***SQL Injection Lab***

**Introduction**

In this lab, we are going to be familiar with the most important vulnerability of web applications , SQL injection. In this vulnerability we can see that web applications get the inputs from users and before checking them construct the queries based on them and send them directly to database. And so attackers will use such a vulnerability to put malicious query into database. and of course still is on of the common mistakes of most developers.

For this task we will use our ubuntu Virtual machine which we can go through a vulnerable site

URL: <http://www.SEEDLabSQLInjection.com>

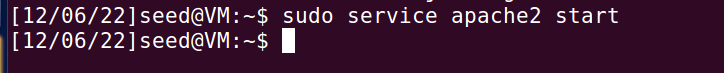
Folder: /var/www/SQLInjection/

This site is just accessible through our Seedubuntu because earlier/etc/hosts has been modified and mapped to to this URL :

Text

Description automatically generated

Before going to the task need to just start Apache2 with its default configuration to start it locally:



Graphical user interface, text, application

Description automatically generated

**Task 1: Get Familiar with SQL** **Statements**

In this task we just want to play with the database that is in our ubuntu and get familiar with query and these stuffs.

First need to connect to “mysql” which is an open-source database management system in ubuntu

-u for username

-p for password

Text

Description automatically generated

show stored databases in system:

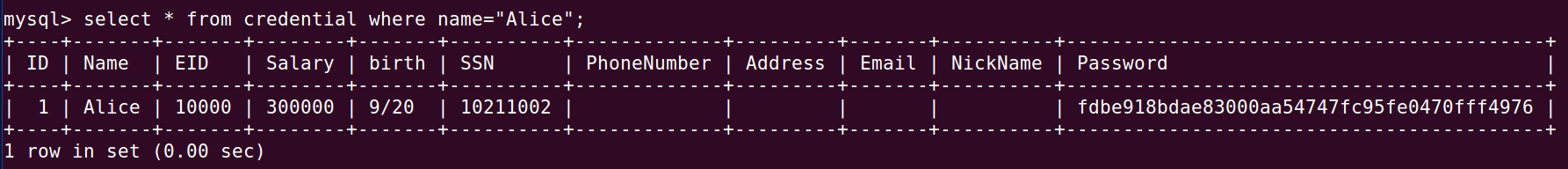
Text

Description automatically generated

Text

Description automatically generated

SQL command to print all the profile information of the employee Alice.



**Task 2: SQL Injection Attack on SELECT Statement**

In SQL injection, attacker will use the vulnerability and with sql statement, steal information from our database or even update our database. for this task we will use the login page of the website which needs to input username and password for authentication. So for entering the site needs to know password. For the attack we are going to log in to the site without knowing credentials.

// just to know that there are two privileges in this database: 1. Admin: can see all of the employees data 2. Employee: can see its own data only

**Task 2.1: SQL Injection Attack from webpage**

Below is how this site work with database, as you see it is directly query from database by all data get from user :

**$input\_uname = $\_GET[’username’];**

**$input\_pwd = $\_GET[’Password’];**

**$hashed\_pwd = sha1($input\_pwd);**

**...**

**$sql = "SELECT id, name, eid, salary, birth, ssn, address, email,nickname, Password**

**FROM credential WHERE name= ’$input\_uname’ and Password=’$hashed\_pwd’";**

**$result = $conn -> query($sql);**

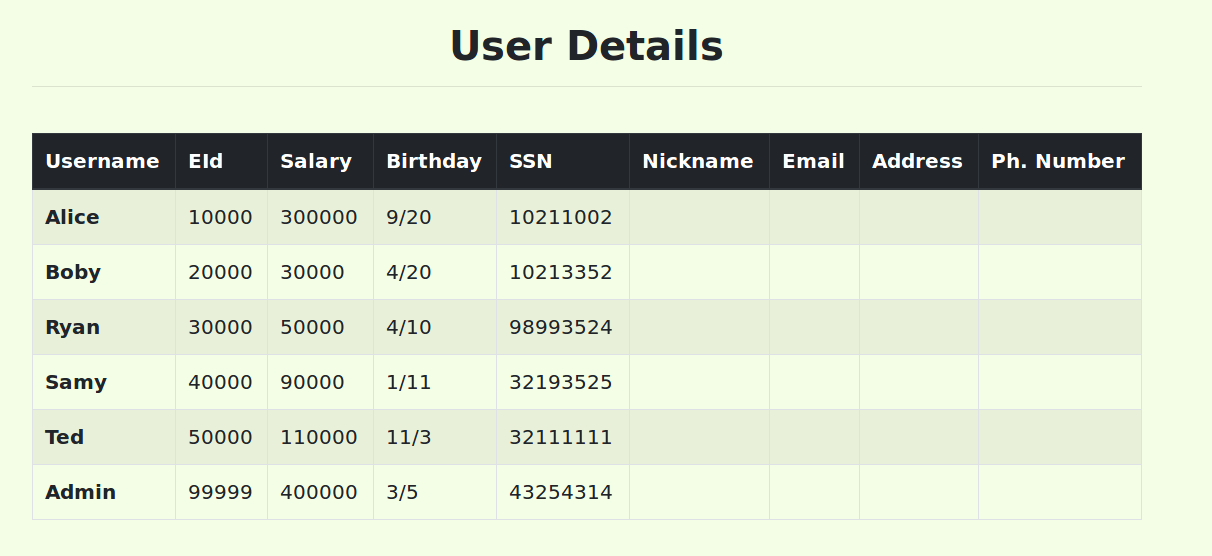
**// The following is Pseudo Code**

**if(id != NULL) {**

if we put below in username as below picture and without inserting any password ,press the log in , **input\_uname** will be replaced by “**admin**” and ‘#’ makes all after discarded like as comment and not executed So we will log in with admin user and can see all the data with out any password.

Graphical user interface, application

Description automatically generated



And this is how sql injection works.

**Task 2.2: SQL Injection Attack from command line.**

In this task we want to get the same result just by using command line tools.

By viewing the source of the site by right click in page :

Graphical user interface, application, website

Description automatically generated

We can see below source PHP code which get the username and Password :

Graphical user interface, text

Description automatically generated

The command line tool which we will use is “curl” to send HTTP request

For using this command we need to send URL and all the parameters with & between them ( all between ‘’)

Since the command should be coded in HTTP request

‘ : % 27

# : %23

And below screenshots show that we get the html that has all data in database :

Text

Description automatically generated

Text

Description automatically generated

**Task 2.3: Append a new SQL statement**

As per TA class don’t need to do it

**Task 3: SQL Injection Attack on UPDATE Statement**

We know how to log in with any user , just by entering name of the user followed by **‘#** to bypass the password

Graphical user interface, application

Description automatically generated

Alice can see her Profile

Table

Description automatically generated

There is a button that can edit the profile

Shape

Description automatically generated with medium confidence

But by clicking in edit profile, user can only edit “NickName” , “Address”, “Email”, “Phone Number” and “Password” .

**Task 3.1: Modify your own salary**

So with employee privilege user can not modify the “salary”..

Nickname is the field that get its data from user so we can exploit using this field. As below with **‘** we close the nickname parameter and then put our own parameter to update and save.

Graphical user interface, application

Description automatically generated

And with below picture it shows that SQL injection is successfully done.

Table

Description automatically generated

**Task 3.2: Modify other people’ salary**

We can also do the same for other employee by just add condition

Graphical user interface, application

Description automatically generated

If we log in as admin and check Boby’s salary we see that salary has been changed to “1”

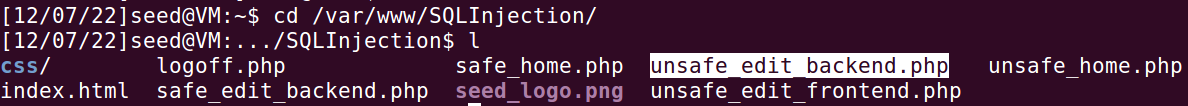
Table

Description automatically generated

**Task 3.3: Modify other people’ password**

Just to have more and more interesting things and can damage other users details, we can change their password as well. Database store all passwords in hash format of them instead of plain text

By checking below directory and reading the php code we can understand that this web application hash all passwords with SHA1



Text

Description automatically generated

First we can find hash of our considered password with below command

Text

Description automatically generated

Then we can put this hash code instead of Samy’s password by editing NickName in Alice’s Profile :

',Password='561c0f70de39daa50a56969195f22947a7dfb0eb' where name='Samy';#

Graphical user interface, application

Description automatically generated

For checking this password, just connect to samy’s profile with considered password here is “task3’ and we can successfully log in

Graphical user interface, application

Description automatically generated

Table

Description automatically generated

**Task 4: Countermeasure — Prepared Statement**

In order to fix this vulnerability, need to change and edit the “unsafe\_home.php” which we have seen through this lab, it gets all data directly from user and put them into the query and pass it to database.

This file is in below path:

Text

Description automatically generated

The code that must be changed is:

Text, timeline

Description automatically generated

For this change we need to go to below path and with sudo command change this part :

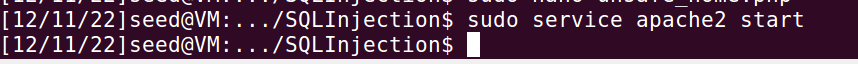


Because it doesn’t have any prepared statement, so below you can find corrected format of that which needs to authenticate the user:

Chart, text

Description automatically generated

After this change first we start the apache2 service again to load the change and then we try to attack same as task2.



if we try to do the attack we can’t see anything, as this bug has been patched

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated with low confidence