Họ và tên : Hà Hoàng Vũ

MSSV: 1050070054

Lớp: 10\_ĐH\_TMĐT

**BÁO CÁO BÀI THỰC HÀNH SỐ 8**

**KIỂM THỬ LỖ HỔNG SQL INJECTION**

**Câu 1(3.5 điểm)**

Kiểm thử lỗ hổng SQL Injection trên các trang web :

- Xác định các tham số đầu vào và cách thức chúng được gửi tới server (0.25 điểm)

- Phân tích và phán đoán kiểu truy vấn (0.25 điểm)

- Phân tích và phán đoán vị trí của các giá trị tham số đầu vào được sử dụng trong câu truy vấn

(0.5 điểm)

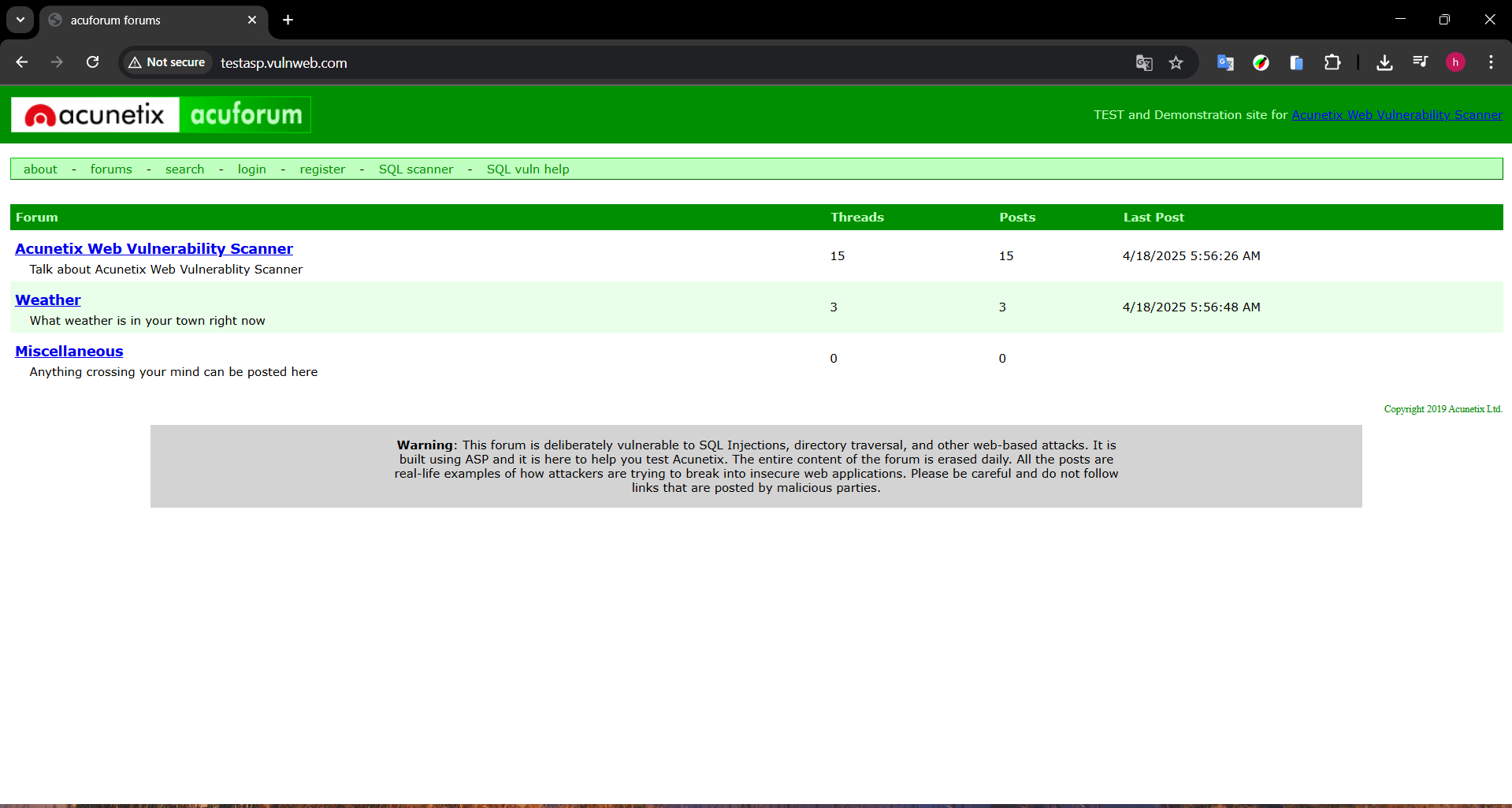
- Xác định mức độ ảnh hưởng của lỗ hổng trên mỗi giá trị đầu vào (2.5 điểm)

Lưu ý:

- Nói rõ mục đích sử dụng các giá trị kiểm thử

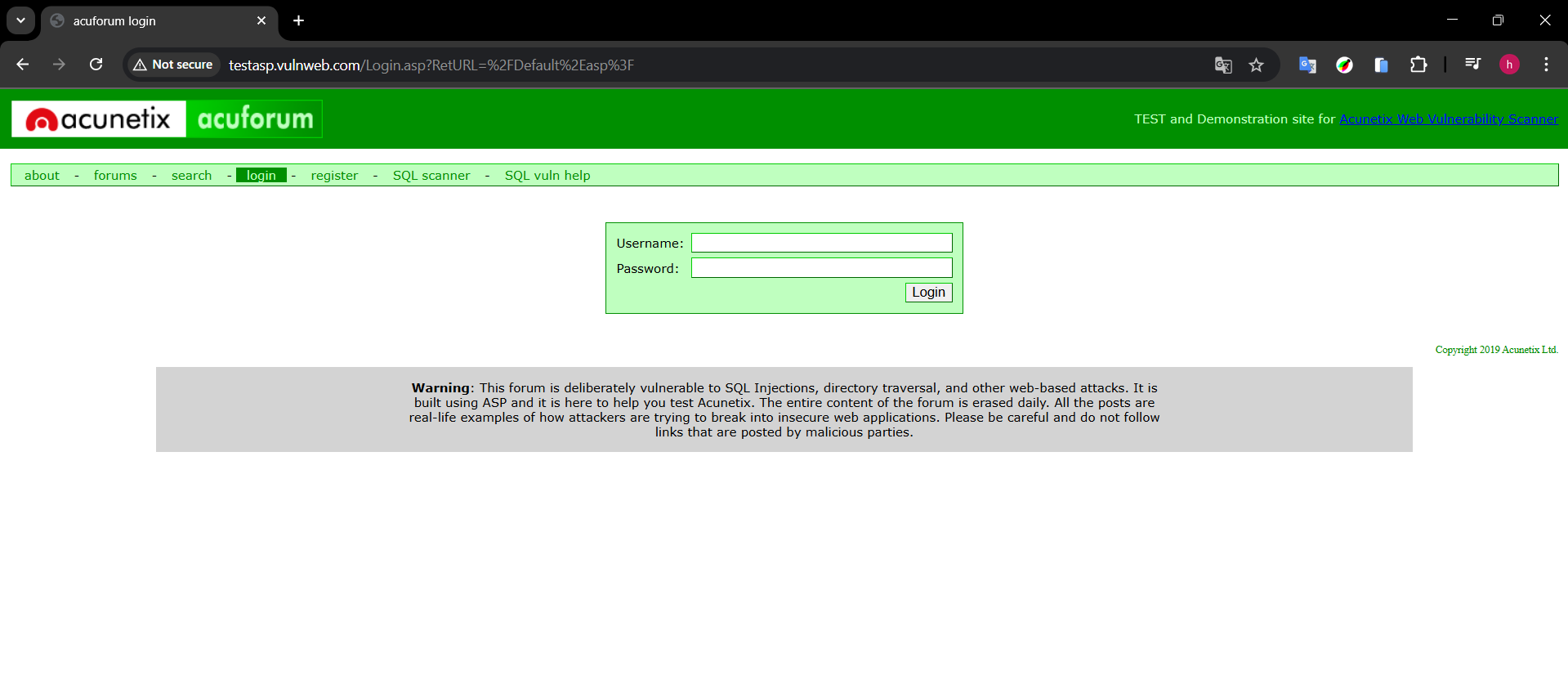
- Giải thích rõ các phán đoán và suy luận từ kết quả khi sử dụng mỗi giá trị kiểm thử

Trang web kiểm thử : [**http://testasp.vulnweb.com**](http://testasp.vulnweb.com)

Truy cập vào địa chỉ: http://testasp.vulnweb.com 

Khám phá trang web và quan sát URL khi thực hiện các thao tác

Ví dụ : click vào login

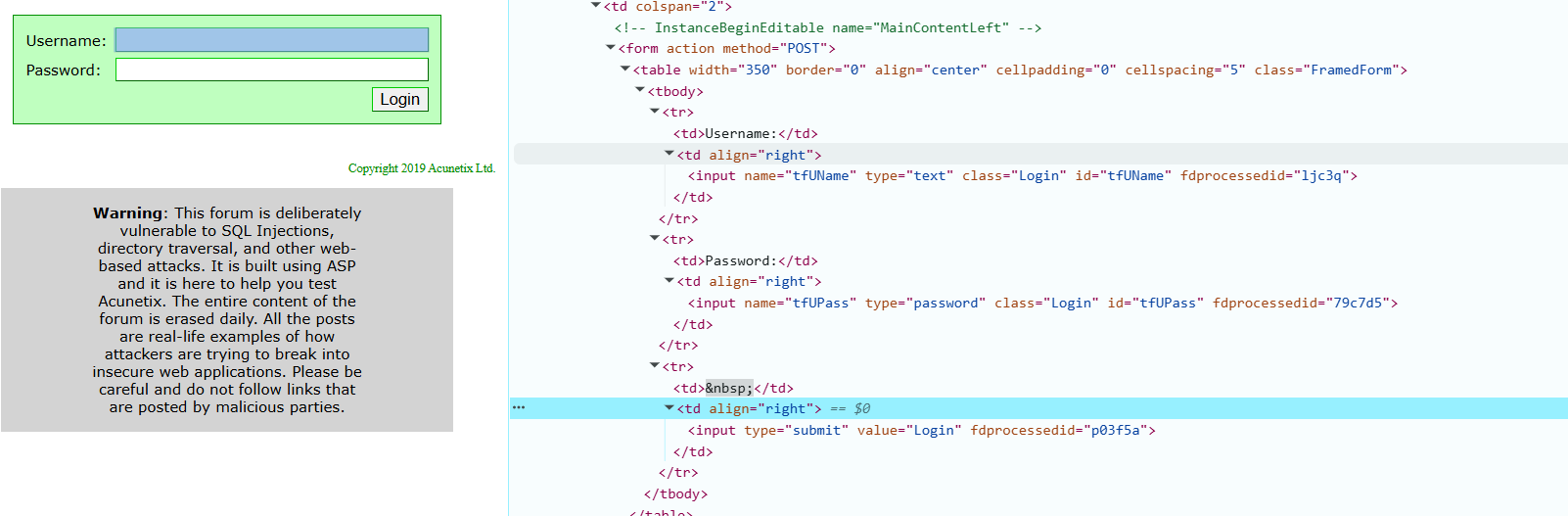


**Form login có 2 trường:**

* Username
* Password

**Cách thức gửi dữ liệu:**

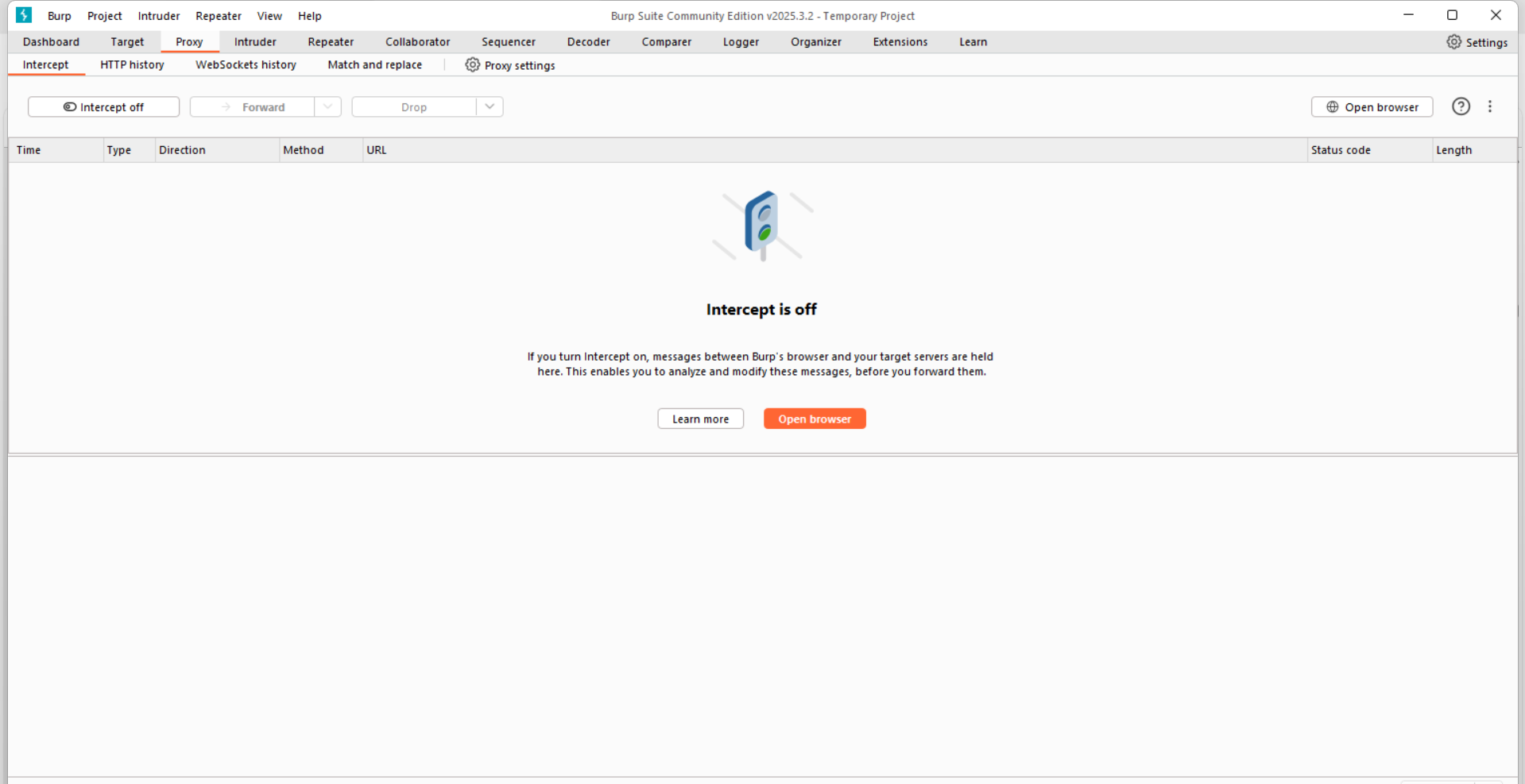
* Dữ liệu được gửi thông qua **form POST** đến Login.asp.
* Đây là biểu mẫu HTML sử dụng thẻ <input> và nút <button> dạng submit



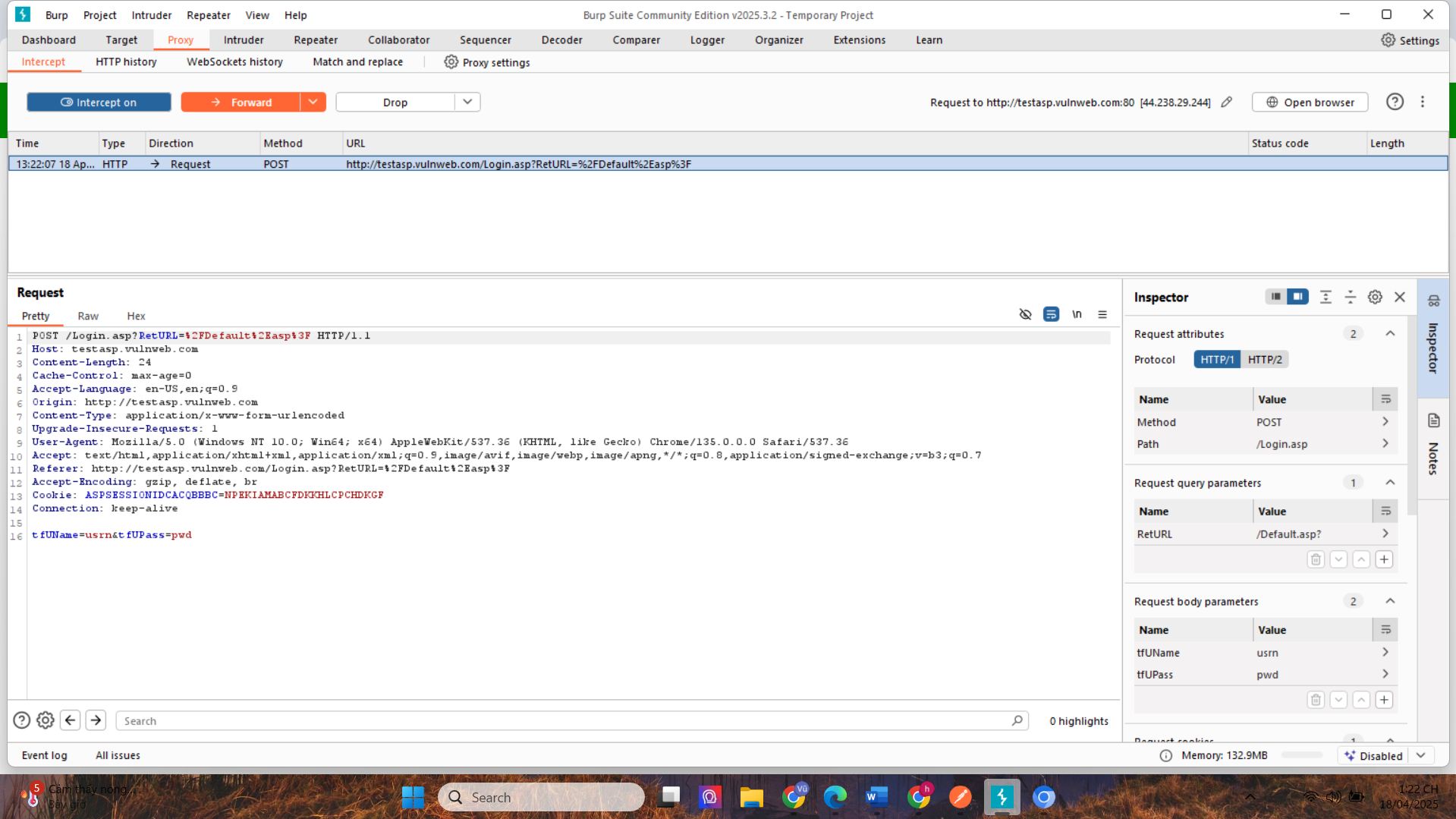
Dựa trên chức năng đăng nhập, có thể phán đoán rằng server thực hiện truy vấn SQL để xác thực người dùng:

SELECT \* FROM users WHERE username = 'tfUName' AND password = 'tfUPass'

Dùng **Burp Suite** (Proxy → Intercept) để chặn và xem chi tiết body:



Gửi Request và xem lại burp suite:



POST /Login.asp?RetURL=%2FDefault%2Easp HTTP/1.1

Host: testasp.vulnweb.com

Content-Type: application/x-www-form-urlencoded

Content-Length: 24

tfUName=usrn&tfUPass=pwd

Dựa vào chức năng login, suy đoán server đang chạy ASP với câu:

SELECT \*

FROM Users

WHERE username = '[tfUName]'

AND password = '[tfUPass]';

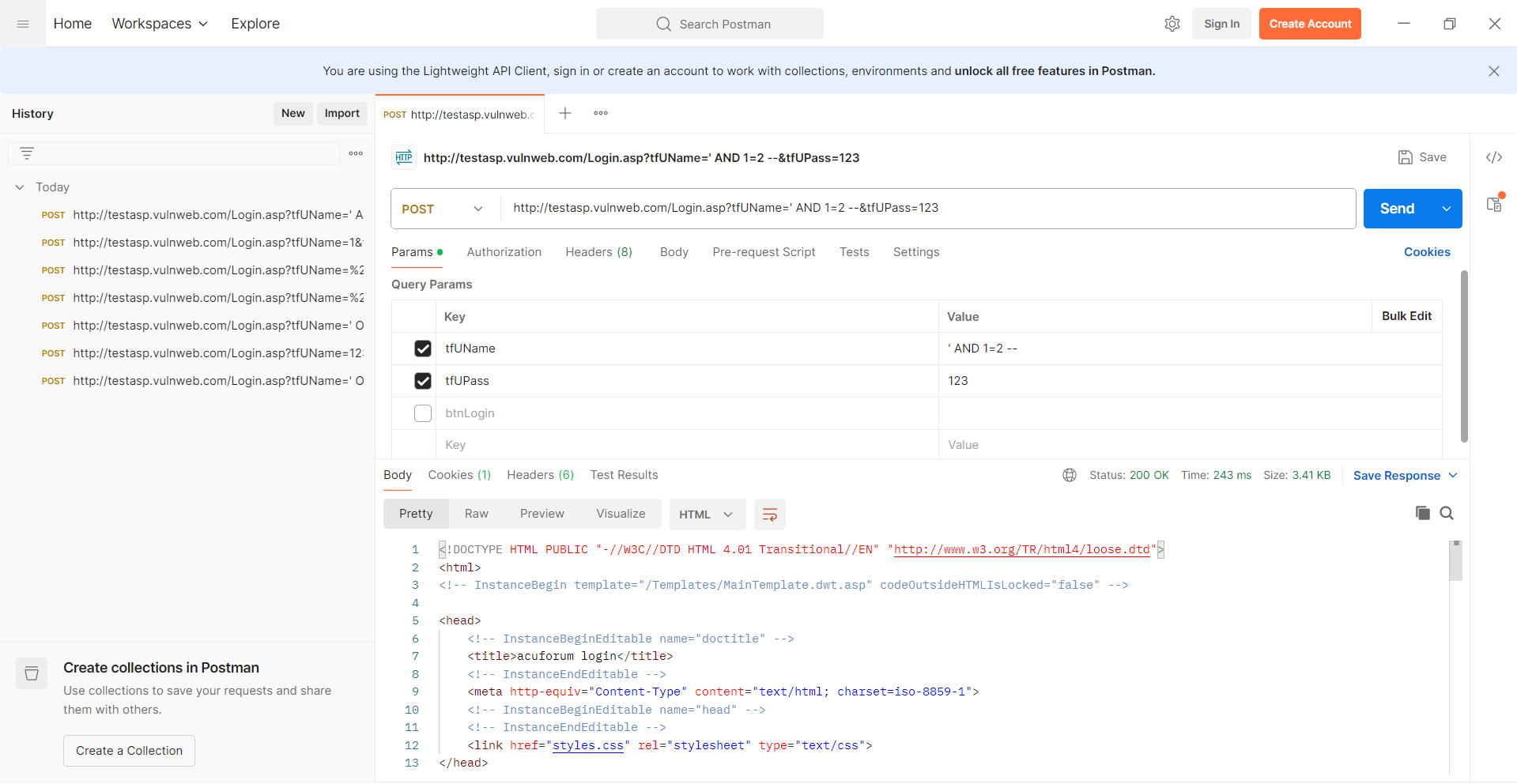
Hai tham số được đóng trong dấu nháy đơn → cho phép chèn chuỗi, toán tử, comment…

* **tfUName** → nằm trong username = '…'
* **tfUPass** → nằm trong password = '…'

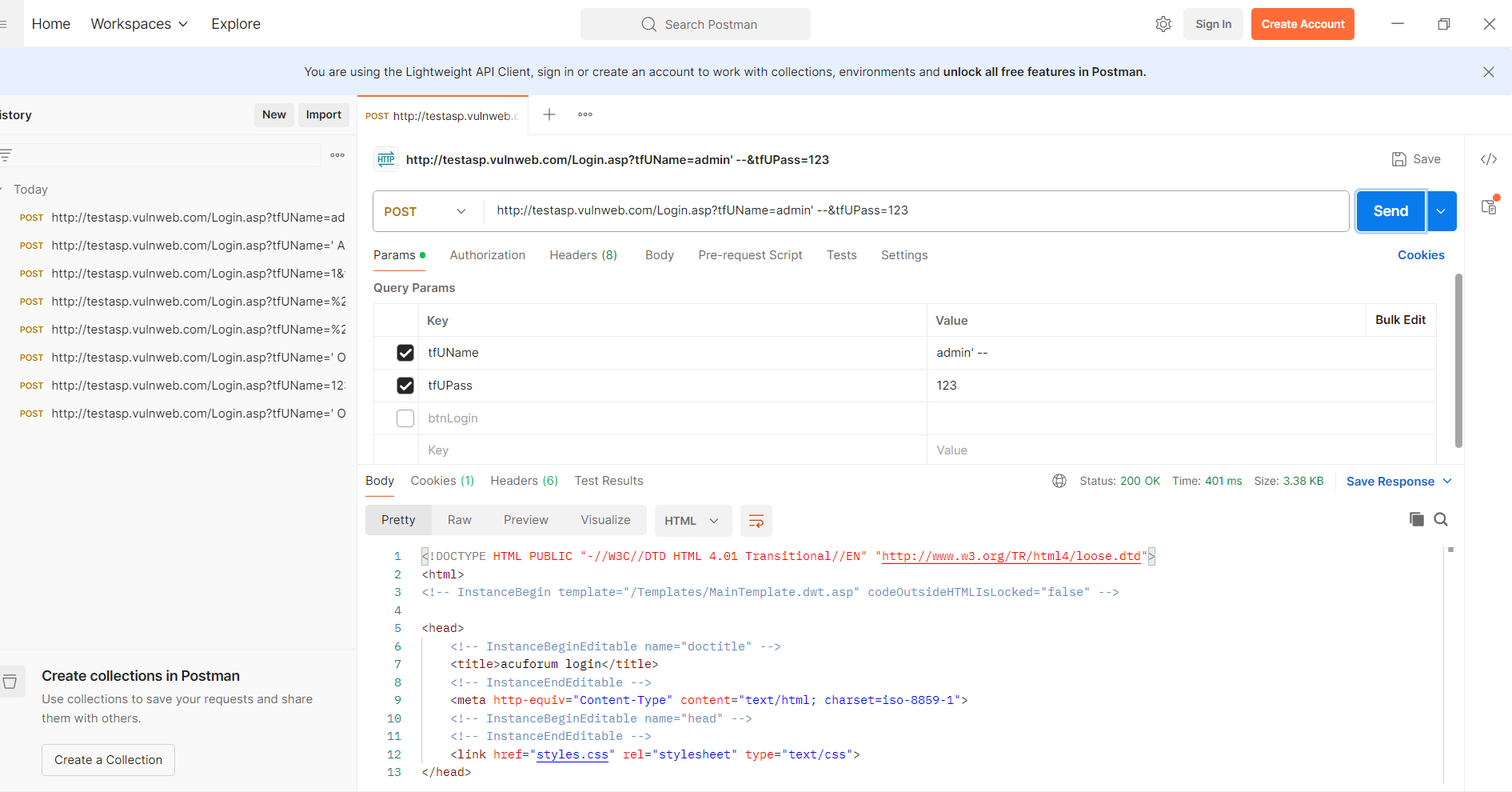
Vì cả hai đều được bao bởi '…', bất kỳ dấu ' nào trong input sẽ **phá vỡ cú pháp** và cho phép chèn thêm điều kiện hoặc comment.

| **Payload (tfUName)** | **Mục đích kiểm thử** | **Kết quả quan sát** | **Phán đoán & Suy luận từ kết quả** |
| --- | --- | --- | --- |
| **1**. ' OR '1'='1 | Tautology‑based: bỏ qua điều kiện username | **302 Redirect** | Điều kiện '' OR '1'='1' luôn đúng ⇒ **vẫn login được** ⇒ tham số không được lọc, dễ bị SQLi. |
| **2**. admin' -- | Comment‑based: bỏ qua luôn phần kiểm tra password | **302 Redirect** | Phần AND password='…' bị comment (--) ⇒ nếu có user “admin” ⇒ login thành công ⇒ lỗ hổng nghiêm trọng. |
| **3**. ' AND 1=2 -- | Boolean-based blind: phân biệt True/False | **200 OK + form** (không login) | 1=2 sai ⇒ không login. Kết quả khác với 1=1 (login được) ⇒ có thể dò dữ liệu từng bit qua True/False. |
| **4**. '; WAITFOR DELAY '0:0:5' -- | Time-based blind: kiểm tra chèn lệnh delay | **Response chậm ~5s** | Lệnh SQL WAITFOR DELAY được thực thi ⇒ server hỗ trợ time‑based injection. |

' OR '1'='1



admin' --

****

Postman cho thấy đăng nhập thành công và trả về trang Home

**Đánh Giá Mức Độ Ảnh Hưởng**

* **Endpoint** /Login.asp hoàn toàn **dễ bị SQL Injection** ở cả tfUName và tfUPass.
* **Mức độ nghiêm trọng**: Rất cao → bypass login, leo quyền, rò rỉ dữ liệu, thậm chí chiếm toàn quyền.
* **Khuyến nghị**:
  1. Dùng **Prepared Statements** (parametrized queries).
  2. Escape & validate đầu vào (loại bỏ ', --, ;, WAITFOR, v.v.).
  3. Tắt hiển thị lỗi chi tiết ra client.
  4. Sử dụng account DB với **least privilege**.

**Câu 2 (3.5 điểm)**

Kiểm thử lỗ hổng SQL Injection trên trang web <http://php.testsparker.com/auth/login.php>

- Xác định các tham số đầu vào và cách thức chúng được gửi tới server (0.25 điểm)

- Phân tích và phán đoán kiểu truy vấn, phân tích và phán đoán vị trí của các tham số đầu vào được

sử dụng trong câu truy vấn (0.25 điểm)

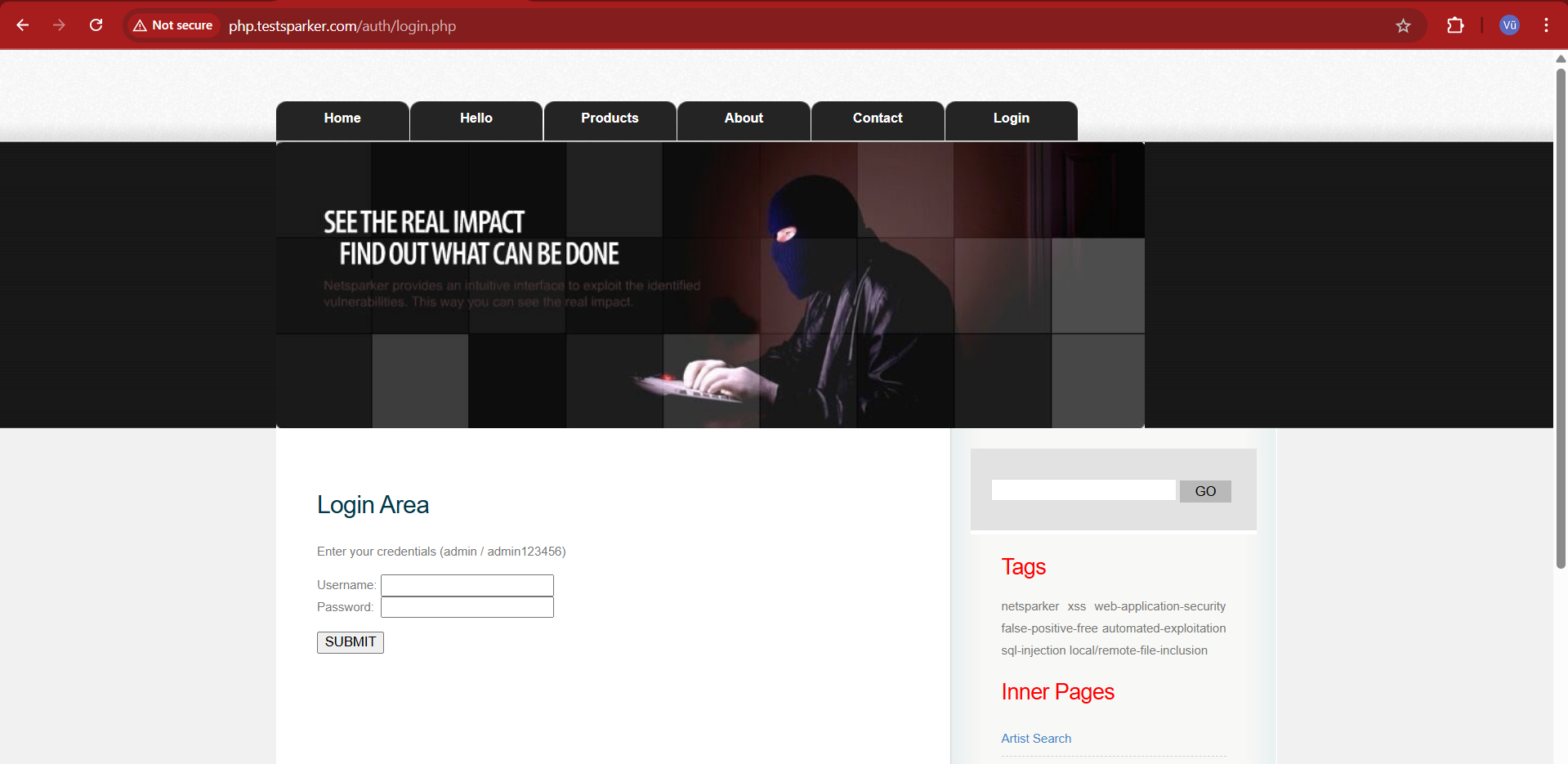
- Xác định mức độ ảnh hưởng của lỗ hổng trên mỗi giá trị đầu vào (3 điểm)

Lưu ý:

- Nói rõ mục đích sử dụng các giá trị kiểm thử

- Giải thích rõ các phán đoán và suy luận từ kết quả khi sử dụng mỗi giá trị kiểm thử

Truy Cập Vào trang [**http://php.testsparker.com/auth/login.php**](http://php.testsparker.com/auth/login.php)



Phân tích form:



**Endpoint** : /auth/control.php

**Tham số đầu vào :**

Username

Password

**Phương thức** : Post

Gửi về sever bằng submit qua value Login

**Qua quan sát cho thấy có thể sever xử lý bằng lệnh :**

SELECT \*

FROM users

WHERE username = '[username]'

AND password = '[password]';

Nếu không có cơ chế escape/parametrized, việc chèn ' sẽ phá vỡ cú pháp và cho phép thêm mệnh đề, comment, hoặc UNION.

Kiểm thử với input

Username = admin

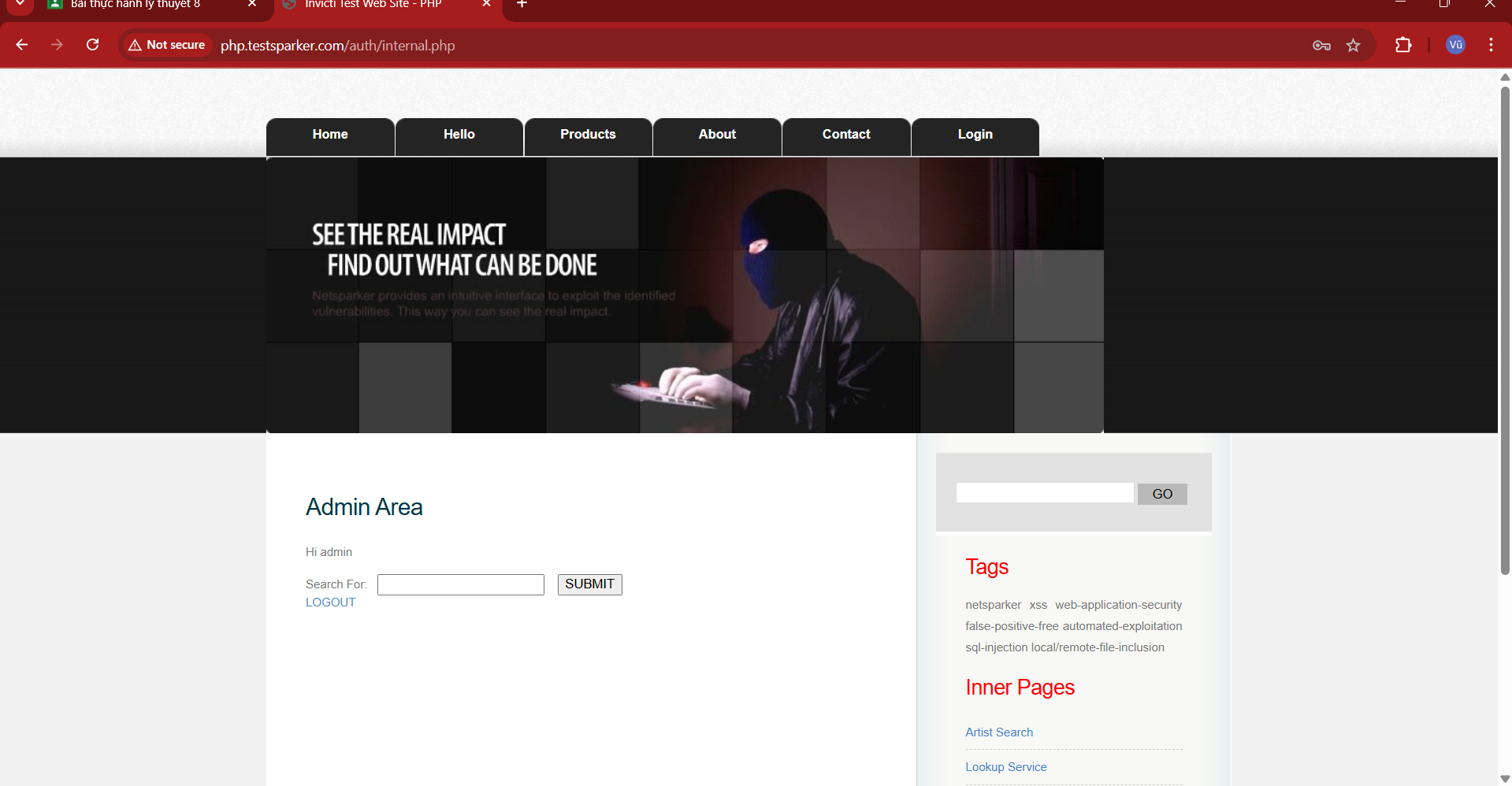
Password = ' OR 1=1--

**Có thể truy vấn thành**

SELECT \* FROM users

WHERE username = admin

AND password = ' OR 1=1--



Kết quả cho thấy đăng nhập thành công Admin và bỏ qua mật khẩu.

Đánh giá mức độ ảnh hưởng :  
 Endpoint login.php dễ bị mọi kiểu SQL Injection: Tautology, Comment, Boolean‑Blind, Time‑Blind, Union‑Based, Error‑Based.

**Mức độ nghiêm trọng**: Rất cao → bypass login, chiếm quyền admin, dump toàn bộ CSDL.

**Câu 3 (3 điểm)**

Kiểm thử lỗ hổng SQL Injection trên trang web http://aspnet.testsparker.com/

- Xác định các tham số đầu vào và cách thức chúng được gửi tới server (0.25 điểm)

- Phân tích và phán đoán kiểu truy vấn, vị trí của các tham số đầu vào được sử dụng trong câu truy

vấn (0.25 điểm)

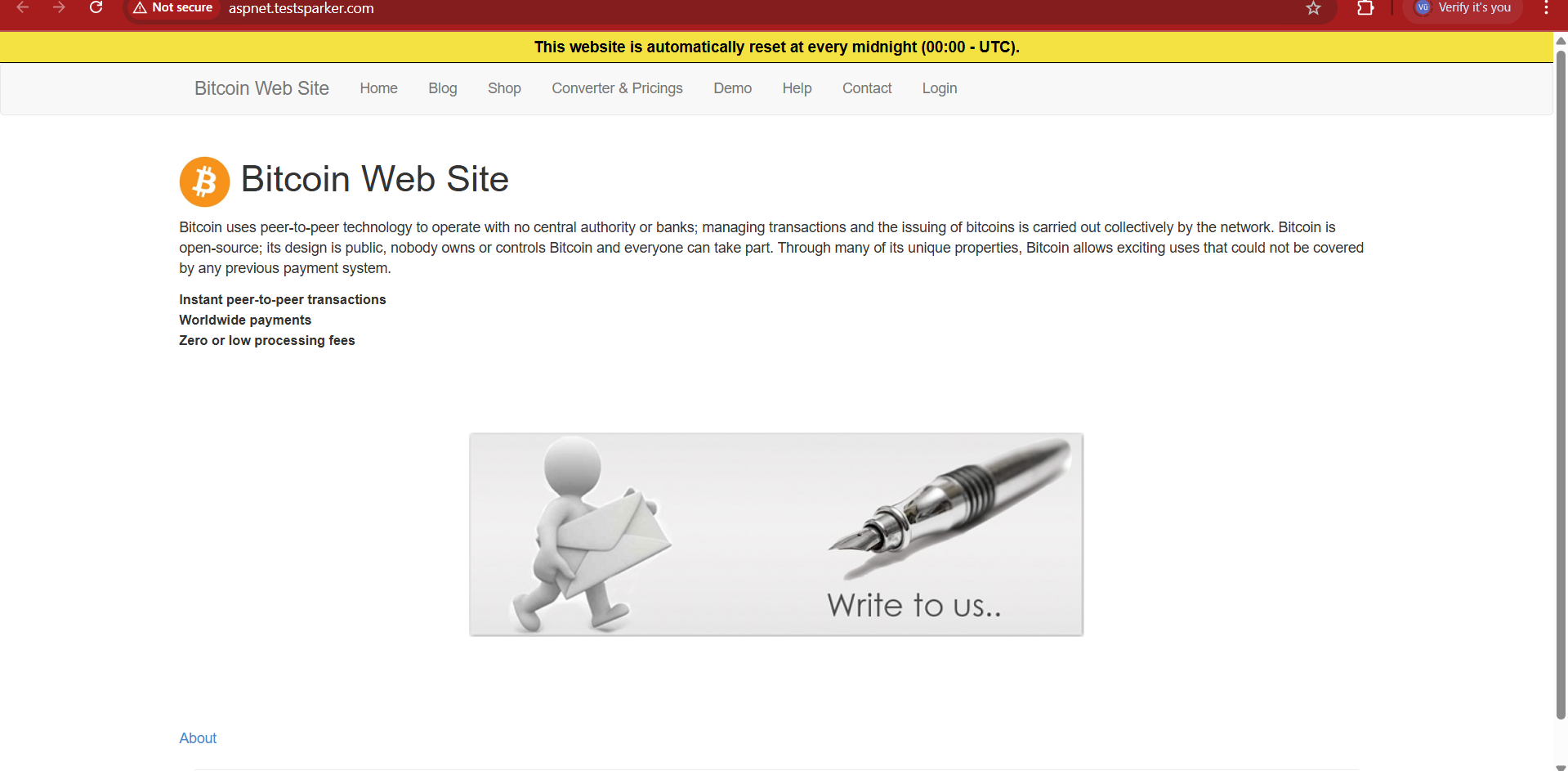
- Xác định mức độ ảnh hưởng của lỗ hổng trên mỗi giá trị đầu vào (2.5 điểm)

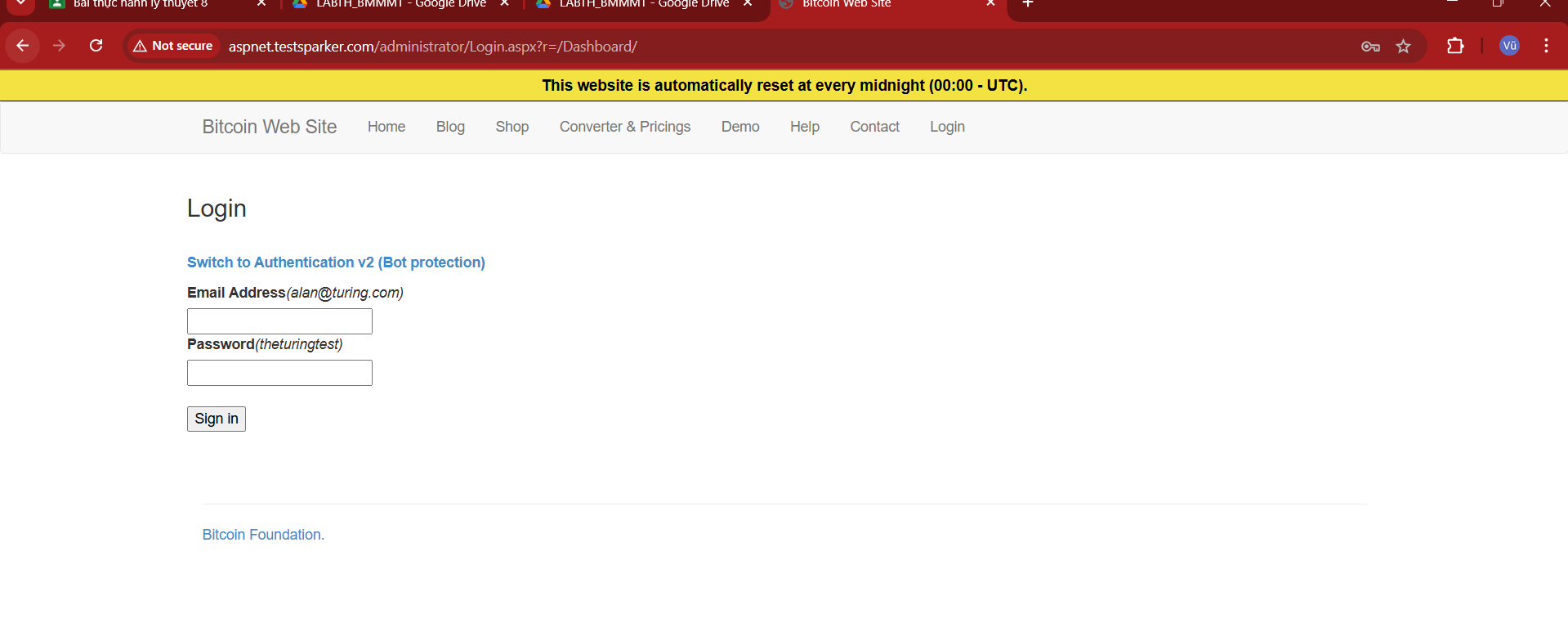
Lưu ý:

- Nói rõ mục đích sử dụng các giá trị kiểm thử

- Giải thích rõ các phán đoán và suy luận từ kết quả khi sử dụng mỗi giá trị kiểm thử\

Truy cập vào trang web :



Chọn chức năng ví dụ login :  


Kiểm tra bằng dev tool:



Tham số đầu vào :

Email

Password

Endpoint : POST /administrator/Login.aspx?r=%2fDashboard%2f

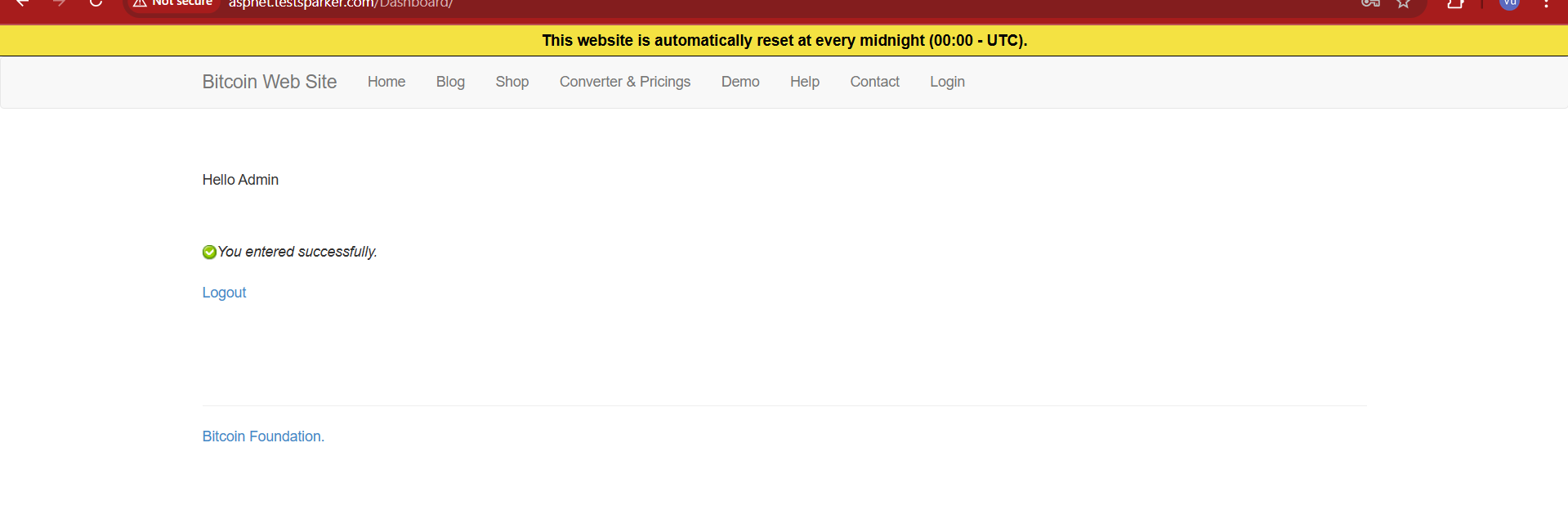
Gỉa định sever xử lý :  
SELECT \* FROM Admins

WHERE UserName = '[UserName]'

AND Password = '[Password]'

### ****Payload 1 – Bypass login (Tautology)****

* **Username**: ' OR '1'='1
* **Password**: abc123
* **Mục đích**: Làm điều kiện WHERE luôn đúng



Kết quả : Đăng nhập thành công

Đánh giá mức độ ảnh hưởng :

| **Tiêu chí** | **Mức độ** | **Giải thích** |
| --- | --- | --- |
| **Tác động hệ thống** | 🔴 **Cao** | Bypass, chiếm tài khoản admin, đọc dữ liệu |
| **Khả năng khai thác** | 🟡 **Trung bình - Dễ** | Gửi thủ công bằng Postman hoặc Burp |
| **Tác động người dùng** | 🔴 **Cao** | Rò rỉ thông tin nhạy cảm của user |
| **Tác động bảo mật** | 🔴 **Rất cao** | Mở đường tấn công chiếm toàn bộ hệ thống |