Tên: Lê Hải Đăng

Mã số sinh viên: 20110243

**BÁO CÁO THỰC HÀNH LAB 5**

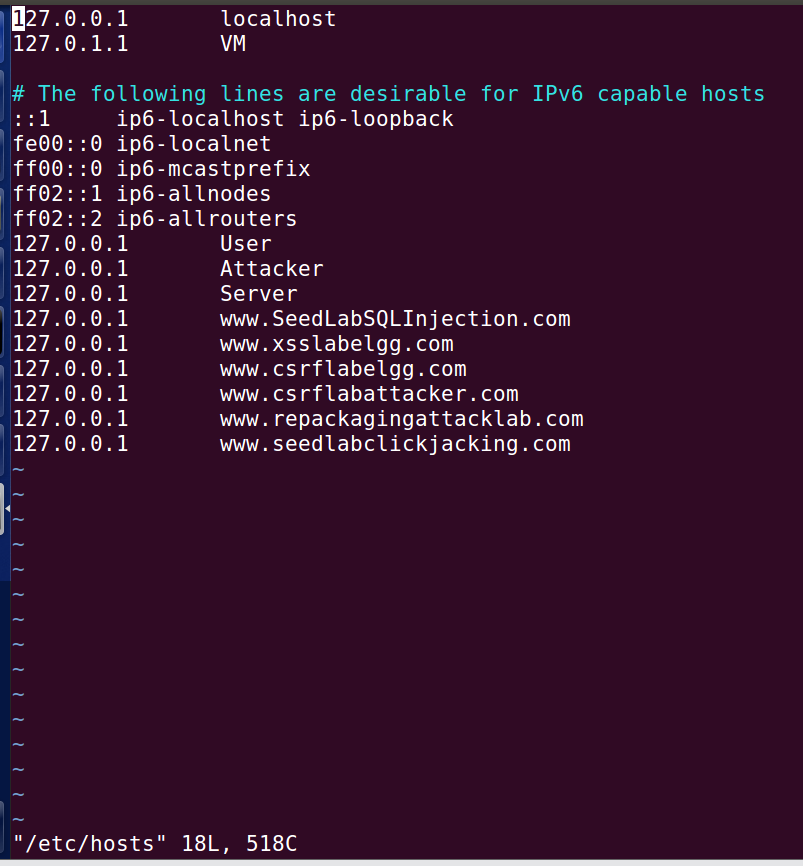
**SQL Injection**

**1. Chuẩn bị**

Chuẩn bị máy ảo ubuntu 32bit, được tải ở trang web seed (<https://seedsecuritylabs.org/lab_env.html>)

Để xem được cấu hình của host, em sử dụng câu lệnh: *sudo vi /etc/hosts*



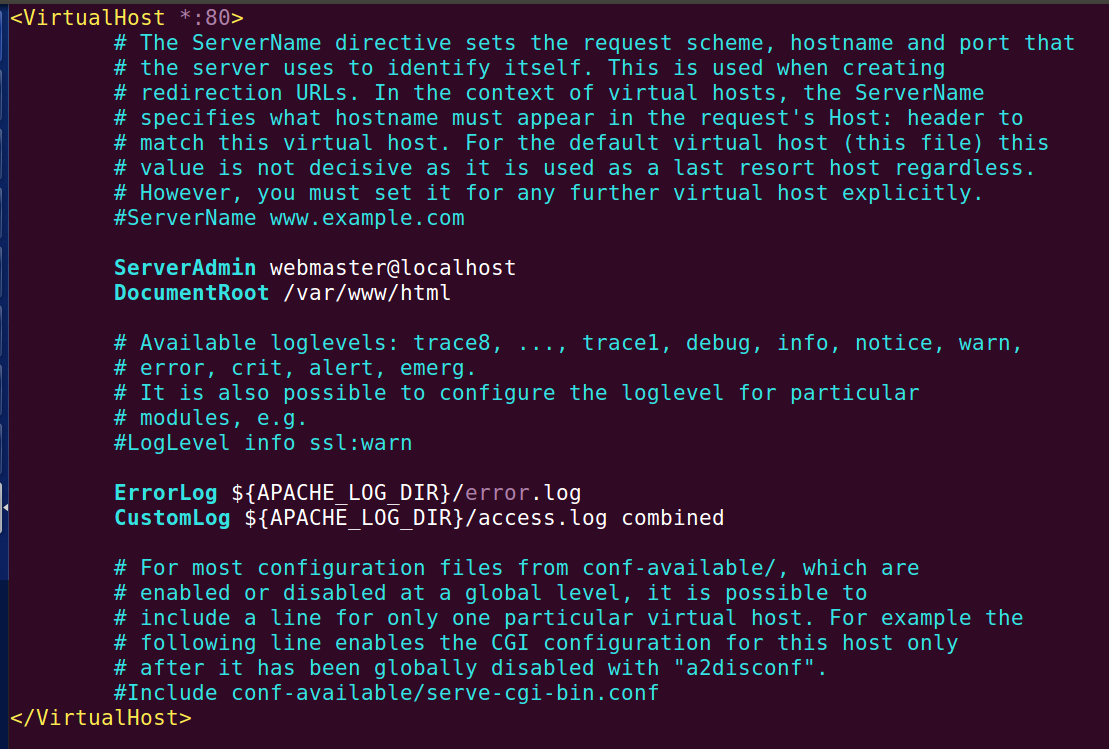
Do đây là một máy ảo được cấu hình sẵn của seedlab, nên mặc định sẽ có những đường dẫn tới các bài lab khác nhau, điển hình là [www.SeedLabSQLInjection.com](http://www.SeedLabSQLInjection.com) mà trong bài lab này em sẽ thực hiện sql injection

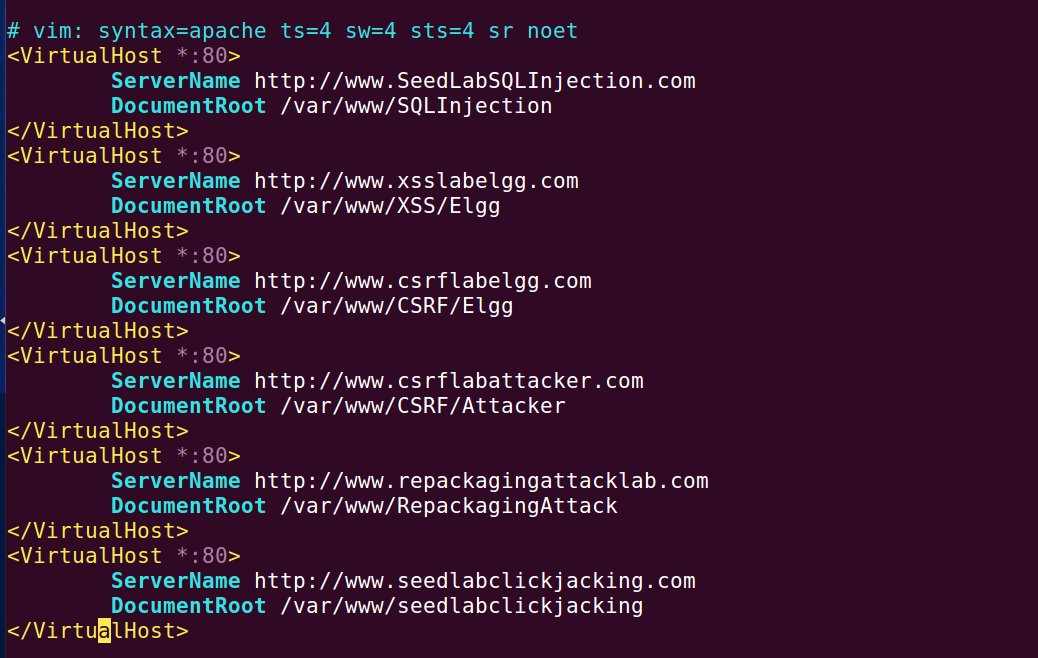
Bên cạnh đó, em tiến hành xem cấu hình trong file 000-default.conf bằng câu lệnh: *sudo vi /etc/apache2/sites-available/000-default.conf*



Ở đây, có thể thấy được DocumentRoot hay còn được gọi là PhysicalRoot ( window ).

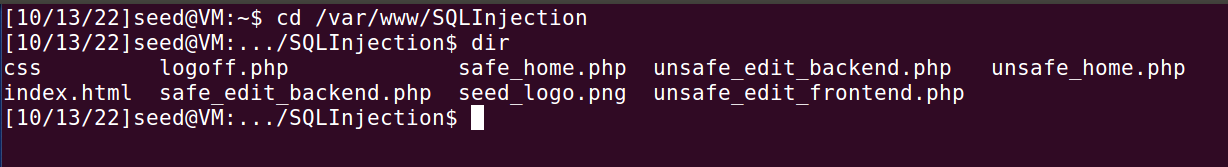
DocumentRoot chứa các file để cấu thành trang web trên ServerName, trong đó có cả front-end và back-end code





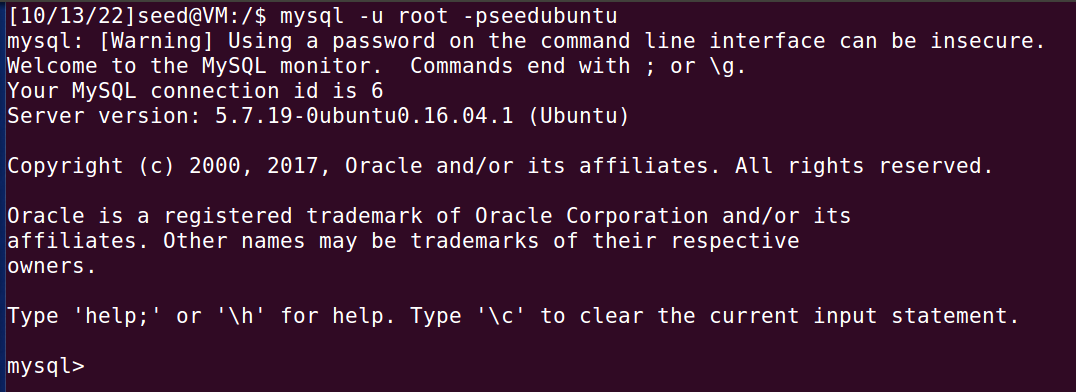
Tiến hành truy cập vào thư mục chứa server SQLInjection bằng câu lệnh: *cd /var/www/SQLInjection*

Sử dụng câu lệnh “dir” hay “ls”, một danh sách các file sẽ được hiện ra, những file này sẽ rất quan trọng trong quá trình exploit trang web.



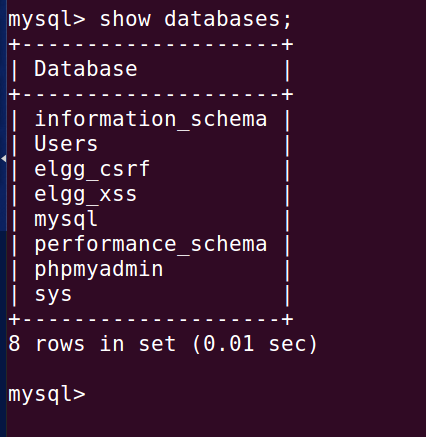
**2. Sử dụng Mysql**

Truy cập vào MySQl đã được cài đặt sẵn thông qua câu lệnh: *mysql -u root -pseedubuntu*

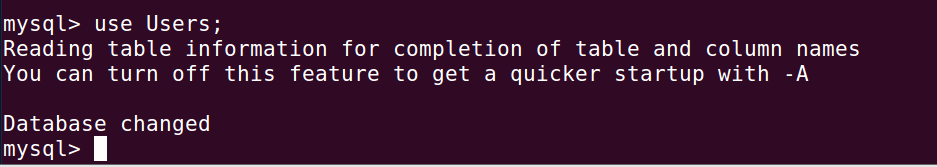


Sau đó, em đã có quyền sử dụng mysql thông qua thẻ mysql> (Trường hợp cần thoát chế độ mysql, ta có thể dùng lệnh “exit”).

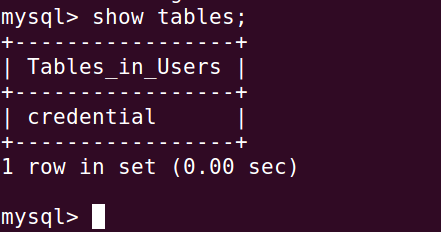
Trong mysql, seedlab đã cài đặt sẵn các dữ liệu mẫu, sử dụng cậu lệnh: *show databases;* để liệt kê các databasae đang có.



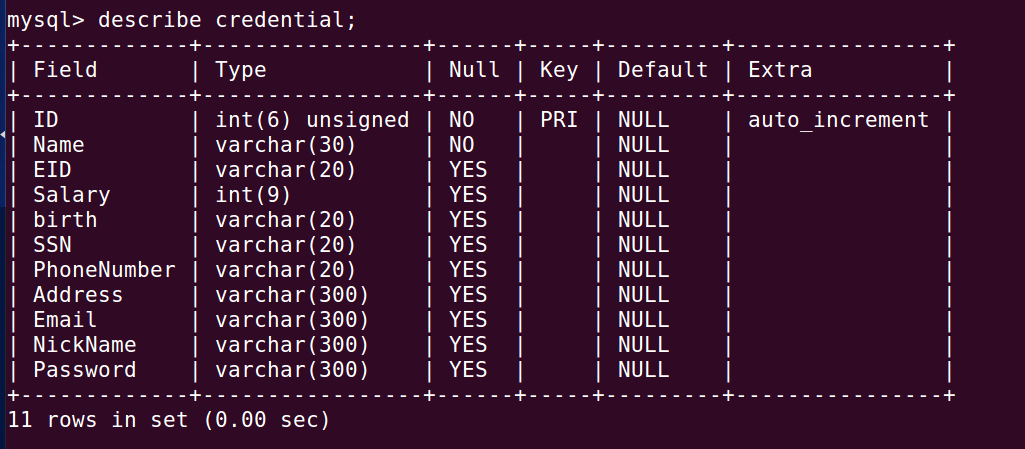
Trong bài lab này, em tiến hành tấn công sql injection đối với database “Users”, để truy cập vào database Users sử dụng câu lệnh: *use Users;*



Trong database Users, sử dụng câu lệnh: *show tables;* để liệt kê các bảng có trong cơ sở dữ liệu này.



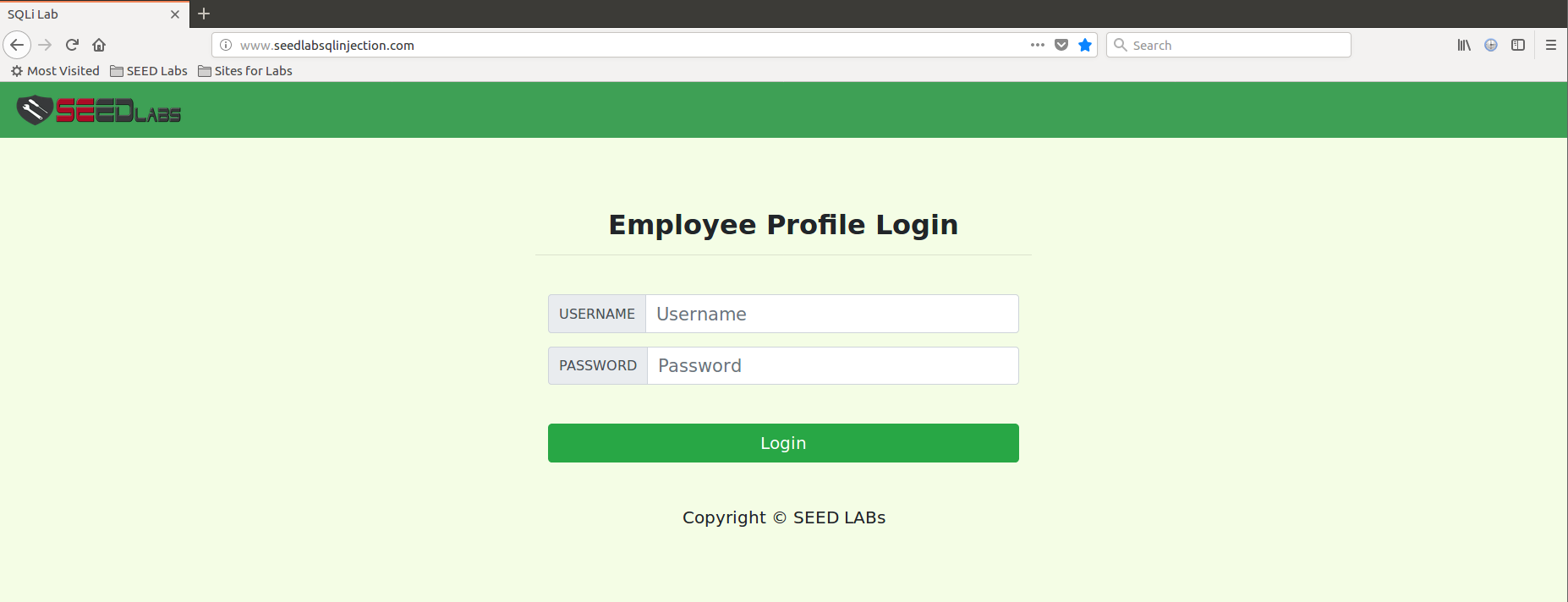
Tiếp theo, sử dụng câu lệnh: *describe credential;* để xem các thông tin có trong bảng credential. Trong credential, có các trường thông tin như Name, Salary, Password,....Trong đó, em sẽ tiến hành tấn công để thay đổi Salary và Password theo yêu cầu của bài lab.

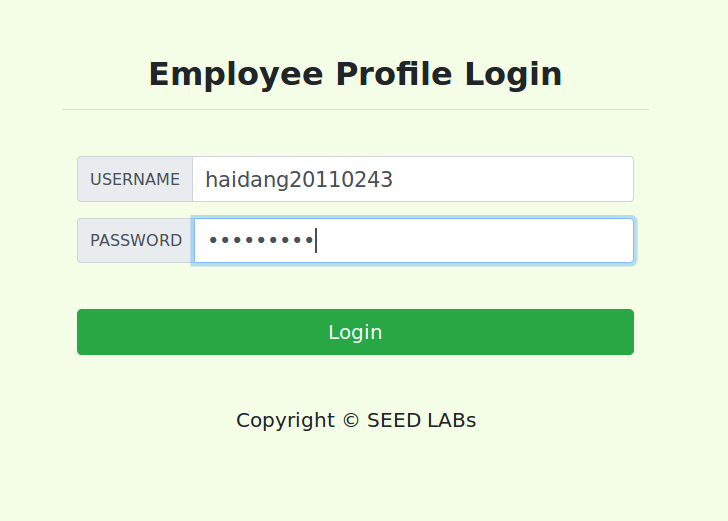


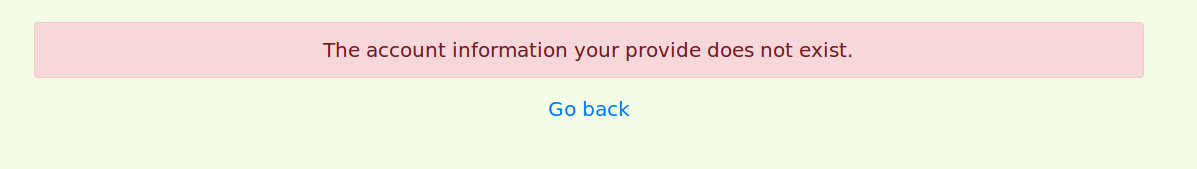
**3. SQL injection bằng câu lệnh select**

Truy cập vào website ***“seedlabsqlinjection.com”***, đây là một web site chứa nhiều lỗ hỏng nhằm để kiểm thử và làm những bài lab luyện tập.

Trong đó server của website này được lữu trữ trong máy em tại đường dẫn “/var/www/SQLInjection/”



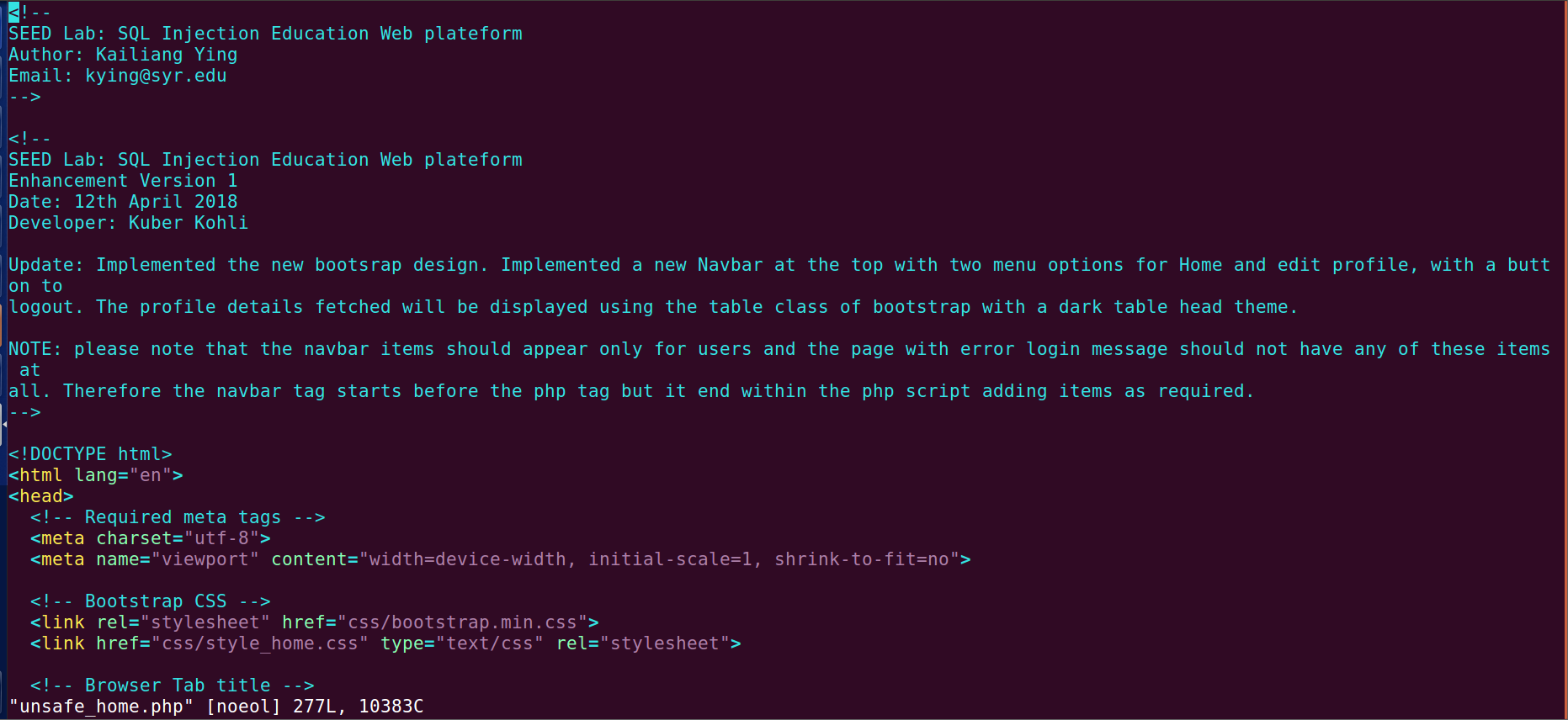
Tại website này, nếu nhập tên hay mật khẩu sai, tất nhiên sẽ không thể đăng nhập vào được

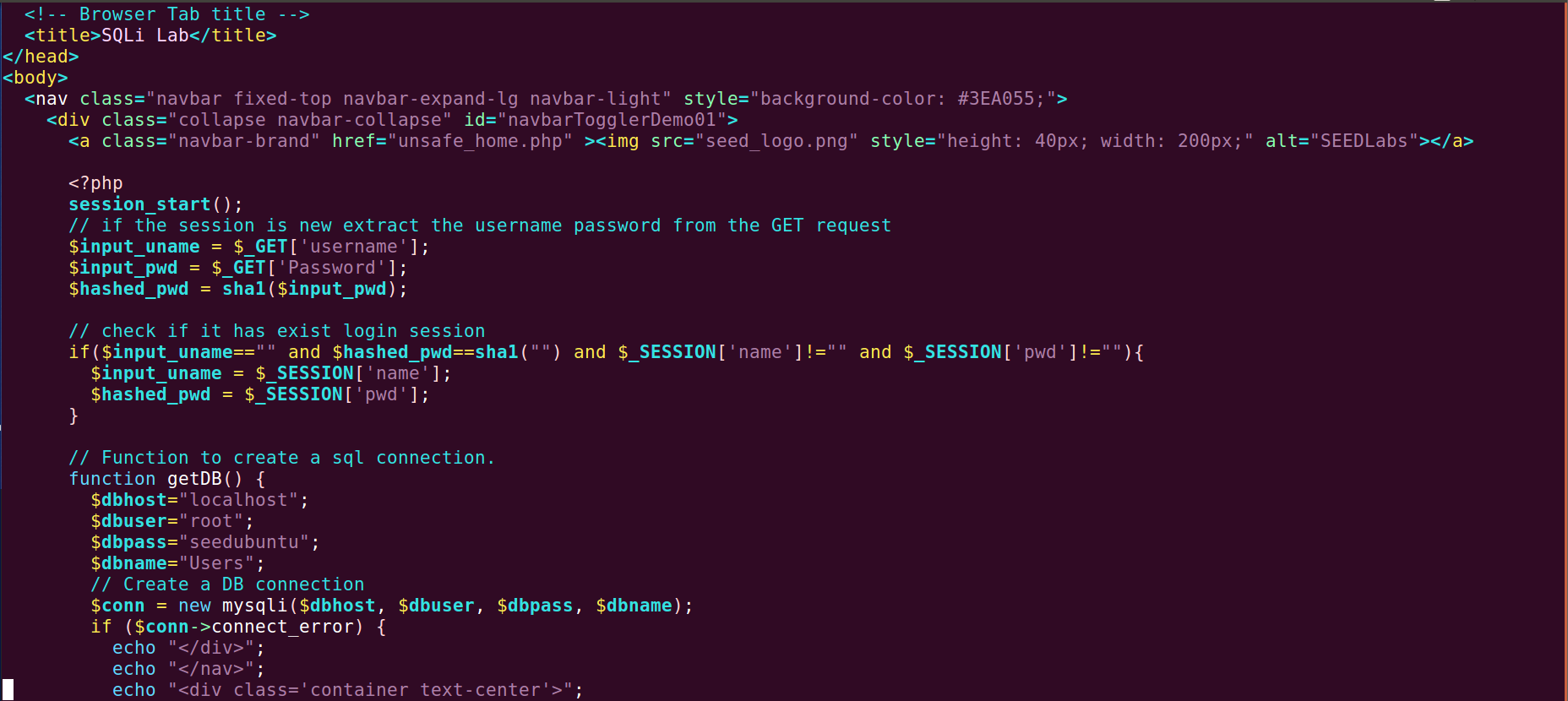


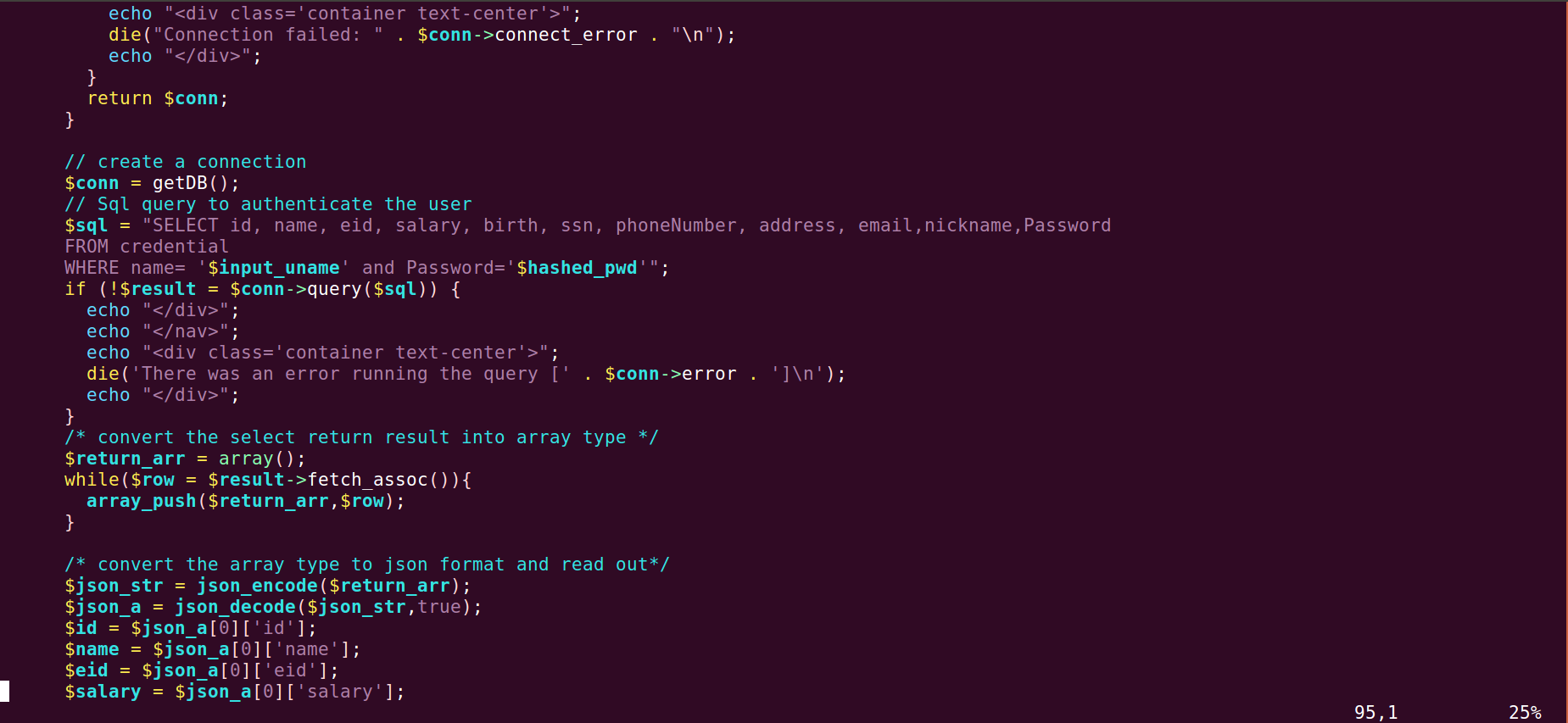
Để có thể truy cập vào được website theo phương thức sql injection. Hãy xem qua file code backend của trang web. Trong thư mục SQLInjection ta gõ lệnh: *sudo vi unsafe\_home.php*

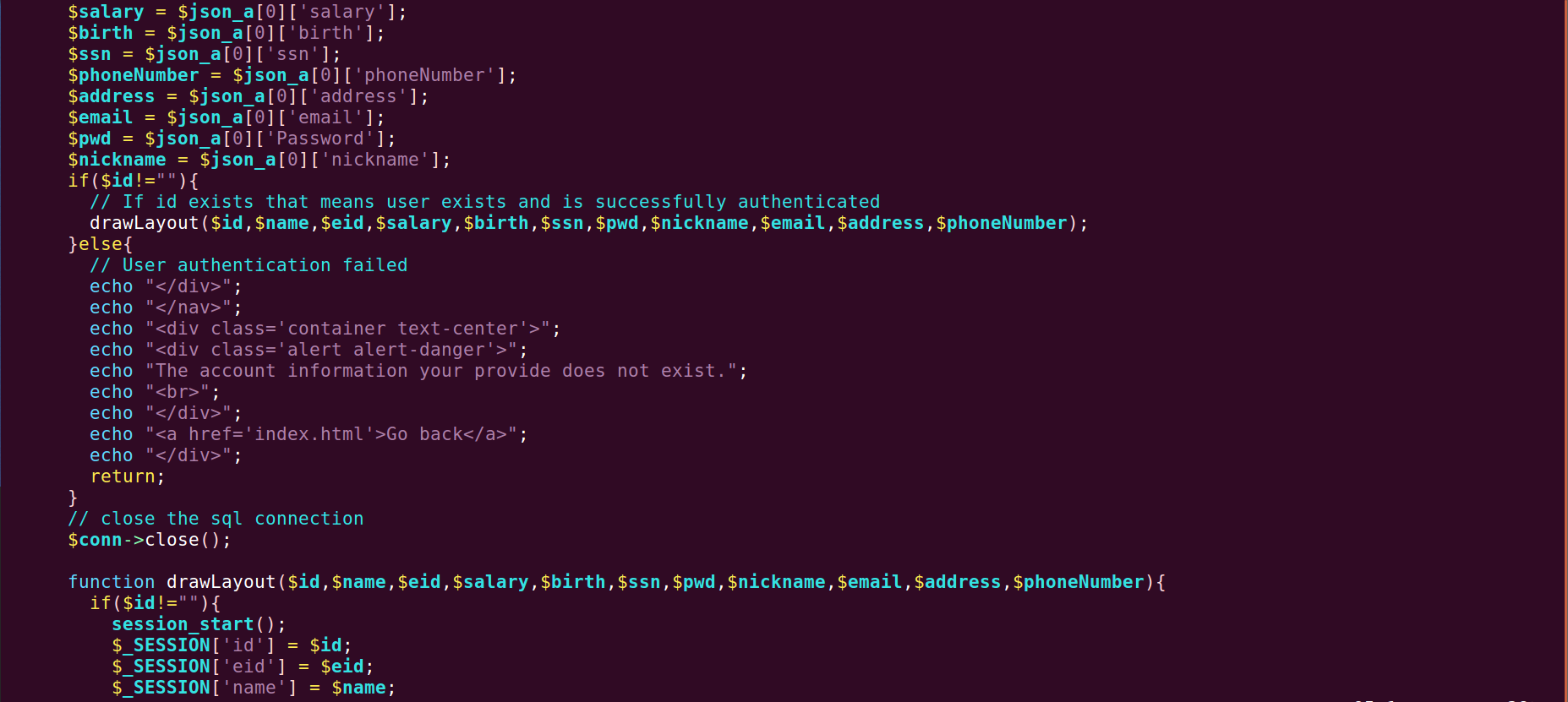


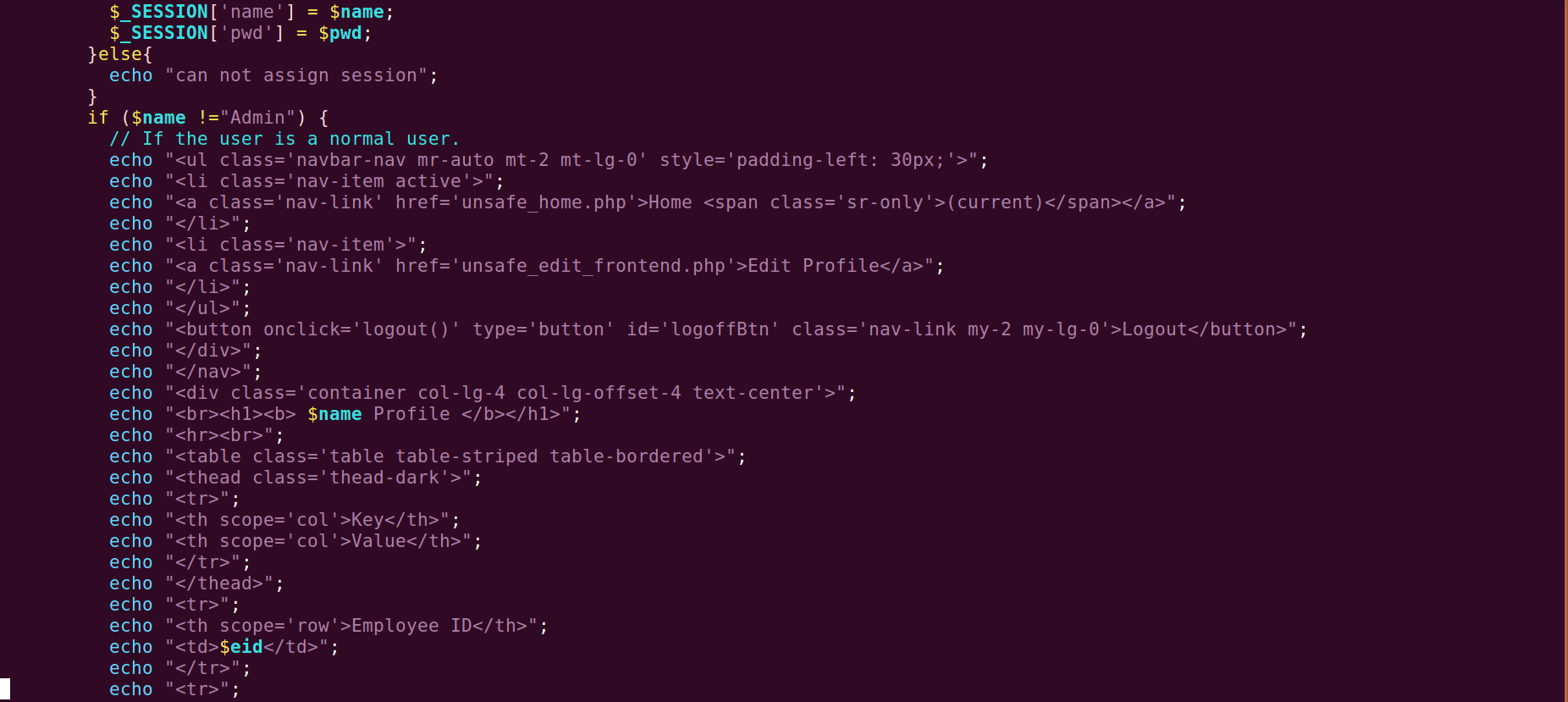
Em thấy, trang web này sử dụng ngôn ngữ php để viết và nội dụng bao gồm các hình bên dưới.

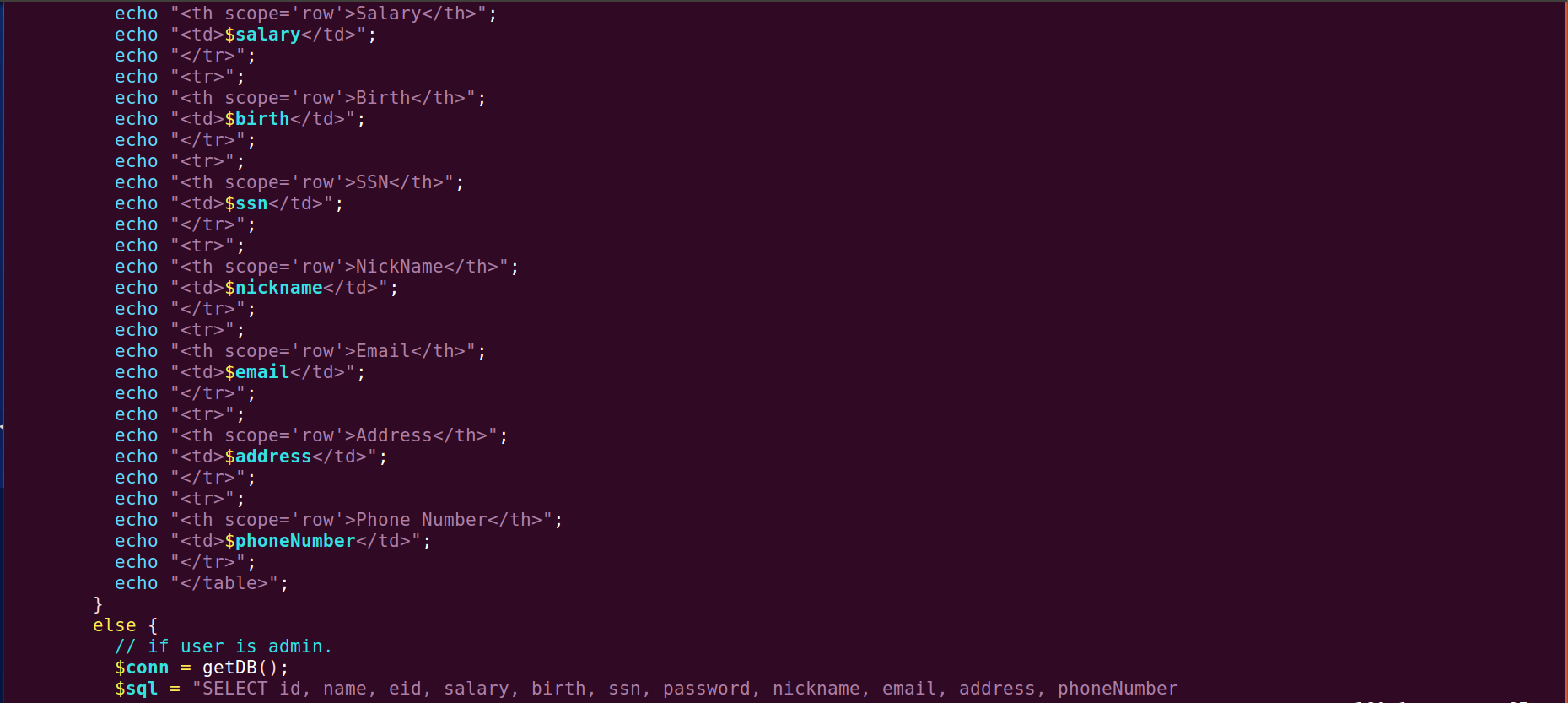


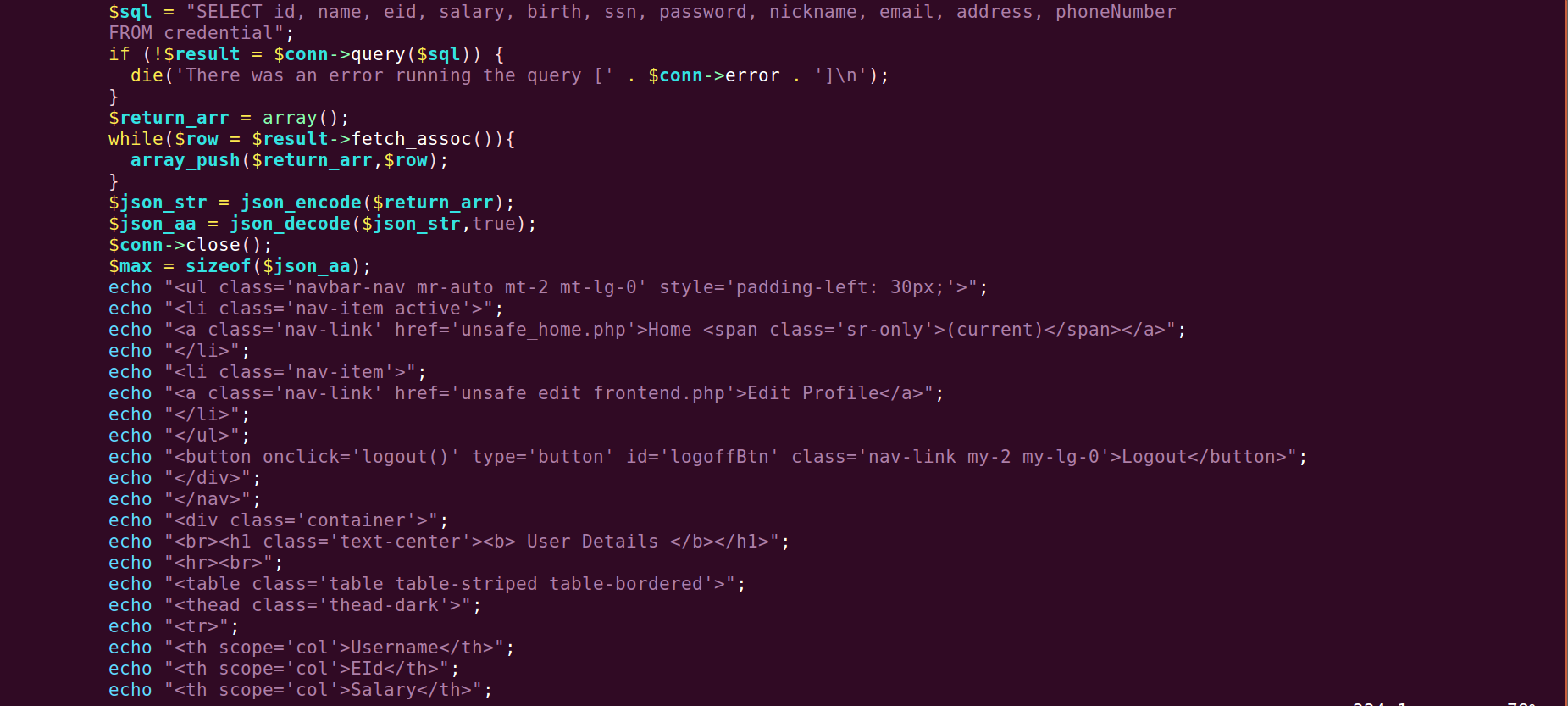


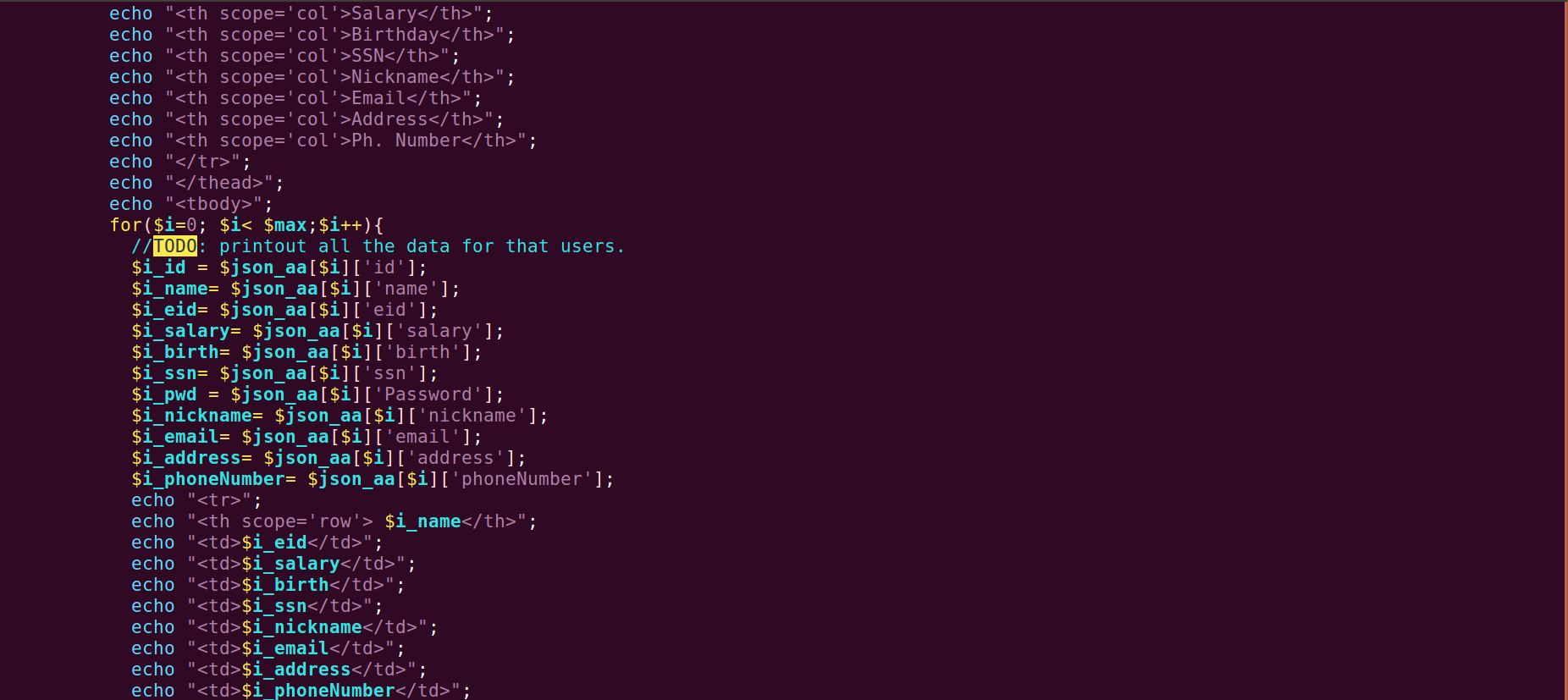






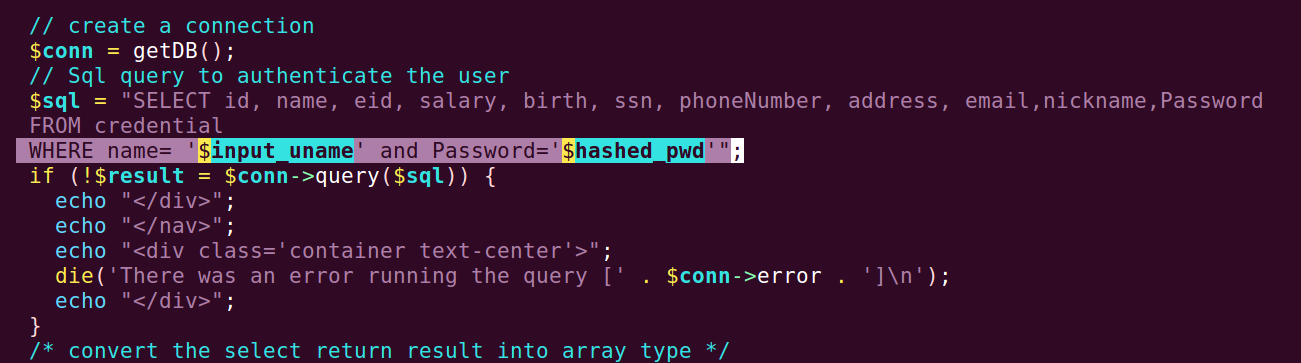








Trong đó, việc bảo mật lỏng lẻo được thể hiện qua dòng:***Where name = ‘$input\_name’ and Password=’$hashed\_pwd’***

******

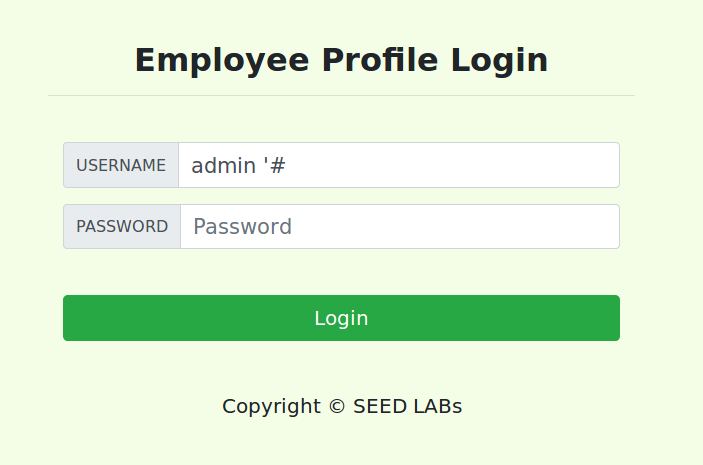
Nếu như ở biến “input\_name”, thông tin được nhập từ form không chỉ có thông tin người dùng mà còn chèn thêm code sql như ký tự để comment “#”, thì tự động, những đoạn code phía sau sẽ bị vô hiệu hóa

Ví dụ chèn thêm chuỗi [[ admin ‘ #]], đoạn code trở thành: *Where name = ‘admin’ #’and Password=’$hashed\_pwd’*

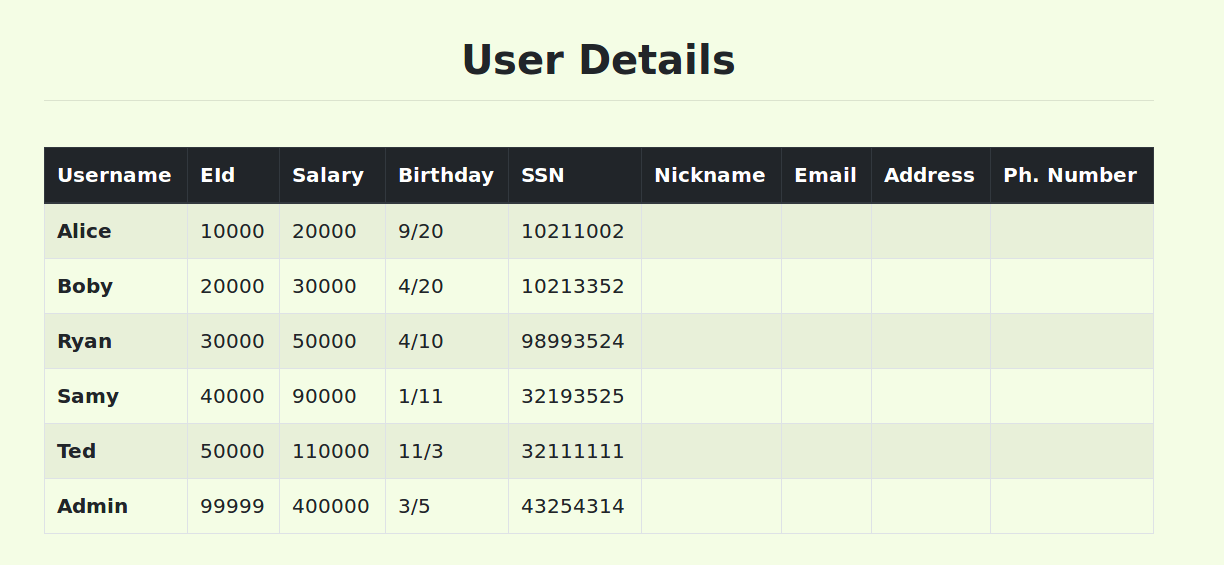
Từ ký tự “#” trở về sau, đoạn code sẽ trở thành một đoạn comment và không được thực thi và chỉ còn: *Where name = ‘admin’*

**4. Tiến hành SQLInjection trên trang web**

Truy cập vào website, chỉ cần nhập thông tin ở username: *admin ‘#*. Không cần quan tâm password là gì, chỉ cần đúng tên đăng nhập thì vẫn có thể đăng nhập vào website.



Và kết quả là em truy cập được vào trang quản lý với tài khoản của admin



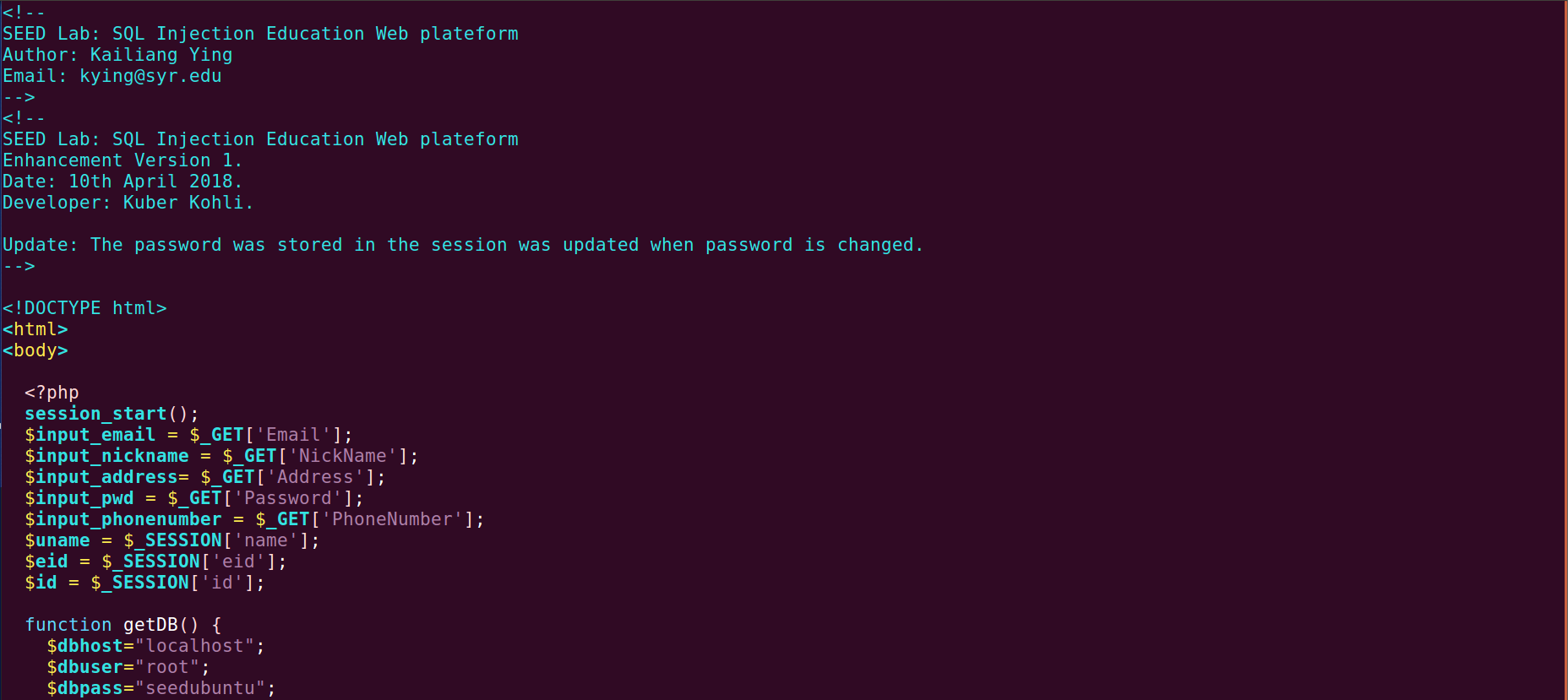
**5. SQL Injection bằng câu lệnh Update**

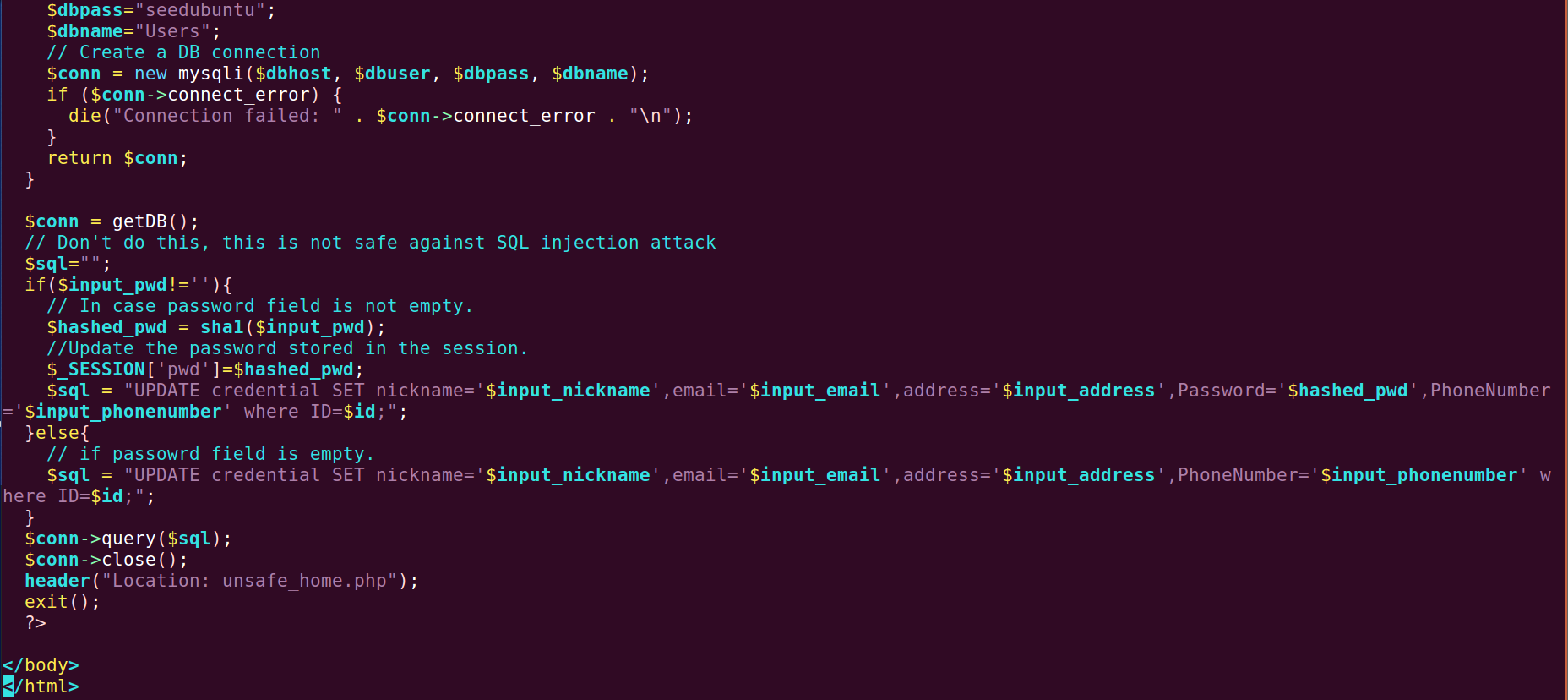
**5.1 Thay đổi lương của Alice**

Truy cập vào đường dẫn chứa code của trang chứa form edit ở thư mục SQLInjection bằng câu lệnh sau: *sudo vi unsafe\_edit\_backend.php*



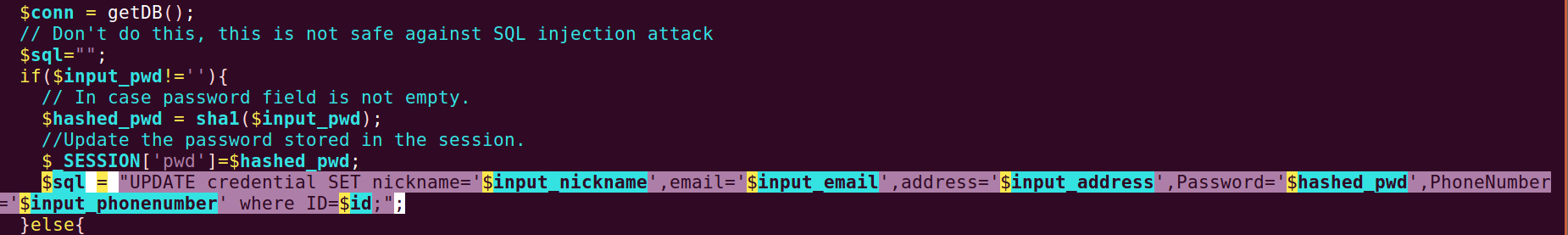
Nội dung của file backend.php như sau:





Em thấy, để có thể update được thông tin, trang web sử dụng câu lệnh

*UPDATE credential SET nickname=’$input\_nickname’, email=...*

**

Nhận thấy lỗ hổng thông qua câu lệnh SET với biến $input\_nickname, em có thể tấn công sql injection thông qua việc chèn một đoạn code tương tự như ở lúc đăng nhập. Nếu ngay trường thông tin “nickname”, em sử dụng dấu nháy đóng [[ ’ ]] và một câu lệnh tùy ý, kết thúc bằng [[ # ]], thì tự động, đoạn code của em sẽ được chạy

Ví dụ: em nhập vào form chỉnh sửa

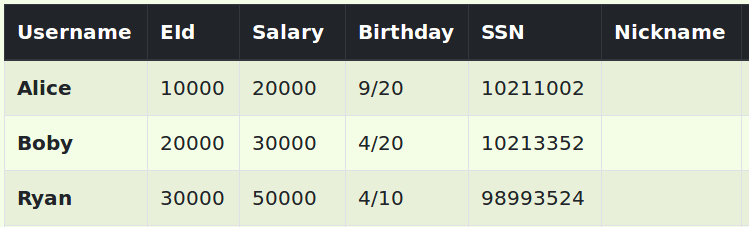
*’, salary=50000 where Name=Boby;#*

Đoạn code sẽ trở thành

*UPDATE credential SET nickname=‘’, salary=50000 where Name=‘Boby’;#’ email = ....*

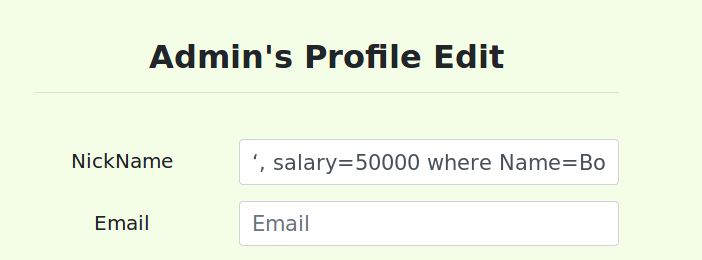
Như vậy, em đã có thể thay đổi trường thông tin salary thành 50000 của Alice, và đoạn code sau dấu “#” sẽ không được thực thi.

Ban đầu, trường thông tin Salary của Boby là 30000

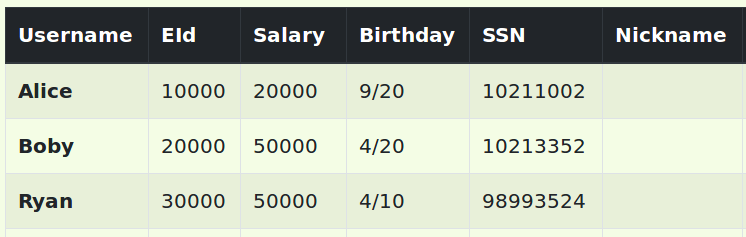


Truy cập vào form chỉnh sửa, tiến hành nhập chuỗi

*', salary=50000 where Name=‘Boby’;#*



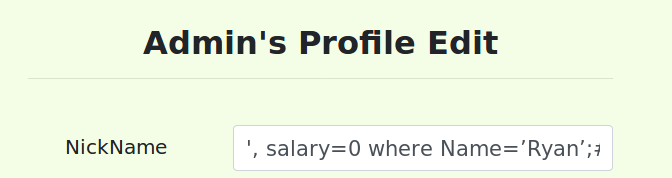
Trường thông tin Salary ( lương) của Boby đã được sửa thành 50000



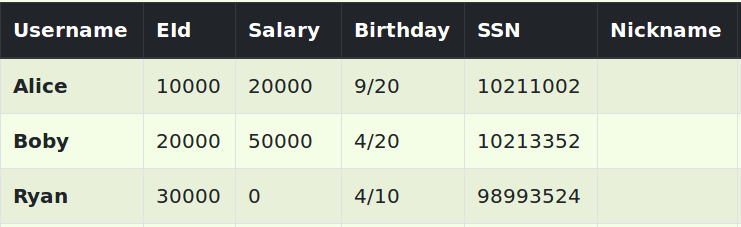
**5.2 Thay đổi lương của Ryan**

Tương tự như 5.1, nhập vào trường nickname chuỗi:

*', salary=0 where Name=’Ryan’;#*



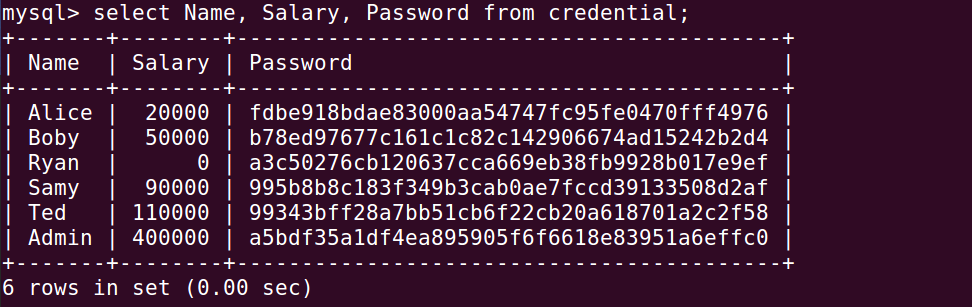
Kết quả là Salary của Ryan bị sửa thành 0



**5.3 Thay đổi mật khẩu của Boby**

Sử dụng Mysql, truy cập vào bảng credential và xem các thông tin Name, Salary, Password bằng câu lệnh: *select Name, Salary, Password from credential;.* Em thấy các trường thông tin Salary của Boby và Ryan đã bị sửa do tấn công sql injection ở *5.1* và *5.2*

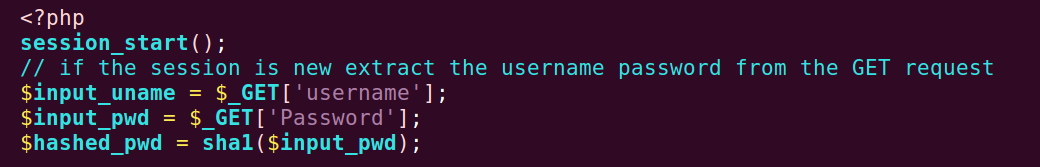
Nhận thấy trường thông tin Password đã bị mã hóa, và để có thể chỉnh sửa Password, em cũng phải mã hóa sẵn và sau đó mới có thể tấn công



Truy cập vào file unsafe\_home.php bằng câu lệnh: *sudo vi unsafe\_home.php*

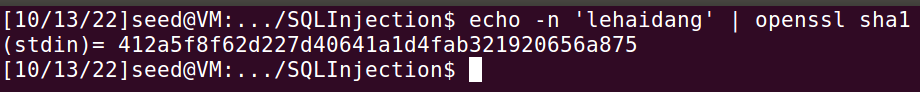


Có thể thấy, website mã hóa mật khẩu bằng thuật toán SHA1



Chuẩn bị sẵn một mật khẩu có tên ‘lehaidang, và sử dụng openssl đã được cài đặt sẵn để tiến hành mã hóa bằng câu lệnh: *echo -n ‘lehaidang’ | openssl sha1*

Mật khẩu được hash thành chuỗi: 412a5f8f62d227d40641a1d4fab321920656a875



Sau đó sử dụng kiểu tấn công SQL Injection như ở mục 5.1 và 5.2

*UPDATE credential SET nickname=’$input\_nickname’, email=...*

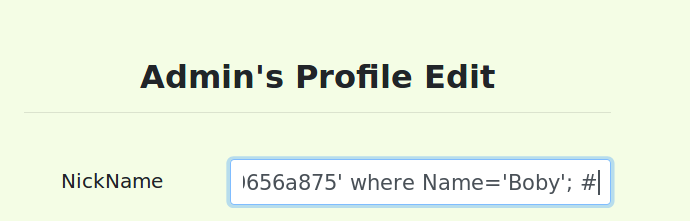
Chèn một chuỗi đã gắn password mã hóa của em vào $input\_nickname:

*', password='*412a5f8f62d227d40641a1d4fab321920656a875*' where Name = 'Boby';#*

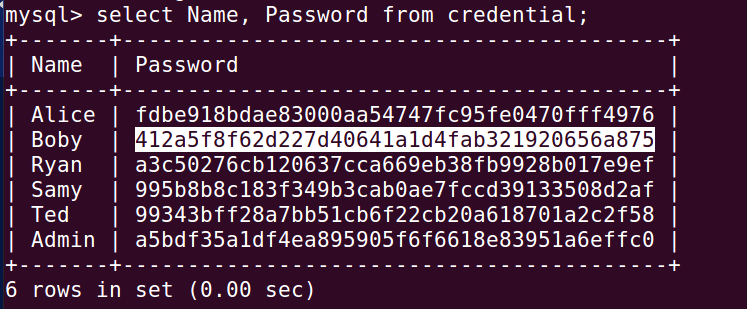
Chuỗi được thực thi sẽ trở thành

*UPDATE credential SET nickname=’’, password='*412a5f8f62d227d40641a1d4fab321920656a875*' where Name = 'Boby';#’ email=...*

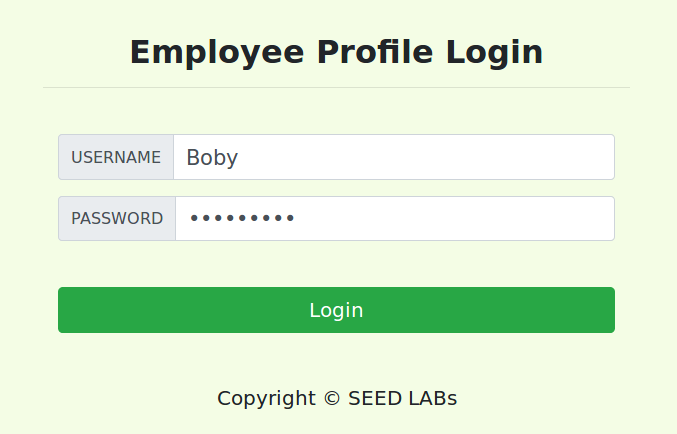
Tiến hành chèn chuỗi trên website:



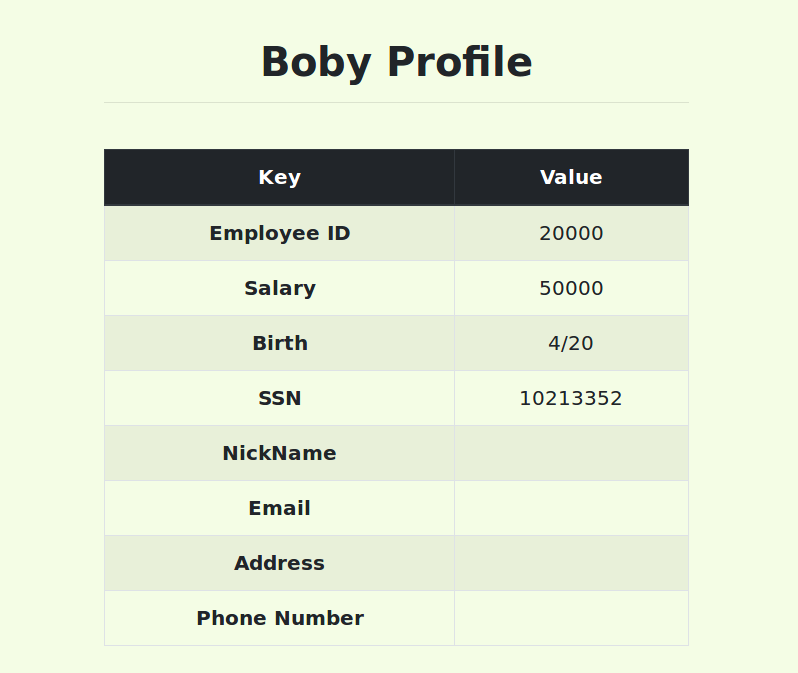
Sử dụng Mysql, xem thông tin Password của Boby đã bị đổi thành mật khẩu đã mã hóa của em bằng câu lệnh: *select Name, Password from credential;*



Tiến hành đăng nhập bằng tài khoản Boby với mật khẩu ‘khainguyen’



Em đã có thể truy cập dưới quyền của Boby



**6. Phòng chống SQL Injection**

Để bảo vệ bản thân trước SQL Injection, em có thể thực hiện các biện pháp sau.

* **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. Viết lại từ đầu vừa tốn thời gian vừa dễ sơ sót.
* **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.
* **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 DB:** Nếu chỉ truy cập dữ liệu từ một số bảng, hãy tạo một account trong DB, 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 xoá dữ liệu.
* **Backup dữ liệu thường xuyên:** Dữ liệu phải thường xuyên được backup để nếu có bị hacker xoá thì ta vẫn có thể khôi phục được