Service-to-service authentication in a microservice deployment

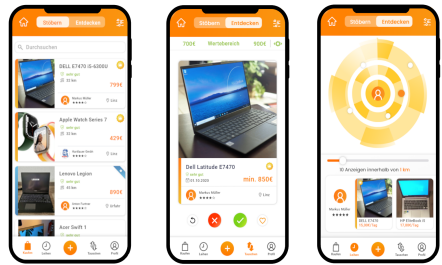
UNIVERSITY
OF APPLIED SCIENCES
UPPER AUSTRIA

FH Hagenberg, WS 2021/2022
Benjamin Ellmer

Bachelor Thesis Presentation

Swapindo

- Flea market app developed for android (so far)
- Easy and playful swapping
- Simple and safe lending and renting
- Fast and secure buying and selling
- Backend based on the microservice architecture developed in C# with ASP.NET

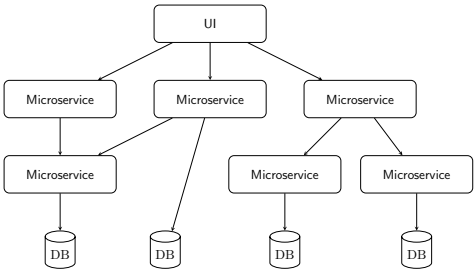


Service-to-service authentication

Term Description

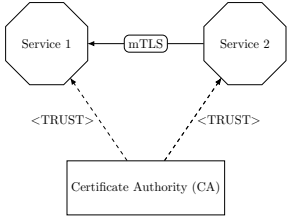
Authentication is the process of identifying the communication partner to protect a system from spoofing.

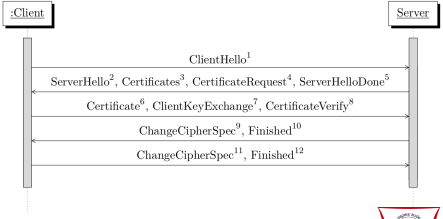
- Function calls become remote calls
- Remote calls have to provide authentication
- TLS authenticates the server to the client, but not the client to the server
- Most popular mechanisms are mutual TLS and self-signed JWTs



Mutual TLS

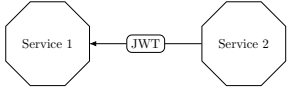
- Each service owns a certificate, signed by a Certificate Authority
- The certificates are exchanged during the TLS-handshake
- When the signature of the certificate is valid, the service is trusted
- Efficient and straightforward
- Based on the TLS protocol, therefore hard to adapt for other purposes like sharing the user-context





Self-signed JWT

- Each service owns a key pair
- A JWT signed with the private key has to be embedded within the Authorization header
- Additional parameters can be embedded within the JWT
- User-context is usually shared using a nested JWT
- Nonrepudiation can be achieved by storing the received JWTs and requests

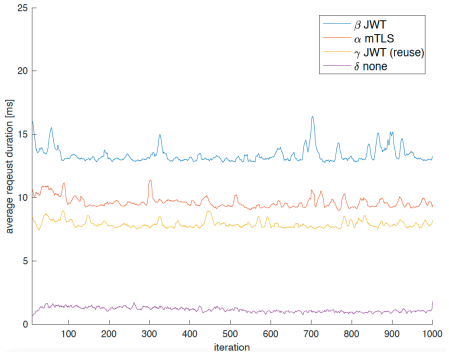


eyJhbGciOiJSUzI1NiIsImtpZCI6IkYwMURFNTBCNTU5MzVBQ0VEOUNDNzdCRjRFMjY5NkVDOTE4MUI5NjkiLCJ0eXAiOiJKV1QifQ.eyJuYmYiOjE2NDU1MjU5MjMsImV4cCI6MTY0NTUyNjIyMywiaXNzIjoic2VydmljZTEuc3dhcGluZG8uY29tIiwiYXVkIjoic2VydmljZTIuc3dhcGluZG8uY29tIn0.

```
{
        "alg": "RS256",
        "kid": "F01DE50B55935ACED9CC77BF4E2696EC9181B969",
        "typ": "JWT"
}
{
        "nbf": 1645525923,
        "exp": 1645526223,
        "iss": "service1.swapindo.com",
        "aud": "service2.swapindo.com"
}
```

Performance Experiment

Authentication Mechanism	Average Request Duration	
	first connection	reuse connection
mTLS	228.03 ms	9.87 ms
self-signed JWT	193.52 ms	13.57 ms
self-signed JWT (reusing token)	184.88 ms	8.10 ms
none	175.78 ms	1.31 ms



Key Management

- Most challenging part of both mechanisms
- Consequence of public key cryptography
- Certification
- Generate/Distribute keys
- Rotate keys
- Simple in the beginning, gets hard as the number of services grows

Choosing the correct mechanism

Hint

Before thinking about the correct authentication mechanism, the developers should evaluate, if the microservice architecture is the correct technology for the project.

- Is nonrepudiation an requirement ? → **self-signed JWT**
- Does your application tend to share the user-context? → **self-signed JWT**
- Does your authentication mechanism require custom adaptations ? → **self-signed JWT**
- Does the project require to work without TLS? → **self-signed JWT**
- None of the previous reasons ? → **mTLS**

Thank you for your attention!