

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**  
**KHOA HỆ THỐNG THÔNG TIN**

-----&-----



**ĐỒ ÁN MÔN HỌC**  
**AN TOÀN VÀ BẢO MẬT HỆ THỐNG THÔNG TIN**

**ĐỀ TÀI SỐ 8**  
**TÌM HIỂU SQL INJECTION**

**Giảng viên:** ThS. Hà Lê Hoài Trung

**Lớp:** IS335.O12.HTCL

**Thành viên:**

Trần Thị Kim Anh – 21520596

Lê Thị Lệ Trúc – 21521586

Lê Minh Chánh – 21521882

*Thành phố Hồ Chí Minh, tháng 12 năm 2023*

## LỜI CẢM ƠN

Đầu tiên, nhóm chúng em xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới giảng viên Hà Lê Hoài Trung – người đã giảng dạy và chia sẻ rất nhiều kiến thức cũng như các ví dụ thực tiễn trong các bài giảng. Thầy đã hướng dẫn cho chúng em làm đề tài, sửa chữa và đóng góp nhiều ý kiến quý báu giúp chúng em hoàn thành tốt báo cáo môn học của mình.

Bộ môn An toàn và Bảo mật Hệ thống Thông tin là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Tuy nhiên, do vốn kiến thức chuyên môn còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều bỡ ngỡ. Mặc dù chúng em đã cố gắng hết sức nhưng chắc chắn bài báo cáo khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, chúng em rất mong nhận được sự góp ý, chỉ bảo thêm của Thầy nhằm hoàn thiện những kiến thức của mình để nhóm chúng em có thể dùng làm hành trang thực hiện tiếp các đề tài khác trong tương lai cũng như là trong học tập và làm việc sau này.

Một lần nữa, nhóm chúng em xin gửi đến Thầy, bạn bè lời cảm ơn chân thành và tốt đẹp nhất!

**Thành phố Hồ Chí Minh, tháng 12 năm 2023**

Nhóm sinh viên thực hiện

Trần Thị Kim Anh

Lê Thị Lệ Trúc

Lê Minh Chánh

## NHẬN XÉT CỦA GIẢNG VIÊN

This image shows a full page of white paper with horizontal dotted lines, typical of primary school writing paper. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

## MỤC LỤC

<b>LỜI CẢM ƠN .....</b>	<b>2</b>
<b>NHẬN XÉT CỦA GIẢNG VIÊN .....</b>	<b>3</b>
<b>DANH MỤC HÌNH ẢNH.....</b>	<b>7</b>
<b>PHÂN CÔNG CÔNG VIỆC .....</b>	<b>9</b>
<b>Phần 1. SQL Injection .....</b>	<b>10</b>
<b>1.1. SQL Injection.....</b>	<b>10</b>
<b>1.1.1. Khái niệm .....</b>	<b>10</b>
<b>1.1.2. Nguyên nhân .....</b>	<b>11</b>
<b>1.1.3. Sự nguy hiểm của SQL Injection .....</b>	<b>11</b>
<b>1.1.4. SQLi hoạt động như thế nào? .....</b>	<b>12</b>
<b>1.2. Phân loại các kiểu tấn công của SQL Injection .....</b>	<b>14</b>
<b>1.2.1. In-band SQL Injection (Classic SQLi) .....</b>	<b>14</b>
1.2.1.1. Error-based SQLi.....	14
1.2.1.2. Union-based SQLi .....	14
<b>1.2.2. Inferential SQL Injection (Blind SQLi) .....</b>	<b>15</b>
1.2.2.1. Blink-boolean-based SQLi .....	15
1.2.2.2. Time-based Blind SQLi.....	15
<b>1.2.3. Out-of-band SQL Injection .....</b>	<b>15</b>
<b>1.3. Một cách phát hiện SQL Injection.....</b>	<b>16</b>
<b>1.4. Một số biện pháp ngăn ngừa SQL Injection .....</b>	<b>17</b>
<b>1.4.1. Câu lệnh được tham số hóa (Parameterized Statements) .....</b>	<b>17</b>
<b>1.4.2. Xác thực dữ liệu đầu vào của người dùng.....</b>	<b>18</b>
<b>1.4.3. Ẩn thông tin của các thông báo.....</b>	<b>19</b>

---

1.4.4.	Hạn chế quyền .....	19
1.4.5.	Cách phòng chống trong PHP (hàm addslashes) .....	20
<b>Phần 2.</b>	<b>THỰC HÀNH.....</b>	<b>21</b>
2.1.	Thực hành trên LAB .....	21
2.1.1.	SQL Injection vulnerability in WHERE clause allowing retrieval of hidden data .....	21
2.1.2.	SQL Injection vulnerability allowing login bypass .....	21
2.1.3.	SQL injection attack, querying the database type and version on Oracle 22	
2.1.4.	SQL Injection attack, querying the database type and version on MYSQL and Microsoft .....	22
2.1.5.	SQL injection attack, listing the database contents on non-Oracle databases .....	23
2.1.6.	SQL injection attack, listing the database contents on Oracle ...	23
2.1.7.	SQL injection UNION attack, determining the number of columns returned by the query .....	24
2.1.8.	SQL injection UNION attack, finding a column containing text	24
2.1.9.	SQL injection UNION attack, retrieving data from other tables 25	
2.1.10.	SQL injection UNION attack, retrieving multiple values in a single column .....	25
2.1.11.	Blind SQL injection with conditional responses.....	26
2.1.12.	Blind SQL injection with conditional errors .....	26
2.1.13.	Visible error-based SQL injection .....	27
2.1.14.	Blind SQL injection with time delays .....	27

---

2.1.15.	Blind SQL injection with time delays and information retrieval	28
2.1.16.	Blind SQL injection with out-of-band interaction .....	28
2.1.17.	Blind SQL injection with out-of-band data exfiltration .....	29
2.1.18.	SQL injection with filter bypass via XML encoding.....	29
2.2.	Thực hành trên WEB .....	30
2.2.1.	Login .....	30
2.2.1.1.	Tấn công .....	30
2.2.1.2.	Phòng thủ .....	31
2.2.2.	Database .....	32
2.2.2.1.	Tấn công .....	32
2.2.2.2.	Phòng thủ .....	35
Phần 3.	KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN .....	37
3.1.	Kết quả đạt được .....	37
3.2.	Hướng phát triển .....	37
Phần 4.	TÀI LIỆU THAM KHẢO .....	38

**DANH MỤC HÌNH ẢNH**

<i>Hình 1.1: Giới thiệu tấn công SQL Injection</i>	10
<i>Hình 1.2: Form đăng nhập</i>	12
<i>Hình 1.3: Thay đổi tên đăng nhập trong form</i>	13
<i>Hình 1.4: Kiểm tra cửa sổ đăng nhập có dễ bị tấn công hay không</i>	16
<i>Hình 1.5: Thông báo lỗi</i>	17
<i>Hình 1.6: Câu SQL bình thường</i>	18
<i>Hình 1.7: Câu SQL khi được tham số hóa</i>	18
<i>Hình 1.8: Phòng thủ bằng hàm <code>addslashes()</code></i>	20
<i>Hình 2.1: Kết quả thực hiện LAB 1</i>	21
<i>Hình 2.2: Kết quả thực hiện LAB 2</i>	21
<i>Hình 2.3: Kết quả thực hiện LAB 3</i>	22
<i>Hình 2.4: Kết quả thực hiện LAB 4</i>	22
<i>Hình 2.5: Kết quả thực hiện LAB 5</i>	23
<i>Hình 2.6: Kết quả thực hiện LAB 6</i>	23
<i>Hình 2.7: Kết quả thực hiện LAB 7</i>	24
<i>Hình 2.8: Kết quả thực hiện LAB 8</i>	24
<i>Hình 2.9: Kết quả thực hiện LAB 9</i>	25
<i>Hình 2.10: Kết quả thực hiện LAB 10</i>	25
<i>Hình 2.11: Kết quả thực hiện LAB 11</i>	26
<i>Hình 2.12: Kết quả thực hiện LAB 12</i>	26
<i>Hình 2.13: Kết quả thực hiện LAB 13</i>	27
<i>Hình 2.14: Kết quả thực hiện LAB 14</i>	27
<i>Hình 2.15: Kết quả thực hiện LAB 15</i>	28
<i>Hình 2.16: Kết quả thực hiện LAB 16</i>	28
<i>Hình 2.17: Kết quả thực hiện LAB 17</i>	29
<i>Hình 2.18: Kết quả thực hiện LAB 18</i>	29
<i>Hình 2.19: Tấn công login trên trang web</i>	30
<i>Hình 2.20: Tấn công thành công vào user admin</i>	30

---

<i>Hình 2.21: Phòng thủ tấn công login bằng hàm <b>addslashes()</b> .....</i>	<i>31</i>
<i>Hình 2.22: Kết quả tấn công login sau khi thực hiện phòng thủ .....</i>	<i>31</i>
<i>Hình 2.23: Tấn công trên URL .....</i>	<i>32</i>
<i>Hình 2.24: Kết quả xem được toàn bộ tên database và columns của trong database trong server .....</i>	<i>32</i>
<i>Hình 2.25: Kết quả được load dữ liệu lên trang web khi xem xong view-source .....</i>	<i>33</i>
<i>Hình 2.26: Kết quả xem được tên database đang được sử dụng hiện tại cho website .....</i>	<i>33</i>
<i>Hình 2.27: Kết quả được load dữ liệu lên trang web khi xem xong view-source .....</i>	<i>34</i>
<i>Hình 2.28: Select thành công dữ liệu trong bảng users .....</i>	<i>34</i>
<i>Hình 2.29: Kết quả được load lên trong view-sources .....</i>	<i>35</i>
<i>Hình 2.30: Thực hiện phòng thủ bằng hàm <b>addslashes()</b> .....</i>	<i>35</i>
<i>Hình 2.31: Tấn công thất bại sau khi phòng thủ .....</i>	<i>36</i>



**PHÂN CÔNG CÔNG VIỆC**

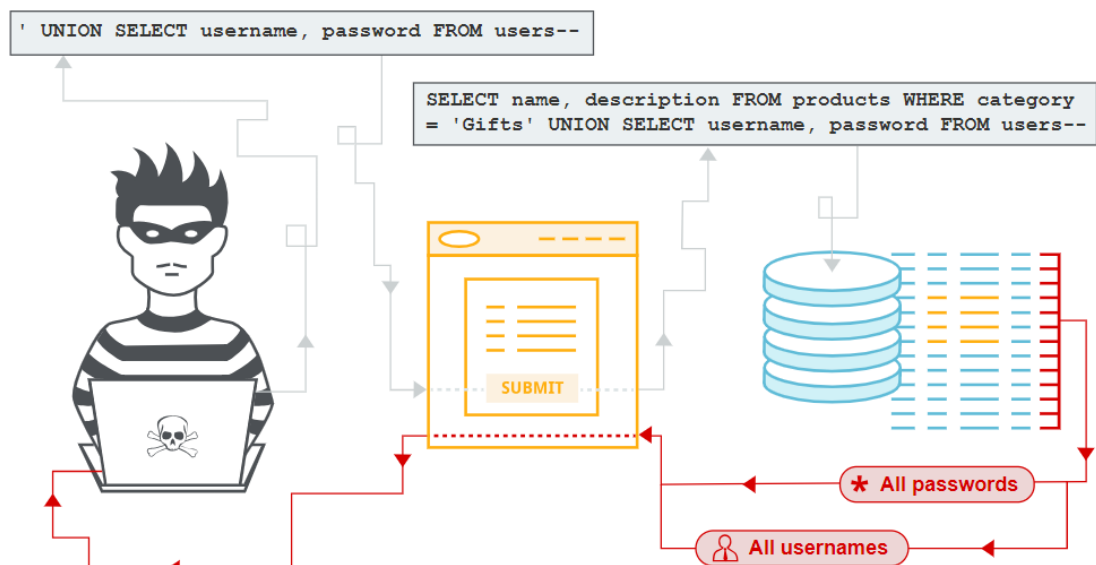
<b>HỌ VÀ TÊN</b>	<b>PHÂN CÔNG</b>	<b>MỨC ĐỘ HOÀN THÀNH</b>
<b>Trần Thị Kim Anh</b>	Thực hiện bài tập lab (18/18) Tìm hiểu lý thuyết về SQLi Chỉnh sửa báo cáo + Slide Demo dự án	100%
<b>Lê Thị Lệ Trúc</b>	Thực hiện bài tập lab (18/18) Tìm hiểu lý thuyết về SQLi Hoàn thành báo cáo + Slide Thuyết trình	100%
<b>Lê Minh Chánh</b>	Thực hiện bài tập lab (18/18) Tìm hiểu lý thuyết về SQLi Chỉnh sửa báo cáo + Slide Demo dự án	100%

## Phần 1. SQL Injection

### 1.1. SQL Injection

#### 1.1.1. Khái niệm

- SQL Injection (SQLi) là một lỗ hổng bảo mật web, cho phép kẻ tấn công can thiệp vào các truy vấn mà ứng dụng thực hiện với cơ sở dữ liệu của nó. Điều này có thể cho phép kẻ tấn công xem dữ liệu mà thông thường chúng không thể truy xuất được. Điều này có thể bao gồm dữ liệu thuộc về người dùng khác hoặc bất kỳ dữ liệu nào khác mà web có thể truy cập.
- Trong nhiều trường hợp, kẻ tấn công có thể sửa, đổi hoặc xóa dữ liệu này, gây ra những thay đổi liên tục đối với nội dung hoặc hành vi của web.
- Trong một số trường hợp, kẻ tấn công có thể nâng cấp cuộc tấn công SQLi để phạm máy chủ cơ bản hoặc cơ sở hạ tầng phụ trợ khác. Nó cũng có thể cho phép họ thực hiện các cuộc tấn công từ chối dịch vụ.



Hình 1.1: Giới thiệu tấn công SQL Injection

### 1.1.2. Nguyên nhân

- **Không kiểm tra dữ liệu đầu vào:** Đây là dạng lỗi SQL Injection xảy ra khi thiếu đoạn mã kiểm tra dữ liệu đầu vào trong câu truy vấn SQL. Kết quả là người dùng cuối có thể thực hiện một số truy vấn không mong muốn đối với cơ sở dữ liệu của ứng dụng.
- **Xử lý không đúng trọng tâm:** Lỗi SQL Injection dạng này thường xảy ra do lập trình viên định nghĩa đầu vào dữ liệu không rõ ràng hoặc thiếu bước kiểm tra và lọc kiểu dữ liệu đầu vào. Điều này có thể xảy ra khi một trường số được sử dụng trong truy vấn SQL nhưng lập trình viên lại thiếu bước kiểm tra dữ liệu đầu vào để xác minh kiểu của dữ liệu mà người dùng nhập vào có phải là số hay không.
- **Lỗi bảo mật bên trong máy chủ:** Lỗi hổng có thể tồn tại trong chính phần mềm máy chủ cơ sở dữ liệu. Điều này cho phép một cuộc tấn công SQL Injection có thể thực hiện thành công dựa trên những ký tự Unicode không thông thường ngay cả khi dữ liệu đầu vào đã được kiểm soát.

### 1.1.3. Sự nguy hiểm của SQL Injection

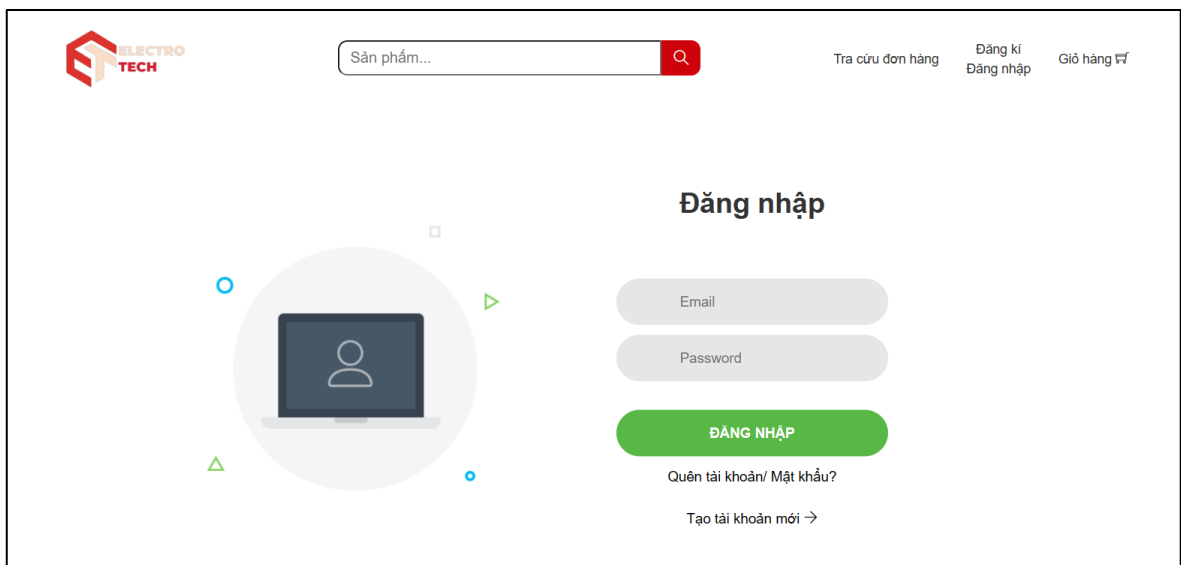
Không phải ngẫu nhiên SQLi được đánh giá là loại tấn công vô cùng nguy hiểm. Đó là bởi tấn công SQLi gây ra những thiệt hại rất lớn cho doanh nghiệp và tổ chức, tiêu biểu như:

- SQLi cho phép kẻ tấn công thực hiện các thao tác xóa, sửa,... do có toàn quyền trên cơ sở dữ liệu của web, gây ra những thiệt hại lớn khiến những dữ liệu trong database bị lộ ra ngoài.
- SQLi không những cho phép kẻ tấn công đọc được, chỉnh sửa dữ liệu. Họ có thể lợi dụng điểm này để xóa toàn bộ dữ liệu và làm cho hệ thống web ngừng hoạt động, tạo các bảng dữ liệu mới, có thể điều khiển toàn bộ hệ quản trị CSDL với quyền hạn rộng lớn

- Việc lộ thông tin gây ảnh hưởng rất lớn đến hình ảnh, uy tín, bảo mật doanh nghiệp và tổ chức. Khi thông tin và dữ liệu của khách hàng bị lộ, họ sẽ không tin tưởng và sử dụng dịch vụ của bạn nữa. Mặt khác, khách hàng họ thường để chung một mật khẩu cho nhiều tài khoản nên khi bị lộ một tài khoản thì những tài khoản khác cũng nguy cơ bị lộ. Bởi vậy khách hàng sẽ bị ảnh hưởng rất lớn, từ đó họ sẽ không sử dụng dịch vụ của công ty dẫn đến doanh nghiệp bị phá sản.

#### 1.1.4. SQLi hoạt động như thế nào?

- Các cuộc tấn công SQLi được thực hiện bằng cách gửi lệnh SQL độc hại đến các máy chủ cơ sở dữ liệu thông qua các yêu cầu của người mà website cho phép. Bất kỳ kênh input nào cũng có thể được sử dụng để gửi các lệnh độc hại.
- Để xem cách nó hoạt động, giả sử bạn có form đăng nhập gồm: email và password:



Hình 1.2: Form đăng nhập

- Khi người dùng nhập thông tin đăng nhập của họ và nhấn vào “Đăng nhập”, thông tin sẽ được gửi lại cho máy chủ web của bạn, ở đó nó sẽ được kết hợp với lệnh SQL. Ví dụ, trong PHP, mã sẽ giống như sau:

```
$sql_command = "SELECT *
                FROM USERS
                WHERE email = " . $_POST['email'] ;
$sql_command . = " AND password = " . $_POST['password'] . "";
```

- Lệnh này sau đó sẽ được gửi đến một máy chủ cơ sở dữ liệu và tập dữ liệu kết quả sẽ được xác định xem email và password có tương ứng với một tài khoản người dùng hợp lệ hay không.
- Ví dụ người dùng nhập “admin@gmail.com” làm email và “123456” làm password thì sẽ chuyển mã trên thành lệnh sau:

```
SELECT *
FROM users
WHERE email = 'admin@gmail.com '
AND password='123456';
```

- Nhưng nếu người dùng cố tình thay đổi email thành như sau:

Hình 1.3: Thay đổi tên đăng nhập trong form

- Lệnh SQL sẽ là như sau:

**SELECT \***

**FROM** users

**WHERE** email = 'admin@gmail.com 'or '1'='0'

**AND** password = '123456';

- Kết quả trả về là thông tin đăng nhập của người dùng có email là “admin@gmail.com” mà không cần mật khẩu chính xác.

⇒ Đây chỉ là một trong những hình thức đơn giản nhất của SQL Injection.

## 1.2. Phân loại các kiểu tấn công của SQL Injection

### 1.2.1. In-band SQL Injection (Classic SQLi)

- Đây là dạng tấn công phổ biến nhất và cũng dễ để khai thác lỗ hổng SQLi nhất.
- Xảy ra khi kẻ tấn công có thể tổ chức tấn công và thu thập kết quả trực tiếp trên cùng một kênh liên lạc.

#### 1.2.1.1. Error-based SQLi

- Là một kỹ thuật tấn công SQLi dựa vào thông báo lỗi được trả về từ Database Server có chứa thông tin về cấu trúc của cơ sở dữ liệu.
- Trong một vài trường hợp, chỉ một mình Error-based là đủ cho hacker có thể liệt kê được các thuộc tính của cơ sở dữ liệu.

#### 1.2.1.2. Union-based SQLi

- Kỹ thuật này lợi dụng toán tử UNION SQL để kết hợp nhiều câu lệnh được tạo bởi cơ sở dữ liệu để nhận được một HTTP response. Response này có thể chứa dữ liệu mà kẻ tấn công có thể sử dụng.

### 1.2.2. Inferential SQL Injection (Blind SQLi)

- Kẻ tấn công sẽ gửi các data payload đến server và quan sát phản ứng, hành vi của server để tìm hiểu về cấu trúc của nó. Phương pháp này được gọi là Blind SQLi vì dữ liệu không được chuyển từ cơ sở dữ liệu trang web đến kẻ tấn công. Do đó kẻ tấn công không thể nhìn thấy thông tin về cuộc tấn công in-band.
- Blind SQLi dựa trên phản ứng và các hành vi hoạt động của server. Do đó chúng thường thực thi chậm hơn, nhưng có thể gây ảnh hưởng tương tự.

#### 1.2.2.1. Blink-boolean-based SQLi

- Là kỹ thuật tấn công SQL Injection dựa vào việc gửi các truy vấn tới cơ sở dữ liệu bắt buộc web trả về các kết quả khác nhau phụ thuộc vào câu truy vấn là True hay False.
- Tùy thuộc kết quả trả về của câu truy vấn mà HTTP response có thể thay đổi, hoặc giữ nguyên.
- Kiểu tấn công này thường chậm (đặc biệt với cơ sở dữ liệu có kích thước lớn) do người tấn công cần phải liệt kê từng dữ liệu, hoặc mò từng ký tự.

#### 1.2.2.2. Time-based Blind SQLi

- Time-base Blind SQLi là kỹ thuật tấn công dựa vào việc gửi những câu truy vấn tới cơ sở dữ liệu và buộc cơ sở dữ liệu phải chờ một khoảng thời gian (thường tính bằng giây) trước khi phản hồi.
- Thời gian phản hồi (ngay lập tức hay trễ theo khoảng thời gian được set) cho phép kẻ tấn công suy đoán kết quả truy vấn là TRUE hay FALSE.
- Kiểu tấn công này cũng tốn nhiều thời gian tương tự như Boolean-based SQLi.

### 1.2.3. Out-of-band SQL Injection

- Out-of-band SQLi không phải dạng tấn công phổ biến, chủ yếu bởi vì nó phụ thuộc vào các tính năng được bật trên Database Server được sử dụng bởi Web Application.

- Kiểu tấn công này xảy ra khi kẻ tấn công không thể trực tiếp tấn công và thu thập kết quả trực tiếp trên cùng một kênh (In-band SQLi), và đặc biệt là việc phản hồi từ server là không ổn định.
- Kiểu tấn công này phụ thuộc vào khả năng server thực hiện các request DNS hoặc HTTP để chuyển dữ liệu cho kẻ tấn công.

### 1.3. Một cách phát hiện SQL Injection

- Việc kiểm tra lỗ hổng này có thể được thực hiện rất dễ dàng. Đôi khi ta chỉ cần nhập ký hiệu ‘ hoặc “ vào các trường được kiểm tra. Nếu nó trả về bất kỳ thông báo bất ngờ hoặc bất thường, thì ta có thể chắc chắn rằng SQL Injection khả thi cho trường đó.

**Ví dụ:** Nếu nhận được thông báo lỗi như “ Internal Server Error” làm kết quả tìm kiếm, thì ta có thể chắc chắn rằng cuộc tấn công này có thể xảy ra trong phần đó của hệ thống.

- Các kết quả khác, có thể thông báo tấn công bao gồm:

Blank page loaded.

No error or success messages – chức năng và trang không phản ứng với đầu vào

Success message for malicious code -thông báo thành công với mã độc hại

- Giả sử: Kiểm tra cửa sổ đăng nhập có dễ bị tấn công đối với SQL Injection hay không. Trong trường địa chỉ email hoặc mật khẩu, ta gõ ký hiệu ' như hình dưới đây.



Hình 1.4: Kiểm tra cửa sổ đăng nhập có dễ bị tấn công hay không



- Nếu đầu vào như vậy trả về kết quả như thông báo lỗi “Internal Server Error” hoặc bất kỳ kết quả không phù hợp được liệt kê nào khác, thì chúng ta gần như có thể chắc chắn rằng cuộc tấn công này có thể xảy ra cho trường đó.



**Internal Server Error**

*Hình 1.5: Thông báo lỗi*

- Do đó việc kiểm tra SQL Injection với ký hiệu ' là một cách đáng tin cậy đã kiểm tra xem cuộc tấn công này có khả thi hay không.
- Nếu ký hiệu ' không trả lại bất kỳ kết quả không phù hợp nào, thì ta có thể thử nhập các ký hiệu khác như " để kiểm tra kết quả.
- Một số loại dữ liệu khác mà cũng nên thử submit để biết xem trang web có gặp lỗi hay không như:

' or 1=1--

" or 1=1--

or 1=1--

' or 'a'='a

" or "a"="a

') or ('a'='a

## **1.4. Một số biện pháp ngăn ngừa SQL Injection**

### **1.4.1. Câu lệnh được tham số hóa (Parameterized Statements)**

- Ứng dụng tham số hóa truy vấn là một trong những cách tốt nhất để ngăn chặn SQL Injection. Cấu trúc của truy vấn và truyền các tham số giá trị được tách biệt.
- Nếu như bình thường câu SQL là:

```
sqlQuery = SELECT * FROM Users WHERE Email = ' + email + ' AND Password = ' + password + ';
```

*Hình 1.6: Câu SQL bình thường*

- Thì khi tham số hóa nó sẽ có dạng:

```
sqlQuery = SELECT * FROM Users WHERE Email =? AND Password =?';  
parameters.add("Email", email)  
parameters.add("Password", password)
```

*Hình 1.7: Câu SQL khi được tham số hóa*

- Khi sử dụng tham số thì câu SQL sẽ không chỉ nhận giá trị input vào và chỉ việc thay thế. Thay vào đó câu lệnh SQL đã được chuyển đến SQL server, tức là nó đã biết sẵn nó sẽ truy vấn cái gì cùng với một danh sách các thông số và giá trị đã input.
- Khi sử dụng tham số hóa truy vấn ở trên. Cơ sở dữ liệu nó đã biết truy vấn sẽ làm gì. Nó sẽ xem Email = **admin@gmail'or '1'='0** Password = **123456** là các giá trị truyền vào. Khi thực hiện truy vấn sẽ tìm Email = **admin@gmail. 'or '1'='0** Password = **123456** và kết quả này sẽ dẫn tới sai thông tin đăng nhập.

#### 1.4.2. Xác thực dữ liệu đầu vào của người dùng

- Ngay cả khi đã sử dụng tham số hóa truy vấn, thì việc xác thực dữ liệu đầu vào là thực sự cần thiết để đảm bảo các thuộc tính dữ liệu là phù hợp như kiểu dữ liệu: Text, Number, Special characters, độ dài input,... thì những yếu tố này có thể validate ngay ở form input.
- Nếu sai thuộc tính dữ liệu sẽ bị lỗi ngay không cần phải chờ tới lúc SQL chạy nữa. Với cách kiểm soát đơn giản về kiểu dữ liệu như vậy cũng đã hạn chế đáng kể các cuộc tấn công.
- Như ví dụ trên nếu ở form input trường Email được validate chỉ cho phép nhập văn bản và chữ số. Không cho phép nhập ký tự đặc biệt thì khi nhập Email = admin@gmail.com'or '1'='0 nó sẽ trả về lỗi ngay mà không cần thực thi SQL

### 1.4.3. Ẩn thông tin của các thông báo

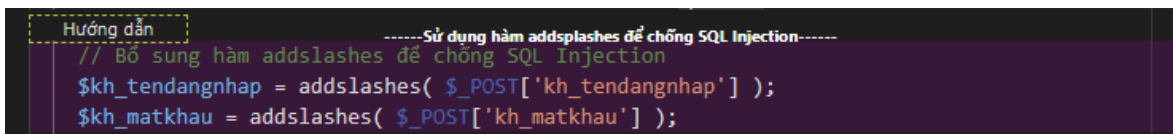
- Thông báo lỗi quá chi tiết cũng là điểm hữu ích cho những kẻ tấn công tìm hiểu cấu trúc CSDL của hệ thống.
- Ví dụ ở form reset password nếu nhập 1 email không tồn tại hệ thống thì nó sẽ có thông báo lỗi Email không tồn tại hoặc email đúng thì nó sẽ thông báo thành công. Trường hợp này thì những kẻ tấn công sẽ dễ dàng dò và biết được trong hệ thống đang có những user nào.
- Thay vào đó có thể xử lý bằng cách, nhập với bất kì email nào nó cũng sẽ thông báo thành công. Nếu email tồn tại trong hệ thống thì sẽ nhận được email reset password. Email nào không tồn tại thì sẽ không nhận được mail. Với cách này thì kẻ tấn công sẽ không thể dò được tài khoản của hệ thống.
- Sẽ tùy từng trường hợp mà việc xử lý mã lỗi làm sao cho hợp lý nhất. Thông báo lỗi chỉ nên hiển thị các thông tin cần thiết. Với những thông tin quan trọng thì tốt hơn là chỉ hiển thị thông báo lỗi chung cho biết có lỗi xảy ra và khuyến khích người dùng liên hệ với bộ phận support

### 1.4.4. Hạn chế quyền

- Hạn chế dùng tài khoản root hoặc sa để truy cập DB. Thay vào đó hãy tạo account và gán một số quyền nhất định. Lúc này lỡ hacker có hack được tài khoản thì cũng sẽ không có toàn quyền truy cập hệ thống, hạn chế được rủi ro.
- Việc xác định người dùng khác nhau với các đặc quyền khác nhau cũng rất là hữu ích trong quá trình phát triển hệ thống. Giảm thiểu rủi ro của cuộc tấn công tiêm nhiễm SQL.

#### 1.4.5. Cách phòng chống trong PHP (hàm addslashes)

- Như đã phân tích ở trên (ví dụ trên): điểm để tấn công chính là tham số truyền vào câu truy vấn. Do vậy phải thực hiện các biện pháp phòng chống để đảm bảo việc kiểm tra dữ liệu truyền vào không thể gây ra sai lệch khi thực hiện truy vấn.
- Giải pháp cho việc kiểm tra này là sử dụng “chuỗi escape”. Khi thực hiện escape một chuỗi, tức là mã hoá các ký tự đặc biệt của chuỗi (như ký tự ‘, &, |, ...) để nó không còn được hiểu là 1 ký tự đặc biệt nữa. Mỗi ngôn ngữ lập trình đều cung cấp các hàm để thực hiện escape chuỗi, với PHP ta sẽ sử dụng hàm **mysqli\_real\_escape\_string()** hoặc cũng có thể dùng **addslashes()** để thực hiện điều này.
- Ví dụ về hàm **addslashes()**: ký tự nháy kép lúc này không còn được hiểu là ký tự điều khiển nữa



```
Hướng dẫn -----Sử dụng hàm addslashes để chống SQL Injection-----
// Bổ sung hàm addslashes để chống SQL Injection
$kh_tendangnhap = addslashes( $_POST['kh_tendangnhap'] );
$kh_matkhau = addslashes( $_POST['kh_matkhau'] );
```

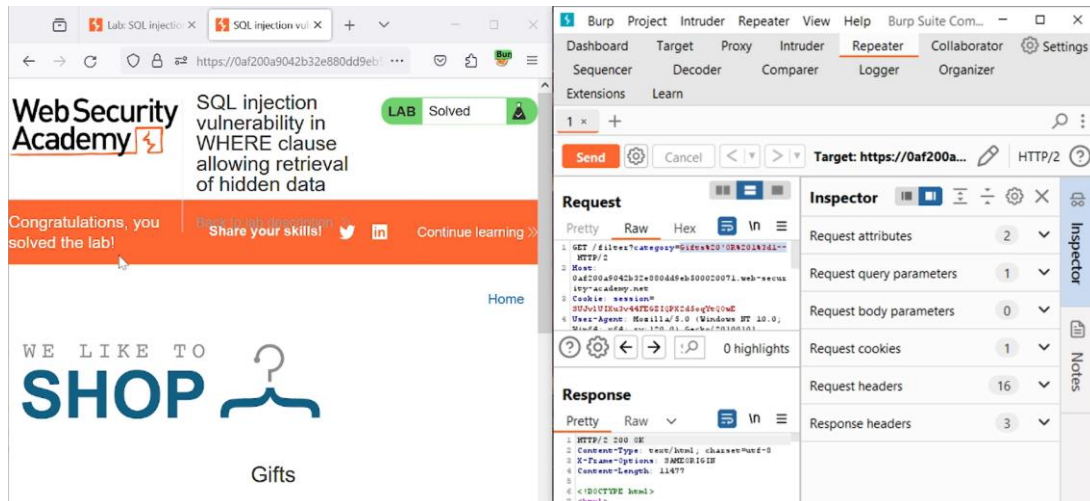
Hình 1.8: Phòng thủ bằng hàm *addslashes()*

## Phần 2. THỰC HÀNH

### 2.1. Thực hành trên LAB

#### 2.1.1. SQL Injection vulnerability in WHERE clause allowing retrieval of hidden data

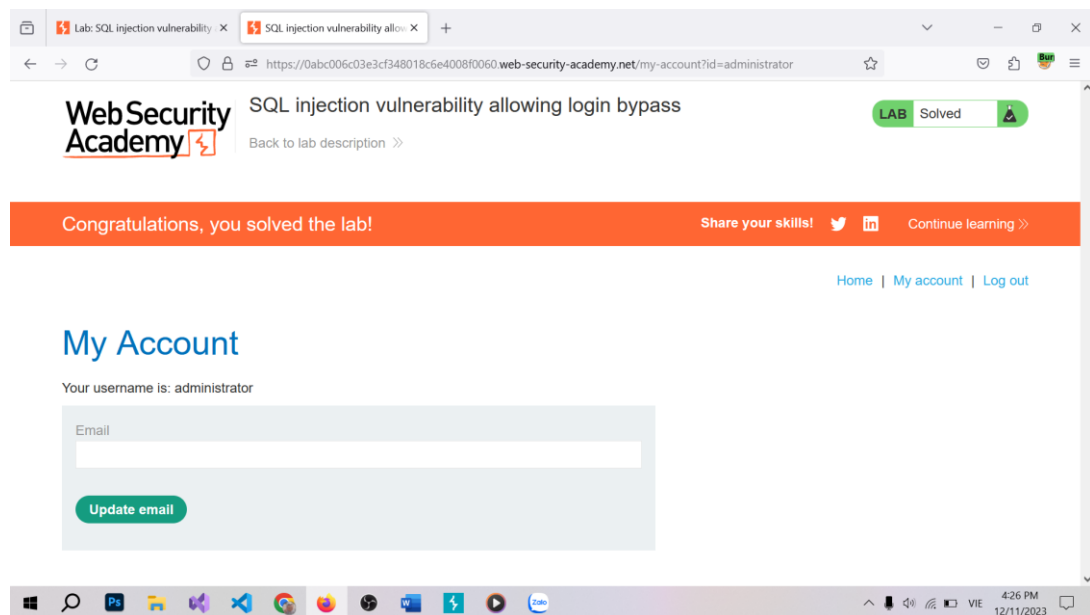
**Mục tiêu:** Hiển thị danh sách sản phẩm đã và chưa phát hành của ứng dụng.



Hình 2.1: Kết quả thực hiện LAB 1

#### 2.1.2. SQL Injection vulnerability allowing login bypass

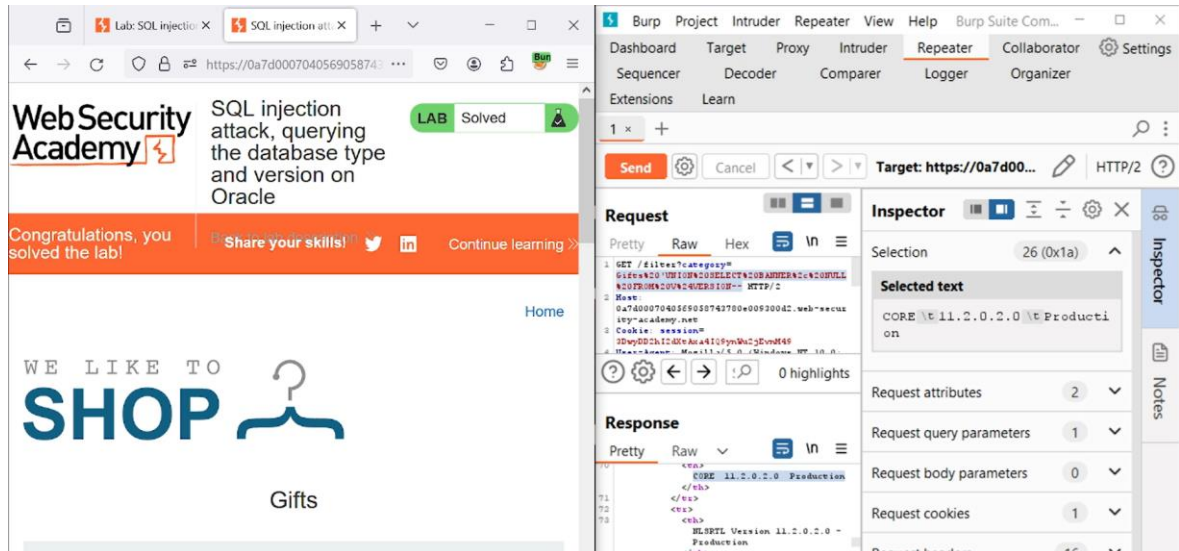
**Mục tiêu:** Đăng nhập ứng dụng thành công với tài khoản 'administrator'.



Hình 2.2: Kết quả thực hiện LAB 2

### 2.1.3. SQL injection attack, querying the database type and version on Oracle

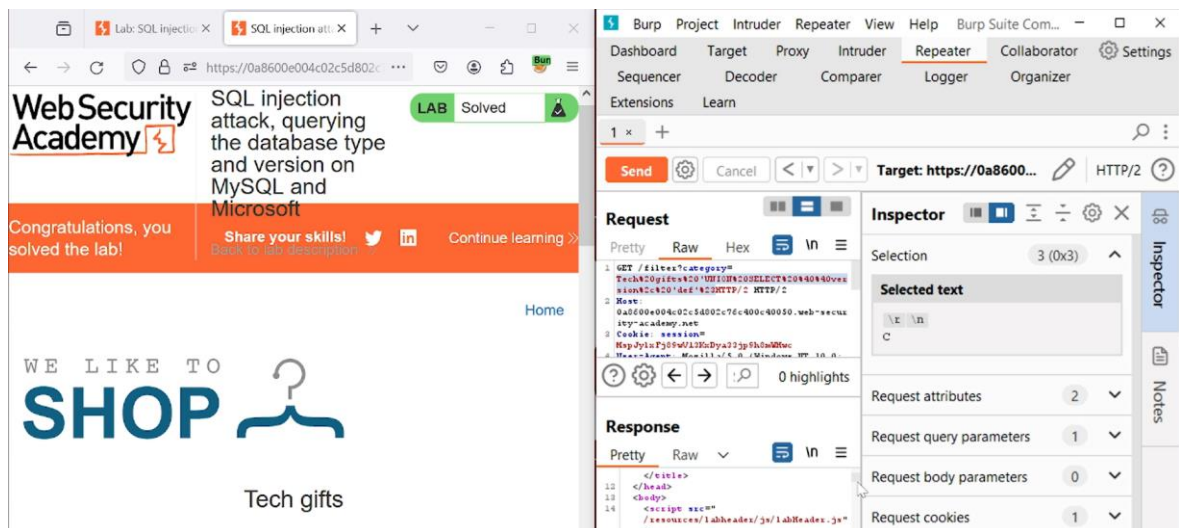
**Mục tiêu:** Hiển thị chuỗi phiên bản của database.



Hình 2.3: Kết quả thực hiện LAB 3

### 2.1.4. SQL Injection attack, querying the database type and version on MYSQL and Microsoft

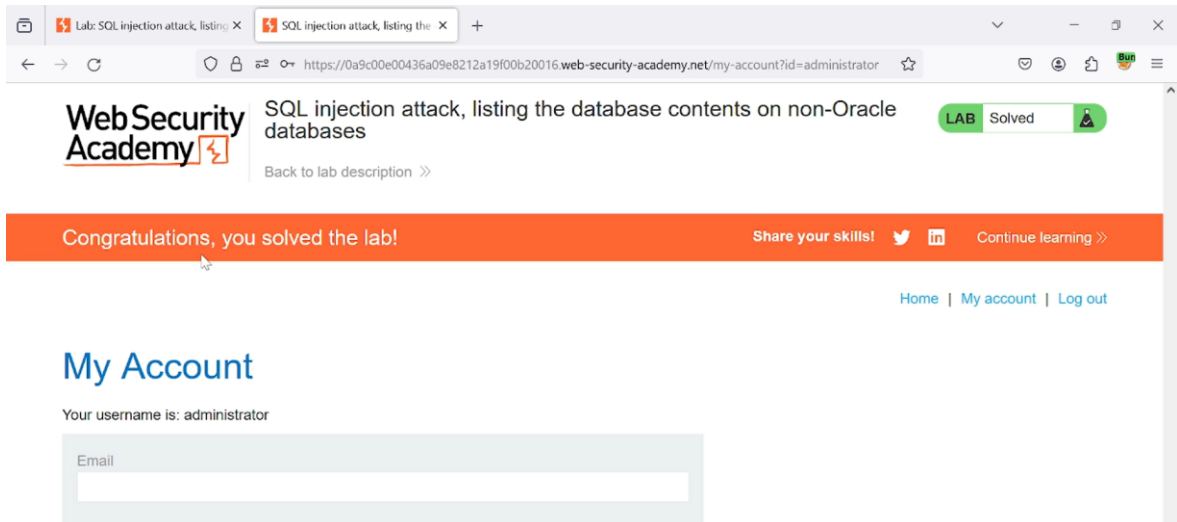
**Mục tiêu:** Hiển thị chuỗi phiên bản của database.



Hình 2.4: Kết quả thực hiện LAB 4

### 2.1.5. SQL injection attack, listing the database contents on non-Oracle databases

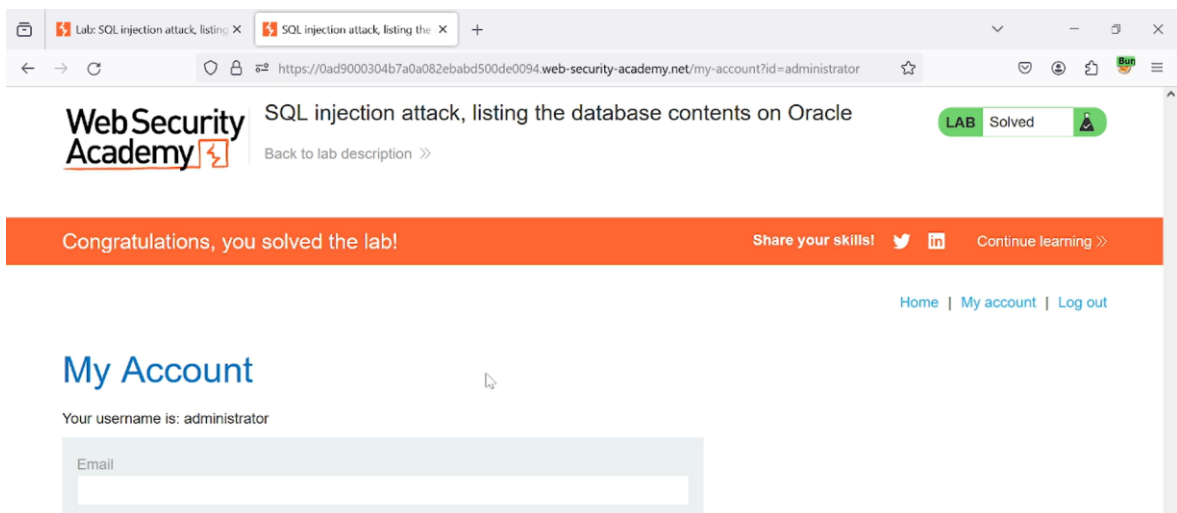
**Mục tiêu:** Đăng nhập thành công ứng dụng với tài khoản ‘administrator’.



Hình 2.5: Kết quả thực hiện LAB 5

### 2.1.6. SQL injection attack, listing the database contents on Oracle

**Mục tiêu:** Đăng nhập thành công ứng dụng với tài khoản ‘administrator’.

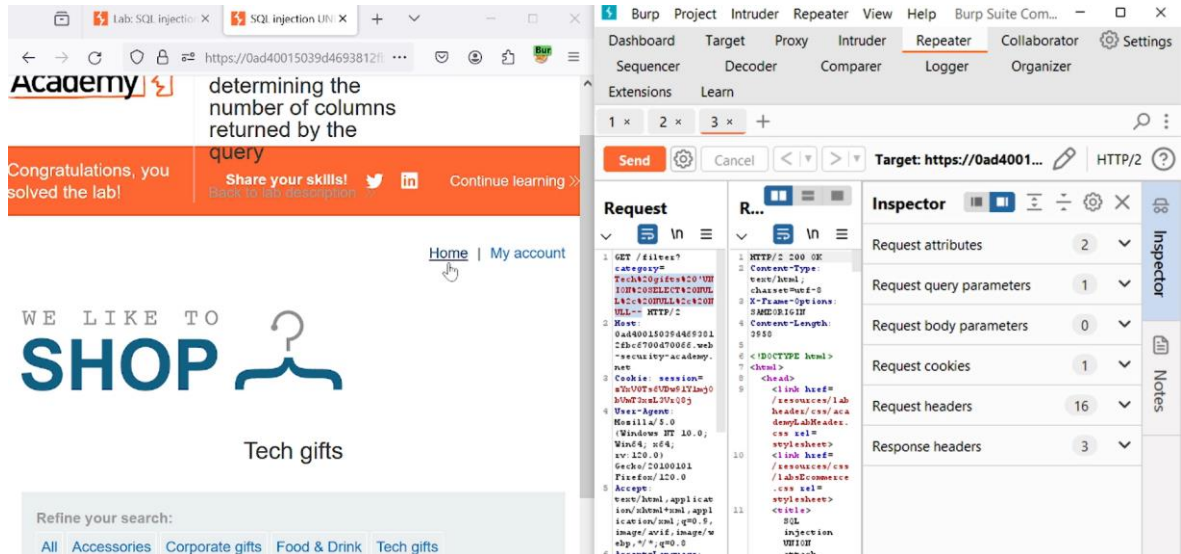


Hình 2.6: Kết quả thực hiện LAB 6



### 2.1.7. SQL injection UNION attack, determining the number of columns returned by the query

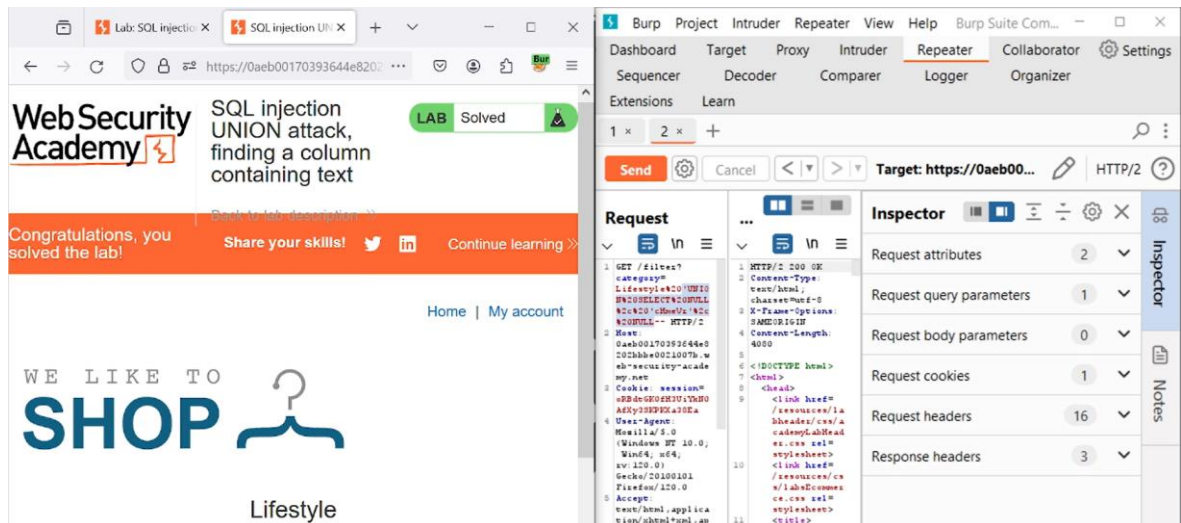
**Mục tiêu:** Xác định được số cột của bảng được truy vấn.



Hình 2.7: Kết quả thực hiện LAB 7

### 2.1.8. SQL injection UNION attack, finding a column containing text

**Mục tiêu:** Dùng UNION để xác định số cột và kiểu dữ liệu của chúng.

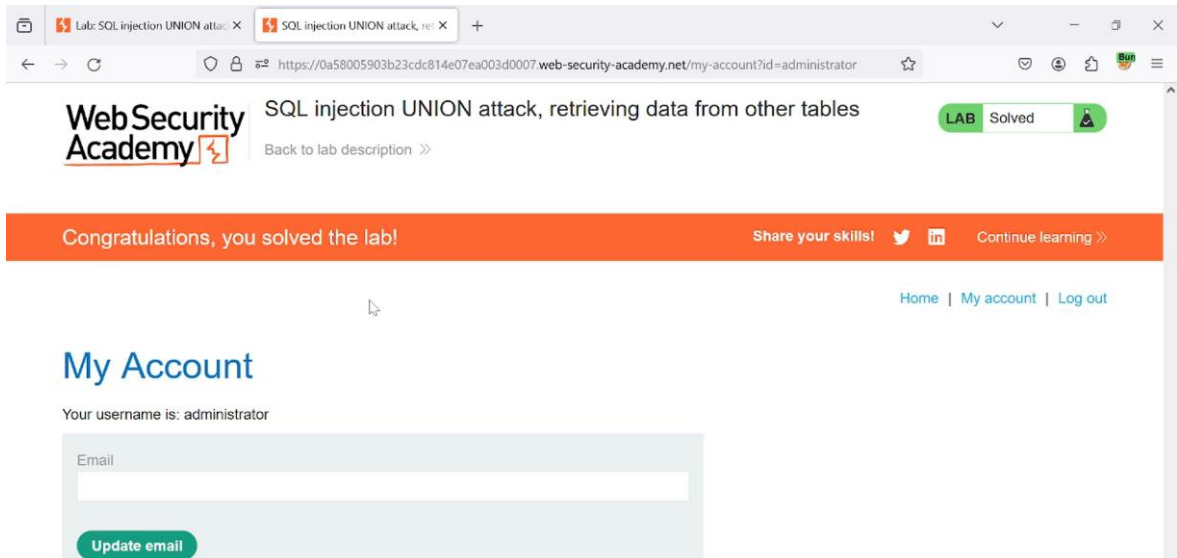


Hình 2.8: Kết quả thực hiện LAB 8



### 2.1.9. SQL injection UNION attack, retrieving data from other tables

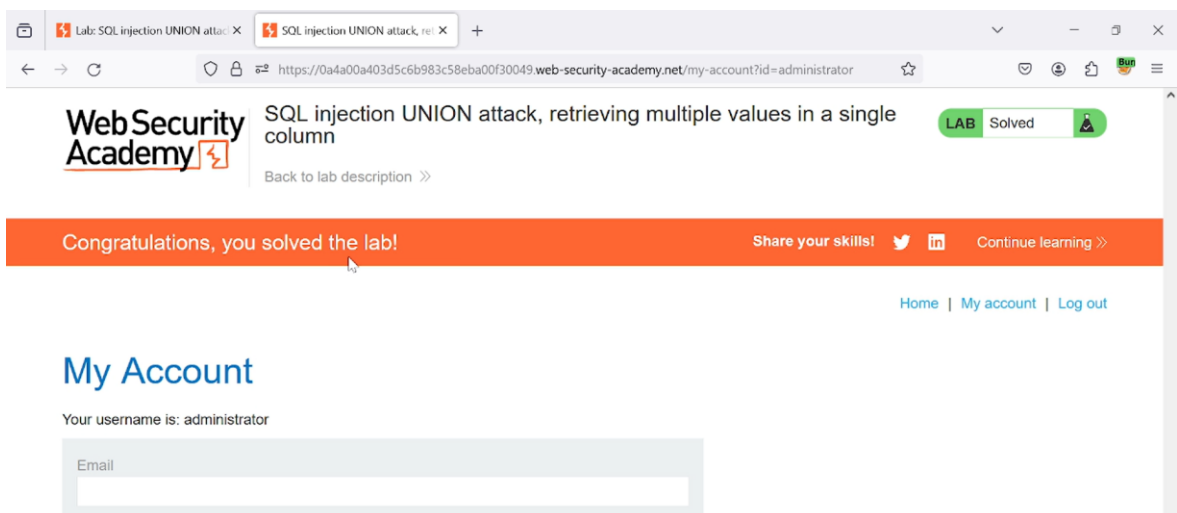
**Mục tiêu:** Hiển thị được tất cả tài khoản người dùng và mật khẩu. Đồng thời đăng nhập thành công ứng dụng với tư cách là quản trị viên ‘administrator’.



Hình 2.9: Kết quả thực hiện LAB 9

### 2.1.10. SQL injection UNION attack, retrieving multiple values in a single column

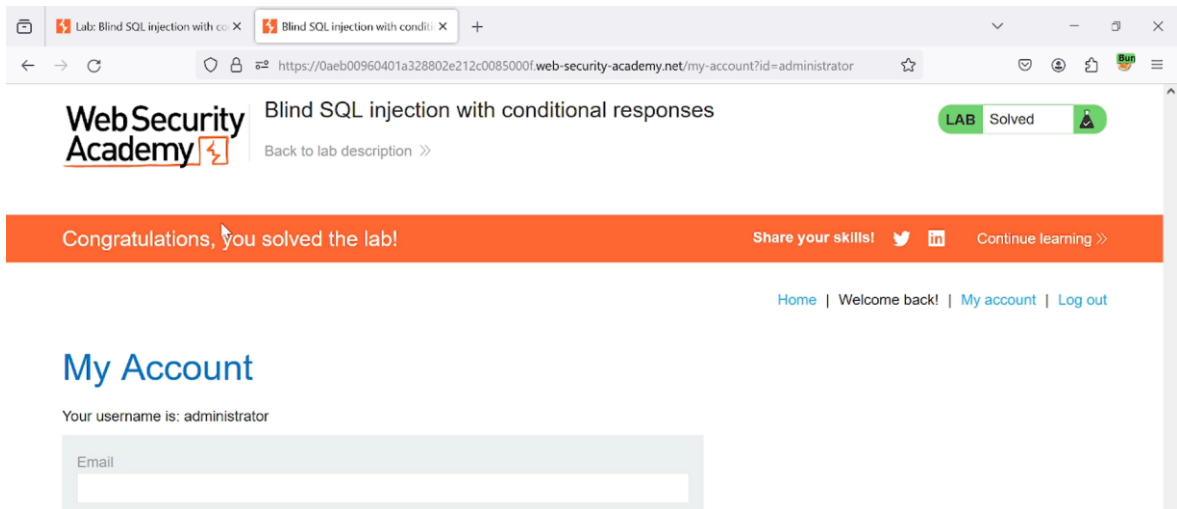
**Mục tiêu:** Hiển thị được tất cả tài khoản người dùng và mật khẩu. Đồng thời đăng nhập thành công ứng dụng với tư cách là quản trị viên ‘administrator’.



Hình 2.10: Kết quả thực hiện LAB 10

### 2.1.11. Blind SQL injection with conditional responses

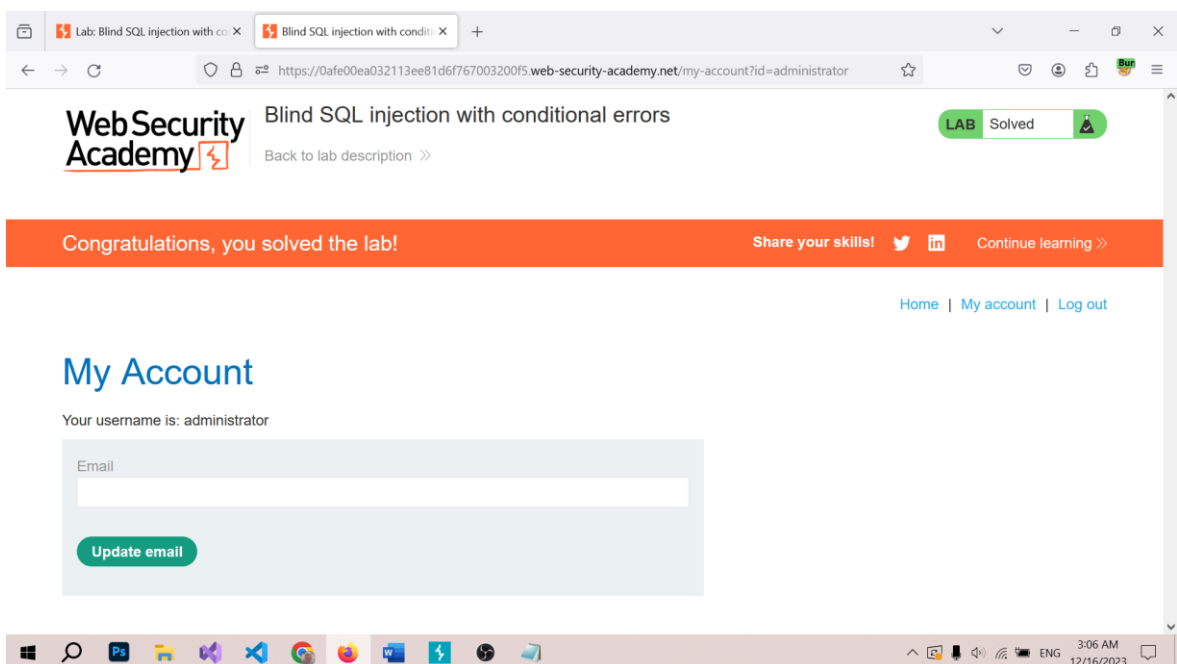
**Mục tiêu:** Đăng nhập thành công ứng dụng với tài khoản ‘administrator’.



Hình 2.11: Kết quả thực hiện LAB 11

### 2.1.12. Blind SQL injection with conditional errors

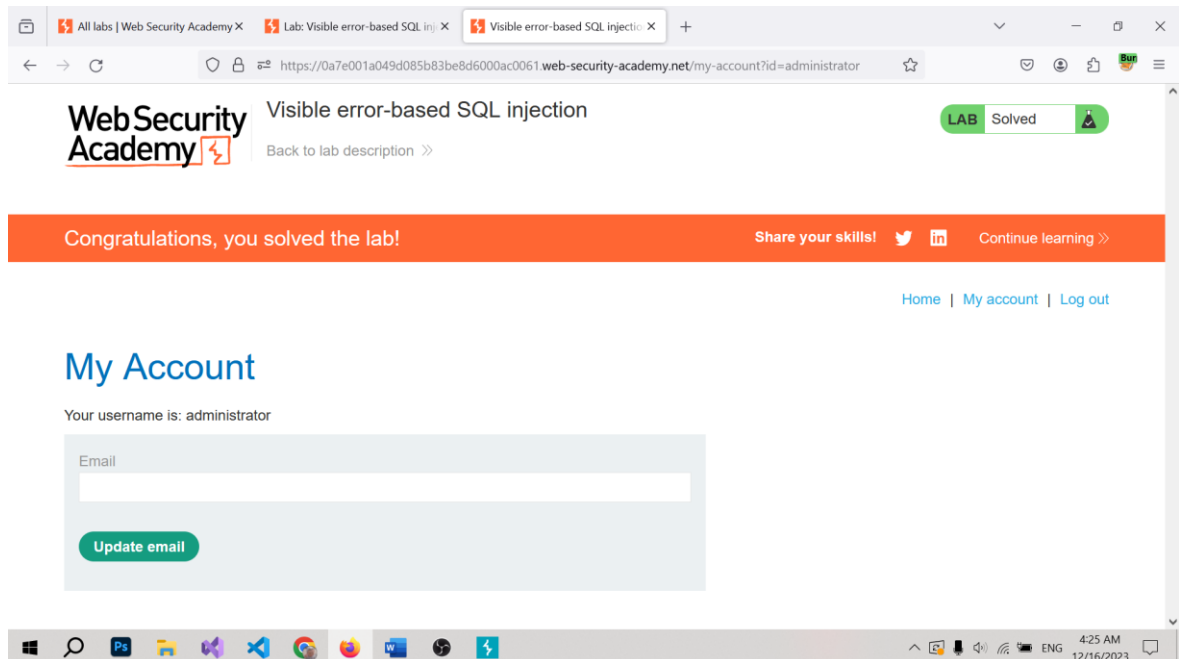
**Mục tiêu:** Đăng nhập thành công ứng dụng với tài khoản ‘administrator’.



Hình 2.12: Kết quả thực hiện LAB 12

### 2.1.13. Visible error-based SQL injection

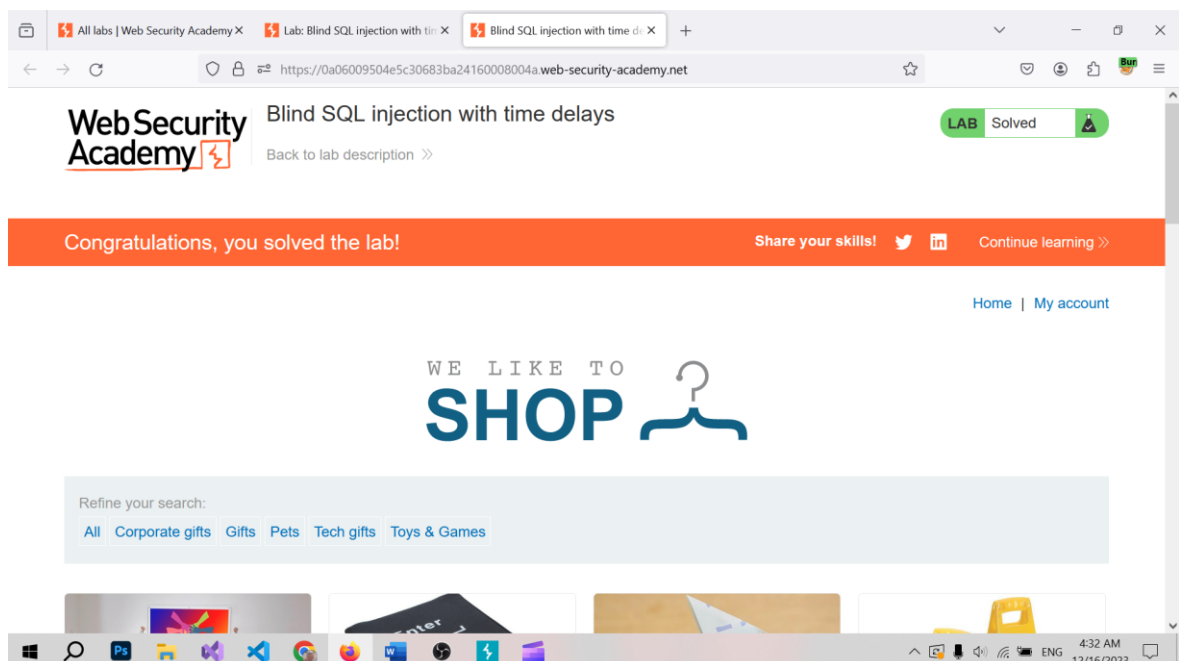
**Mục tiêu:** Đăng nhập thành công ứng dụng với tài khoản ‘administrator’.



Hình 2.13: Kết quả thực hiện LAB 13

### 2.1.14. Blind SQL injection with time delays

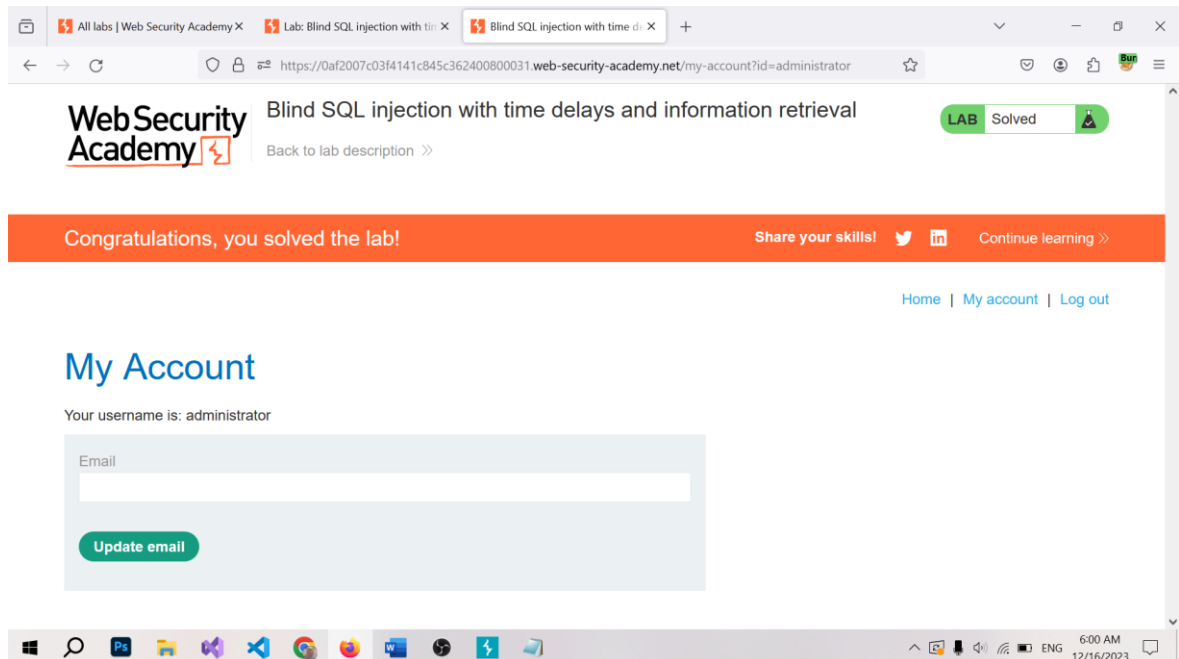
**Mục tiêu:** Khai thác lỗ hổng để gây ra độ trễ 10s.



Hình 2.14: Kết quả thực hiện LAB 14

### 2.1.15. Blind SQL injection with time delays and information retrieval

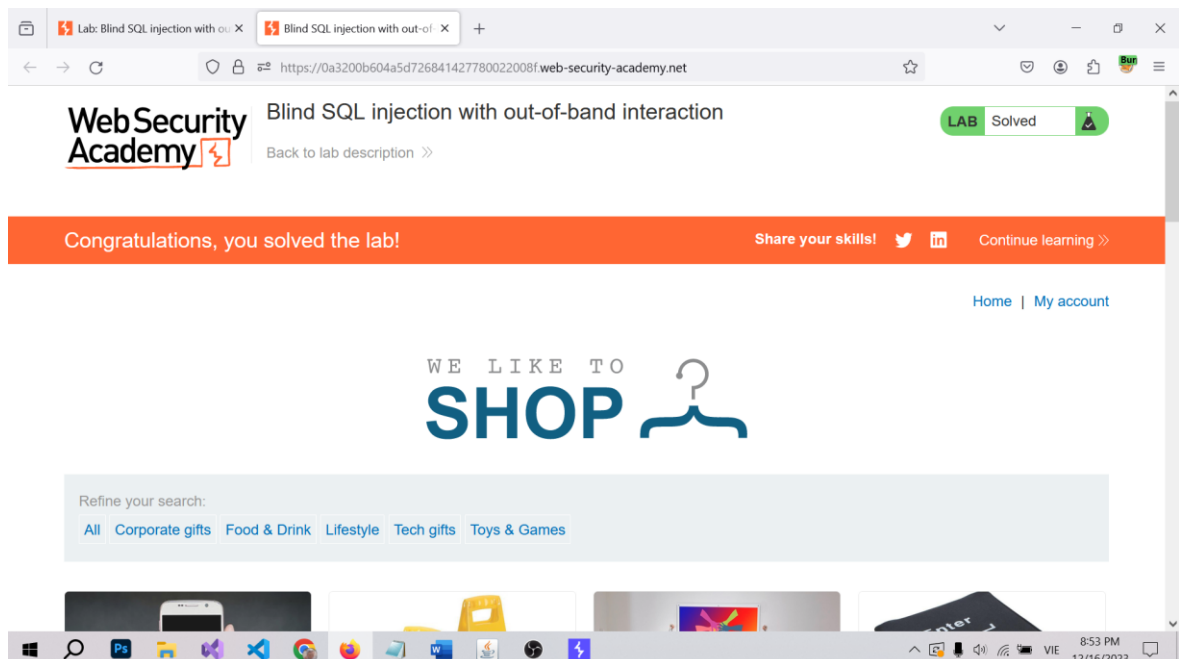
**Mục tiêu:** Đăng nhập thành công ứng dụng với tài khoản ‘administrator’.



Hình 2.15: Kết quả thực hiện LAB 15

### 2.1.16. Blind SQL injection with out-of-band interaction

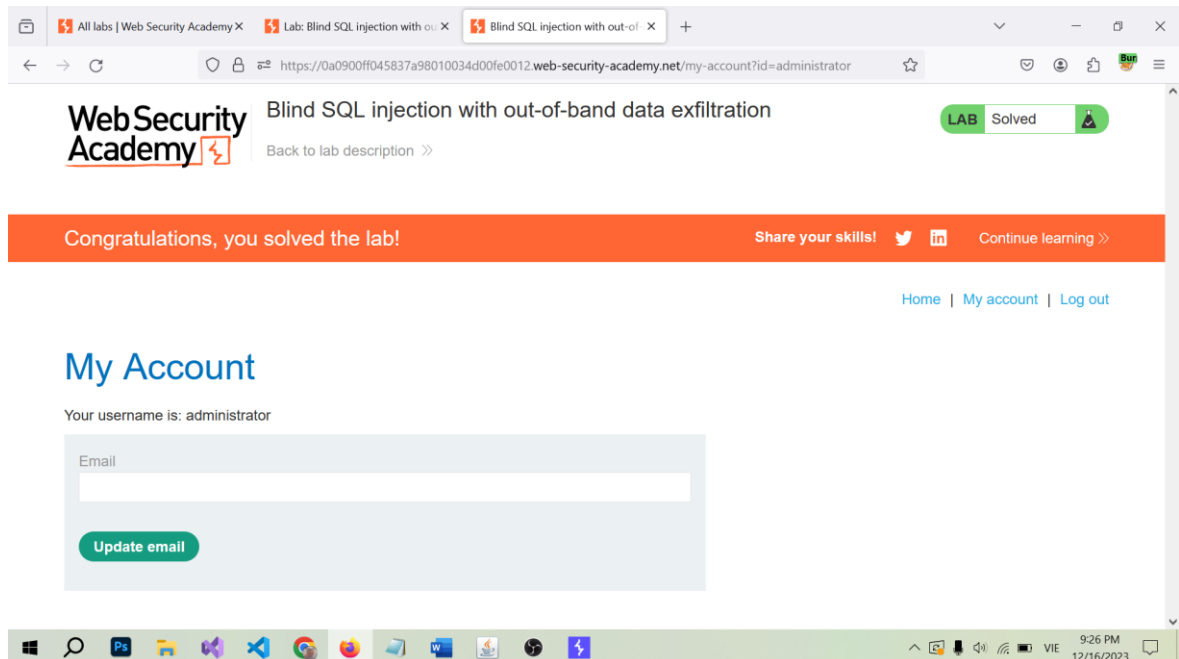
**Mục tiêu:** Thực hiện tra cứu DNS cho Burp Collaborator



Hình 2.16: Kết quả thực hiện LAB 16

### 2.1.17. Blind SQL injection with out-of-band data exfiltration

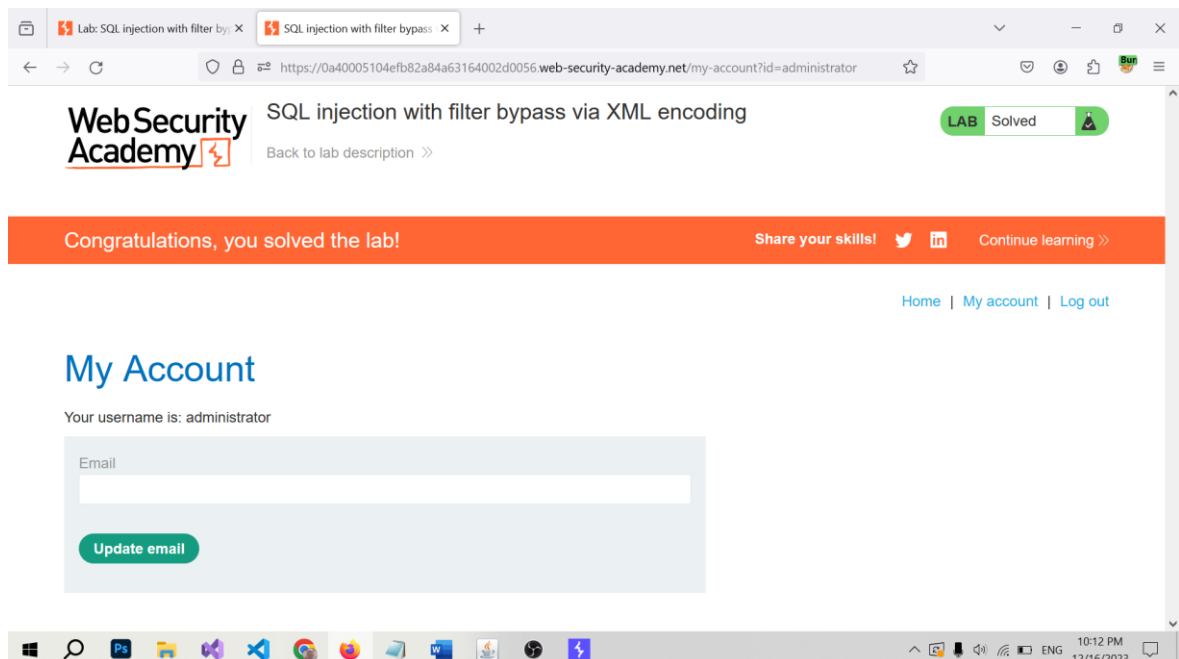
**Mục tiêu:** Đăng nhập thành công ứng dụng với tài khoản ‘administrator’.



Hình 2.17: Kết quả thực hiện LAB 17

### 2.1.18. SQL injection with filter bypass via XML encoding

**Mục tiêu:** Đăng nhập thành công ứng dụng với tài khoản ‘administrator’.



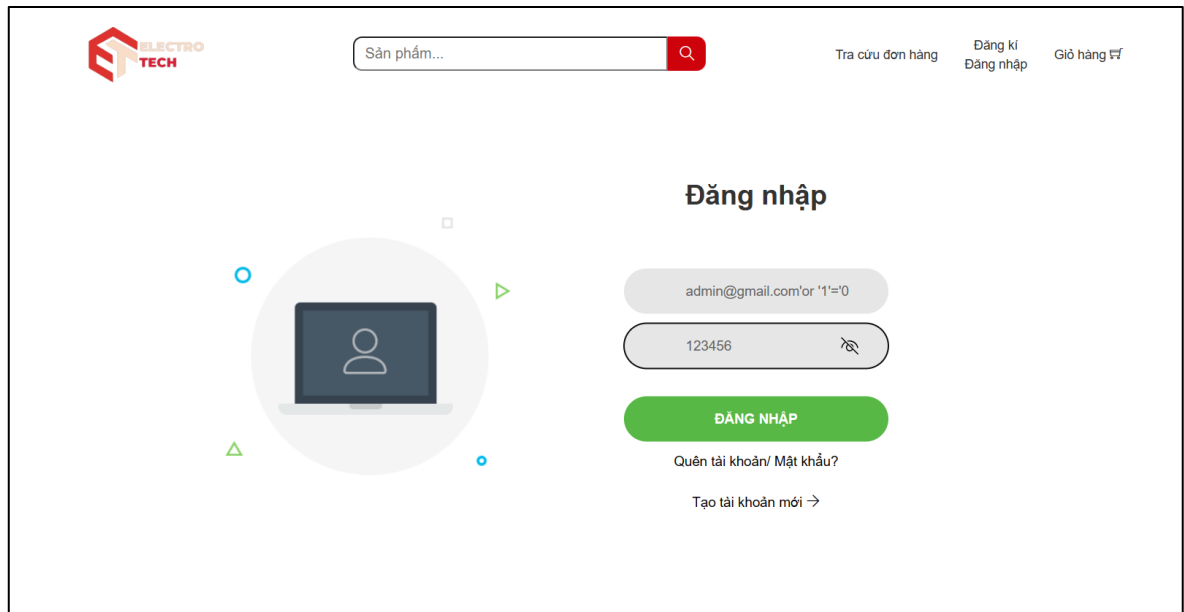
Hình 2.18: Kết quả thực hiện LAB 18

## 2.2. Thực hành trên WEB

### 2.2.1. Login

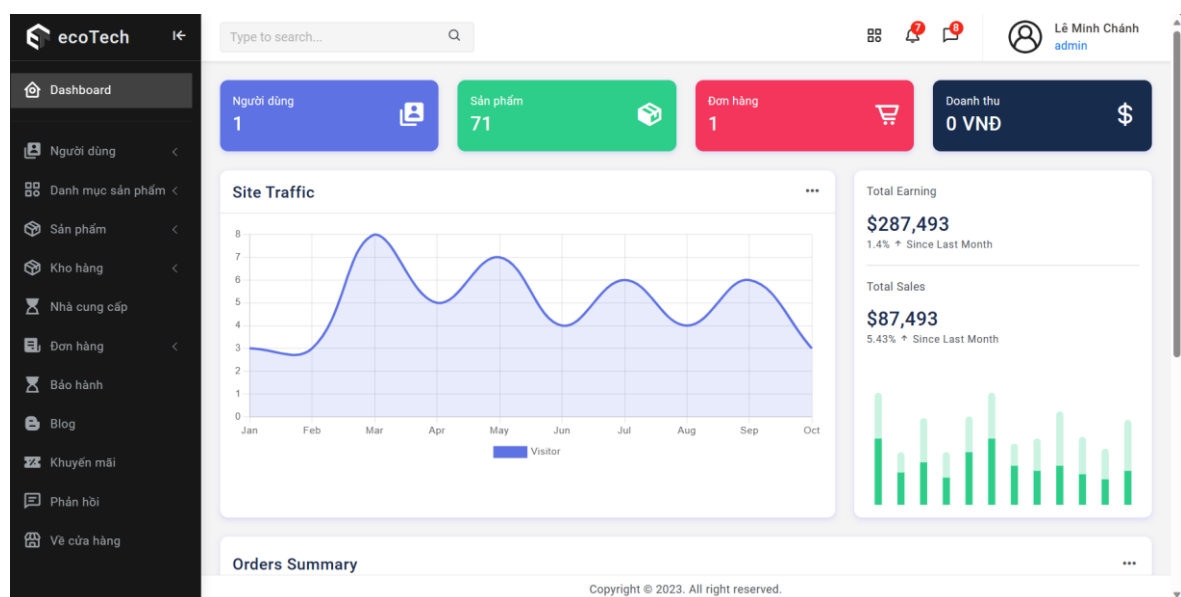
#### 2.2.1.1. Tấn công

Chèn câu lệnh '**OR '1'='0**' vào sau địa chỉ email và nhập 1 mật khẩu bất kì để đăng nhập vào trang web.



Hình 2.19: Tấn công login trên trang web

### Kết quả:



Hình 2.20: Tấn công thành công vào user admin

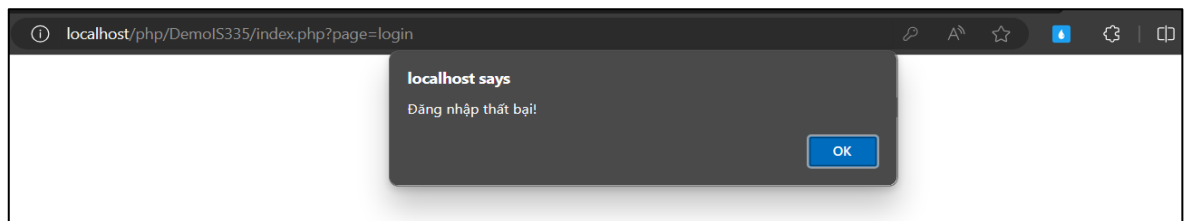
### 2.2.1.2. Phòng thủ

Sử dụng hàm **addslashes()**

```
// $email = ($_POST['email']);  
// $password = ($_POST['password']);  
  
// fix  
$email = addslashes($_POST['email']);  
$password = addslashes($_POST['password']);  
  
$sql = "SELECT * FROM USERS WHERE USER_EMAIL='$email' AND USER_PASSWORD='$password'";  
// $sql = "SELECT * FROM USERS WHERE USER_EMAIL='$email' AND USER_PASSWORD='$password'";  
$result = mysqli_query($connect, $sql);  
  
if (!mysqli_num_rows($result)) {  
    echo "<script>alert('Đăng nhập thất bại!');</script>";  
    die();  
}  
$row = mysqli_fetch_array($result);  
  
$_SESSION['email'] = $row['USER_EMAIL'];
```

Hình 2.21: Phòng thủ tấn công login bằng hàm **addslashes()**

Sau khi phòng thủ, thực hiện lại thao tác tấn công SQL Injection.

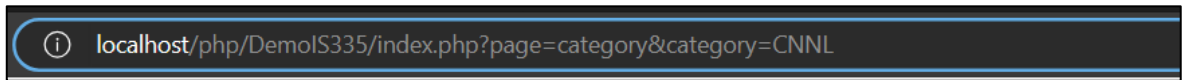


Hình 2.22: Kết quả tấn công login sau khi thực hiện phòng thủ

## 2.2.2. Database

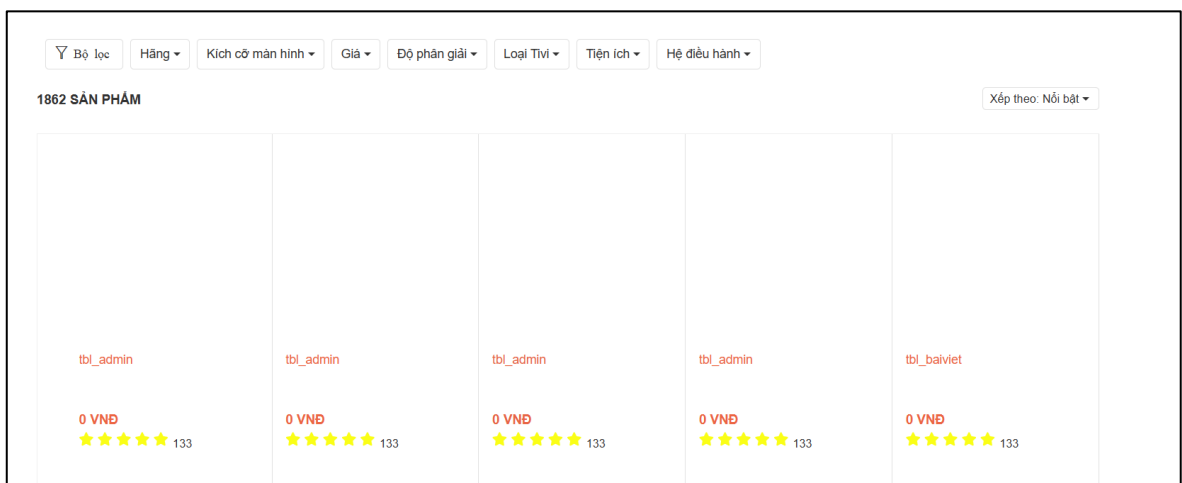
### 2.2.2.1. Tấn công

Thực hiện tấn công ở URL:



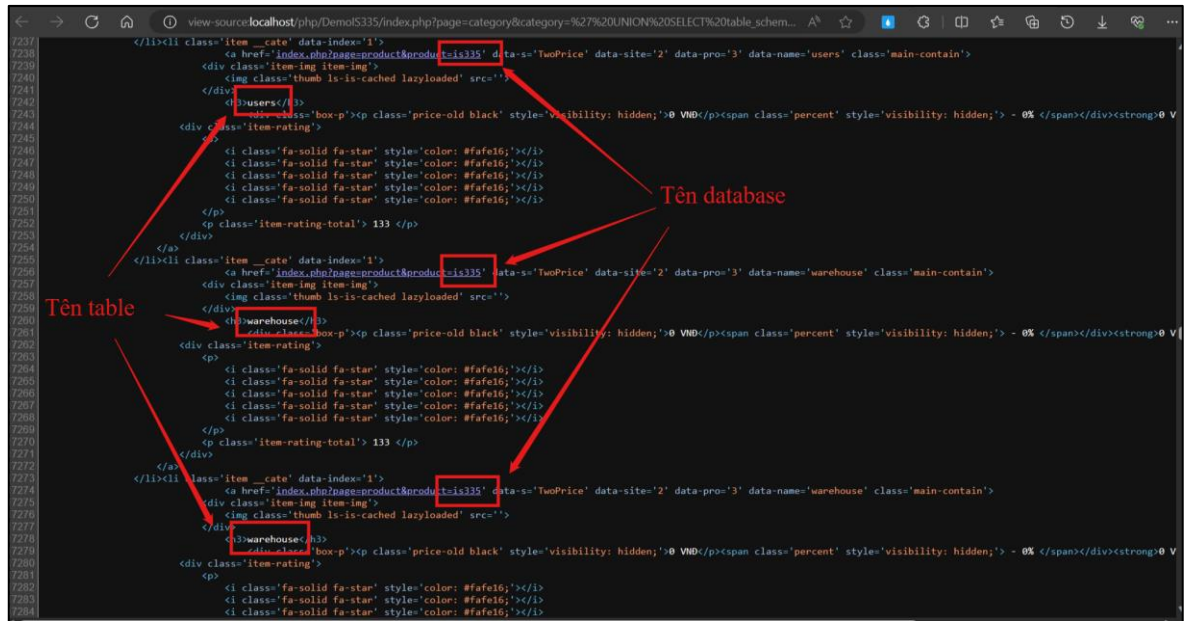
Hình 2.23: Tấn công trên URL

- Xem được toàn bộ tên database và columns của database có trong server



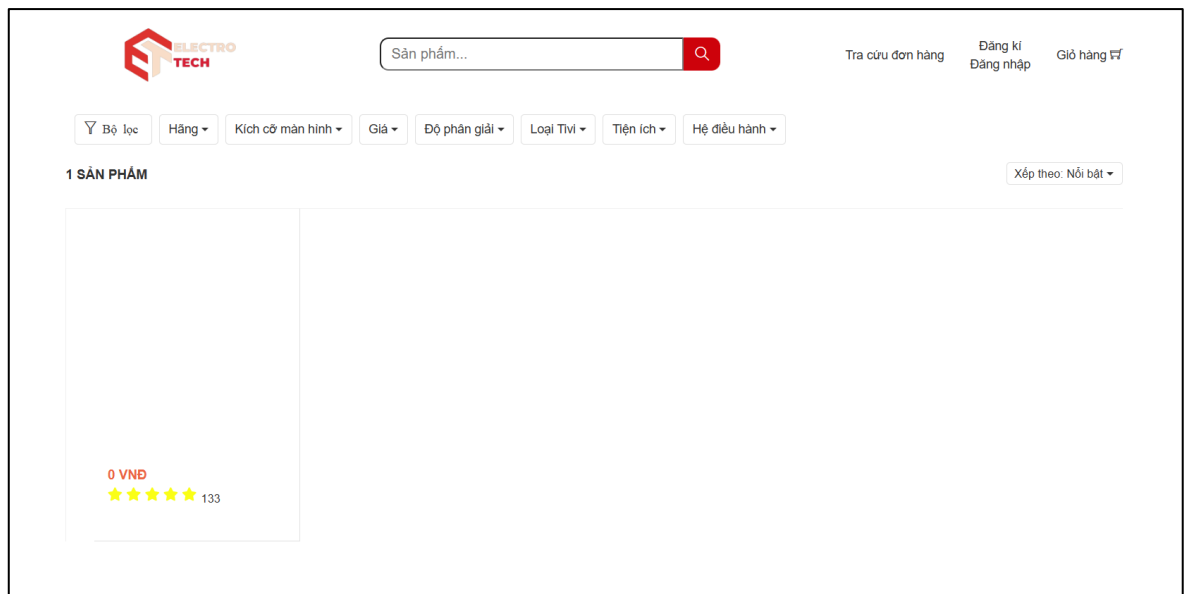
Hình 2.24: Kết quả xem được toàn bộ tên database và columns của trong database trong server



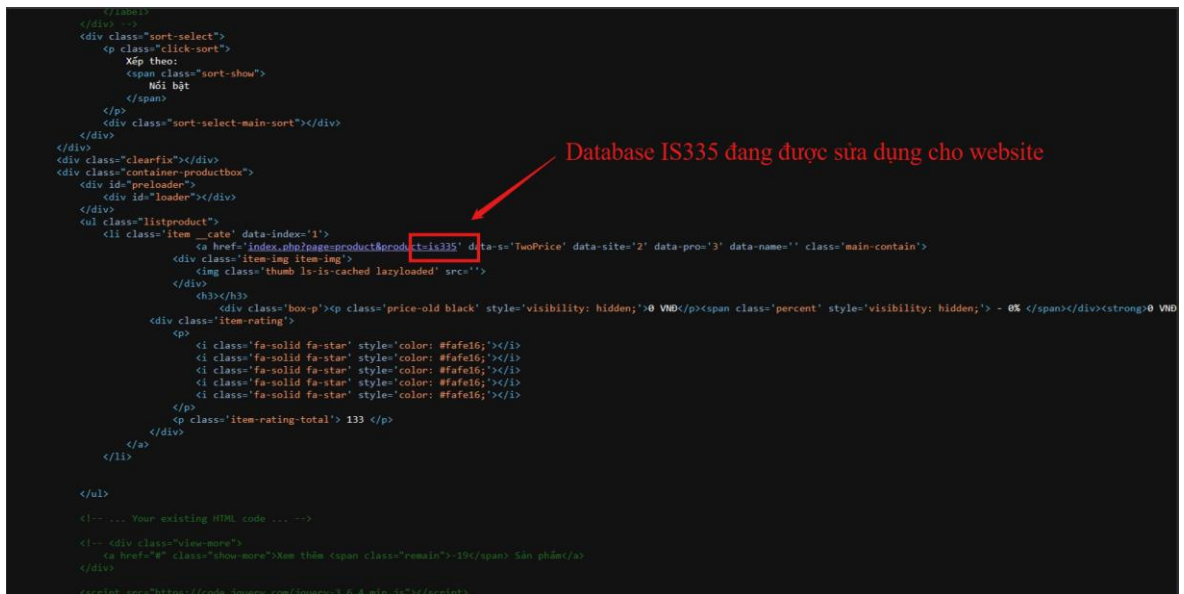


Hình 2.25: Kết quả được load dữ liệu lên trang web khi xem xong view-source

- Select được database hiện tại đang được sử dụng cho trang web



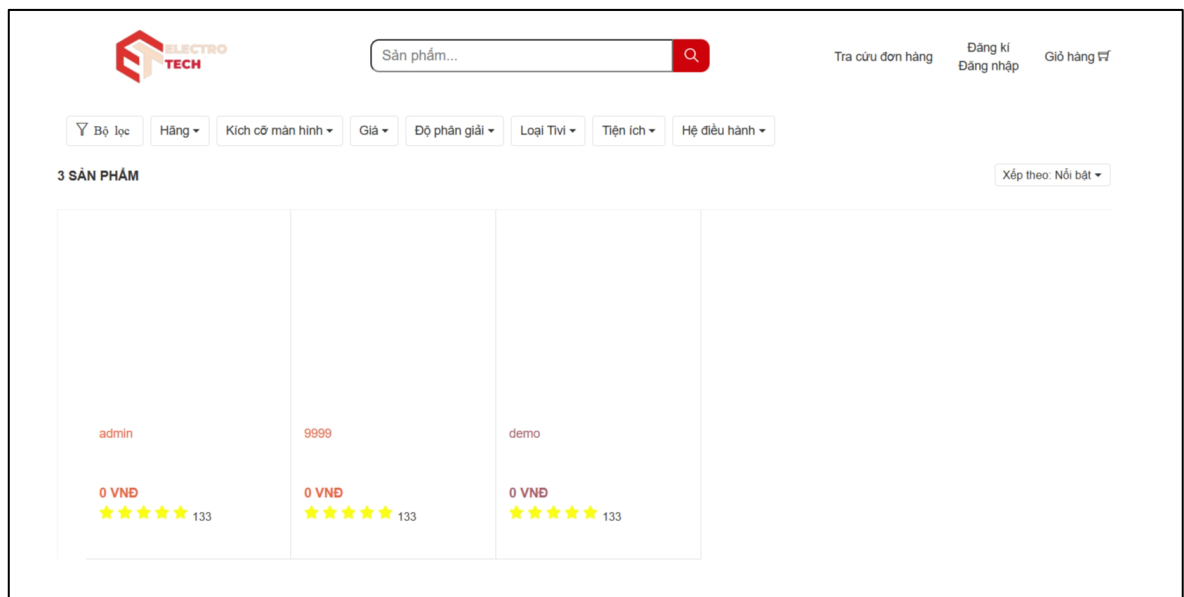
Hình 2.26: Kết quả xem được tên database đang được sử dụng hiện tại cho website



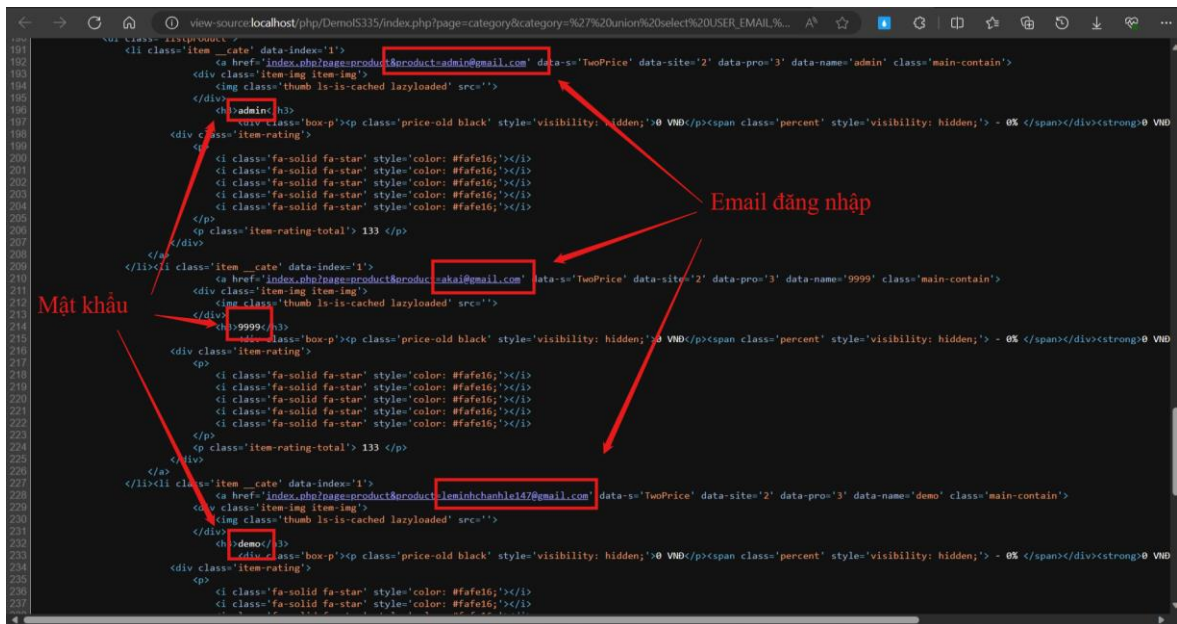
Hình 2.27: Kết quả được load dữ liệu lên trang web khi xem xong view-source

- Lấy được toàn bộ thông tin trong các database có trong server:

**ví dụ:** table **USERS** trong database hiện tại: **IS335**



Hình 2.28: Select thành công dữ liệu trong bảng users



Hình 2.29: Kết quả được load lên trong view-sources

### 2.2.2.2. Phòng thủ

#### Sử dụng hàm addslashes()

```
// $id = ($_GET['category']);

//fix
$id = addslashes($_GET['category']);

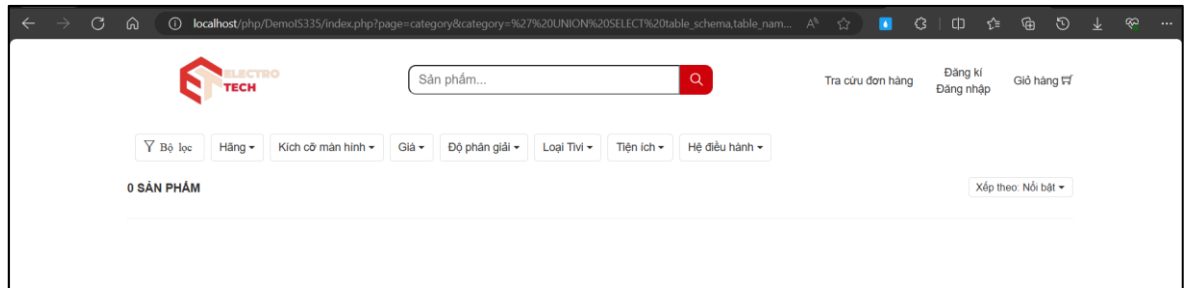
$name;
$count;

$sql = "SELECT * FROM products WHERE CATEGORY_ID = '$id' and PRODUCT_STATUS = 'active'";
$result = mysqli_query($connect, $sql);

if ($result) {
    $count = mysqli_num_rows($result);
    echo "<span> " . $count . " </span>";
}
```

Hình 2.30: Thực hiện phòng thủ bằng hàm addslashes()

- Sau khi phòng thủ, thực hiện lại thao tác tấn công SQL Injection
- Tất cả các mã cũ đều không hoạt động và không query được kết quả nào



*Hình 2.31: Tấn công thất bại sau khi phòng thủ*

Các thao tác thực hiện chi tiết xin tham khảo tại [đây](#).

### **Phần 3. KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN**

#### **3.1. Kết quả đạt được**

- Cả nhóm hiểu được kiến thức cơ bản cũng như nâng cao về An toàn và Bảo mật Hệ thống Thông tin.
- Cả nhóm làm việc hiệu quả, hoàn thành tốt môn học cũng như bài tập lớn.
- Hoàn thành 18 Lab và thực hiện được trên web của nhóm về SQL Injection.
- Hiểu rõ về lý thuyết cũng như cách thực hiện phòng thủ tấn công về SQL Injection.

#### **3.2. Hướng phát triển**

- Vì kiến thức chuyên môn còn hạn chế nên nhóm còn thực hiện những thao tác đơn giản, nhóm sẽ tìm hiểu và phát triển kỹ năng hơn.
- Ngoài SQL Injection, nhóm sẽ tìm hiểu nhiều kỹ năng trong an toàn và bảo mật hơn.

#### **Phần 4. TÀI LIỆU THAM KHẢO**

- Tìm hiểu về SQL Injection:  
<https://www.hacksplaining.com/prevention/sql-injection>  
<https://portswigger.net/web-security/sql-injection>
- Slide của Giảng viên Hà Lê Hoài Trung, môn An toàn và Bảo mật HTTP, khoa Hệ thống Thông tin, Trường Đại học Công nghệ Thông tin – ĐHQG TP.HCM