[**CS336 LAB4-new.pdf (uidaho.edu)**](https://bblearn.uidaho.edu/bbcswebdav/pid-2748482-dt-content-rid-34487405_1/courses/202110_29302_merge/CS336%20LAB4-new.pdf)

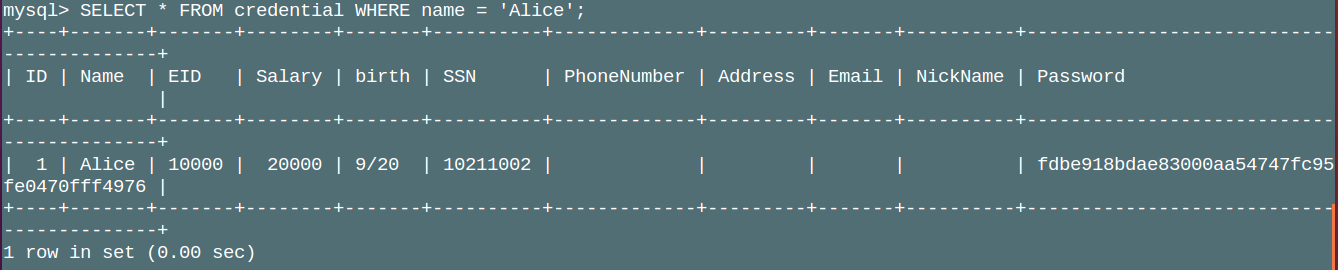
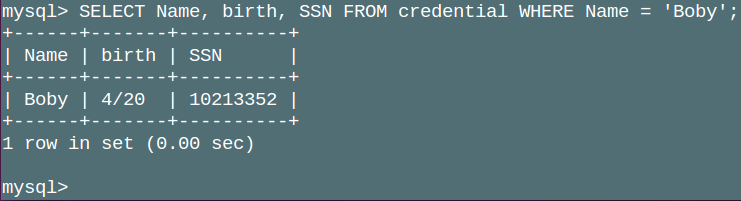
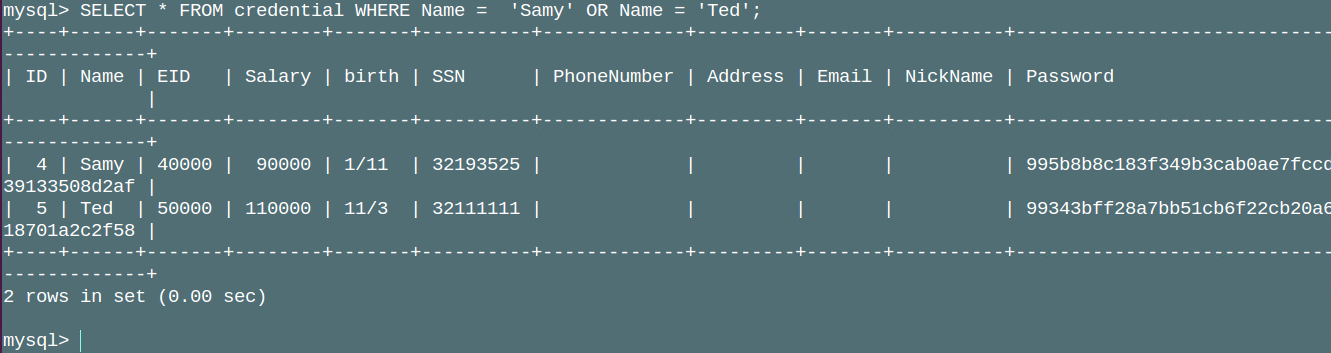
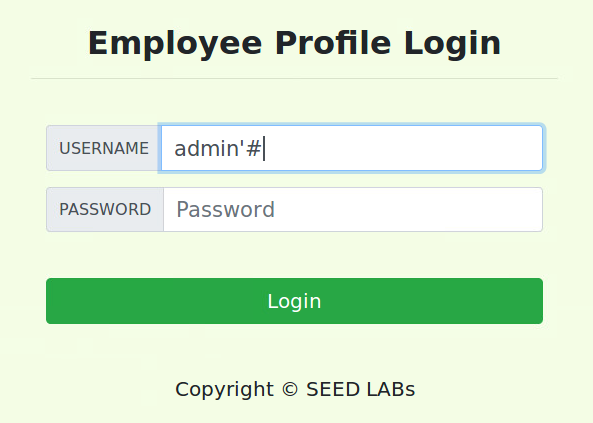
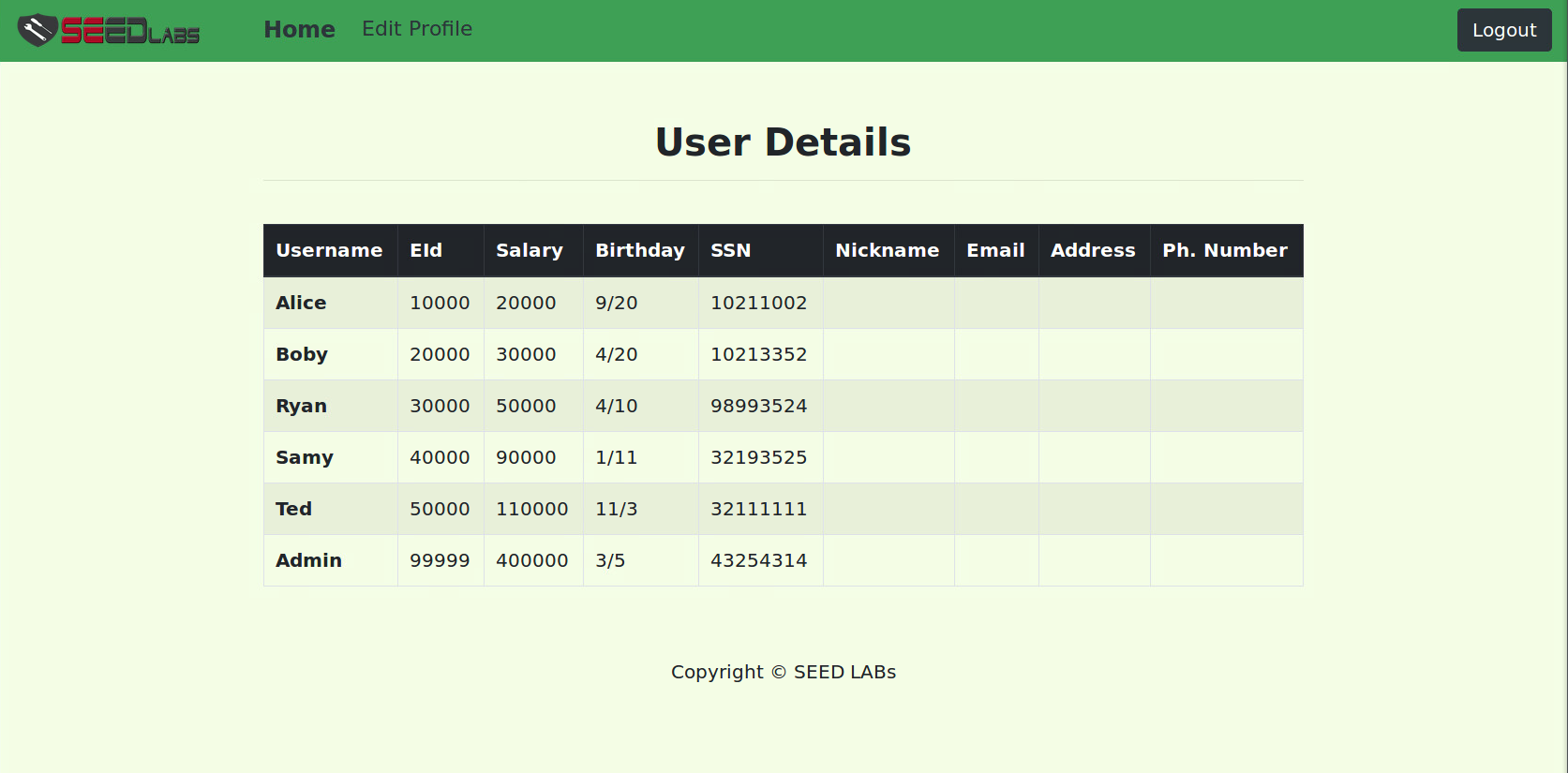
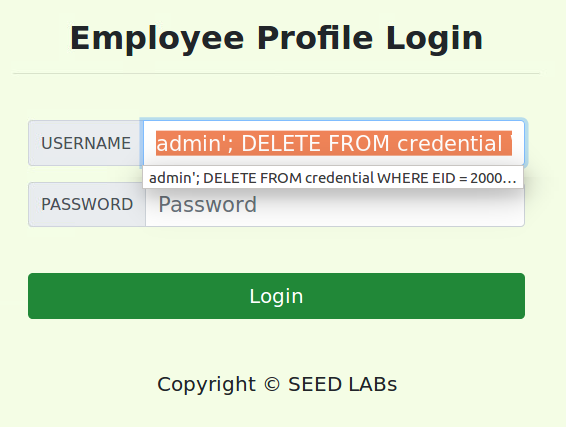
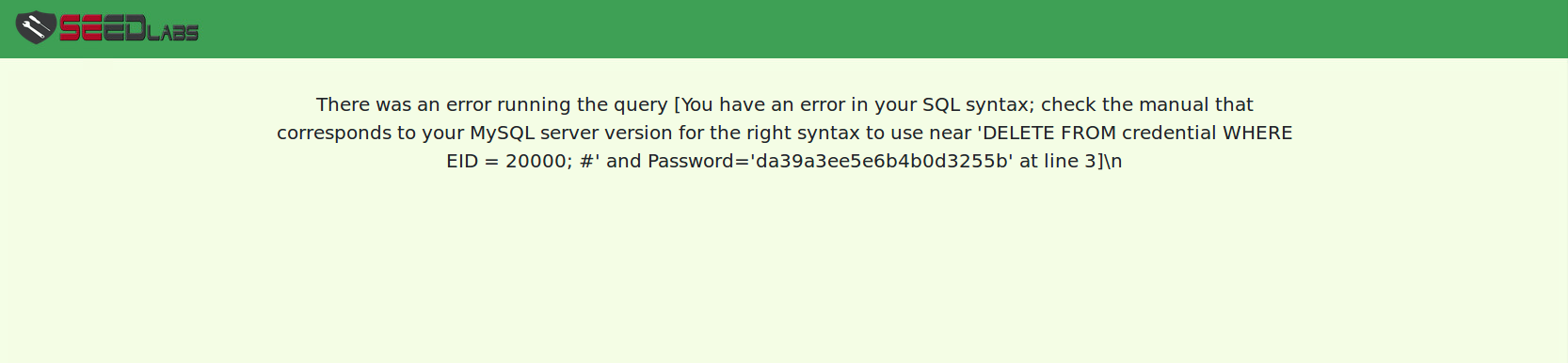
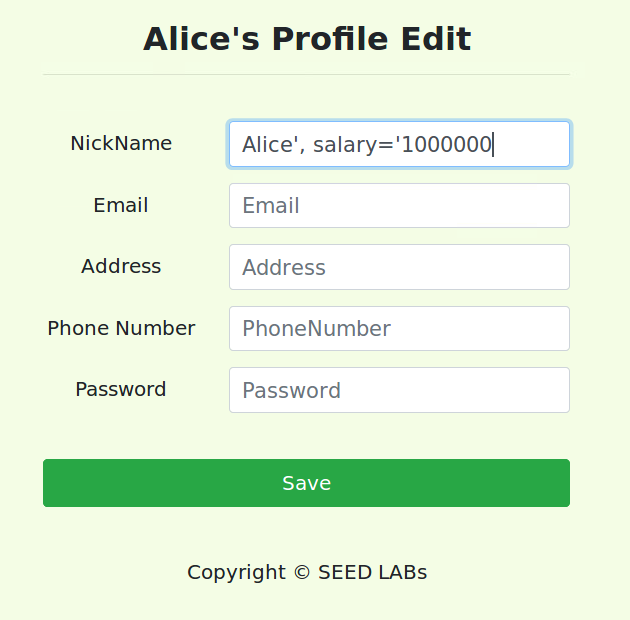
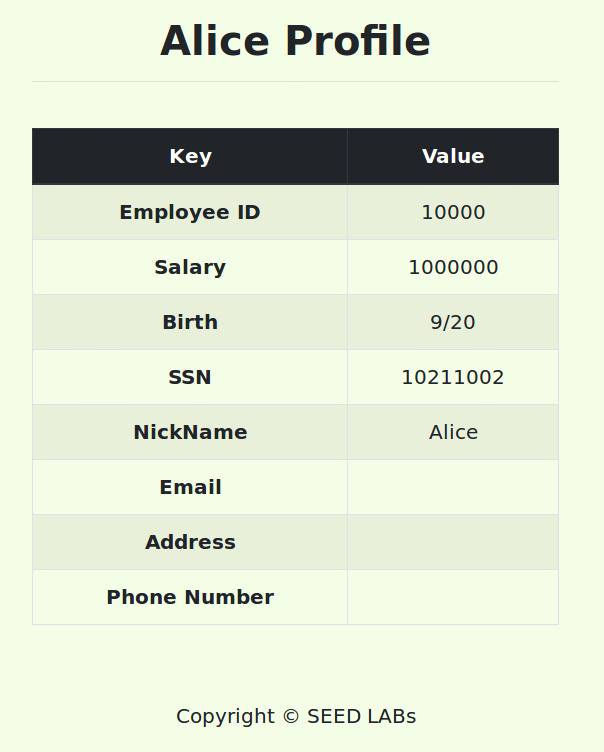
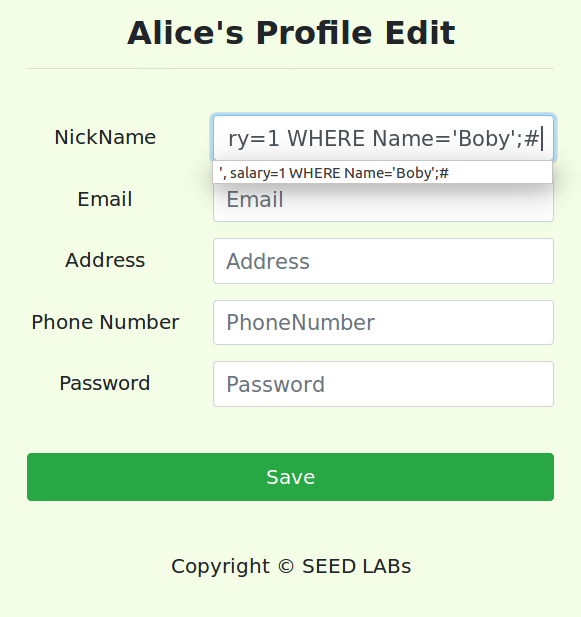
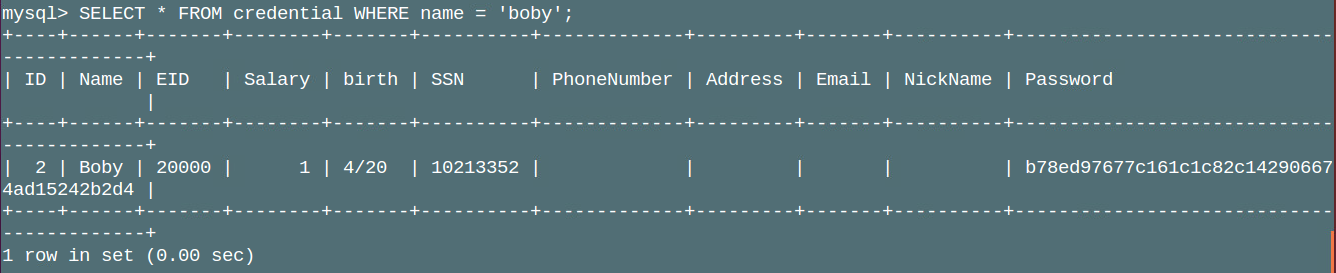
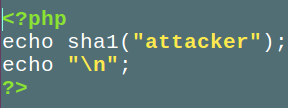
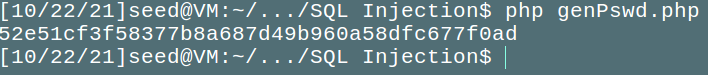
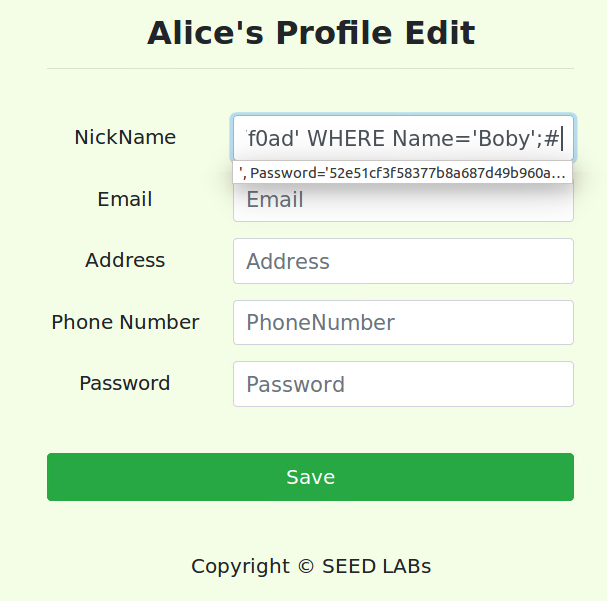
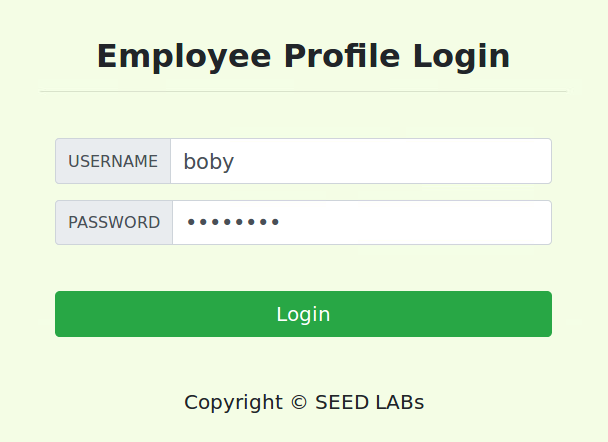
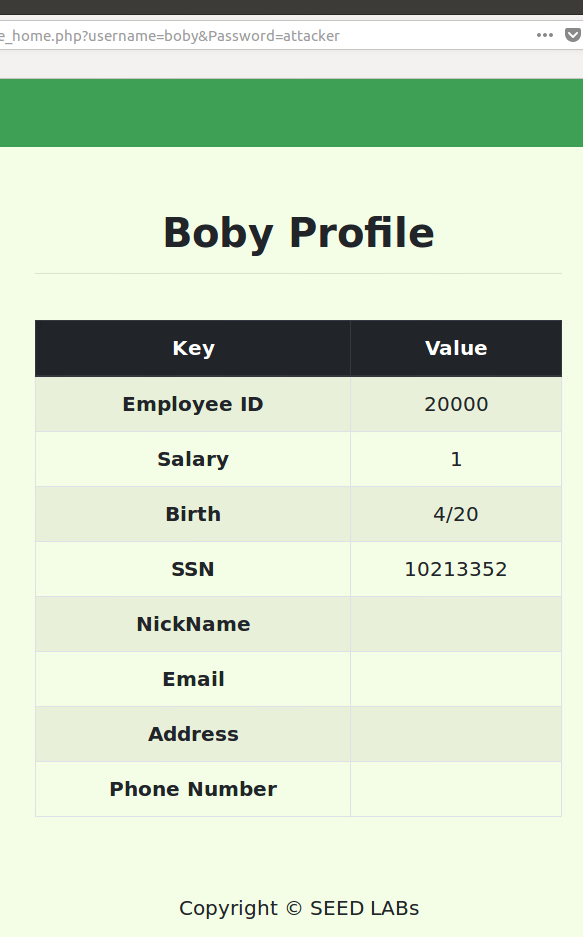
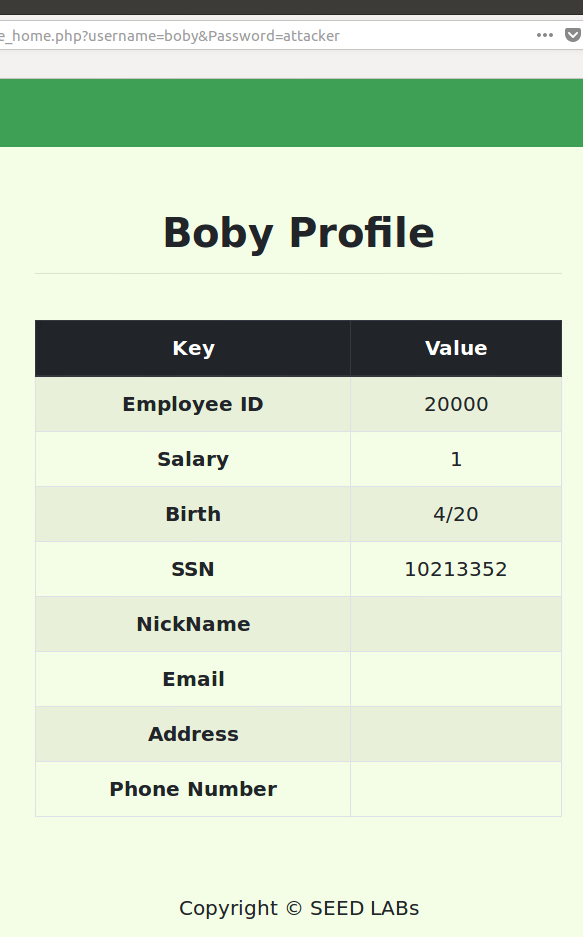
**CS336-LAB 4 SQL INJECTION LAB**

Due: Wednesday Oct 27th, 2:30pm.

Turn in: a lab report

Points: 70 pts

(70 points) Write a detailed report about the SQL injection attack lab. Include step by step screenshots and explanations of each task (Task1, 2 and 3).

* (15 pts) Task 1 (detailed tasks are listed on slide 15)
  + First we used the SQL command:
  + SELECT \* FROM credential WHERE name = ‘Alice’;
  + to print out the information of Alice:
  + 
  + Then we used the command SELECT Name, birth, SSN FROM credential WHERE Name = ‘Boby’; to print out the selected information of the employee Boby.
  + Finally, we used the command SELECT \* FROM credential WHERE Name = ‘Samy’ OR Name = ‘Ted’; to print out all the information of Samy and Ted.
* (20 pts) Task 2 (tasks 2.1, 2.2, and 2.3, requirements are on the handout)
  + Task 2.1: Log into the webpage as admin. To do this, we use the simple string admin’;# (Note: I realize as I write the lab report that this command and some of the others in this section should have semicolons to be valid sql; however, the commands still worked as written).
    - As demonstrated below, this comments out the check for the password and successfully logs us into the system:
  + Task 2.2: Repeat the same attack from the command line. To do this, we use the curl command, replacing the symbols #, ‘, and ; with %23, %27, and %3B respectively. We use the command:
    - curl http://www.seedlabsqlinjection.com/unsafe\_home.php?username=admin%27%23%3b&Password=
    - This returns the html of the admin’s login page, which we shouldn’t have access to.
  + Task 2.3: Run two commands from user login form. To do this, we just type our next SQL command after the semicolon of the login command: admin’; Insert\_Command\_Here; #
  + 
    - In this case, we try to delete an employee.
    - This didn’t seem to work for me, but according to the slides that may be expected.
* (25 pts) Task 3 (tasks 3.1, 3.2, and 3.3, requirements are on the handout)
  + Task 3.1: Edit Alice’s salary. In this case, the program will put our input between to single quotes. By using two quotes, we can splice a command into what the program is expecting, as shown below:
    - 
    - If done correctly, Alice’s salary should now be $1,000,000.
    - 
    - Logging in to Alice’s account, we can see the attack succeeded.
  + Task 3.2: Change Boby’s salary to $1. For this, we can use the profile edit page. By default, the backend of the Nickname editing box uses WHERE to specify the current employee. We can comment that out and put in our own WHERE Name=’Boby’# to target Boby. Then, we just have to set salary to 0:
  + 
    - Looking back at the mysql command line interface, we can see that the attack succeeded and Boby’s salary is now $1.
  + Task 3.3: Change Boby’s password. In this sql database, passwords are stored as a SHA1 Hash. So, we can’t directly repeat our last attack setting Password to whatever we want. First we have to hash the password we want to use:
    - 
    - This short PHP script will output the hash of the string “attacker”.
    - 
    - Now, we can use *this* hash as the input for our attack. Into Alice’s profile edit, we write:
    - ‘, Password=’52e51cf3f58377b8a687d49b960a58dfc677f0ad’ WHERE Name=’Boby’;#
    - 
    - If this worked as expected, we should be able to log in to Boby’s account using our chosen password ‘attacker’:
    - 
    - As you can see in the url of the screenshot below, the string ‘attacker’ worked for the login:
    - 
    - 
* (10 pts) Include a conclusion of the lab, what have you learned?
  + This was a very interesting lab. First of all, I’ve never used sql before so I learned some new sql commands, such as SELECT, INSERT, UPDATE, DELETE, FROM, WHERE, and AND and OR. This lab again highlights the importance of treating all user input as malicious. The sql behind the scenes on this site was so insecure that we were able to carry out all these attacks in less than two hours. I can imagine that in more complex systems there would be even more possible avenues of attack, so it is that much more important to make sure the inputs are properly handled.