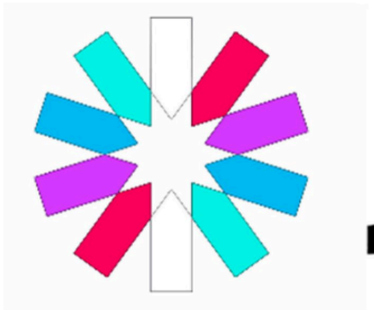


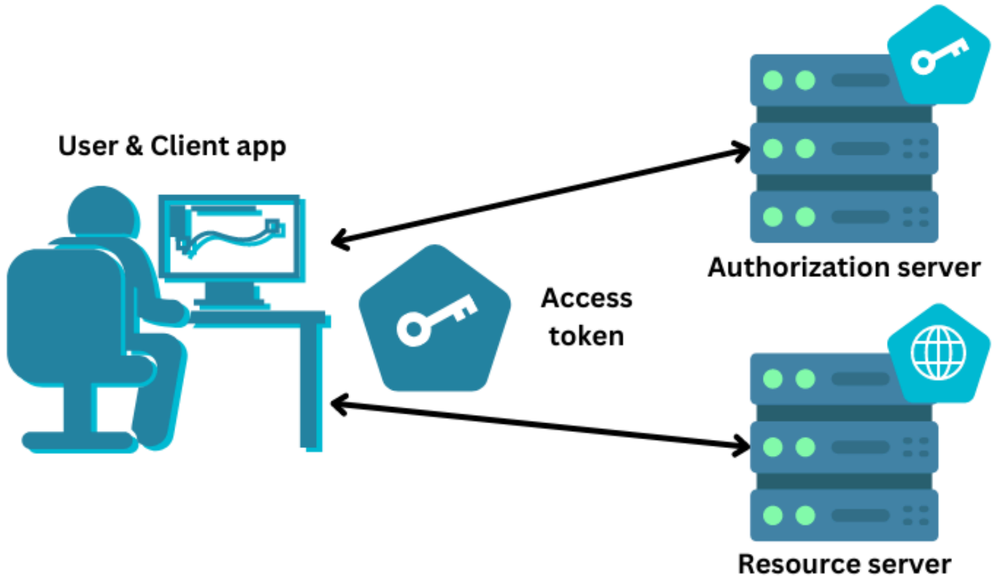
BẢO MẬT RESTFUL API VỚI JWT

MỤC TIÊU CỦA BÀI THỰC HÀNH

- Hiểu cơ chế bảo mật của JWT
- Biết cách tạo, mã hóa và giải mã JWT
- Áp dụng JWT để bảo vệ API trong dự án PHP

GIỚI THIỆU TỔNG QUÁT VỀ JWT

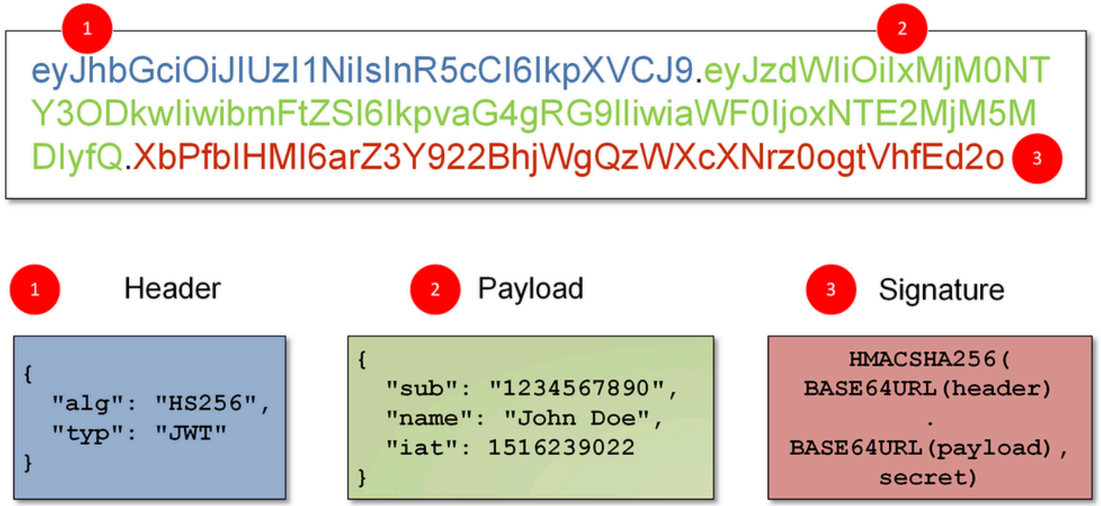
- JWT (JSON Web Token) là một tiêu chuẩn mở (RFC 7519).
- Dùng để truyền tải thông tin giữa các bên một cách an toàn và nhỏ gọn (compact & secure).
- Thường dùng để xác thực (authentication) và phân quyền truy cập (authorization) trong các ứng dụng web, mobile, API RESTful.



CẤU TRÚC CỦA JWT

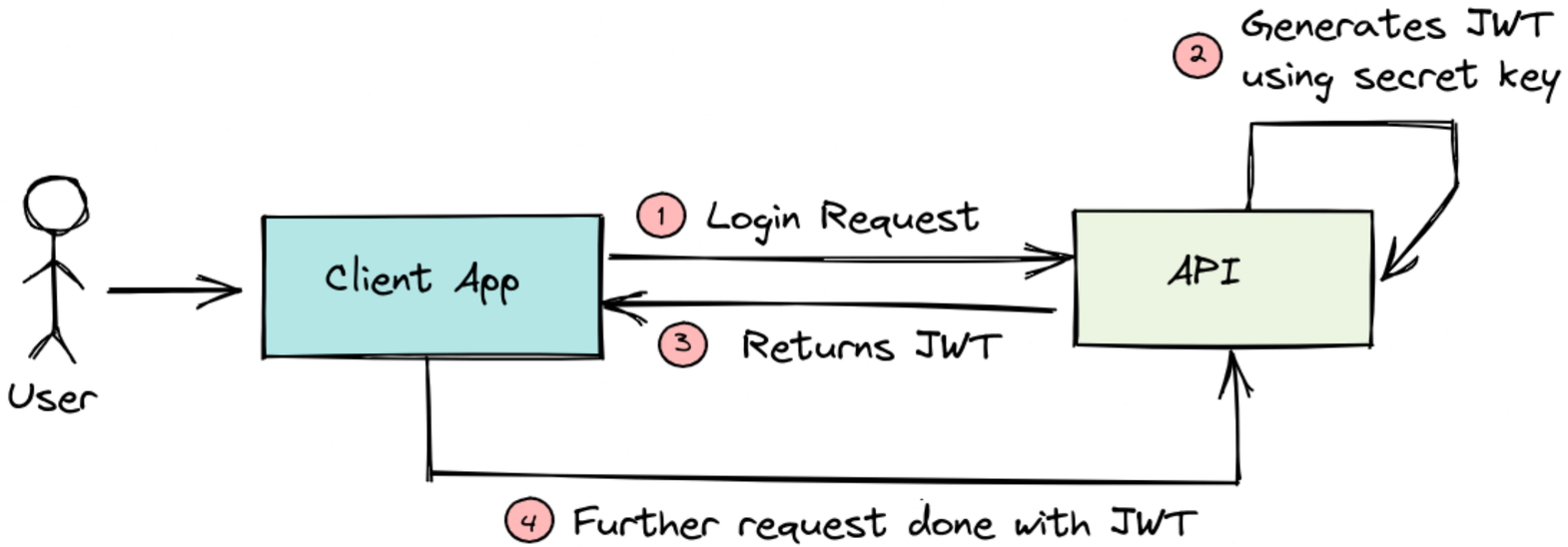
Cấu trúc của JWT gồm 3 phần chính:

1. **Header** – Thông tin về thuật toán mã hóa (ví dụ: HS256)
2. **Payload** – Dữ liệu muốn truyền đi (ví dụ: user_id, role, exp, ...)
3. **Signature** – Chữ ký số đảm bảo tính toàn vẹn



JWT ĐƯỢC DÙNG ĐỂ LÀM GÌ?

- Thay thế session trong xác thực người dùng
- Truyền thông tin user giữa client ↔ server
- Không cần lưu trạng thái người dùng (stateless) trên server



GIỚI THIỆU VỀ COMPOSER

Composer là một trình quản lý thư viện (**dependency manager**) dành cho ngôn ngữ PHP. Cho phép bạn:

- Cài đặt thư viện bên ngoài dễ dàng.
- Quản lý phiên bản thư viện.
- Tự động tải (autoload) class.
- Cấu trúc dự án chuyên nghiệp hơn.

Composer hoạt động như thế nào?

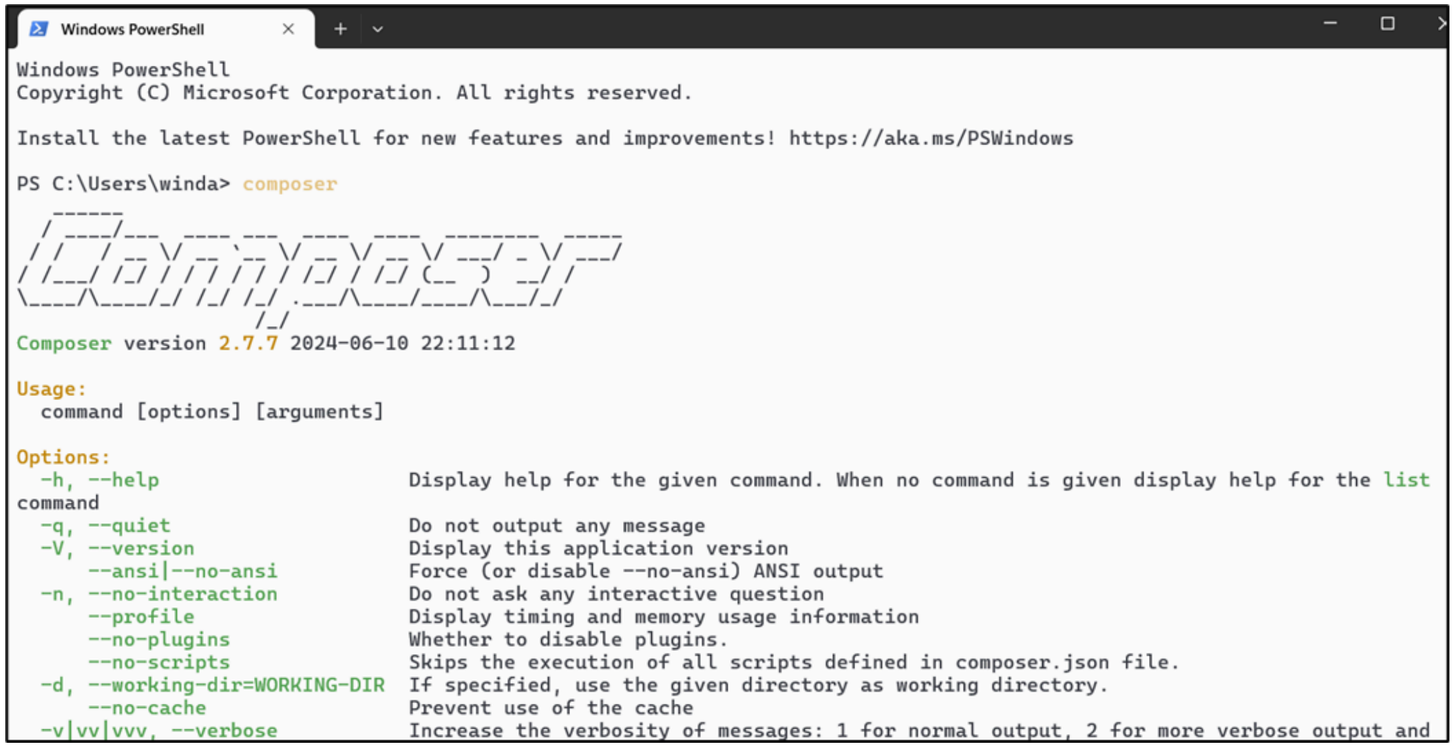
- Dựa vào tệp composer.json để xác định các thư viện cần thiết.
- Tự động tải thư viện và các phụ thuộc vào thư mục vendor.
- Tạo file composer.lock để đảm bảo phiên bản ổn định.

🔗*Truy cập trang chính thức: https://getcomposer.org*

CÀI ĐẶT COMPOSER (WINDOWS/LINUX/MACOS)

- Truy cập trang chính thức: *https://getcomposer.org*

  Cài đặt và kiểm tra bằng lệnh: composer



CÀI ĐẶT THƯ VIỆN JWT TRONG PHP

Cài đặt thư viện JWT với Composer

```
// Sử dụng Composer để cài đặt:
composer require firebase/php-jwt

// Nếu chưa có composer.json, bạn có thể khởi tạo bằng:
composer init

```

Cài đặt thư viện JWT với Composer

```
project/
├── vendor/
│   └── firebase/
│   └── php-jwt/
├── composer.json
├── index.php
```

TẠO JWT SAU KHI ĐĂNG NHẬP

Khi nào tạo JWT?

- Sau khi người dùng đăng nhập thành công (kiểm tra tài khoản + mật khẩu đúng).
- Thay vì tạo session, ta sẽ tạo JWT và gửi về client.
- Client lưu token trong localStorage hoặc Authorization header.

Ghi chú quan trọng:

- exp: Hạn dùng của token (tính bằng giây).
- HS256: Thuật toán mã hóa HMAC SHA-256
- JWT nên được lưu ở client và gửi với mỗi request cần xác thực.

```php
<?php

use Firebase\JWT\JWT;

$secret_key = "my_secret_key";
$payload = [
  "iss" => "http://localhost",
  "aud" => "http://localhost",
  "iat" => time(),
  "exp" => time() + 3600, // hết hạn sau 1 giờ
  "data" => [
    "id" => $user["id"],
    "username" => $user["username"]
  ]
];

$jwt = JWT::encode($payload, $secret_key, 'HS256');

```

GỬI TOKEN TRONG HEADER

Vì sao gửi token qua Header?

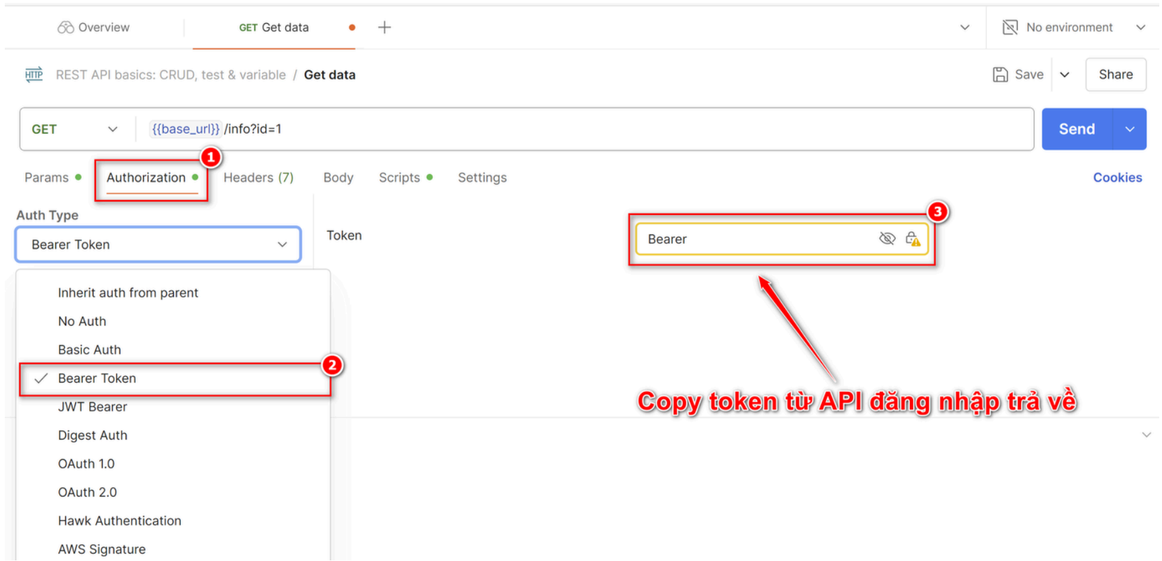
- Bảo mật hơn so với gửi qua query string hoặc body.
- Tuân thủ chuẩn HTTP Authorization Header.
- Dễ kiểm soát trong middleware và gateway API.

Mã kiểm tra Header trong PHP

```php
<?php

$headers = getallheaders();
if (isset($headers['Authorization'])) {
  $authHeader = $headers['Authorization'];
  $token = str_replace('Bearer ', '', $authHeader);
  // tiếp tục kiểm tra token...
}
```

Gửi JWT bằng Postman



XÁC THỰC TOKEN Ở SERVER

Mục tiêu xác thực

- Đảm bảo request đến từ người dùng đã đăng nhập
- Token hợp lệ: chưa hết hạn, được ký bằng secret key đúng

Các bước xác thực JWT

1. Lấy token từ header
2. Giải mã (decode) token
3. Kiểm tra chữ ký (signature) và hạn (exp)
4. Thực hiện chức năng sau:
    - Nếu hợp lệ → cho phép tiếp tục
    - Nếu không hợp lệ → trả lỗi 401 Unauthorized

```php
<?php

use Firebase\JWT\JWT;
use Firebase\JWT\Key;

$jwt = $token_from_header;
$secret_key = "my_secret_key";

try {
  $decoded = JWT::decode($jwt, new Key($secret_key, 'HS256'));
  // Truy cập payload: $decoded->data
} catch (Exception $e) {
  http_response_code(401);
  echo json_encode(["message" => "Token không hợp lệ"]);
  exit;
}

```

LỖI THƯỜNG GẶP KHI XỬ LÝ JWT

1. Token không tồn tại do không gửi token trong Authorization Header hoặc sai định dạng Bearer.

2. Token hết hạn (exp).

=> Cách khắc phục: *Tăng exp* nếu cần lâu hơn, Thông báo người dùng đăng nhập lại.

3. Sai secret key hoặc thuật toán dùng sai secret_key so với lúc mã hóa token.

=> Cách khắc phục: Đảm bảo giải mã *dùng đúng key* và thuật toán như khi tạo token.

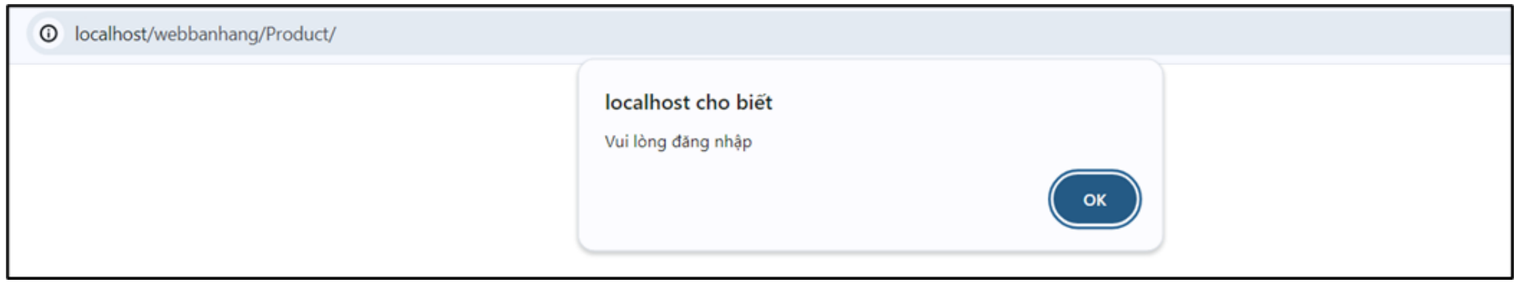
4. Token bị sửa hoặc không hợp lệ signature không còn đúng.

=> Cách khắc phục: Không giải mã thủ công, luôn *dùng thư viện (như Firebase\JWT)*

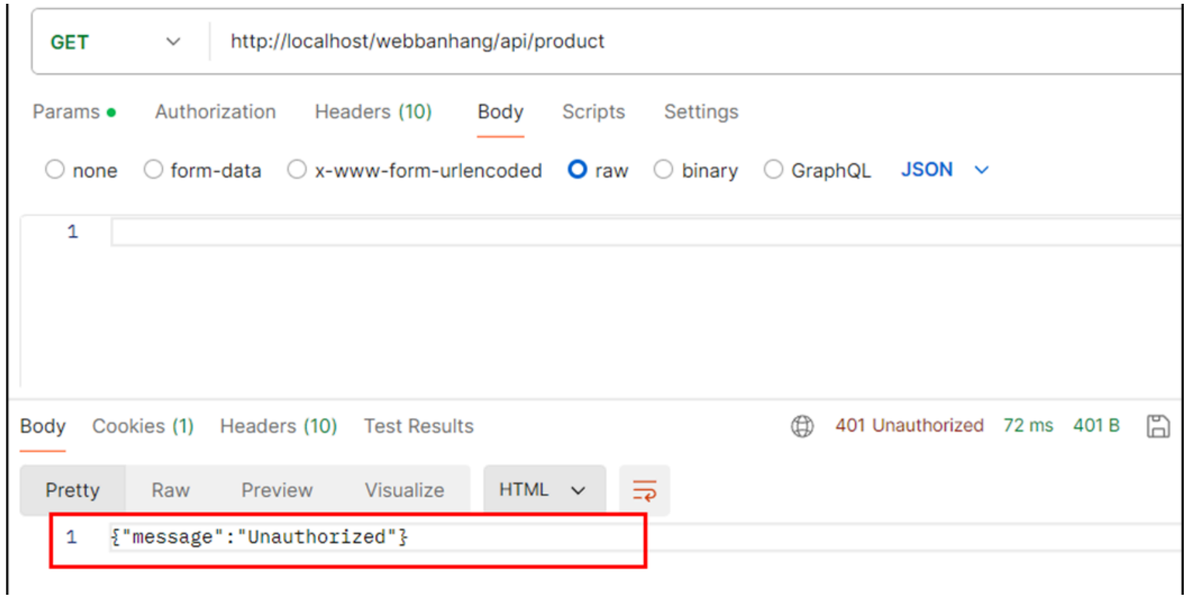
5. Không xử lý ngoại lệ (Exception) gây lỗi trắng hoặc crash server.

=> Cách khắc phục: Cần luôn bọc JWT::decode() trong *try { ... } catch (Exception $e)* { ... }

KẾT QUẢ KHỞI CHẠY DỰ ÁN



Kiểm thử trên trình duyệt



Kiểm thử trong Postman