**VULNBLOG**

**REPORT**

1. **JSON Web Token (JWT) là gì?**

JWT là một chuẩn mở để truyền thông tin an toàn giữa các bên dưới dạng JSON.

JWT gồm 3 thành phần chính:

* Header
* Payload
* Signature

3 thành phần này sẽ được ngăn cách bằng một dấu “.”, và thông thường sẽ được encode ở dạng base64.

Ví dụ:

eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ0ZXN0In0.rqm56w1vfUMSGVe3PxfJqtAiyoIiRnKh1

1. **Các kiểu tấn công vào JWT**

**2.1 JWT authentication bypass via unverified signature**

Đây là lỗ hổng xảy ra do phía server không thực hiện xác nhận tính hợp lệ, kiểm tra chữ ký của JWT, dẫn tới kẻ tấn công có thể tùy ý thay đổi các giá trị trong JWT.

Ví dụ một JWT có payload như sau:

{"iss":"vulnblog","exp":1747482223,"sub":"vodanh"}

Ta thấy trong payload này, **sub** claim đang chứa tên của người dùng hiện tại là “vodanh”, ta có thể đổi nó thành tên của quản trị viên, ví dụ: “sub”:”administrator”. Nếu phía server không xác thực chữ ký của JWT thì ta có thể đăng nhập với tài khoản của quản trị viên.

**2.2 JWT authentication bypass via flawed signature verification**

Đây là lỗ hổng xảy ra khi JWT của ứng dụng có hỗ trợ algorithm **none**, khi algorithm của JWT được đặt giá trị **none** thì phía server sẽ không thực hiện xác thực chữ ký của JWT, dẫn tới kẻ tấn công có thể kiểm soát tùy ý giá trị của JWT.

Ví dụ một JWT như sau:

{“alg”:”RS256”}{“sub”:”vodanh”}signature

Kẻ tấn công có thể đổi giá trị của alg claim thành **none** và xóa đi phần chữ ký của JWT, khi đó kẻ tấn công có thể đăng nhập vào tài khoản của quản trị viên với payload như sau:

{“alg”:”none”}{“sub”:”administrator”}

**2.3 JWT authentication bypass via jwk header injection**

Đây là lỗ hổng xảy ra khi server cho phép xác thực bằng public key được nhúng vào JWT thông qua tham số **jwk** trong headers. Khi này, kẻ tấn công có thể tự tạo một public key và nhúng vào tham số **jwk**. Từ đó, kẻ tấn công có thể kiểm soát được giá trị của JWT.

**2.4 JWT authentication bypass via jku header injection**

Việc sử dụng public key được nhúng trong JWT có thể chứa nhiều rủi ro, bởi vậy một số ứng dụng sử dụng tham số **jku** nhằm xác định một URL tham chiếu tới một bộ khóa công khai được đặt ở server.

Tuy nhiên, việc triển khai không đúng cách có thể tạo ra lỗ hổng bảo mật nghiêm trọng. Một cuộc tấn công tham số **jku** trong self-signed JWTs thường xảy ra một kẻ tấn công giả mạo JWT bằng cách thay đổi giá trị của **jku** để trỏ đến một URL mà kẻ tấn công kiểm soát. Khi truy cập tới URL này, ứng dụng sẽ lấy và sử dụng các public keys do kẻ tấn công tạo ra.

**2.5 JWT authentication bypass via kid header path traversal**

Ngoài việc được sử dụng để xác định khóa công khai (public key) hoặc khóa bí mật (private key) trong xác minh chữ ký của JWT, **kid** có thể được chỉ định giá trị đường dẫn trỏ đến tệp chứa thông tin khóa xác minh. Kẻ tấn công có thể nhắm vào tham số này với kỹ thuật **Path traversal**, khiến trang web tìm tới các tệp tùy ý trong server.

**2.6 JWT authentication bypass via algorithm confusion**

Đây là lỗ hổng xảy ra khi kẻ tấn công có thể “khiến” server xác minh chữ ký của JWT bằng một thuật toán khác với thuật toán mà trang web dự định sử dụng. Nếu không được xử lý đúng cách, nó có thể cho phép kẻ tấn công làm giả JWT hợp lệ chứa các giá trị tùy ý mà không cần biết secret key của server.

1. **VulnBlog Report**
   1. **JWT authentication bypass via unverified signature**

Tiếp cận trang web, ta thực hiện đăng nhập với tài khoản test:

A screenshot of a computer

AI-generated content may be incorrect.

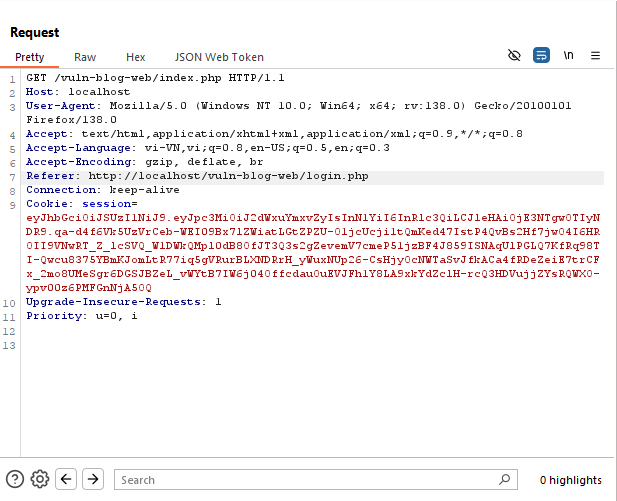
*Đặng nhập với tài khoản test*

Ta được chuyển tiếp tới giao diện trang chủ của ứng dụng web:A screenshot of a computer

AI-generated content may be incorrect.

*Giao diện trang chủ của ứng dụng*

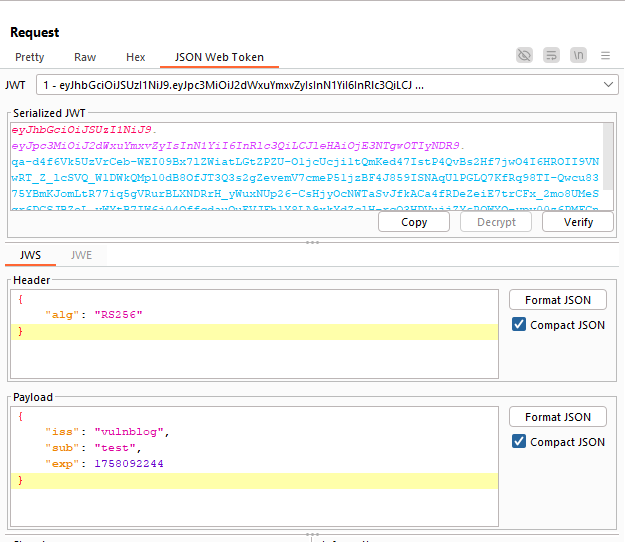
Thực hiện kiểm tra request với Burp Suite:



*Kiểm tra request với Burp Suite*

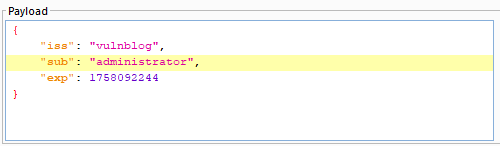
Ta thấy request của chúng ta được gửi kèm với trường Cookie tên là session, session này nhìn qua có thể thấy là định dạng của JWT.

Ta sẽ sử dụng JWT Editor extension để thực hiện decode JWT này để xem các thông tin của nó:



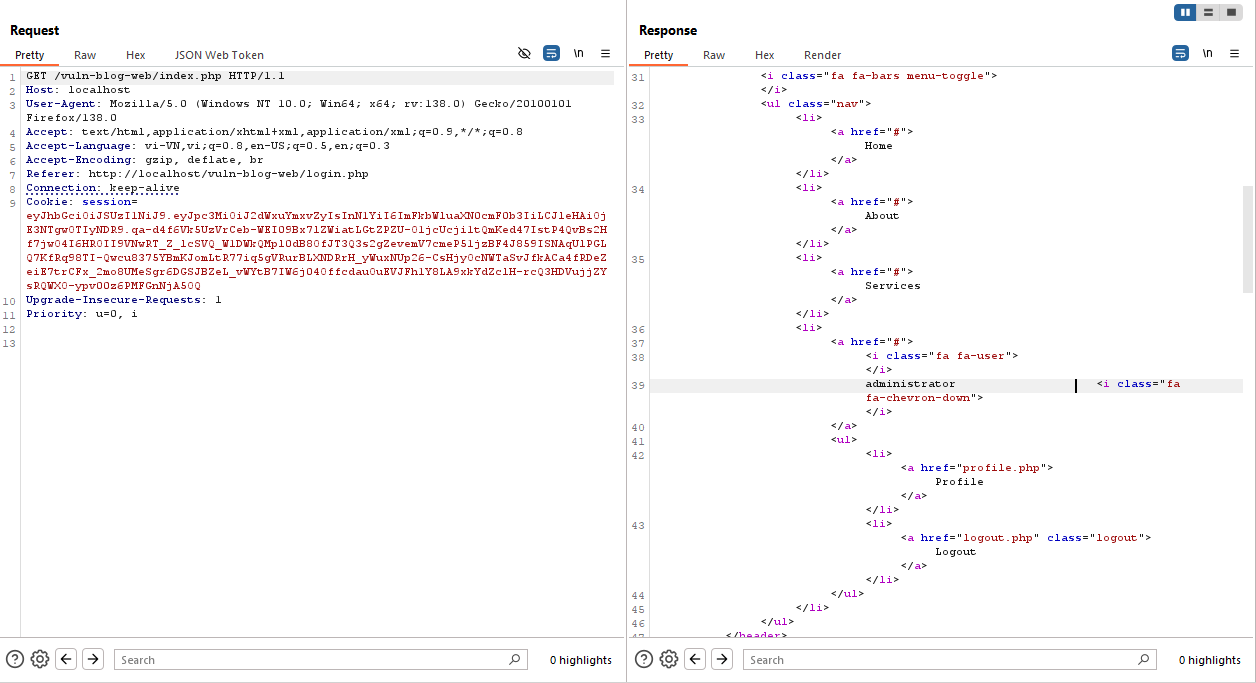
*Sử dụng JWT Editor để coi thông tin JWT*

Ta thấy rằng ở trường payload có sub claim đang chứa tên người dùng của chúng ta, ta sẽ thực hiện thay đổi giá trị của sub claim từ tên người dùng hiện tại thành tên của quản trị viên, ở đây là administrator:



*Thay đổi giá trị của sub claim*

Tiếp đó, ta sẽ thử gửi request với JWT mới này để xem liệu chúng ta có thể đăng nhập với tài khoản quản trị viên hay không:

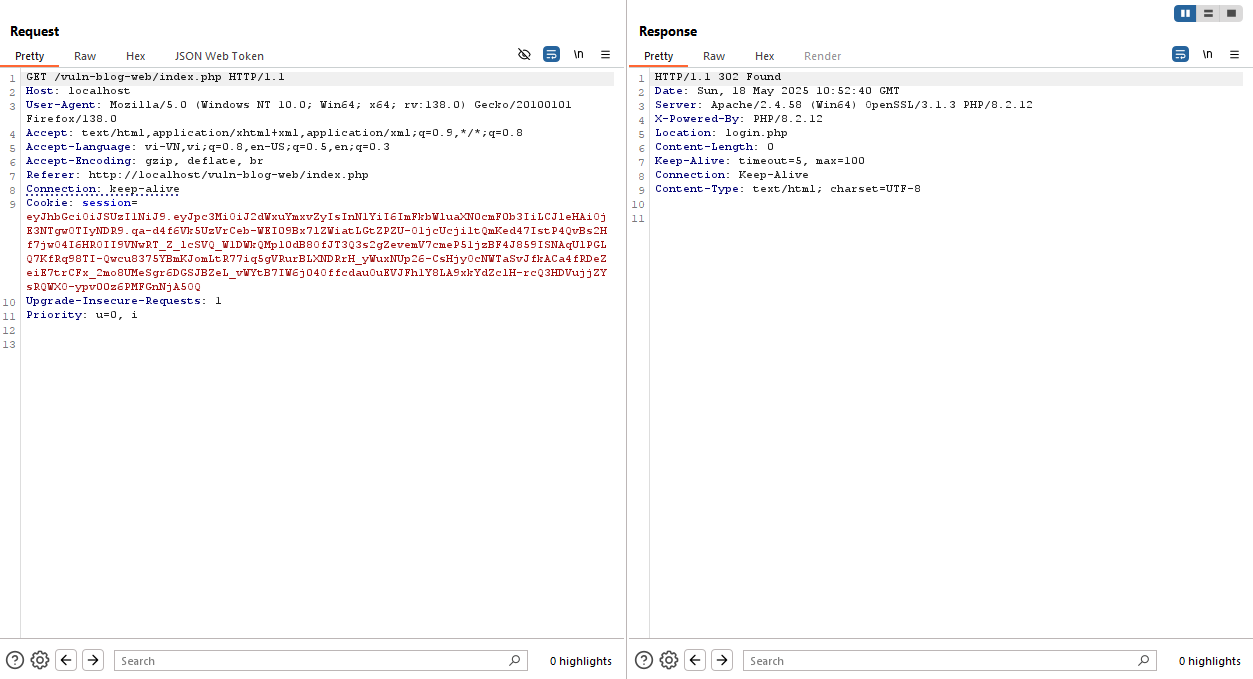


*Đăng nhập thành công tài khoản quản trị viên*

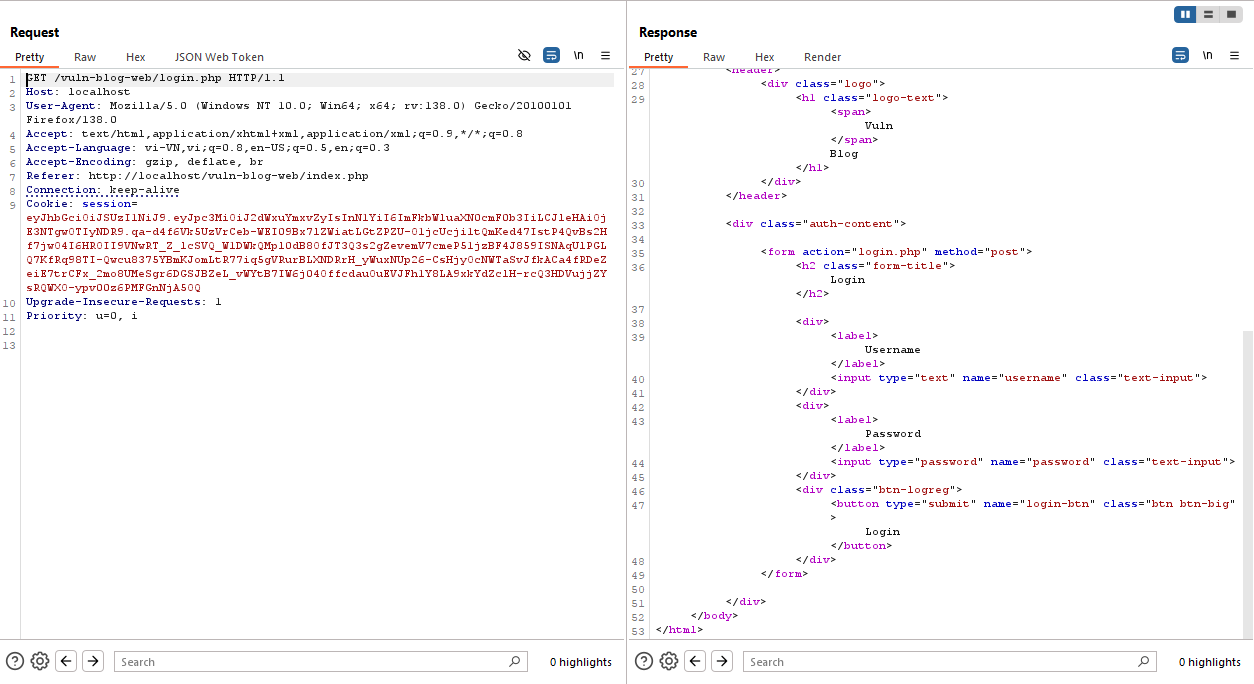
Kết quả là ta đã thành công đăng nhập tài khoản của quản trị viên.

* 1. **JWT authentication bypass via flawed signature verification**

Lần tiếp cận trang web này, ta sẽ lại thử tấn công vào JWT bằng cách thay đổi giá trị của sub claim thành administrator:

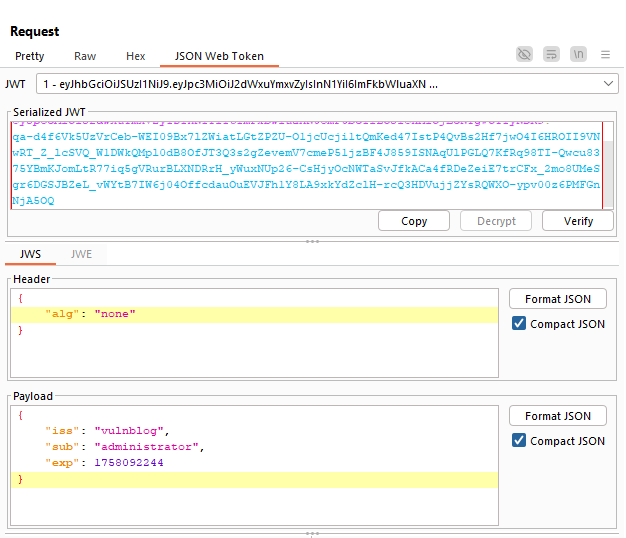


*Ứng dụng trả về status code 302*



*Ứng dụng redirect về trang login*

Kết quả là ứng dụng trả về status code 302 và redirect chúng ta về trang login. Ta sẽ thử tấn công bằng cách khác, ở đây ta sẽ đổi giá trị của **alg** thành **none**:



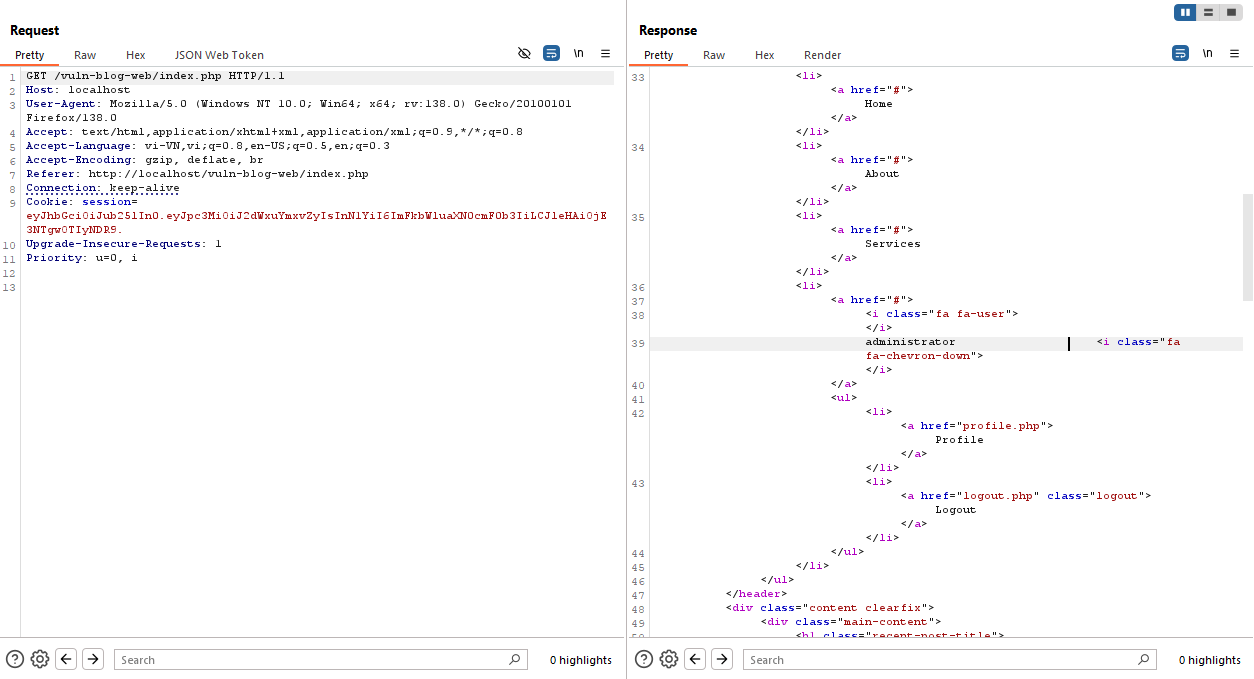
*Đổi giá trị của alg thành none*

Tiếp theo, ta sẽ thực hiện xóa đi phần chữ ký của JWT:



*Xóa bỏ phần chữ ký của JWT*

Sau khi đã xóa phần chữ ký, thực hiện gửi request:



*Đăng nhập thành công tài khoản quản trị viên*

Kết quả là ta đã thành công đăng nhập tài khoản của quản trị viên.