ORPHEUS

Comparing differences in OIDC Providers

Uros Stevanovic, Gabriel Zachmann, Marcus Hardt – uros.stevanovic@kit.edu
Developing new services

- Protocol == OIDC
- All services (OPs, RPs, clients, proxies...) must follow protocol
- Evaluation of OIDC features is a regular activity
Motivation

OpenID Connect (OIDC) is an authentication layer on top of OAuth2
- RESTful+JSON (simpler+lightweight flow comparing to SAML)
- Strives to provide simple API
- „makes simple things simple and complicated things possible“
- Many features (native and mobile apps, delegation,

Many new IdPs/Proxies are OIDC capable
- Google, Microsoft, PayPal, but also EGI, eduTEAMS, B2ACCESS, IAM (among others)
- Most new OPs are OIDC capable (or OIDC only)
- Most new services are OIDC only
- OIDC Federations
However…

- OIDC is complex
  - Variety of flows (authorization, device..)
  - Variety of capabilities (token exchange, ID token vs User Info...)
- OIDC standard leaves many things open
  - Providers do differ among themselves (flows, capabilities)
  - Sometimes not even following standards
  - New feature implemented, typically not advertised
- Developers point of view → what now?
- (Some) Questions to consider:
  - For SPs: Which OP to select? For OPs: How to debug?
  - Capabilities, flows?
- (Some) Capabilities to consider:
  - JWT vs opaque
  - Features (openid-configuration, user interaction)

→ Orpheus
Orpheus

- Oidc ProvidEr featUre Support
- OIDC Client
- Written in Go
  - Cross-platform
  - Local or deployed on a server
- Comparison of different OPs
- Check which features are supported
- OIDC flows (authorization, implicit, device)
- Debug OIDC flows, capabilities
Use Cases – for Developers

Tool to support development, e.g.
- Feature analysis for providers
- Token inspection
- Debug support

Analysis of OIDC flows:
- Where to get which information
- Principal concept of an OIDC flow
- Exchanged information
Use Cases – for Developers / Users

- Support debugging OIDC related problems
- Authorization decision is (partially) based on attributes released by the user’s home IdP
- Problems related to the released attributes can be hard to debug
  - Error might not occur for the developer’s accounts
  - Developer might have different home IdP
  - Accounts linked to real identities
  - …
- Different problems possible:
  - Misconfigured client, home IdP, user account, …
- Solution:
  - User can perform OIDC flow on Orpheus
  - User shares the results in a privacy compliant (GDPR conforming) way
  - Developer can check the released attributes
On Features

- **Automatic**
  - Can be checked automatically by Orpheus
  - Does not require any user interaction
  - Orpheus periodically checks these (e.g. every 5min)
  - Based on the information available from `.well-known/openid-configuration`

- **Examples:**
  - Introspection Endpoint Advertised
  - Supported Scopes Advertised

- **Community**

- **Manual**
On Features

- Automatic
- Community
  - Cannot be checked fully automatically by Orpheus
  - Does require some sort of user interaction, i.e. performing an OIDC flow
  - When user performs a flow, Orpheus checks all linked features
  - Based on the flow or information available from it
- Examples:
  - Device Flow Supported (it does work)
  - Access Token is JWT
  - Token Revocation
- Manual
On Features

- **Automatic**
- **Community**
- **Manual**
  - Cannot be checked by Orpheus
  - Have to be manually provided
  - Usually these do not change
  - Examples:
    - Web interface for client registration
    - Used underlying OIDC implementation
    - Client Registration requires manual approval
DEMO time

https://orpheus.data.kit.edu
Summary

- Extensible and extendable functionalities
- Highly configurable
- Public instance running at: https://orpheus.data.kit.edu
- Run your own instance (MIT License): https://git.scc.kit.edu/oidc/orpheus
Thank you for your time!!

https://orpheus.data.kit.edu