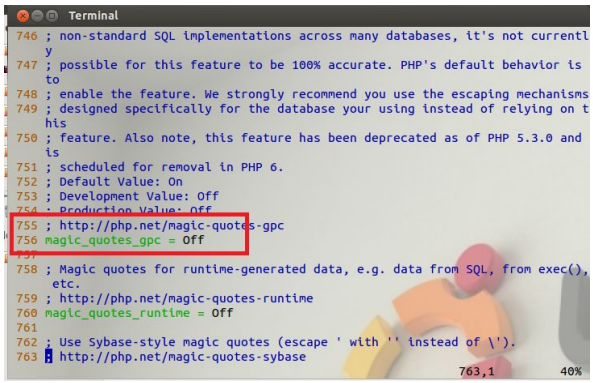
Aaron Cheung
CSC 154
Lab 5 SQL Injection

1) Prior to starting the lab, I had to patch both VMs in order to do this lab. The SEEDLABs website had a patch package for me to download, which I did and followed the instructions included. I also noticed the same instructions for the patch on the pdf for the lab. The commands for the patch are as follows:

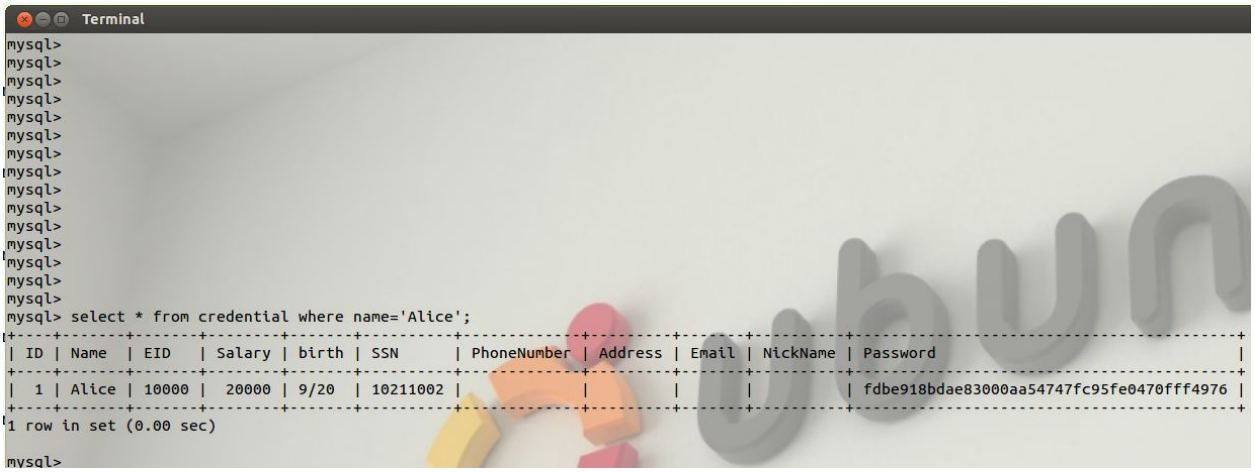
README

```
This is patch folder for exist SEED VM 12.04.
        author: Kailiang  kying@syr.edu

You have two ways to patch the SEED VM 12.04

1) automatic approach:
------------------------

chmod a+x bootstrap.sh
./bootstrap.sh
type seed password "dees"


2) manual approach:
------------------------

In case the automatic approach does not work for you.  Please take a look
at the SQL Injection Attack Lab description Appendix secton.
It contains detail step how to setup the SQL injection lab environment manually.
```

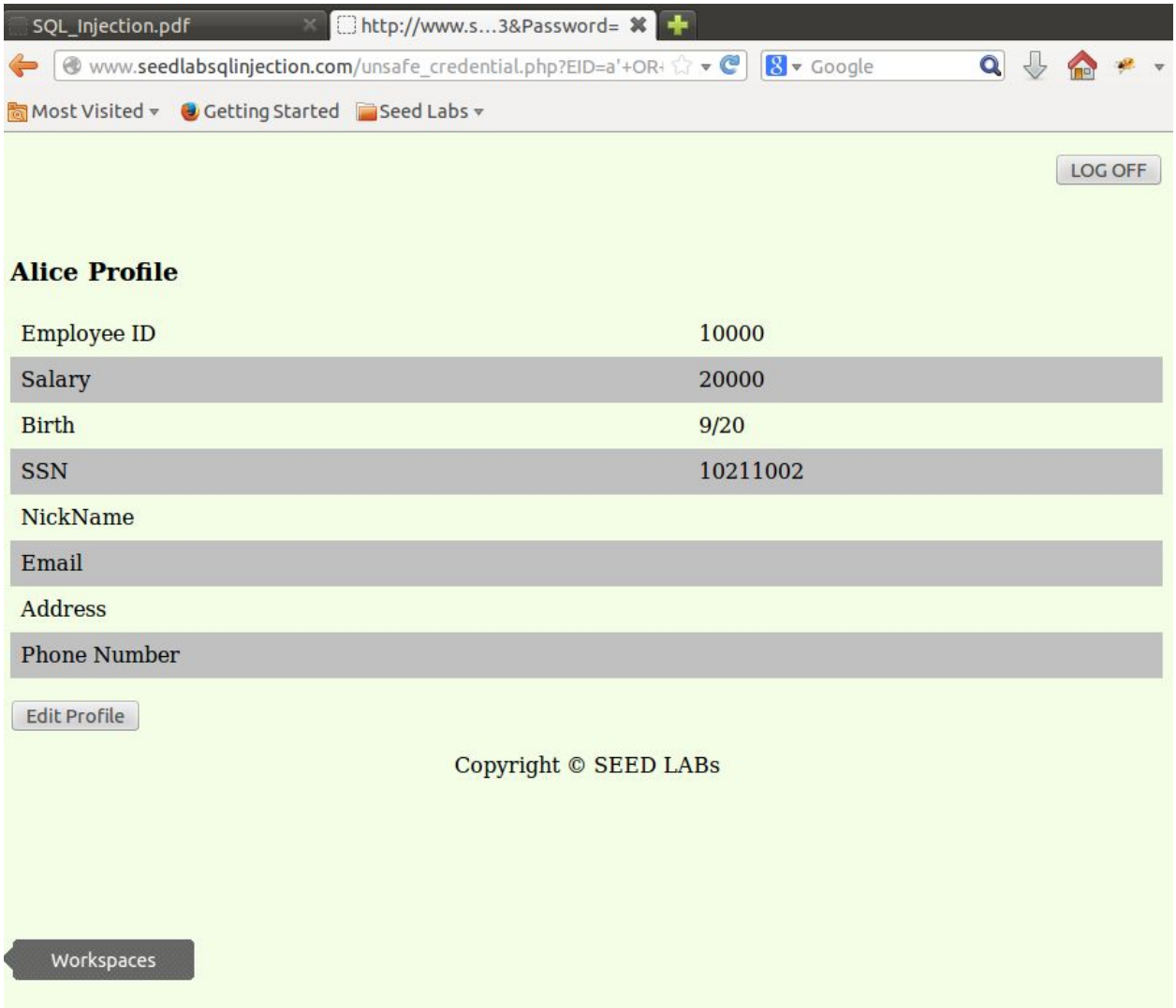
2) This is where I set the magic_quotes_gpc value to off to disable the countermeasure to the SQL injection attack that we are about to perform.



3) With all the prerequisites complete, we can now start the lab. This is task 3.1 where we experiment with the MySQL console. This is a screenshot of all the profile information of the employee Alice.

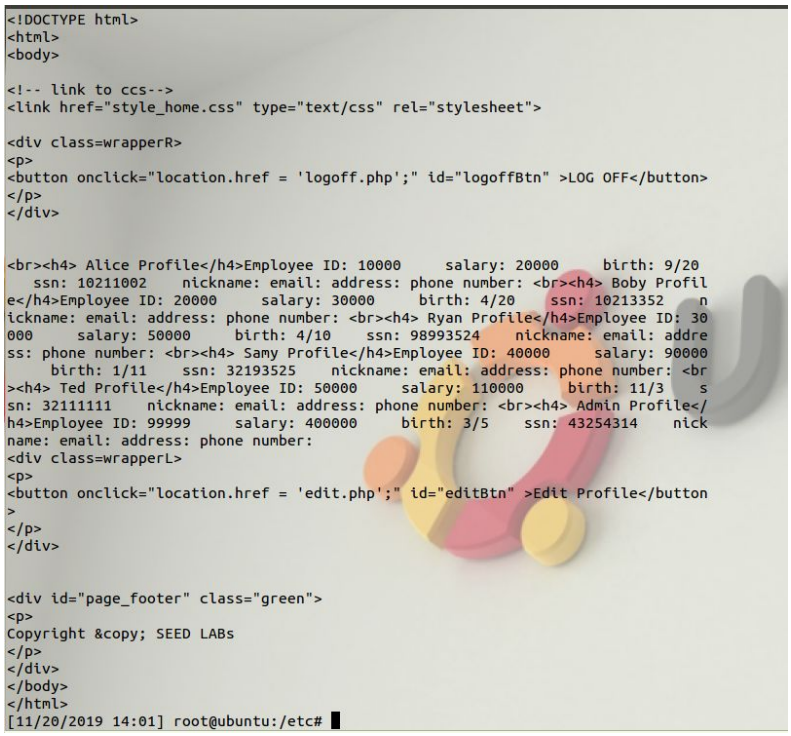


4) Here we actually do the SQL injection at the login page. The most important screenshot is the first one with the login fields. We have input the values “a’ OR 1=1#’. Now, what does this mean? To break this down, I will explain each piece of it below:
   a) The “a” is just a parameter I put in as a string.
   b) The ‘ OR 1=1# is the more important bit of the attack. We put the single quote in to separate our OR statement from any parameters we have entered. The database then evaluates the code after the OR statement, in this case it sees that 1=1 is true and allows us to login. The # is another important piece because it prevents the database from doing any further checks beyond the true statement.


SQL_Injection.pdf
http://www.s...3&Password=
www.seedlabsqlinjection.com/unsafe_credential.php?EID=a'+OR
Google
Most Visited
Getting Started
Seed Labs
LOG OFF
Alice Profile
Employee ID 10000
Salary 20000
Birth 9/20
SSN 10211002
NickName
Email
Address
Phone Number
Edit Profile
Copyright © SEED LABs
Workspaces

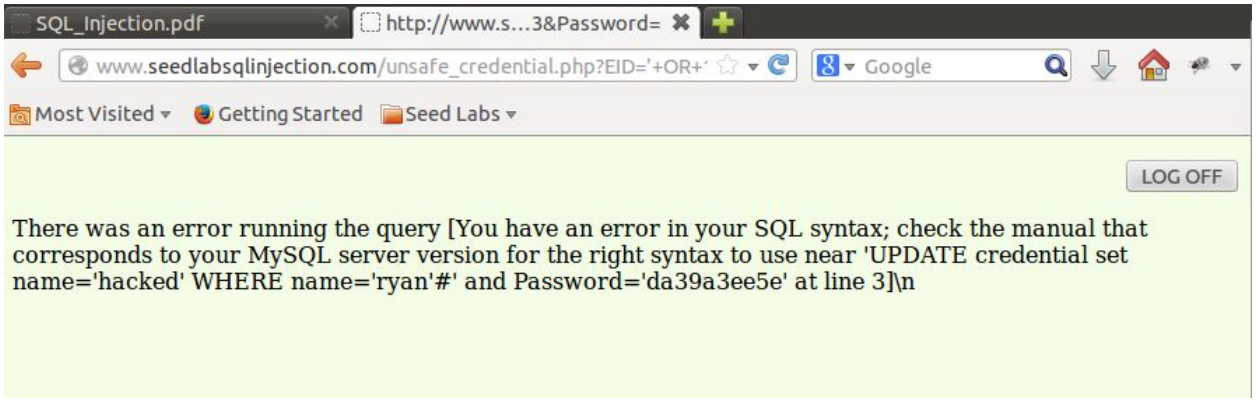
5) Here is a different example, pertaining to task 2.1 where we have to access the admin account that lists everyone's credentials and information. Same deal as the previous two screenshots.

6) Here is the SQL injection attack through the terminal. If we copy paste the url from the previous attack, we get pretty much the same result

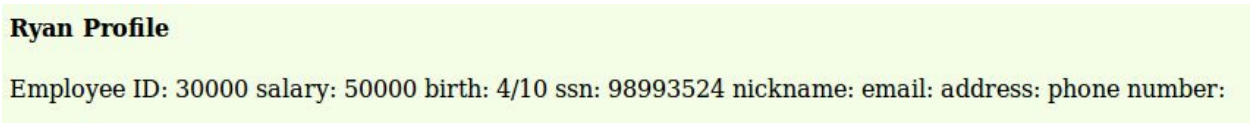


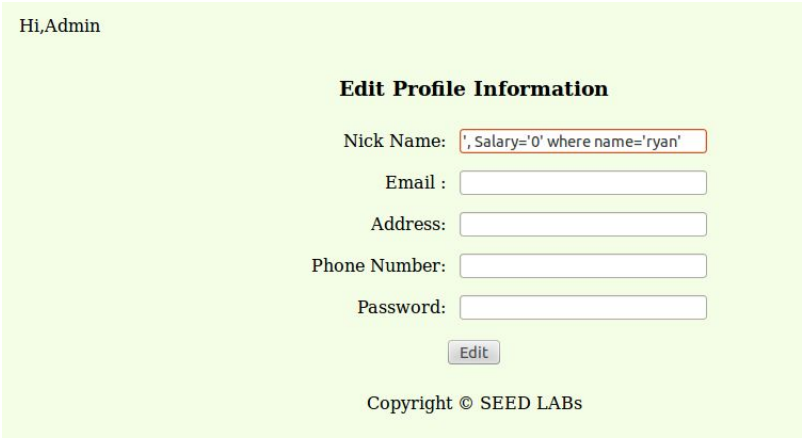
7) Now, we move on to attacking with multiple SQL statements rather than just the true 1=1 statement. Here we use the statement ` OR 1=1; UPDATE credential set name='hacked' where name='ryan';. However, as we can see, that does not work. This is because PHP

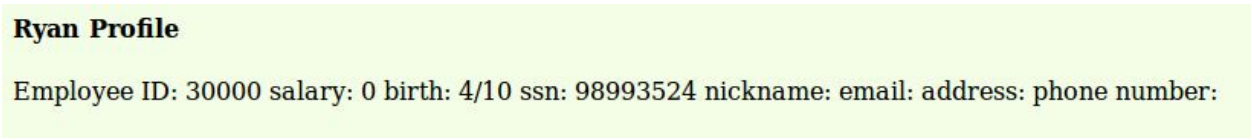
does not support stacked queries, so we can't exactly put multiple SQL statements into the login field. This pertains to task 2.3.



8) Despite our difficulties in the previous step, there is quite another handy way to do this, and that's with the edit profile button found when using the first injection attack where we were able to access the admin account. The first screenshot shows the information ryan had prior to our attack, the second screenshot shows our actual query that we inject (we are missing the # character at the end, which is important but i forgot to add it to the screenshot before I moved on), and the third screenshot shows what happens when the attack was successful.

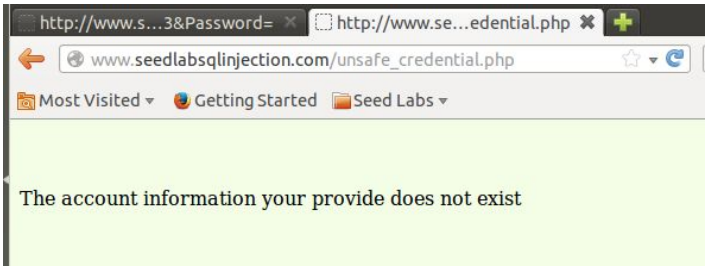






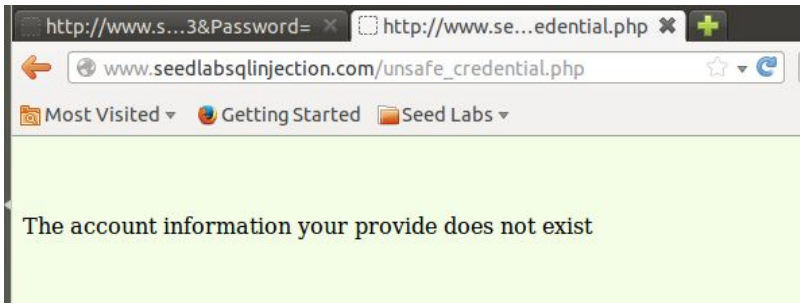
As we can see, the salary field changed to 0. This pertains to task 3.1.

9) Now we change the password for the admin account by using a similar procedure. The injection query we used this time was ‘,Password=sha1(‘hacked’) where name=’admin’#. Upon executing this statement by pressing the edit button, we immediately lose access to the account, as shown in the below screenshots. This is because the password was successfully updated from “seedadmin” to “hacked”, the password that we specified.



10) Now, we patch the victim virtual machine to prevent any further SQL Injection attacks. Here, we modify the “unsafe_credential.php” file to fix the bug. The screenshot below will show the area where the changes took place.

```
  /* start make change for prepared statement */
  $stmt = $conn->prepare ("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,
nickname, Password
       FROM credential
       WHERE eid= ? and Password= ?");
  $stmt->blind_param("ss", $input_eid, $input_pwd);
  $stmt->execute();
  $result = $stmt->get_result();

  /* convert the select return result into array type */
  $return_arr = array();
  while($row = $result->fetch_array(MYSQLI_ASSOC)){
      $return_arr[] = $row;
  }
```



And sure enough, the injection attack no longer works.