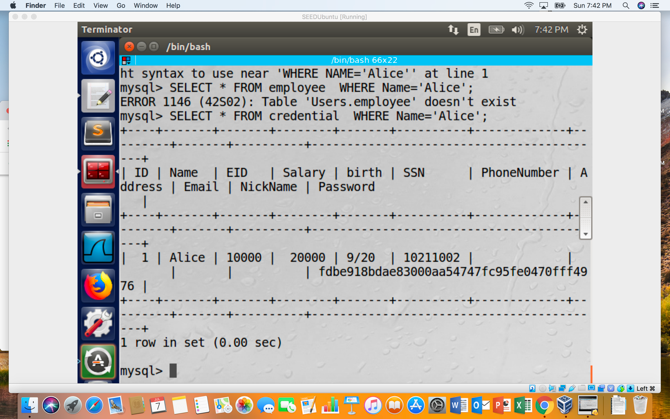
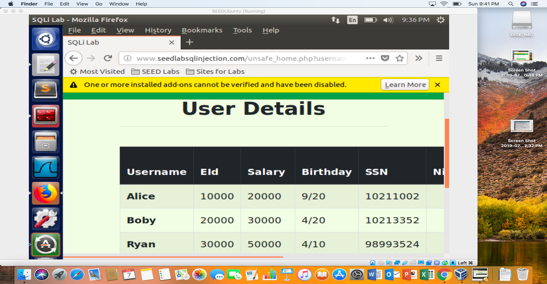
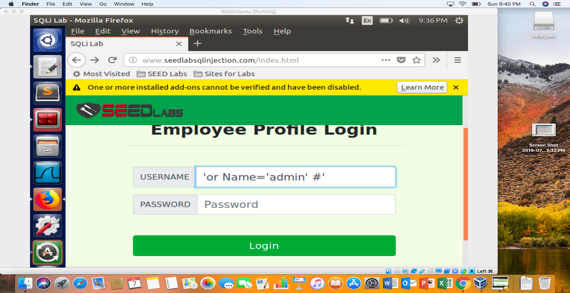
3.1 Task 1: Get Familiar with SQL Statements

After running the commands above, you need to use a SQL command to print all the profile information of the employee Alice. Please provide the screenshot of your results.



3.2 Task 2: SQL Injection Attack on SELECT Statement

• Task 2.1: SQL Injection Attack from webpage. Your task is to log into the web application as the administrator from the login page, so you can see the information of all the employees. We assume that you do know the administrator’s account name which is admin, but you do not the password. You need to decide what to type in the Username and Password fields to succeed in the attack.

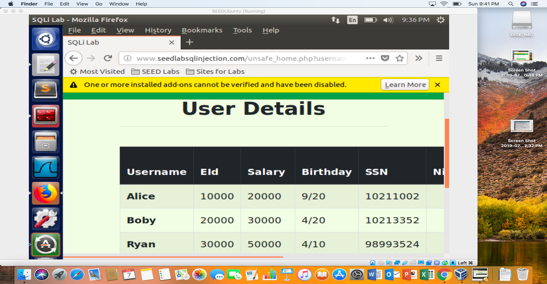
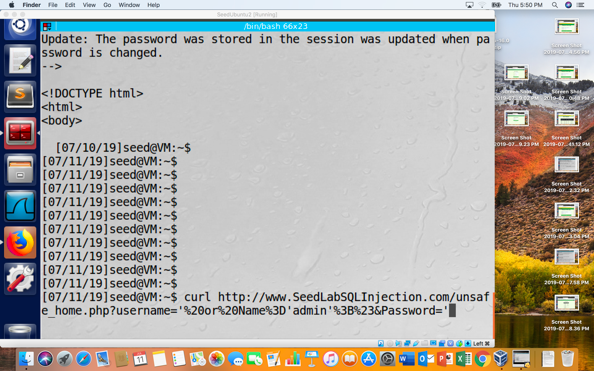


• Task 2.2: SQL Injection Attack from command line. Your task is to repeat Task 2.1, but you need to do it without using the webpage. You can use command line tools, such as curl, which can send HTTP requests. One thing that is worth mentioning is that if you want to include multiple parameters in HTTP requests, you need to put the URL and the parameters between a pair of single quotes; otherwise, the special characters used to separate parameters (such as &) will be interpreted by the shell program, changing the meaning of the command. The following example shows how to send an HTTP GET request to our web application, with two parameters (username and Password) attached:

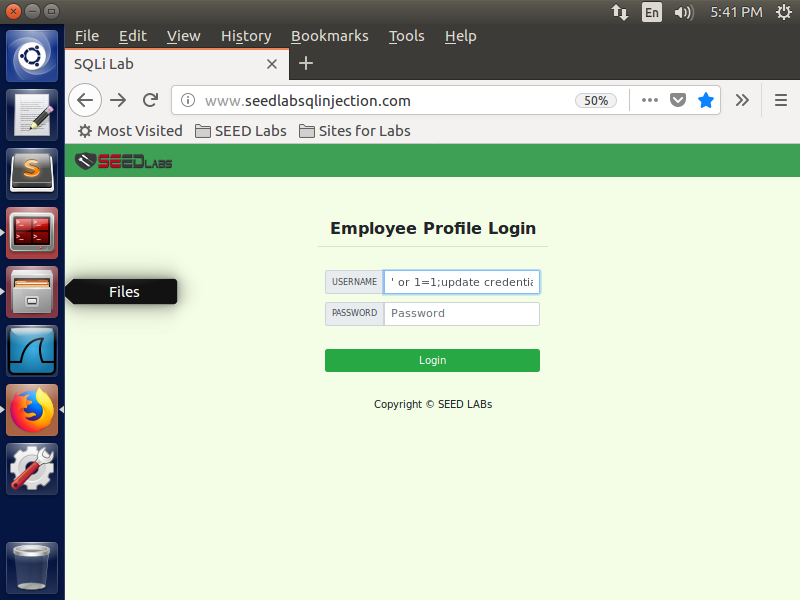
$ curl ’www.SeedLabSQLInjection.com/index.php?username=alice&Password=111’

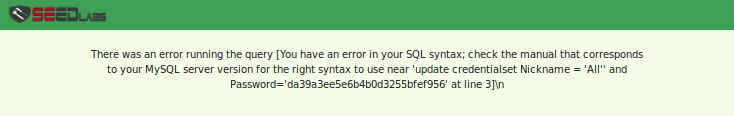
If you need to include special characters in the username or Password fields, you need to encode them properly, or they can change the meaning of your requests. If you want to include single quote in those fields, you should use %27 instead; if you want to include white space, you should use %20. In this task, you do need to handle HTTP encoding while sending requests using curl.

To help you started with this task, we explain how authentication is implemented in the web application. The PHP code unsafe home.php, located in the /var/www/SQLInjection directory, is used to conduct user authentication. The following code snippet show how users are authenticated.



• Task 2.3: Append a new SQL statement. In the above two attacks, we can only steal information from the database; it will be better if we can modify the database using the same vulnerability in the login page. An idea is to use the SQL injection attack to turn one SQL statement into two, with the second one being the update or delete statement. In SQL, semicolon (;) is used to separate two SQL statements. Please describe how you can use the login page to get the server run two SQL statements. Try the attack to delete a record from the database, and describe your observation.





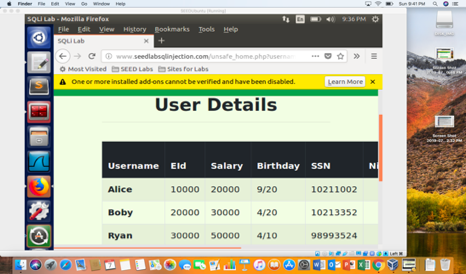
**Observation:** We append an update statement after the semicolon as show in the above the screenshot, however it is unsuccessful.

**Explanation:** The attack fails because there is a countermeasure in MySQL that prevents multiple statements from being executed when it is invoked from php.

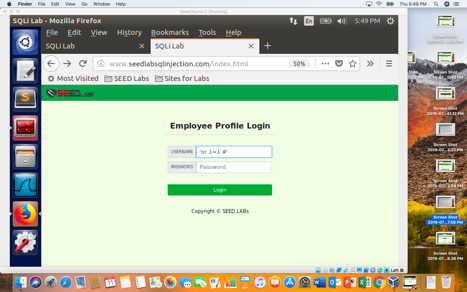
3.3 Task 3: SQL Injection Attack on UPDATE Statement

• Task 3.1: Modify your own salary. As shown in the Edit Profile page, employees can only update their nicknames, emails, addresses, phone numbers, and passwords; they are not authorized to change their salaries. Assume that you (Alice) are a disgruntled employee, and your boss Boby did not increase your salary this year. You want to increase your own salary by exploiting the SQL injection vulnerability in the Edit-Profile page. Please demonstrate how you can achieve that. We assume that you do know that salaries are stored in a column called salary.

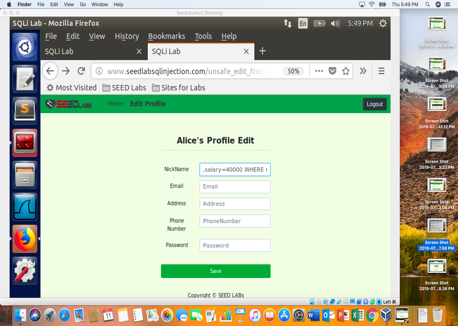
Alice salary is 20000 before



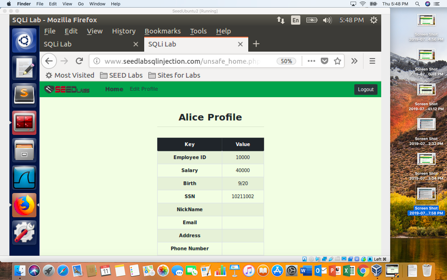
Log in using ‘or 1=1 #’



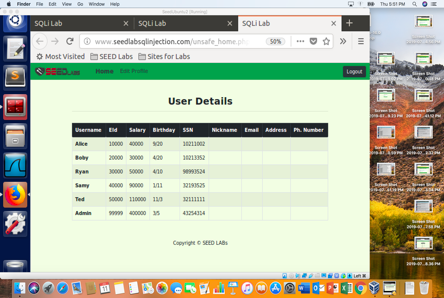
Type : ‘, salary=40000 WHERE name=’Alice’; --



Results



Alice salary after:

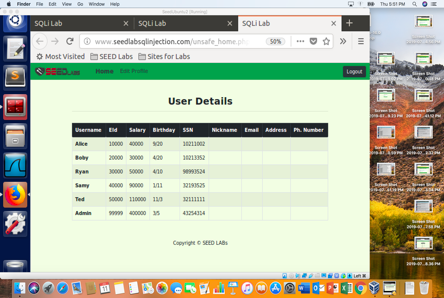


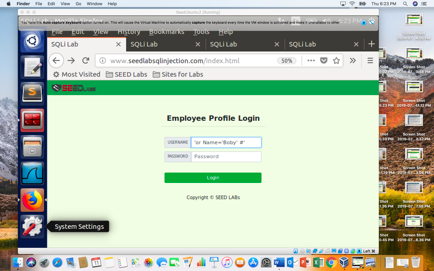
• **Task 3.2:** Modify other people’ salary. After increasing your own salary, you decide to punish your boss Boby. You want to reduce his salary to 1 dollar. Please demonstrate how you can achieve that.

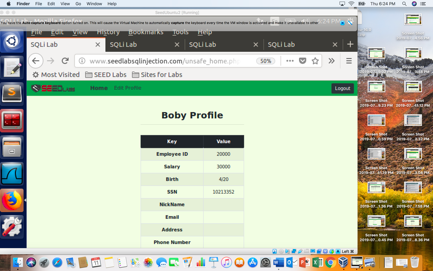
Type:

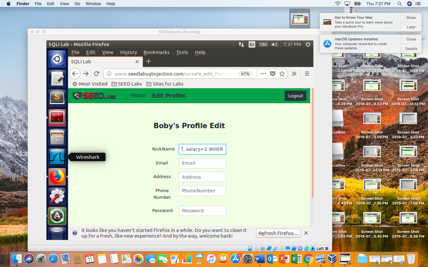
1.) ‘ or Name=’Boby’ #

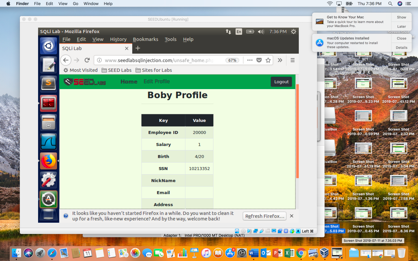
2.) ’ , salary=1 WHERE name=’Boby’ ; --







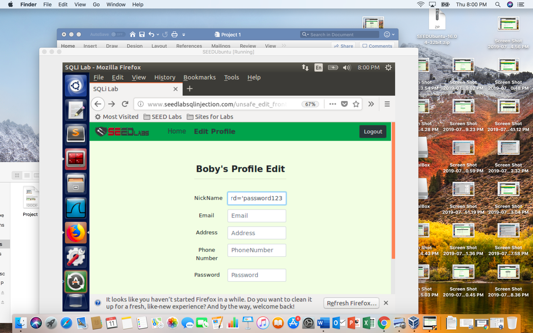


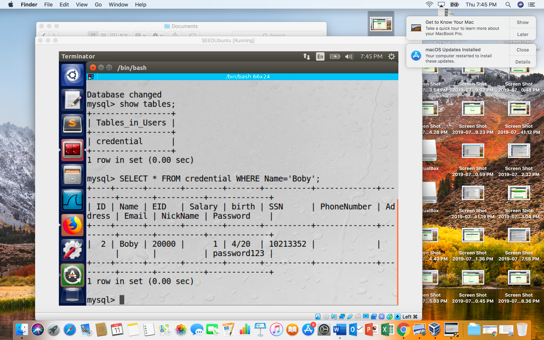


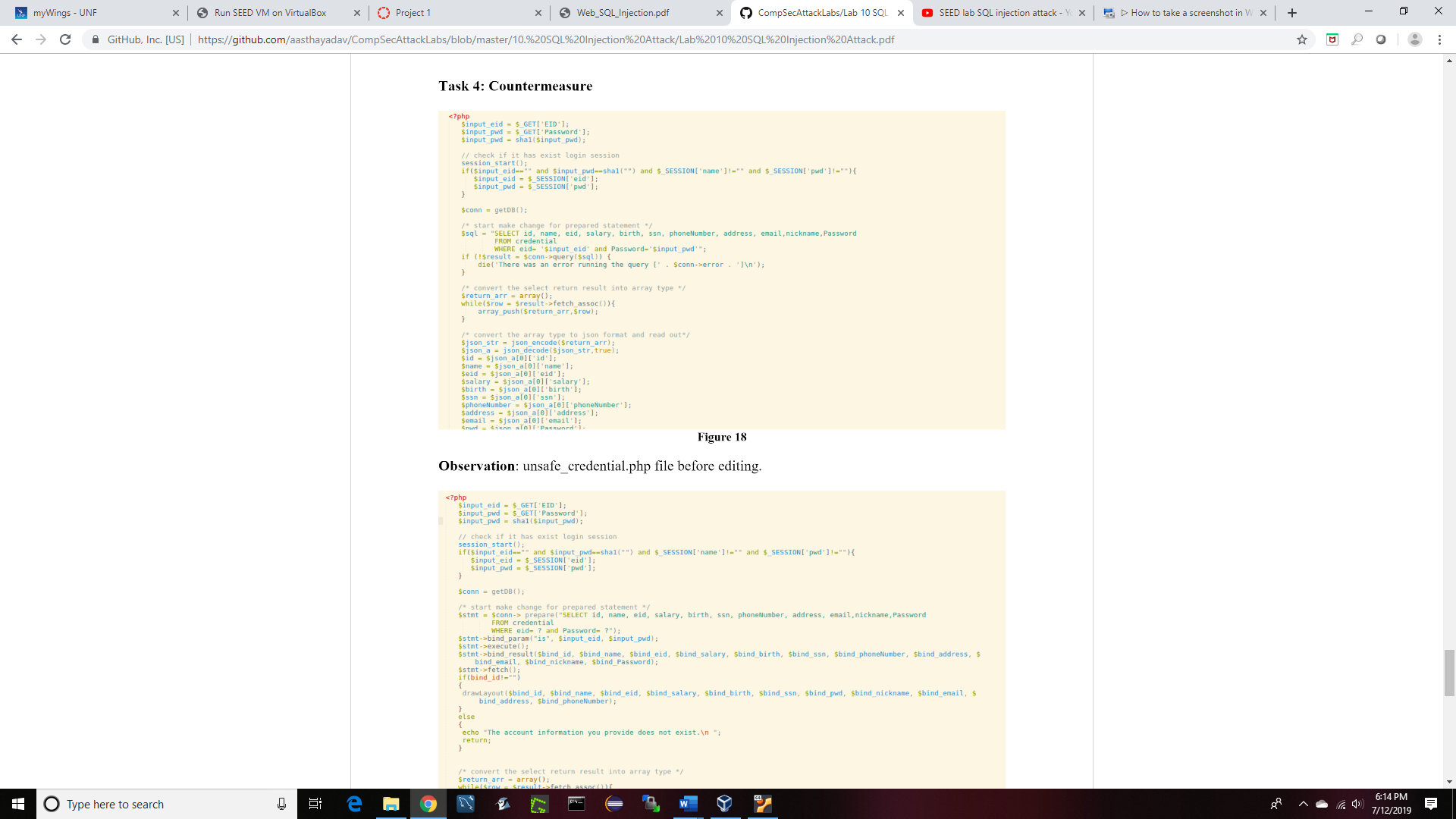
• **Task 3.3:** Modify other people’ password. After changing Boby’s salary, you are still disgruntled, so you want to change Boby’s password to something that you know, and then you can log into his account and do further damage. Please demonstrate how you can achieve that. You need to demonstrate that you can successfully log into Boby’s account using the new password.

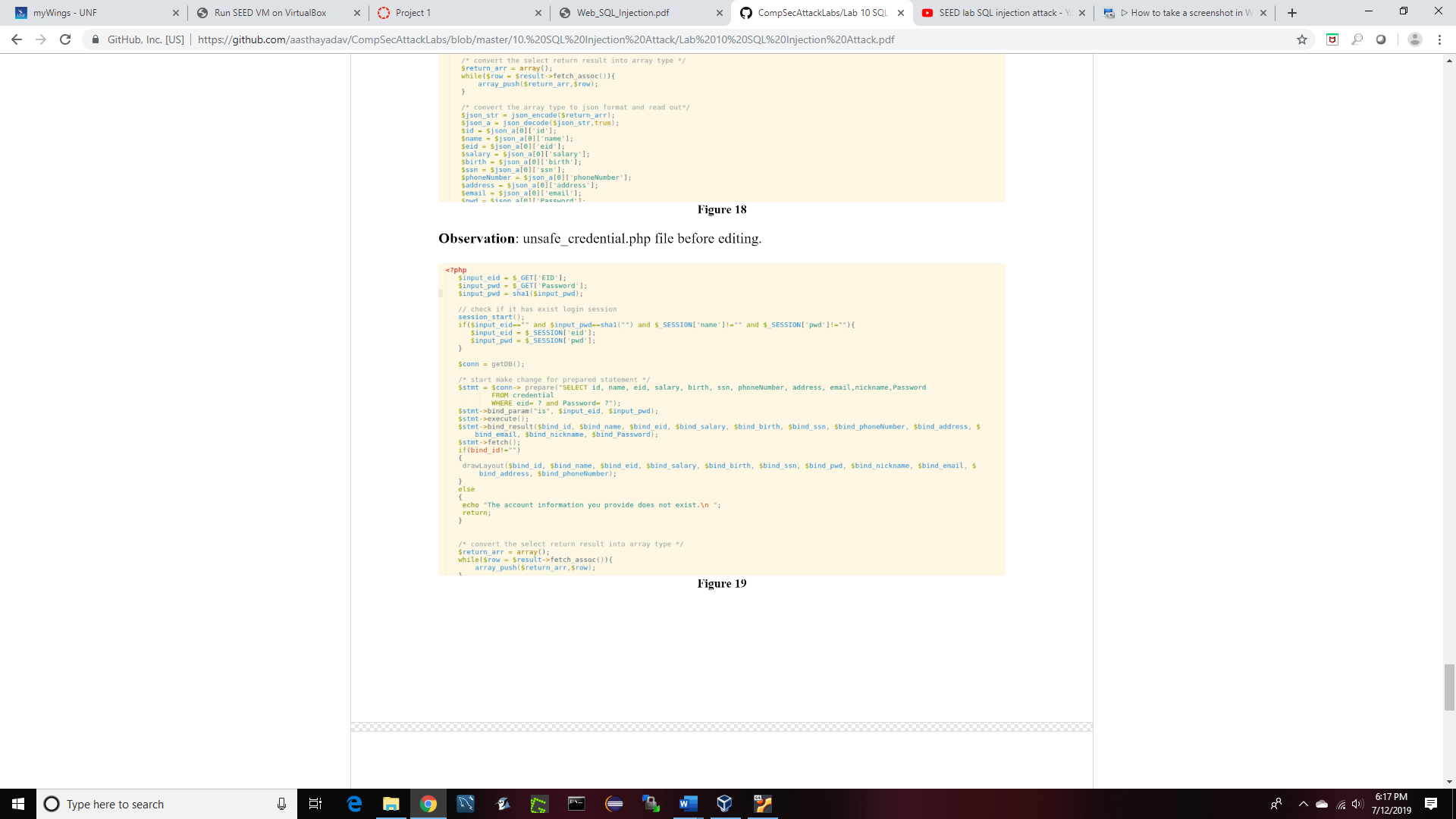
Type:

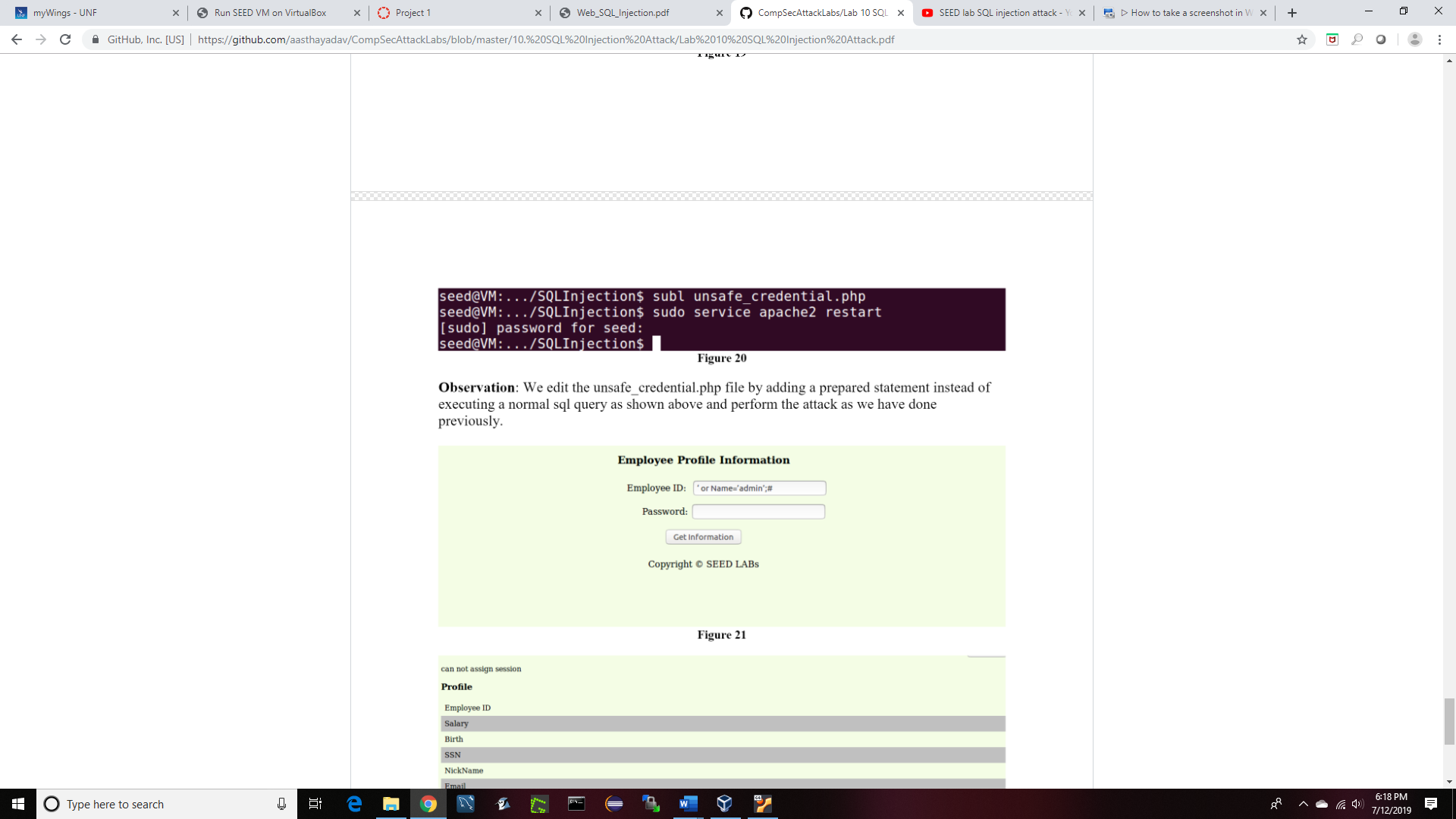
1. ‘ , password=’password123’ WHERE name=’Boby’; #
2. SELECT \* FROM credential WHERE Name=’Boby’;

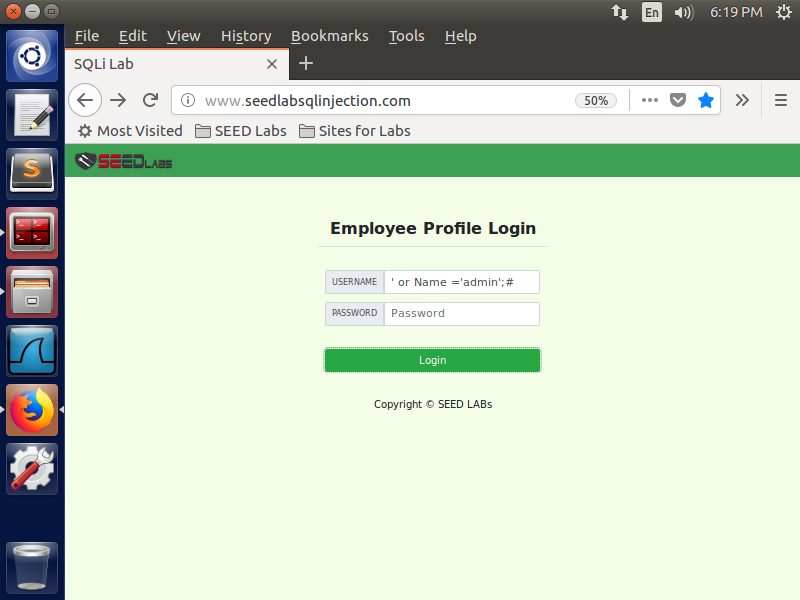


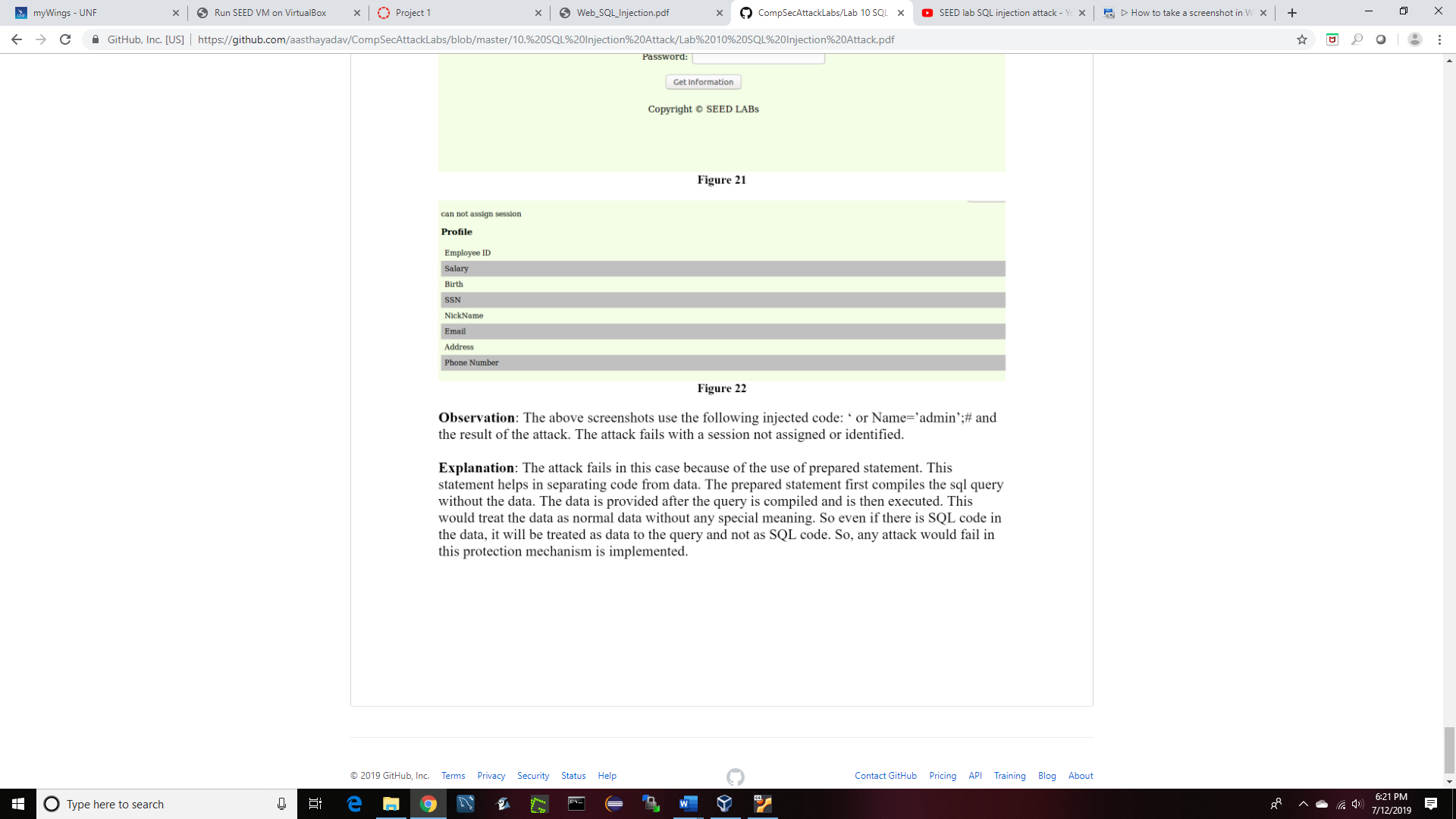


**Task 4: Countermeasure — Prepared Statement**









**Observation:** The attack fails and a session is not assigned.

**Explanation:** This attack fails due to the use of prepared statements. Prepared statements separate code from data by first compiling the SQL query without the data and then adding the data later on after the compilation process. This would cause the SQL queries to be treated as normal data and not as SQL code.