**BÁO CÁO BÀI THỰC HÀNH SỐ 8**

**KIỂM THỬ LỖ HỔNG SQL INJECTION**

***Họ và tên: Nguyễn Khánh Duy***

***Yêu cầu kết quả thực hành:***

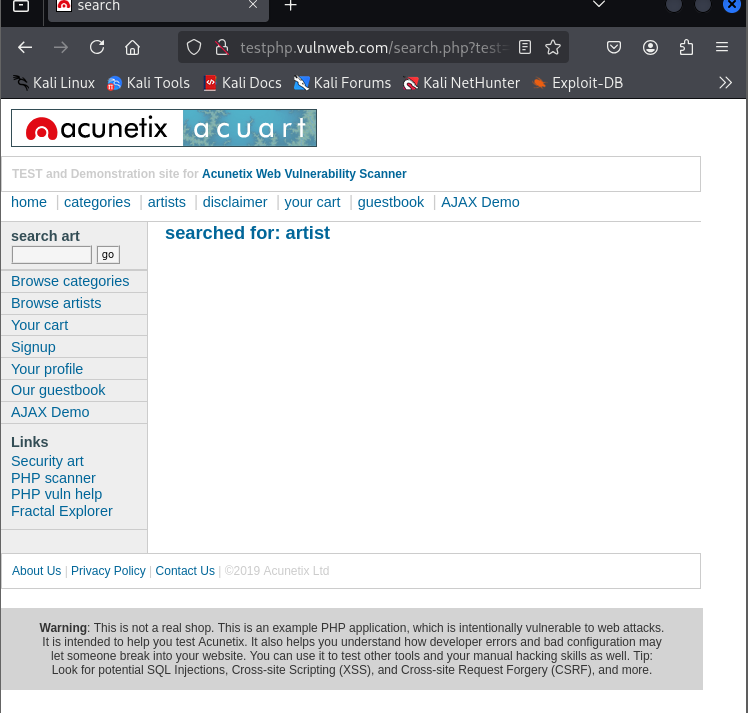
Báo cáo kết quả thực hành có tên theo định dạng HoTenSV\_MSSV\_Lab08.docx

**Câu 1*(3.5 điểm)***

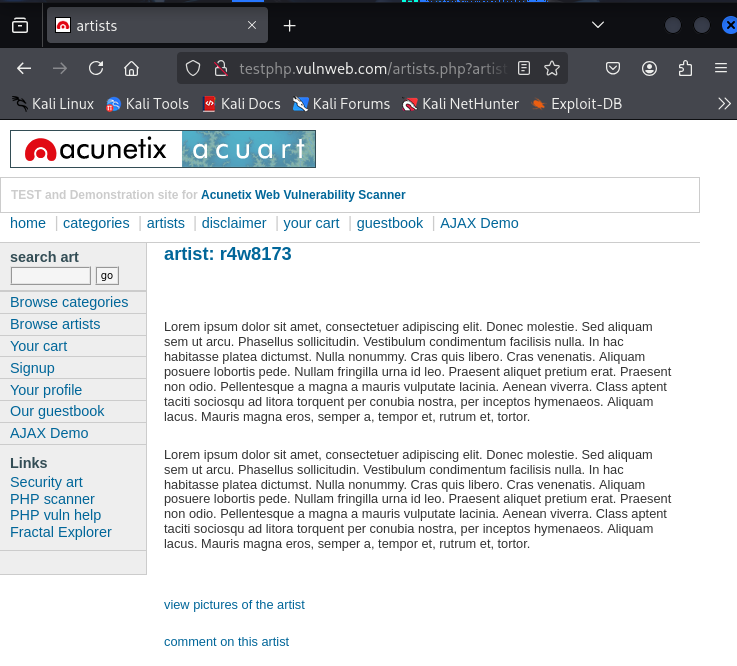
Kiểm thử lỗ hổng SQL Injection trên các trang web :

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

Tham số **artist** được truyền thông qua phương thức **GET** trên URL. Đây là đầu vào có thể điều chỉnh dễ dàng trên trình duyệt, dễ bị tấn công nếu không được kiểm soát.



http://testphp.vulnweb.com/artists.php?artist=1



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

SELECT \* FROM artists WHERE id = '1';

* Dựa trên URL và nội dung hiển thị, có thể suy đoán rằng giá trị artist=1 được dùng để truy vấn thông tin nghệ sĩ có id=1
* 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)**

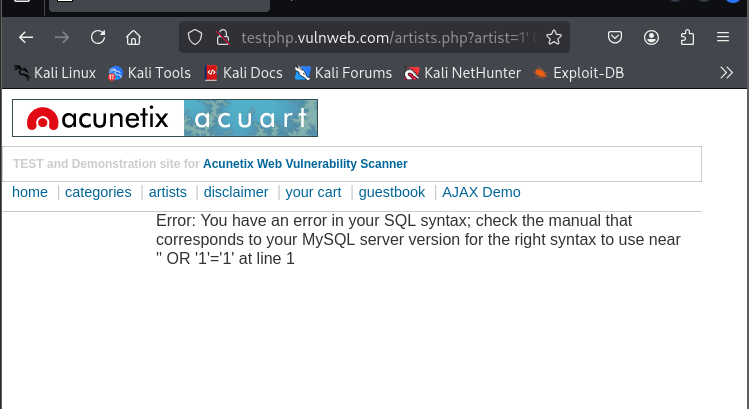
Tham số artist nằm trong mệnh đề WHERE của câu truy vấn SQL.

Nó ảnh hưởng trực tiếp đến việc **lọc dữ liệu** trong bảng artists.

→ Nếu bị thay đổi, hacker có thể kiểm soát dữ liệu trả về hoặc làm thay đổi logic truy vấn.

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

http://testphp.vulnweb.com/artists.php?artist=1' OR '1'='1

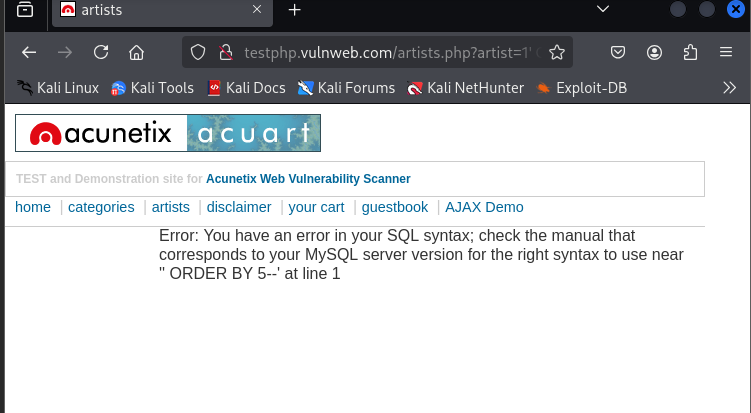


Dùng để kiểm tra xem ứng dụng có lọc dữ liệu đầu vào hay không. Nếu bị lỗi, sẽ truy vấn trả về toàn bộ.

Hệ thống hiển thị tất cả thông tin nghệ sĩ, thay vì chỉ một ID cụ thể.

→ Có thể thực hiện truy vấn luôn đúng (TRUE) ⇒ Bypass điều kiện WHERE. Chứng tỏ ứng dụng **không lọc dữ liệu đầu vào** ⇒ Dễ bị tấn công SQLi.

http://testphp.vulnweb.com/artists.php?artist=1' ORDER BY 5--

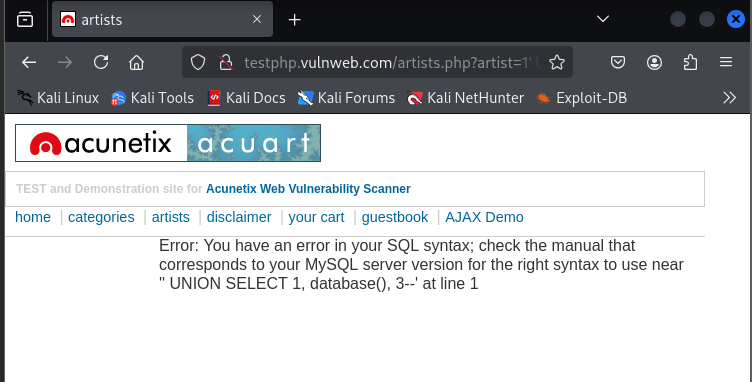


Kiểm tra số lượng cột trong câu lệnh SELECT phía backend.

Nếu số lượng cột không đúng → trả về lỗi SQL. Nếu đúng → trang vẫn hiển thị bình thường.

→ Từ kết quả, ta có thể suy đoán được **số cột** trong truy vấn → phục vụ cho khai thác SQLi kiểu UNION.

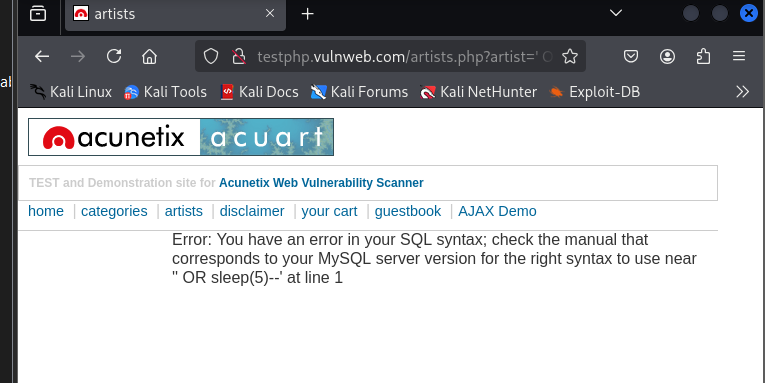
http://testphp.vulnweb.com/artists.php?artist=1%27%20UNION%20SELECT%201,%20database(),%203--



Kiểm tra khả năng khai thác SQL Injection thông qua UNION SELECT để **lấy dữ liệu từ hệ thống**

Tên cơ sở dữ liệu (ví dụ: acuart) được hiển thị ở vị trí tên nghệ sĩ trên trang.

→ Truy vấn UNION được thực thi ⇒ chứng minh hệ thống **bị lỗi SQLi kiểu UNION-based**. Từ đó kẻ tấn công có thể truy vấn bảng, cột, dữ liệu nhạy cảm.



Kiểm tra **SQL Injection dạng Blind** bằng kỹ thuật time-based (dựa vào thời gian phản hồi).

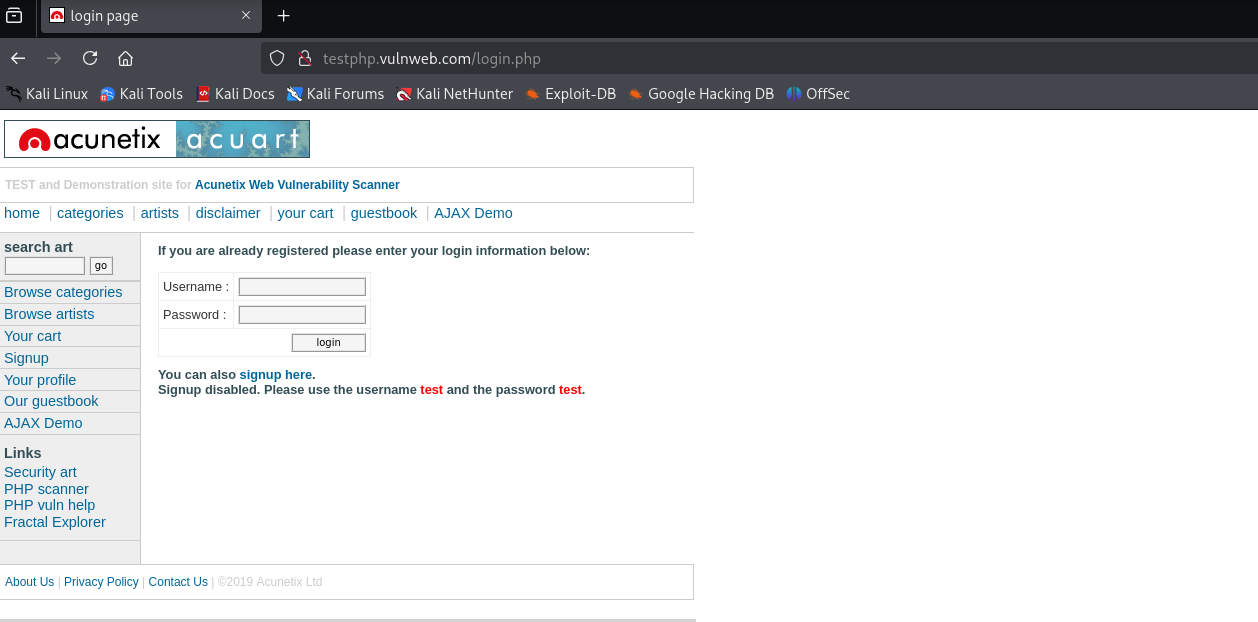
Trang web phản hồi chậm hơn rõ rệt (≈5 giây).

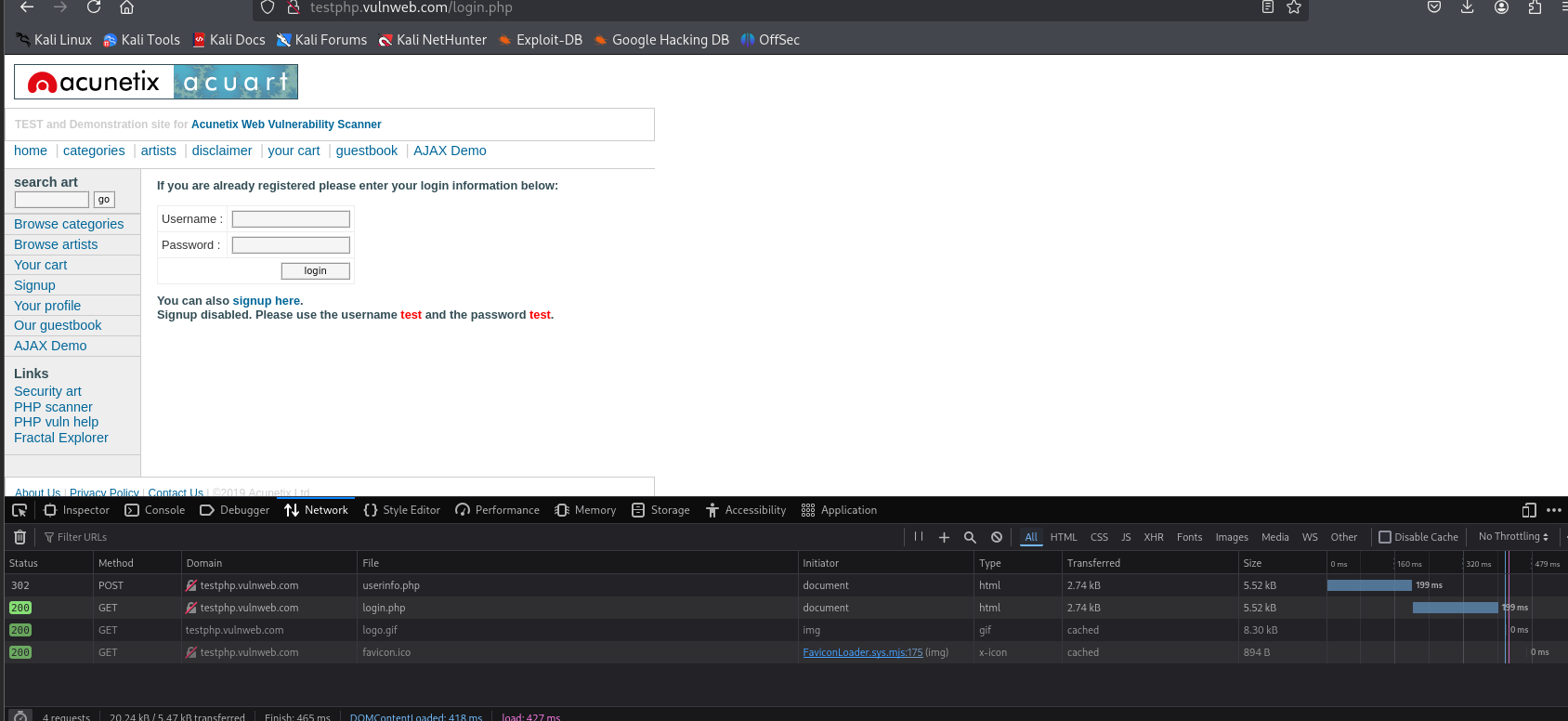
→ Cho thấy truy vấn độc hại đã được thực thi ngầm mà **không cần trả dữ liệu ra**, chứng tỏ có thể **tấn công Blind SQLi**.

**Câu 2 *(3.5 điểm)***

Kiểm thử lỗ hổng SQL Injection trên trang web http://webvul.bkcs.vn/login.php

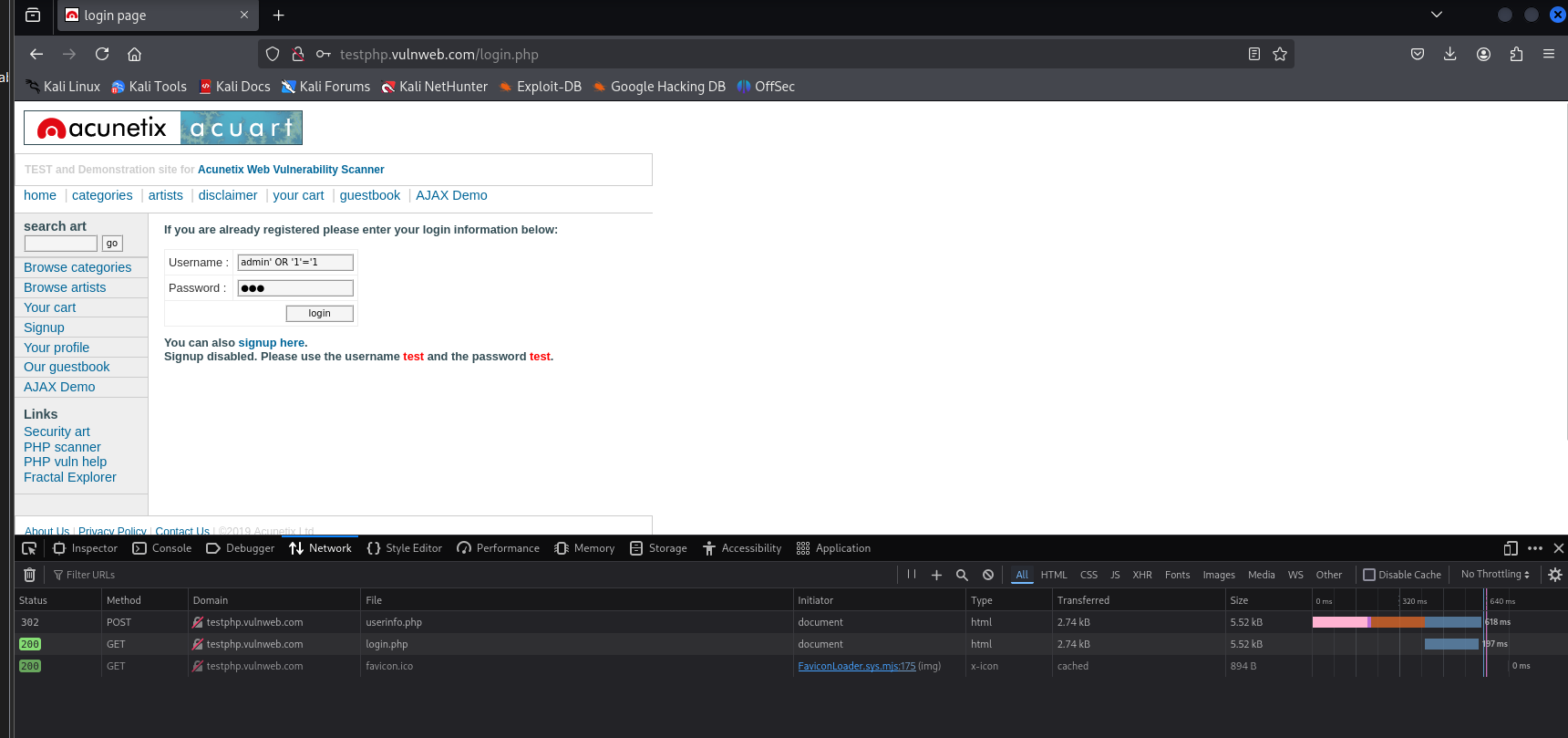
* Xác định các tham số đầu vào và cách thức chúng được gửi tới server **(0.25 điểm)**





* 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)**

SELECT \* FROM users WHERE username = '$\_POST[username]' AND password = '$\_POST[password]';



* 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*

**Câu 3 *(3 điểm)***

Kiểm thử lỗ hổng SQL Injection trên trang web http://webvul.bkcs.vn/index.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, 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*