Stateless Authentication for REST and Microservices with JSON Web Token

YANNICK MORTIER – JAVA SOFTWARE DEVELOPER AT TOPDESK

Agenda

- Presentation
- Demo
- Code Tour

Remember: Goals in Microservices

- Scalability
- Replaceabilty
- Technology-Independence

A big obstacle: Authentication

- One central authority
- Potential Bottleneck
- Always a trade-off

Central Problem

- How can we have an independent authentication micro service while maintaining security?

Possible Solution JWT

- „JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object."

- JWT offers tokens that are cryptographically signed
- Trusted without auth service involved with each request
- Demo: https://jwt.io/

Problems with Token-Based Auth

- It is a trade off
- Not possible to immediately invalidate a session
- Choose shorter expiration date to increase security
- Use refresh tokens to find a balance
- Cryptographic key needs to be securely stored
- Very vulnerable to MITM attacks -> SSL absolutely necessary

Acceptance Criteria

- Cryptographic key is safely stored
- Public key can be easily accessed by other services
- Services can pass along token to do requests on users‘ behalf
- User can log out sessions remotely (as soon as token expires)
- Tokens can be invalidated in case of security breach
- Still missing: Authorization (only partial implementation)

Proposed Solution: Auth Service

- Auth service starts up, generates keys for its instance
- Private key is stored only in memory, never written to disk
- Public key is stored in database, can be requested by other services

- When token is refreshed, it is verified if the token is still valid before a new bearer token is provided

Proposed Solution: Application

- „Integration on the Glass“ or Server-Side Rendering
- Application asks user for credentials
- Application submits credentials to auth service
- Auth service creates bearer & refresh token, stores refresh token information
- Application takes care of state management (refreshing bearer token)
- Application uses bearer token to request data from other micro services

Proposed Solution: Micro Service

- Micro Service receives request from application with bearer token
- Token is verified by fetching key from auth server and caching it
- If token is verified successfully data and service is provided
- Primitive Authorization can be used by looking at the roles stored in the token

Proposed Solution: In case of breach

- Delete all (compromised) keys from auth server
- Restart auth servers
- Wait for caches in micro services to expire or restart them
- All tokens up to this point are invalidated

Problems with the solution

- Proper authentication is still missing
- Size of JWTs can be problematic (need to be sent with every request)

- Remember: Micro Services are not an architecture, but an implementation detail
- „Eventual Consistency“
- Using JWT for authorization is still hotly debated – often without presenting a better solution.

Bonus Fact

- This is almost following the OAuth 2 „Framework“
- See the Implicit Flow
- Requires a few more fields in token & application registration

Questions?

- Source Code available at https://github.com/mortiery/jsonwebtoken
- Dockerfiles will be available soon