



3

Lab

Tấn công SQL Injection

Thực hành Bảo mật web và ứng dụng

Lưu hành nội bộ

A. TỔNG QUAN

A.1. Mục tiêu

- Tìm cách để khai thác lỗ hổng SQL injection để thấy được sự nguy hiểm.
- Làm chủ kỹ thuật để giúp chống lại tấn công SQL injection.

B. CHUẨN BỊ MÔI TRƯỜNG

Môi trường thực hành là một máy Linux có sẵn ứng dụng có lỗ hổng SQL Injection.

B.1. Chuẩn bị máy ảo có ứng dụng có lỗ hổng SQL Injection

B.1.1. Phần 1: Chuẩn bị máy ảo có cài Docker

▪ Cách 1: Tạo mới máy ảo tích hợp sẵn

Tạo một máy ảo với disk được cung cấp sẵn ở file **Seed-Ubuntu20.04.vmdk** ([Google Drive](#), [Digital Ocean](#)).

▪ Cách 2: Tùy chỉnh máy ảo sẵn có

- **Bước 1:** Tùy chỉnh máy ảo đã có sẵn để cài đặt **Docker** và **Docker compose**. Tham khảo các hướng dẫn: [Install Docker on Ubuntu](#), [Install Docker-compose on Ubuntu](#) (yêu cầu version 1.13.0 trở lên)
- **Bước 2:** Chỉnh sửa tập tin hosts như bên dưới ở các đường dẫn:
 - Với Windows: C:\windows\system32\drivers\etc\hosts
 - Với Linux và MacOS: /etc/hosts.

```
# For SQL Injection Lab
10.9.0.5      www.SeedLabSQLInjection.com
```

B.1.2. Phần 2: Cấu hình và cài đặt môi trường ứng dụng có lỗ hổng SQL Injection

GVTH cung cấp sẵn cho sinh viên tập tin nén **LabsetupSQLi.zip** chứa các file cần thiết để thiết lập môi trường cho kịch bản tấn công SQL Injection.

▪ **Bước 1:** Tải và giải nén tập tin, sau đó truy cập vào thư mục

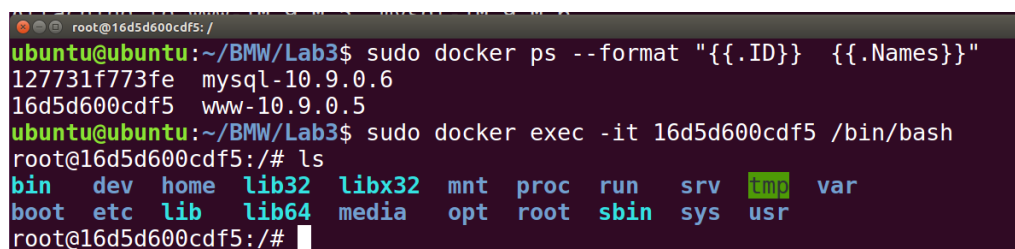
```
unzip LabsetupSQLi.zip
cd Labsetup
```

▪ **Bước 2:** Sử dụng file *docker-compose.yml* để tạo môi trường thực hành.

```
sudo docker-compose build # Build the container image
sudo docker-compose up    # Start the container
# Shut down the container when not used anymore
sudo docker-compose down
```

- Cách thức vào shell của một container

```
sudo docker ps --format "{{.ID}} {{.Names}}" # list all containers
sudo docker exec -it <id> /bin/bash        # Get the shell of given id
```



```
root@16d5d600cdf5: /
ubuntu@ubuntu:~/BMW/Lab3$ sudo docker ps --format "{{.ID}} {{.Names}}"
127731f773fe mysql-10.9.0.6
16d5d600cdf5 www-10.9.0.5
ubuntu@ubuntu:~/BMW/Lab3$ sudo docker exec -it 16d5d600cdf5 /bin/bash
root@16d5d600cdf5:/# ls
bin  dev  home  lib32  libx32  mnt  proc  run  srv  tmp  var
boot  etc  lib  lib64  media  opt  root  sbin  sys  usr
root@16d5d600cdf5:/#
```

▪ Bước 3: Kiểm tra môi trường

- Sau các bước cấu hình và chạy docker, trên máy Linux của sinh viên có một Ứng dụng Quản lý nhân sự, một ứng dụng web có URL www.SEEDLabSQLInjection.com. Đây là một ứng dụng quản lý nhân viên. Nhân viên có thể xem và cập nhật thông tin cá nhân của mình vào cơ sở dữ liệu thông qua ứng dụng web.
- Có 2 quyền chính trong ứng dụng: Administrator quản lý thông tin profile của các nhân viên khác, Employee có thể xem và cập nhật thông tin profile của mình.

User	UserName	Password
Admin	admin	seedadmin
Alice	alice	seedalice
Boby	boby	seedboby
Charlie	charlie	seedcharlie
Samy	samy	seedsamy

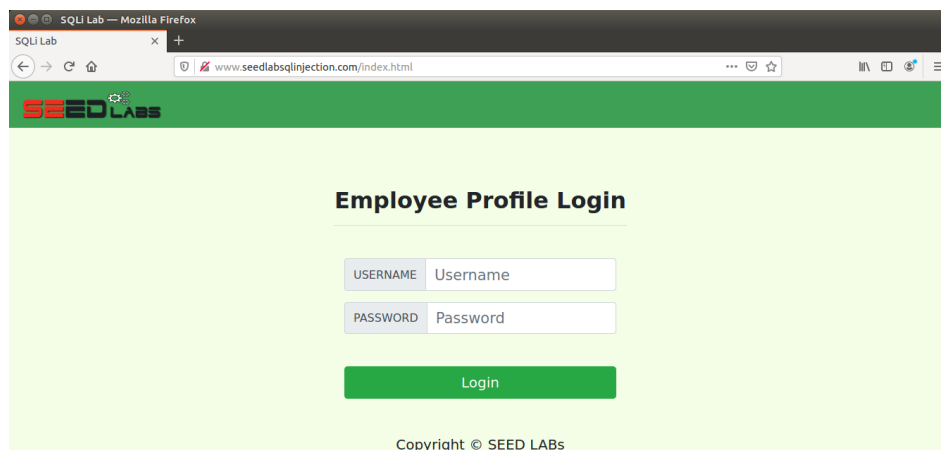
User	Employee ID	Password	Salary	Birthday	SSN	Nickname	Email	Address	Phone#
Admin	99999	seedadmin	400000	3/5	43254314				
Alice	10000	seedalice	20000	9/20	10211002				
Boby	20000	seedboby	50000	4/20	10213352				
Ryan	30000	seedryan	90000	4/10	32193525				
Samy	40000	seedsamy	40000	1/11	32111111				
Ted	50000	seedted	110000	11/3	24343244				

Một số cấu hình DNS đã được cấu hình sẵn trong môi trường thực hành.

URL	Thư mục
http://www.SEEDLabSQLInjection.com	/var/www/SQL_Injection/

Lưu ý khi sử dụng các container:

- Khi thực hiện tấn công, sinh viên cần đảm bảo container liên quan đến kịch bản tấn công đó đang chạy. Sinh viên có thể truy cập vào các URL tương ứng để kiểm tra ứng dụng đã chạy thành công trên môi trường chưa.
- Sinh viên nên dừng các container khác không liên quan đến bài lab để tránh xung đột nếu có (ví dụ xung đột với môi trường đã set up cho Lab 2).



C. THỰC HÀNH

C.1. Sử dụng MySQL console

- **Mục tiêu:** làm quen với câu lệnh SQL qua cơ sở dữ liệu được cung cấp.

Ứng dụng được cài đặt sử dụng Cơ sở dữ liệu MySQL, trong đó có một database tên **sqllab_users** chứa table tên **credential**. Table này lưu thông tin cá nhân (như: name, password, salary, ssn,...) của nhân viên. Administrator được phép thay đổi thông tin profile của các nhân viên, nhưng nhân viên chỉ có thể thay đổi thông tin của chính mình.

MySQL đã được cài đặt sẵn trên một container riêng biệt sau khi cài đặt xong môi trường. Với Tên đăng nhập là **root**, mật khẩu **dees**.

- **Bước 1:** Truy cập vào container của MySQL, đăng nhập vào MySQL từ console

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

```
ubuntu@ubuntu: ~/BMW/Lab3$ sudo docker ps --format "{{.ID}} {{.Names}}"
127731f773fe mysql-10.9.0.6
16d5d600cdf5 www-10.9.0.5
ubuntu@ubuntu:~/BMW/Lab3$ sudo docker exec -it 127731f773fe /bin/bash
root@127731f773fe:/# 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 18
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>
```

- **Bước 2:** Vào cơ sở dữ liệu **sqllab_users**

Sau khi đăng nhập, sử dụng lệnh sau để vào cơ sở dữ liệu có sẵn **sqllab_users**:

```
mysql> use sqllab_users;
```

```
mysql> use sqllab_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>
```

- **Bước 3:** Hiển thị các bảng có trong cơ sở dữ liệu

Để hiển thị những table có trong cơ sở dữ liệu **sqllab_users**, có thể sử dụng lệnh sau để hiện tất cả table của cơ sở dữ liệu đang chọn.

```
mysql> show tables;
```

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

○ **Bước 4:** Xem các cột có trong bảng

Sử dụng lệnh sau để xem các cột có thể truy vấn trong bảng

```
mysql> desc credential;
```

```
mysql> desc credential;
```

Field	Type	Null	Key	Default	Extra
ID	int unsigned	NO	PRI	NULL	auto_increment
Name	varchar(30)	NO		NULL	
EID	varchar(20)	YES		NULL	
Salary	int	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 (0.06 sec)

○ **Bước 5:** Đọc thông tin trong bảng

Yêu cầu 1. Với danh sách các cột trong bảng, sinh viên dùng một lệnh SQL để hiện tất cả thông tin profile của nhân viên Alice. Viết lệnh và chụp lại màn hình kết quả.

Kết quả mong muốn:

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
1	Alice	10000	20000	9/20	10211002					fdbe918bdae83000aa54747fc95fe0470fff4976

1 row in set (0.00 sec)

C.2. Thực hiện tấn công SQL Injection trên câu lệnh SELECT

SQL Injection là một kỹ thuật mà qua đó người tấn công có thể thực hiện những câu lệnh SQL độc (xem như những payload độc hại). Qua những câu lệnh SQL độc hại này, người tấn công có thể đánh cắp thông tin từ cơ sở dữ liệu nạn nhân, hoặc tệ hơn có thể thay đổi cơ sở dữ liệu. Ứng dụng web quản lý nhân viên có những lỗ hổng SQL Injection được bắt chước những lỗi thông thường của các lập trình viên.

Khi vào trang www.SEEDLabSQLInjection.com/index.html, người dùng được yêu cầu cung cấp Username và Password để đăng nhập, chỉ những nhân viên biết username và mật khẩu của họ mới được phép xem và cập nhật thông tin của mình.

Ở phần này, sinh viên tìm cách đăng nhập vào ứng dụng mà không cần biết trước bất kỳ thông tin gì về username hay password.

Gợi ý: File `unsafe_home.php` trong thư mục `/var/www/SQL_Injection` bên dưới được dùng để xử lý xác thực người dùng.

```
$input_username = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);
$conn = getDB();
// Sql query to authenticate the user
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname,
        Password
        FROM credential
        WHERE name='$input_username' and Password='$hashed_pwd'";
$result = $conn->query($sql)
// The following is psuedo code
if(name=='admin'){
    return All employees information.
} else if(name!=NULL){
    return employee information.
} else {
    authentication fails.
}
```

Phân tích: Câu lệnh SQL trên hiện thông tin cá nhân của nhân viên gồm id, name, salary, ssn,... từ bảng **credential**. Biến **`input_username`** và **`input_pwd`** chứa chuỗi được nhập từ người dùng qua trang đăng nhập, được dùng để dựng câu lệnh SQL để kiểm tra có hay không record khớp với username và password. Nếu có, người dùng xác thực thành công và được cung cấp thông tin tương ứng. Nếu không thì xác thực không thành công.

C.2.1. Tấn công SQL Injection từ trang web

Yêu cầu 2. Sinh viên sử dụng giao diện web, thực hiện đăng nhập vào với vai trò của administrator để xem thông tin của tất cả nhân viên. *Giả sử biết administrator có username dạng 'admin' nhưng không biết password.* Xác định nội dung cần nhập ở form đăng nhập để tạo câu lệnh SQL phù hợp.

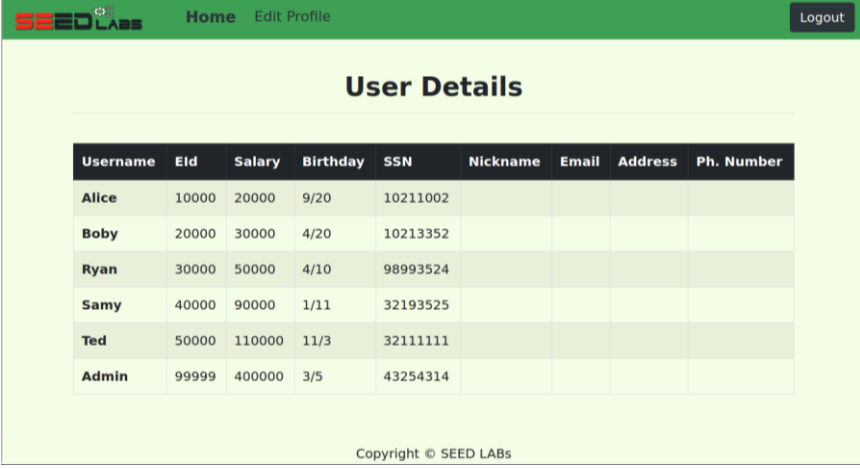
- **Gợi ý:** Câu lệnh SQL cần tạo sẽ có đặc điểm và có dạng như bên dưới:
 - Để thấy password sẽ được hash trước khi dùng để dựng câu lệnh SQL. Do đó, ta sẽ lợi dụng trường username để chèn những nội dung mong muốn vào lệnh SQL.
 - Trong mọi trường hợp nhập username, lệnh SQL này sẽ chỉ dựa trên điều kiện trên.
 - Cần bỏ qua các điều kiện truy vấn phía sau (như điều kiện kiểm tra password màu xanh dương) bằng cách sử dụng ký hiệu comment # trong MySQL.
 - Cần nhập gì để điền phần còn thiếu màu đỏ trong câu truy vấn?

```
SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname,
Password
```

```
FROM credential
WHERE name='<some thing here to query admin> #' and Password='$hashed_pwd'
```

- **Kết quả mong đợi**

Khi đăng nhập vào tài khoản administrator thành công sẽ có được giao diện xem tất cả thông tin của các nhân viên như bên dưới.



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				

C.2.2. Tấn công SQL Injection từ cửa sổ dòng lệnh

Mục đích tương tự **Yêu cầu 2.1**, tuy nhiên sinh viên không sử dụng giao diện web mà sử dụng command line để gửi request đăng nhập.

Gợi ý: Sinh viên có thể sử dụng công cụ **curl** để gửi HTTP request.

Cách sử dụng:

```
curl '<URL>'
```

Truyền tham số cho GET request: các tham số sẽ nằm trên URL, mỗi tham số phân cách với nhau bằng dấu **&** và có định dạng **name=value**. Ví dụ bên dưới có 2 tham số SUID và Password được truyền cho URL:

```
curl 'www.SeedLabSQLInjection.com/index.php?SUID=10000&Password=111'
```

Lưu ý: Nếu cần sử dụng những ký tự đặc biệt trong các tham số của request, sinh viên cần mã hóa chúng sang dạng thích hợp để tránh tác động đến quá trình phân tách tham số. Ví dụ: dấu nháy đơn ' sẽ thay thế bằng %27, khoảng trắng sẽ thay thế bằng %20.

Yêu cầu 3. Sinh viên sử dụng cửa sổ dòng lệnh, thực hiện đăng nhập vào ứng dụng với vai trò của administrator để xem thông tin của tất cả nhân viên.

Gợi ý các bước:

- **Bước 1: Xác định URL sẽ tấn công**

File xử lý tác vụ đăng nhập là **unsafe_home.php**, do đó cần tấn công URL của file này: http://www.seedlabsqlinjection.com/unsafe_home.php

- **Bước 2: Chuẩn bị chuỗi SQL Injection**

Tham số tương tự như cách tấn công ở **Yêu cầu 2** (username và Password), tuy nhiên cần thay thế các ký tự đặc biệt như dấu nháy đơn, khoảng trắng trong chuỗi.

• Bước 3: Thực hiện tấn công

Nhập lệnh sau vào terminal với <url> là chuỗi xây dựng ở bước 1.

```
curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?<Tham số>'
```

Quan sát kết quả trên màn hình khi thực hiện tấn công bằng command line, nếu kết quả trả về chứa thông tin của nhiều nhân viên thì tấn công thành công.

C.2.3. Tấn công SQL Injection thực thi thêm 1 lệnh SQL

Trong 2 tấn công trên, chúng ta chỉ đánh cắp được thông tin từ cơ sở dữ liệu; yêu cầu này tìm cách chỉnh sửa cơ sở dữ liệu dựa trên cùng lỗ hổng trang đăng nhập.

Ý tưởng: sử dụng tấn công SQL injection để chuyển một câu lệnh SQL thành 2 câu lệnh, với câu lệnh thứ 2 là lệnh cập nhật hoặc xóa. Trong SQL, dấu ; được dùng để tách 2 câu lệnh SQL. Thử tấn công để xóa một record từ cơ sở dữ liệu và mô tả quan sát.

a) Tấn công trên MySQL console

Yêu cầu 4. Sử dụng dấu ; để thực hiện 2 câu lệnh SQL trong cùng 1 truy vấn trên MySQL console: câu lệnh 1 – SELECT để truy vấn thông tin, câu lệnh 2 – UPDATE hoặc DELETE một record nào đó.

- **Bước 1:** Sinh viên tạo thêm 1 record với **thông tin của các thành viên trong nhóm** trong **credential** dùng để thử nghiệm.
- **Bước 2:** Mở rộng thêm câu lệnh xóa cho câu lệnh SELECT để xóa record đã thêm ở Bước 1. Hoàn thành phần còn thiếu trong **<?>**

```
SELECT * FROM credential WHERE eid='' or name='admin'; DELETE FROM credential WHERE <?>;
```

- **Bước 3:** Quan sát thông tin trong bảng trước khi thực hiện lệnh. Thực hiện câu SQL trên với MySQL console.
- **Bước 4:** So sánh kết quả sau khi thực thi lệnh với trước khi thực thi.

b) Tấn công trên ứng dụng web

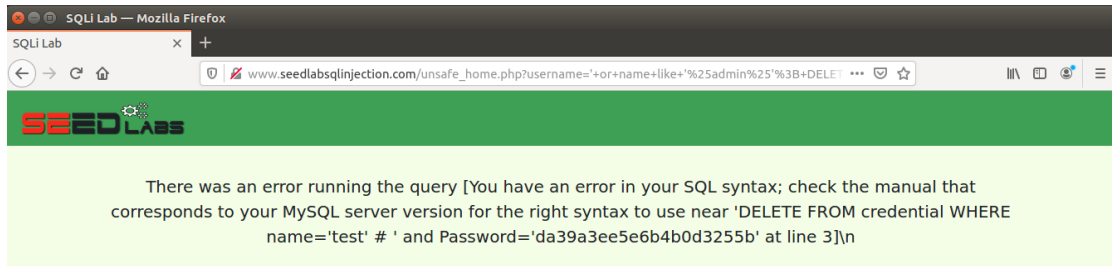
Thực hiện tương tự các bước trên khi tấn công trên MySQL console

- **Bước 1:** Mở trang <http://www.seedlabsqlinjection.com/> và nhập nội dung giá trị dùng để tấn công vào ô username.

Hoàn thành nội dung dùng để tấn công bên dưới.

```
?username=admin'; DELETE FROM credential WHERE <?>
```

- **Bước 2:** Nhấn Get Information. Kết quả:

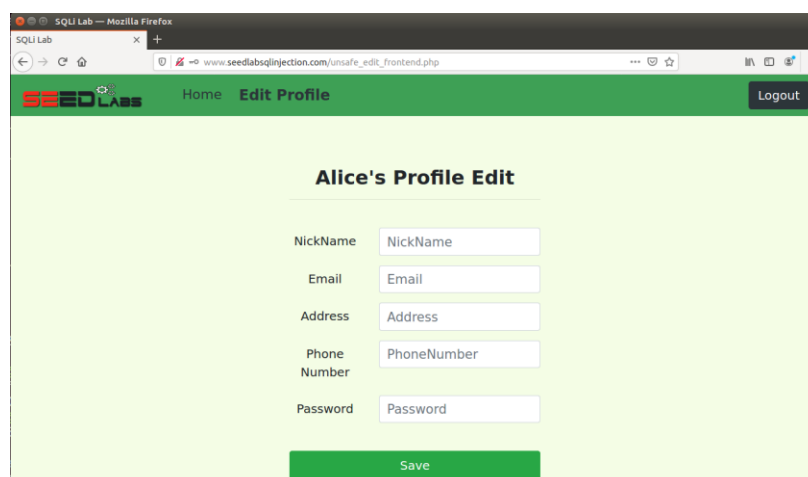


Yêu cầu 5. Lý giải lý do vì sao thực hiện tấn công tương tự trên giao diện web không thành công.

Gợi ý: xem lại các hàm có trong đoạn mã nguồn xử lý SELECT ở trên.

C.3. Tấn công SQL Injection trên câu lệnh UPDATE

Một lỗ hổng SQL injection xảy ra ở câu lệnh Update sẽ càng nguy hiểm hơn vì người tấn công có thể sử dụng lỗ hổng để chỉnh sửa cơ sở dữ liệu. Trong ứng dụng quản lý nhân viên, có một trang **Edit Profile** cho phép nhân viên cập nhật thông tin profile của họ, gồm: nickname, email, address, phone number và password. Chỉ nhân viên đã đăng nhập mới có thể vào trang này.



Khi nhân viên cập nhật thông tin của họ qua trang **Edit Profile**, câu truy vấn SQL Update sau sẽ được thực thi. Mã PHP thực thi trong tập tin **unsafe_edit_backend.php** tại thư mục `/var/www/SQL_Injection` được dùng để cập nhật thông tin profile của nhân viên.

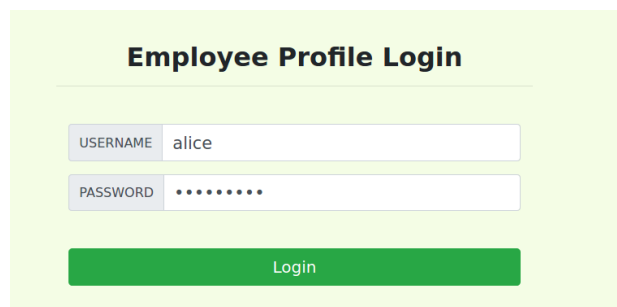
```
$hashed_pwd = sha1($input_pwd);  
$sql = "UPDATE credential SET  
    nickname='$input_nickname',  
    email='$input_email',  
    address='$input_address',  
    Password='$hashed_pwd',  
    PhoneNumber='$input_phonenumber'  
WHERE ID=$id;";  
$conn->query($sql);
```

C.3.1. Tấn công SQL Injection trên câu lệnh Update để chỉnh sửa lương

Như đã thấy trong trang **Edit Profile**, nhân viên chỉ có thể cập nhật nickname, email, address, phone number, password; họ không có quyền để thay đổi lương. Chỉ quản trị viên mới được phép thực hiện thay đổi lương. Tuy nhiên, giả sử một nhân viên biết được lương được lưu trong cột **Salary**, và xác định mục tiêu là tìm cách tăng lương cho mình qua trang **Edit Profile**.

Yêu cầu 6. Với vai trò là 1 nhân viên, sử dụng SQL Injection trên câu lệnh Update để tìm cách chỉ chỉnh sửa lương của chính mình.

- **Bước 1:** Đăng nhập vào 1 tài khoản nhân viên bất kỳ, ví dụ Alice.



The form is titled "Employee Profile Login". It contains two input fields: "USERNAME" with the value "alice" and "PASSWORD" with masked characters ".....". Below the fields is a green "Login" button.

Thông tin hiện tại của Alice, có lương là 20000.

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

- **Bước 2:** Sử dụng chức năng chỉnh sửa profile bằng cách bấm vào nút **Edit Profile**.

Alice's Profile Edit

NickName

Email

Address

Phone Number

Password

Nhập nội dung để thực hiện tấn công qua việc chỉnh sửa thông tin rồi nhấn **Save**. Câu lệnh SQL được tạo từ input của form cần chỉnh sửa luôn cả cột **Salary**, dưới dạng:

```
UPDATE credential SET nickname=..., email=..., address=..., Password=...,
PhoneNumber=..., Salary=... WHERE ID = <id>
```

- **Bước 3:** Xem kết quả sau khi chỉnh sửa. Lưu ý Alice chỉ muốn chỉnh lương của chính mình.

Alice Profile

Key	Value
Employee ID	10000
Salary	1000000
Birth	9/20
SSN	10211002
NickName	alice
Email	
Address	
Phone Number	

C.3.2. Tấn công SQL Injection trên câu lệnh Update để sửa mật khẩu của người khác

Sử dụng lỗ hổng tương tự trong câu lệnh Update ở trên, 1 nhân viên cũng có thể thay đổi dữ liệu của người dùng khác.

Yêu cầu 7. Với vai trò là 1 nhân viên, sử dụng SQL Injection trên câu lệnh Update để tìm cách chỉnh sửa mật khẩu của 1 nhân viên khác có tên Ryan.

Gợi ý:

- Lệnh SQL cần tạo sẽ có dạng:

```
UPDATE credential SET ..., Password=<new_password> WHERE name='Ryan'
```

- Lưu ý: cơ sở dữ liệu lưu giá trị hash của mật khẩu thay vì chuỗi mật khẩu rõ ràng. Sinh viên có thể xem lại mã nguồn **unsafe_edit_backend.php** để thấy cách mật khẩu được lưu. Mã hash của mật khẩu được dùng để tạo câu lệnh SQL.

Hướng dẫn:

- **Bước 1:** Tạo chuỗi hash của mật khẩu. Sử dụng lệnh command line sau:

```
echo -n "new password" | shasum
```

```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ echo -n "wonderful" | shasum
51f90c38d7935a8f617aeb3aae618937eb6bc34b -
ubuntu@ubuntu:~$
```

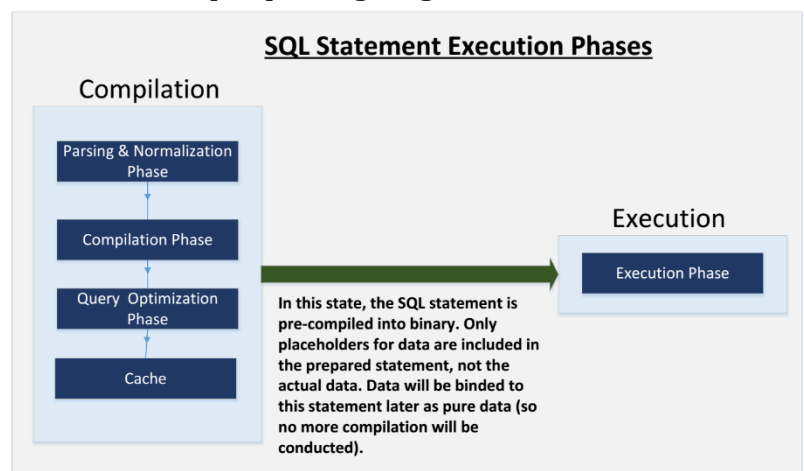
- **Bước 2:** Thực hiện tấn công tương tự **Yêu cầu 3.1** để thay đổi trường Password của tài khoản Ryan.
- Đăng nhập vào 1 tài khoản bất kỳ (khác Ryan) và sử dụng chức năng Edit profile.
- Thêm phần lệnh SQL thay đổi password vào 1 trong các trường của form edit profile.
- Làm thế nào để bỏ đi điều kiện hiện tại của lệnh SQL là WHERE ID= <id> và thay bằng WHERE name='Ryan'?
- **Bước 3:** Đăng nhập vào tài khoản của Ryan bằng mật khẩu mới để kiểm tra kết quả.

C.4. Biện pháp ngăn chặn – prepared statement

Vấn đề cơ bản của lỗ hổng SQL Injection là vấn đề tách biệt giữa mã nguồn và dữ liệu người dùng nhập vào. Khi xây dựng câu lệnh SQL, chương trình (PHP) biết được phần nào là dữ liệu, phần nào là mã nguồn. Tuy nhiên, khi câu lệnh SQL được tạo và gửi đến hệ quản trị cơ sở dữ liệu có thể có sự nhập nhằng, góc nhìn của trình thông dịch SQL có thể khác nguyên gốc được thiết lập bởi lập trình viên. Để giải quyết vấn đề này, cách bảo mật nhất là sử dụng prepared statement, nhằm nhất quán góc nhìn của trình thông dịch SQL và lập trình viên.

Khi SQL Server nhận một câu truy vấn, quy trình thực thi gồm nhiều bước:

- Giai đoạn phân tích và chuẩn hóa: kiểm tra cú pháp và ngữ nghĩa.
- Giai đoạn thông dịch: các từ khóa (như SELECT, FROM, UPDATE,...) được chuyển thành định dạng máy tính có thể hiểu.
- Giai đoạn tối ưu câu truy vấn: lựa chọn giải pháp tốt nhất để thực thi câu truy vấn. Giải pháp này được lưu trong bộ nhớ đệm, để có thể được sử



dụng trong những lần truy vấn kế tiếp.

- Giai đoạn thực thi để thực hiện truy vấn.

Prepared statement nằm ở sau bước biên dịch nhưng trước bước thực thi. Ở đó, prepared statement đã trải qua bước biên dịch và chuyển thành câu truy vấn đã được xử lý với dữ liệu trống. Để chạy câu truy vấn này, dữ liệu cần được cung cấp nhưng sẽ không đi qua bước biên dịch; thay vào đó, chúng được thêm trực tiếp vào câu truy vấn đã xử lý và được gửi đến nơi thực thi. Vì vậy, thậm chí trong trường hợp dữ liệu có chứa mã SQL nhưng không đi qua bước biên dịch thì mã nguồn cũng được xem như một phần của dữ liệu mà không có ý nghĩa đặc biệt gì khác. Đây là cách prepared statement giúp phòng tránh tấn công SQL injection.

Ví dụ đoạn mã dưới đây có thể xảy ra tấn công SQL Injection ở câu lệnh SELECT.

```
$conn = getDB();
$sql = "SELECT name, local, gender
      FROM USER_TABLE
      WHERE id = '$id' AND password = '$pwd' ";
$result = $conn->query($sql)
```

Áp dụng prepared statement để viết lại đoạn mã trên nhằm tránh lỗ hổng SQL Injection:

```
$conn = getDB();
$stmt = $conn->prepare("SELECT name, local, gender
                      FROM USER_TABLE
                      WHERE id = ? and password = ? ");
// Bind parameters to the query
$stmt->bind_param("is", $id, $pwd);
$stmt->execute();
$stmt->bind_result($bind_name, $bind_local, $bind_gender);
$stmt->fetch();
```

Sử dụng cơ chế prepared statement, ta thực hiện 2 bước:

- **Bước 1:** tạo và gửi phần mã nguồn (câu lệnh SQL) mà không có dữ liệu thực tế với **prepare()**. Trong lệnh SQL, vị trí sẽ có dữ liệu được thay thế bằng dấu ?.
- **Bước 2:** Gửi dữ liệu đến hệ quản trị cơ sở dữ liệu sử dụng **bind_param()**, kết nối dữ liệu với các dấu ? tương ứng của lệnh SQL. Trong hàm **bind_param()**, mỗi ký tự trong chuỗi tham số đầu tiên lần lượt chỉ kiểu dữ liệu của các tham số phía sau: "i" nghĩa là dữ liệu trong \$id là loại số nguyên (integer) và "s" nghĩa là dữ liệu \$pwd là loại chuỗi (string).

Hệ quản trị cơ sở dữ liệu sẽ xem mọi thứ được gửi đến ở bước này là dữ liệu được và không có bất kỳ mã nguồn nào.

Yêu cầu 8. Sử dụng cơ chế prepared statement để sửa những lỗ hổng SQL Injection được khai thác ở các câu trước ở trang

<http://www.seedlabsqlinjection.com/defense>

- **Bước 1:** Thực hiện chỉnh sửa mã trong tập tin **unsafe.php** trong thư mục **/var/www/SQL_Injection/defense** sử dụng prepared statement.
- **Bước 2:** Thực hiện lại **Yêu cầu 2** với trang này và ghi lại kết quả.

D. YÊU CẦU & ĐÁNH GIÁ

- Sinh viên tìm hiểu và thực hành theo hướng dẫn, theo nhóm tối đa 2 sinh viên.
- Nộp báo cáo kết quả gồm chi tiết những việc bạn đã quan sát và thực hiện kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).

Báo cáo:

- Trình bày trong file **.PDF**.
- Đặt tên theo định dạng: [Mã lớp]-LabX_MSSV1-MSSV2-MSSV3.
Ví dụ: [NT213.L21.ATCL.1]-Lab3_18520xxxx_18520yyy_18520zzzz.
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Bài sao chép, trễ, ... sẽ được xử lý tùy mức độ vi phạm.

HẾT