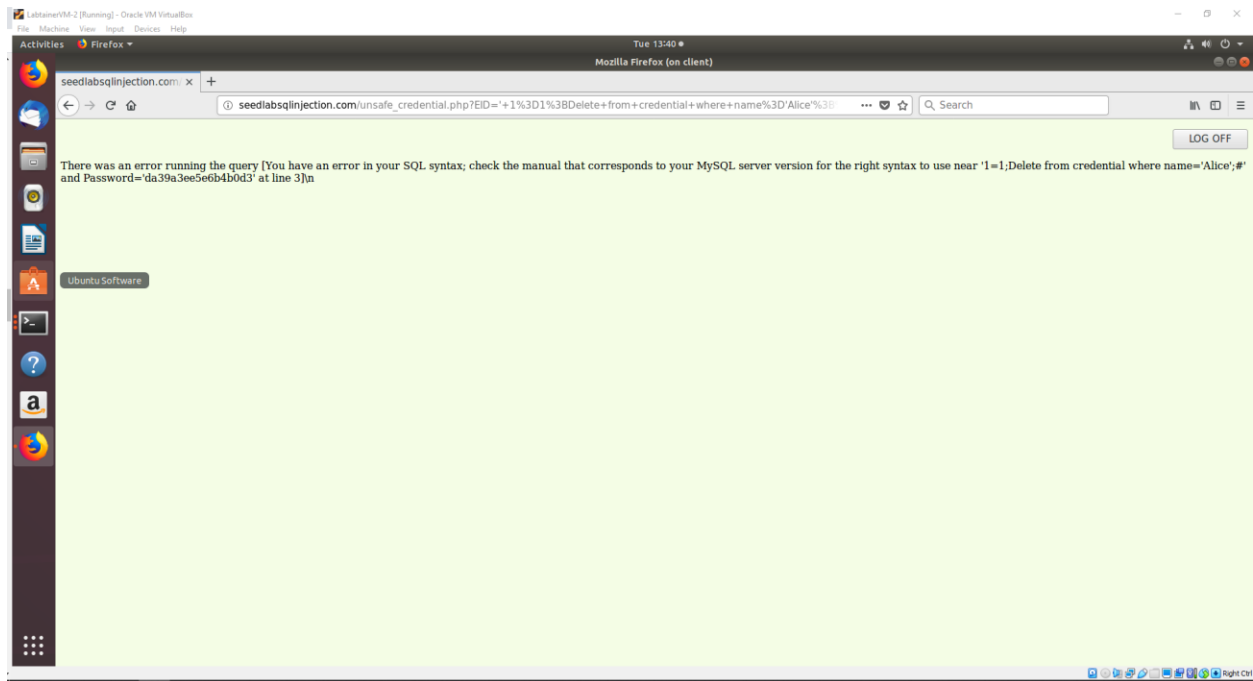


Use the code ' or Name='admin';# to exploit the attack. The single quotation closes the input id parameter, and the OR statement that follows permits us to login as admin. The # is added at the end to comment out everything that comes after it, allowing the password input to be bypassed.

The image shows a Kali Linux desktop environment. In the foreground, a terminal window titled 'student@client: ~' displays the output of a curl command. The command is: `curl -s 'http://www.secdatabase.com/unsafe_credentials.php?EID=N274or+Nanex3DXZ7adnIn%2738N238Password'`. The output shows a JSON response with a status of 200 and a body containing employee profile data for Alice, Bobby, Ryan, Ted, and Admin. To the right, a web browser window shows the 'Request URL: http://www.secdatabase.com/style\_home.css' and the 'Response Headers' section, which includes 'Content-Type: text/css', 'Date: Tue, 10 Aug 2021 20:12:13 GMT', 'ETag: '72c-55d1bfba5cac0'', 'Keep-Alive: timeout=5, max=99', 'Last-Modified: Fri, 03 Nov 2017 22:54:11 GMT', and 'Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16'. The terminal window also shows the 'body' of the response, which is a list of employee profiles in JSON format.

### 2.3: Append a new SQL statement.

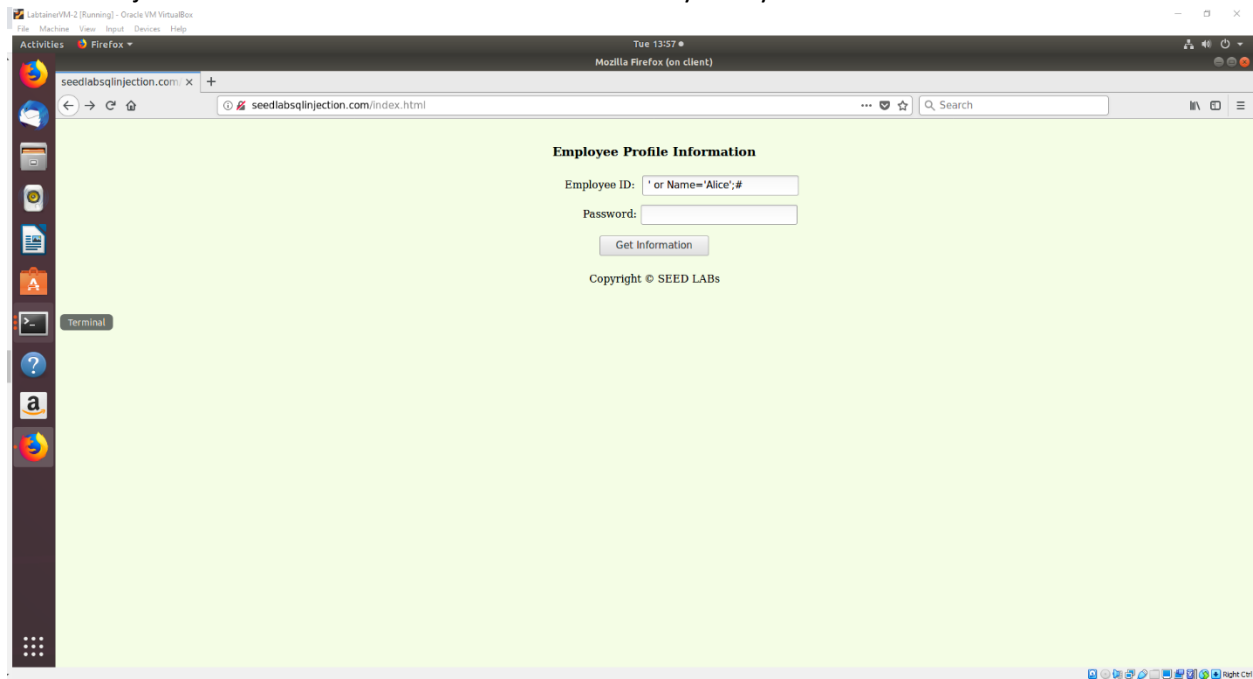


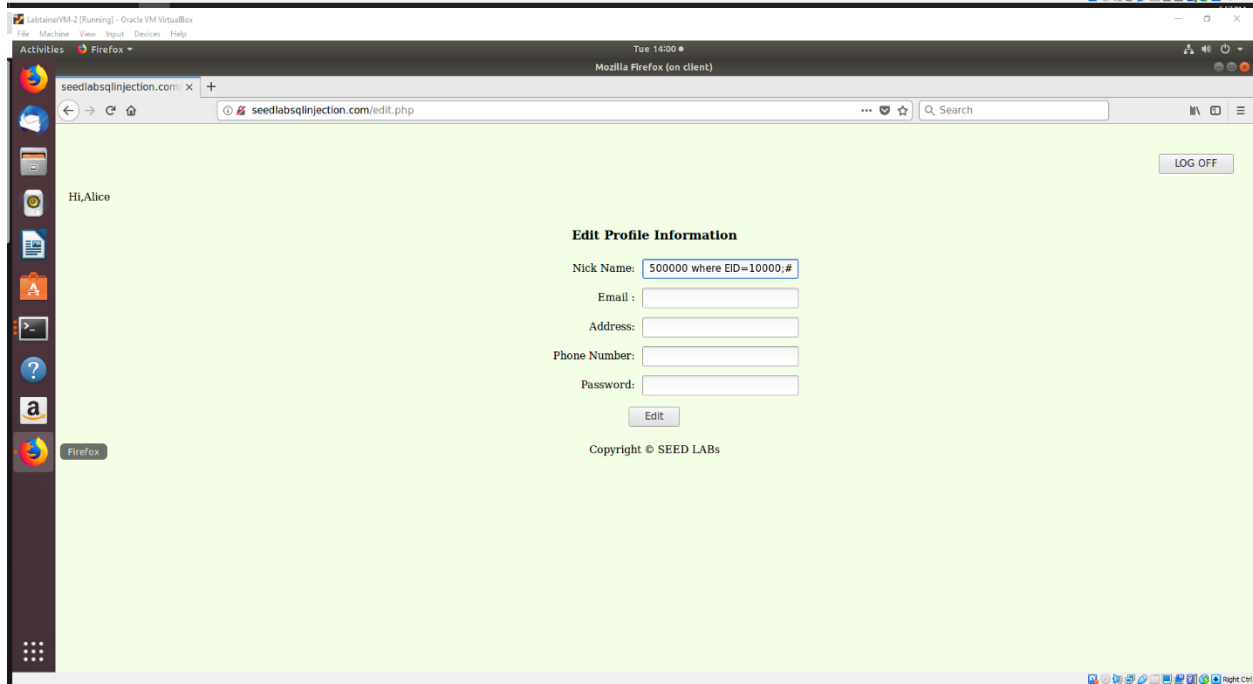
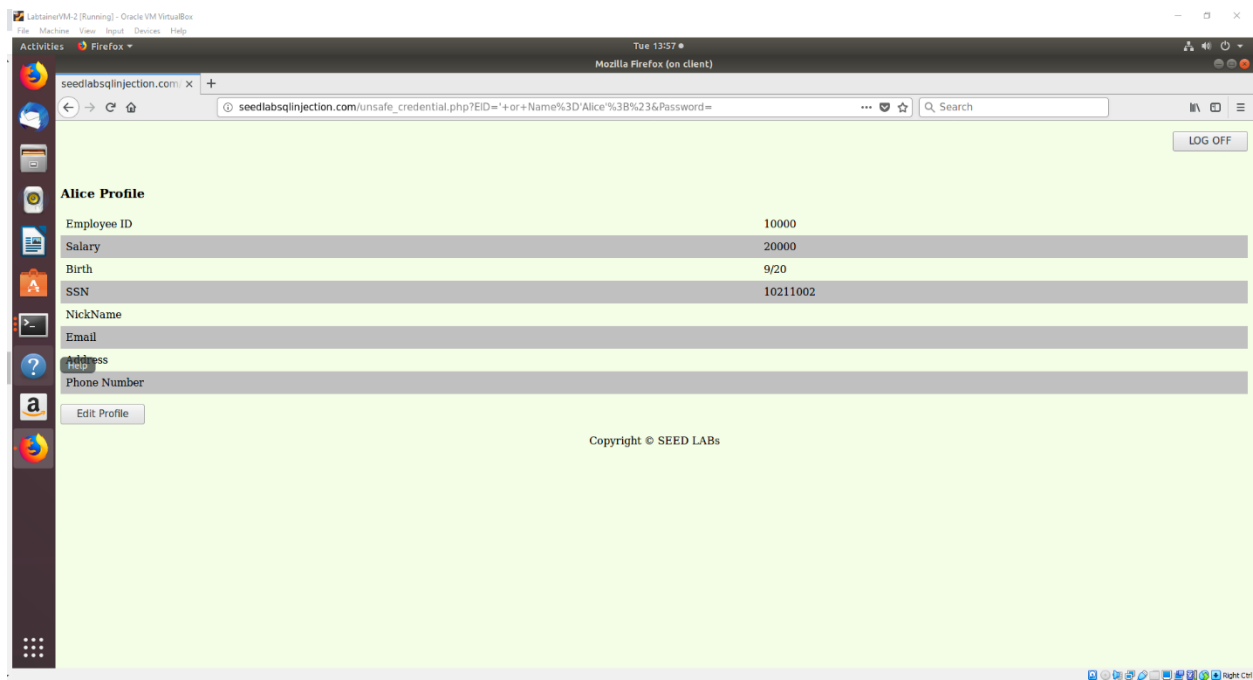


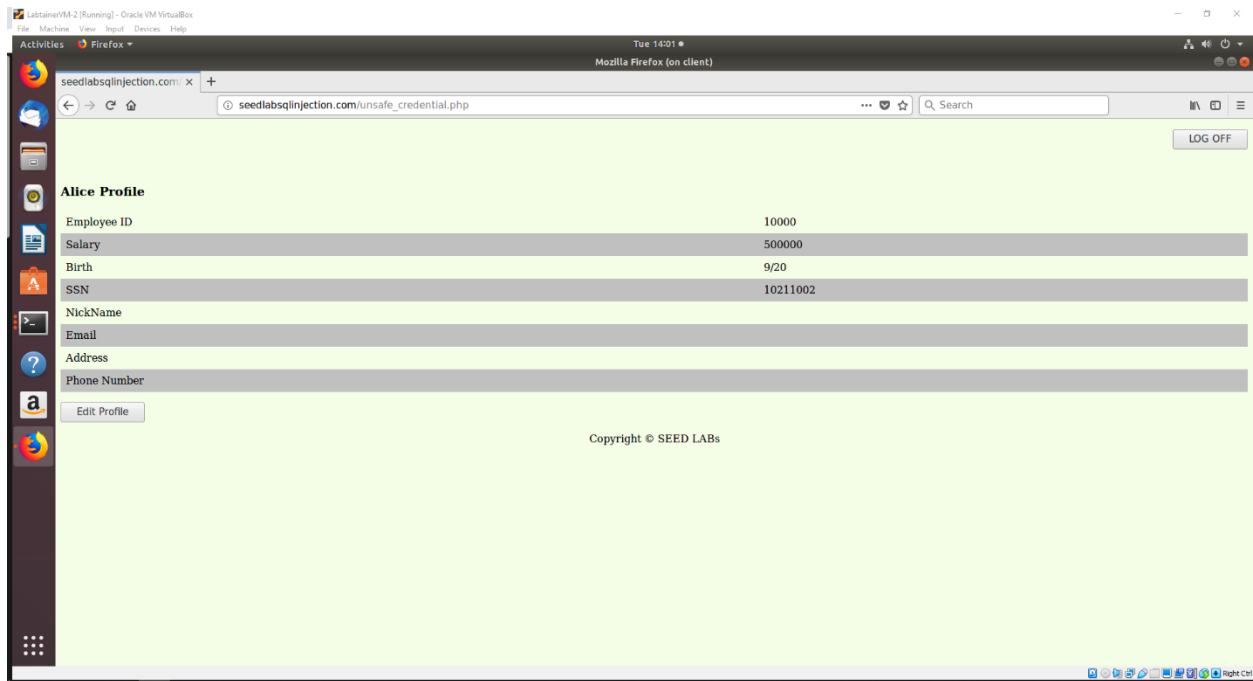
The attack fails due to a MySQL countermeasure that prevents numerous statements from being executed when called from PHP.

### Task 3: SQL Injection Attack on UPDATE Statement

#### 3.1: SQL Injection Attack on UPDATE Statement — modify salary







We're attempting to attack a SQL injection vulnerability by injecting code into the edit profile page to change the current employee's pay. We use a "#" at the end to comment out any subsequent values, ensuring that we don't run into any issues with null or erroneous input values from other fields. And we carry out this attack and change the salary field, which is hidden since the employee is not permitted to alter it. It can only be edited by the administrator. The pay of Alice has been adjusted because of the successful attack.

The screenshot displays a virtual machine environment with the following components:

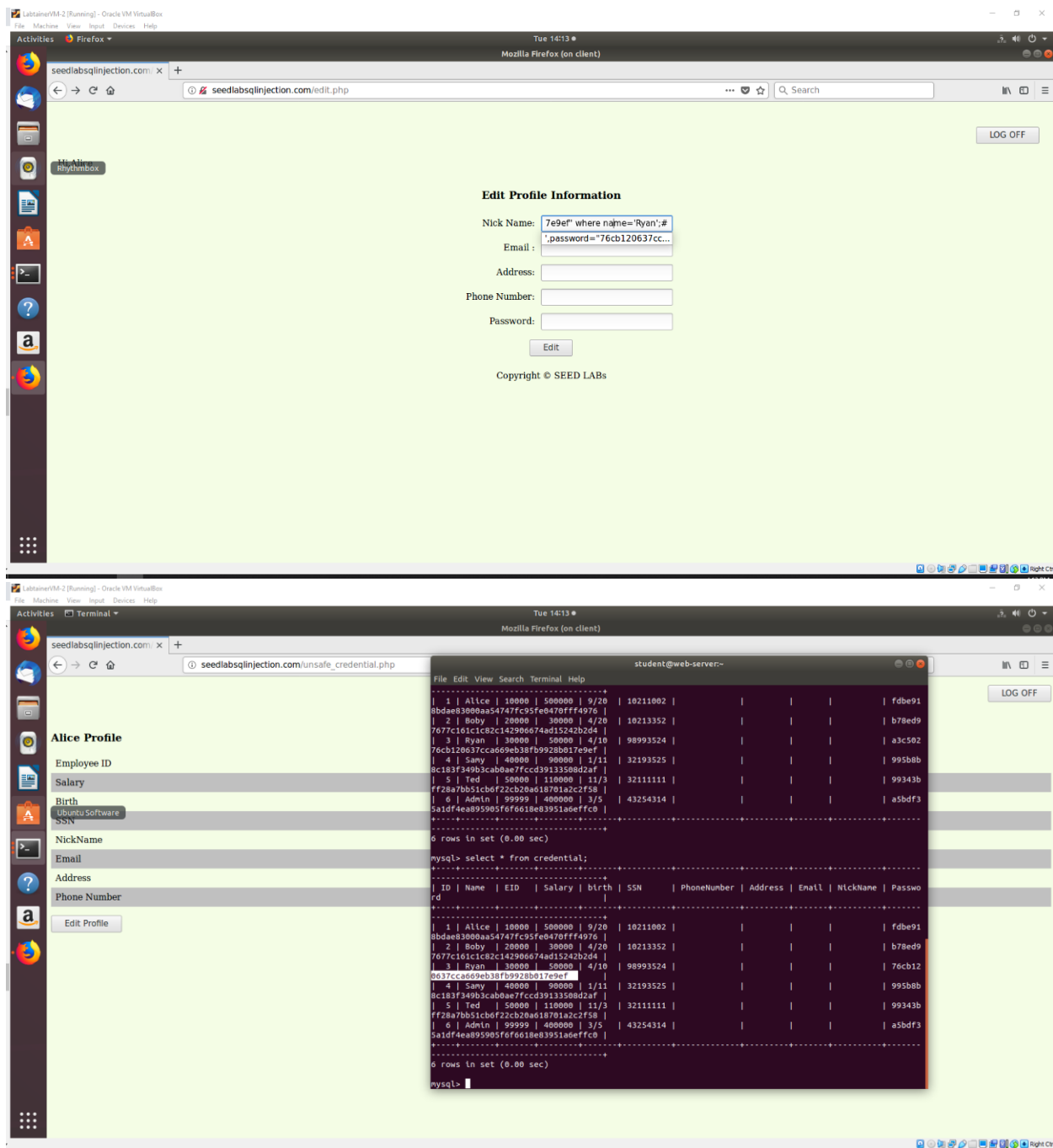
- Top Bar:** Shows the VM name "Lecturer\VM-2 [Running]" and the host name "Oracle VM VirtualBox".
- Activities Panel:** Includes icons for "Terminal", "Firefox", and "Amazon".
- Terminal Window:**
  - Tab: "seedlabsqlinjection.com: x"
  - Address Bar: "seedlabsqlinjection.com/unsafe\_credential.php"
  - Content: A table titled "Alice Profile" with columns: Employee ID, Salary, Birth, SSN, NickName, Email, Address, and Phone Number. The table contains 5 rows of data.
- Terminal Window (Background):**
  - Tab: "student@web-server-"
  - Content: A terminal session showing a MySQL query: `mysql> select * from credential;` and the resulting output table.

**Alice Profile Table Data:**

Employee ID	Salary	Birth	SSN	NickName	Email	Address	Phone Number
10000	500000	9/20	10211002				
10001	500000	9/20	10211002				
10002	500000	9/20	10211002				
10003	500000	9/20	10211002				
10004	500000	9/20	10211002				

**Terminal Window (Background) Table Data:**

ID	Name	EID	Salary	Birth	SSN	PhoneNumber	Address	Email	NickName	Password
1	Alice	10000	500000	9/20	10211002					fdbe91
2	Bob	10001	30000	4/20	10213352					b78ed9
3	Byan	10000	50000	4/10	98993524					a3c502
4	Sany	10000	50000	4/11	32193525					995b0b
5	Ted	10000	110000	11/3	32111111					99343b
6	admin	99999	400000	3/5	43254314					a5bdf3



The update command is used to change the password of a different account (Ryan) from another account (Alice). The SQL Injection vulnerability is exposed because of this. This demonstrates how hazardous it may be. We attempt to edit Alice's profile by logging into her account. The password of Ryan is changed when we enter the attack vector into the nickname box, and if the attack is successful. The update statement is used on the edit profile page to update the fields of an account; however, we utilize the injected code to change the information of another account. The # sign at the conclusion of the attack vector is used to comment out all subsequent code in the original code, ensuring that the attack is not hampered.



## CONCLUSION

Please provide a conclusion based on these prompts:

- What are the goals of the techniques and tools used in this lab? How are they relevant to real-world security?  
MySQL and HTML are both involved in this lab. As the most used database language, knowing how to control the safety of the database is crucial.
- How effective are these techniques and tools, in your opinion?  
Very effective, but some extra research were needed
- What is your opinion of this lab section in terms of difficulty and relevance? What other aspects of this topic would you like to explore?  
Its challenging.

## REFERENCES

### Internet Resources

[https://www.youtube.com/watch?v=\\_P8HCLkDInA&feature=youtu.be](https://www.youtube.com/watch?v=_P8HCLkDInA&feature=youtu.be)

