JWT

Naelah Alshibani

- **What is it?**
  JWT, or JSON Web Token, is a small and self-contained way to securely transfer information between two parties, such as a server and a client. It is a token formatted as a JSON object, which includes all the necessary information needed for the server to verify and authorize a user. Because it is self-contained, it doesn't rely on the server to store session data, making it a more efficient and flexible method for managing authentication and authorization.
- **How does It work?**
  Authorization on servers is typically handled using sessions. In this method, the server stores user information and assigns a unique session ID. When a user makes a request, the server validates their identity by checking this session ID.
  With JSON Web Tokens (JWT), authorization is handled differently. Instead of storing user information in the server's session memory, JWT encodes and serializes the user's data using a secret key/signature. As shown in *figure (1),* the client (application) first sends a token request to the authorization server. The server generates an access token (JWT) and sends it back to the client. The browser stores the token and includes it in later requests when accessing protected resources on the resource server. This allows the server to verify the user without keeping track of their session in memory.

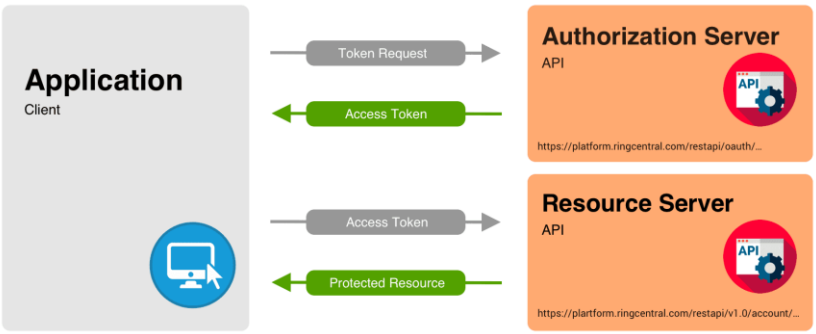

Figure (1) JWT Authorization Flow

- **What Dose JWT Consist Of?**
  A JSON Web Token (JWT) consists of three main parts: the header, payload, and signature. The header contains data about the token, including the algorithm used

for signing (e.g., HS256) and the token type (JWT). The payload holds the actual data, which might include information such as the user ID, roles, or permissions. Finally, the signature ensures the token's integrity and authenticity. It is generated by combining the encoded header and payload with a secret key and then hashing them. As shown in *figure (2),* the JWT structure is simple and compact. Each component is encoded in Base64 format and separated by dots (.), making it easy to transmit and verify. This design allows JWTs to securely carry information without the need for server-side storage.

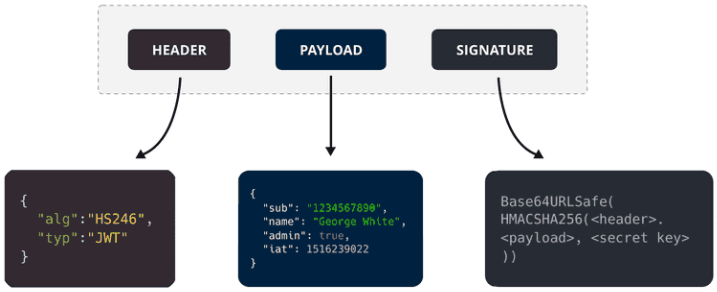

Figure (2) JWT Structure

- **What Makes JWT Powerful?**

  JWTs are particularly powerful because they enable seamless communication across multiple servers or services. For example, if an institution stores its data across two different servers, a user logged into one server would typically need to log in again to access resources on the other server. This happens because the session ID is stored on the first server, and the second server has no access to it.

  However, with JWT, the user's information is stored on the client side as a token. This allows the user to access resources on any server, as long as all servers share the same secret key to verify the token. This makes JWT especially useful in systems using microservices, where different services can validate the token without requiring centralized session management.