JWT AND HTTP BASIC AUTHENTICATION

DECEMBER 22, 2024

BUSHRA ALI ARISHI

JWT (JSON Web Token) and HTTP Basic Authentication

This research outlines the concepts of JWT and HTTP Basic Authentication, explains the differences between them, provides a comparison table, and offers examples of real-life systems using each. The content is tailored for backend developers and students.

1. JWT (JSON Web Token)

Definition:

JWT is an open standard for securely transmitting information between parties as a JSON object. It is commonly used for authentication and authorization in web applications.

Structure:

- Header: Includes metadata (e.g., token type and hashing algorithm).
- Payload: Contains claims, such as user details and permissions.
- Signature: Ensures the token's integrity and authenticity using a secret key.

How it Works:

1. A client logs in with their credentials.
2. The server verifies credentials and generates a JWT.
3. The JWT is sent to the client and stored (usually in localStorage or a cookie).
4. For subsequent requests, the client includes the JWT in the Authorization header (`Bearer <token>`).

Key Features:

- Stateless: The server does not store session data.
- Secure: Uses algorithms like HMAC SHA256 or RSA.
- Portable: Can be easily shared across domains.

Example Use Case:

JWT is widely used in Single Page Applications (SPAs), such as systems like Gmail.

2. HTTP Basic Authentication

Definition:

HTTP Basic Authentication is a simple authentication mechanism where the client sends a username and password encoded in Base64 in the Authorization header.

How it Works:

1. The client sends a request with the Authorization header (`Basic <Base64(username:password)>`).
2. The server decodes the credentials and verifies them.
3. If valid, access is granted.

Key Features:

- Stateless: Credentials are sent with every request.
- Less secure: The Base64 encoding can be easily decoded; must be used over HTTPS.
- Simple: Requires no additional libraries or setup.

Example Use Case:

HTTP Basic Authentication is often used in internal systems or REST APIs that are not exposed to the public.

3. Comparison Table

Feature	JWT	HTTP Basic Authentication
Authentication Type	Token-based	Credentials-based
Security	More secure with signature and expiration	Less secure; relies on HTTPS encryption
Statefulness	Stateless (no server storage of sessions)	Stateless
Ease of Use	Requires token generation	Simple; no setup needed

	and validation	beyond Base64
Performance	Slightly more overhead due to token size	Lightweight due to minimal data
Expiration	Supports expiration and revocation	No built-in expiration
Use Case	Modern APIs, SPAs	Simple systems, internal APIs
Transmission	`Bearer <token>` in Authorization header	`Basic <Base64(username:password)>`

4. When to Use Each

JWT

- When the system needs stateless authentication.
- If there is a need to share tokens across domains or services.
- For modern web apps with multiple clients (web, mobile).

Example:

A banking web app using JWT to manage sessions for its users.

HTTP Basic Authentication

- For small or internal systems with minimal security requirements.
- When simplicity is prioritized over flexibility or scalability.

Example:

An internal company API for accessing employee directories.

5. Real-Life Examples

1. JWT:

- System: eCommerce platforms (e.g., Amazon).
- Scenario: A user logs in, and their JWT token is used to access their shopping cart and order history.

2. HTTP Basic Authentication:

- System: Internal REST API for DevOps tools like Jenkins.
- Scenario: Developers authenticate to the API using their credentials.