Introduction to Authentication with Replyke

Authentication is a cornerstone of Replyke, enabling all its functionalities to work seamlessly together. By associating actions and data with users, authentication facilitates user interactions, ensures data consistency, and enforces authorization rules within your application.

What Replyke Offers

Replyke provides a straightforward, built-in authentication system, allowing developers to easily implement email and password authentication. With user-friendly hooks or endpoints for signing up, signing in, and signing out, developers can quickly get started without the need to build these features from scratch.

If you already have an existing user management system or plan to use a different system in your project, Replyke is designed to integrate seamlessly. Developers can pass a JSON Web Token (JWT) containing user details, signed with a private key provided by Replyke. This ensures that Replyke can trust the user data and associate actions with the appropriate users.

Authorization and Security

Replyke includes basic authorization to secure user actions. For example:

- Only authenticated users can vote on entities or comments.
- Comments and replies are restricted to logged-in users.

For more advanced authorization needs, such as restricting access to specific pages or sections of your app, developers have full flexibility to implement custom rules on top of Replyke’s foundational capabilities.

Integrating Replyke with an External User System

When integrating Replyke with an external user system, two primary challenges arise:

1. **Associating actions and content with external users:** Replyke needs to connect its features to users managed in a separate system.

2. **Ensuring data security:** Simply passing user details as a plain object is insecure, as it could be manipulated by malicious actors on the client-side.

To address these challenges, Replyke uses JSON Web Tokens (JWTs) signed with a private key. Below, we outline the logic behind this integration.

Understanding JSON Web Tokens (JWTs)

JWTs are a secure method for transmitting information between two parties. They consist of three parts:

- **Header:** Contains metadata about the token, such as the signing algorithm.
- **Payload:** Holds the data being transmitted (e.g., user details).
- **Signature:** A cryptographic hash generated using the payload and a secret key, ensuring the token's authenticity.

By verifying the signature with the public key, Replyke can trust that the data hasn't been tampered with and was signed by the correct private key.

Overview of the Process

1 Generating JWT Keys

- Developers generate a pair of JWT keys (public and private) from the Replyke Dashboard.
- The public key is stored with Replyke and associated with the project.
- The private key is shown only once during generation and must be securely stored by the developer, typically as an environment variable on their server.

2 Obtaining and Signing User Data

- When a logged-in user is detected in your app, the app makes a request to the developer's server (not to Replyke's server, but to your own) to obtain a signed JWT.
- The server retrieves the user's data (e.g. from cookies or authentication headers - depends on your own auth implementation), signs the JWT with the private key, and returns it to the client. This ensures that user data is never directly sent from the client to the server, reducing the risk of tampering.
- By using the private key to sign the JWT, the data's integrity and authenticity are guaranteed. The public key on Replyke's side can then verify the JWT, ensuring that it comes from a trusted source.

3 Passing the JWT to Replyke

- The signed JWT is sent back to the client and passed to Replyke for verification either via an API call, or as a prop to the `ReplykeProvider` which is wrapping your project (React / React Native implementation).
- Replyke verifies the JWT using the public key.
- If verification succeeds, Replyke creates a user in its system linked to the external system's user ID. All actions performed by this user are now securely associated with them.

Benefits of this Approach

- **Security:** Ensures user data integrity and prevents unauthorized actions.
- **Flexibility:** Allows developers to use any user management system with Replyke.
- **Seamless Integration:** Replyke transparently handles user association and data linking without compromising security.

For a detailed step-by-step guide for implementing this integration in your project, follow the respective guides in the integration approach you of your choice.

Last updated on May 6, 2025