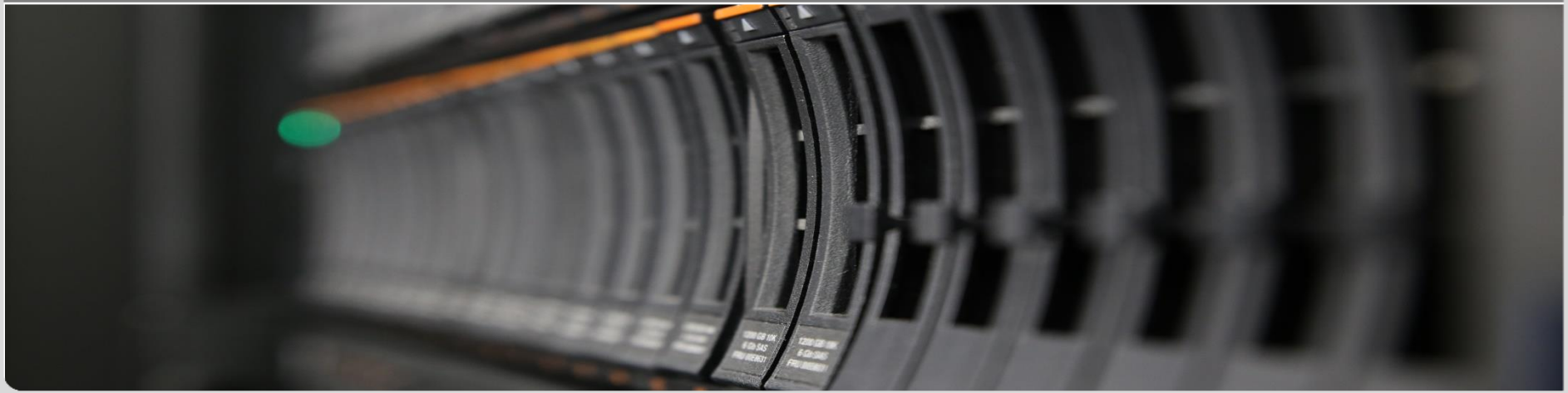
oidc-agent

Steinbuch Centre for Computing
Dr. Uros Stevanovic, Dr. Marcus Hardt, Gabriel Zachmann



oidc-agent

- Tool to obt
  - Suppor ange)
  - CLI cap emon and forward
- C os?

The problem

- OIDC works well for web-browser based applications (with caveats)

- But! There are use cases requiring non-web-(browser) flows
  - Applications that use REST and federated identification
    - Data management (webDav/OIDC, remote job/container/vm submission)
  - Delegated authentication (Server acts on my behalf, e.g. get data from yet another server)
  - Development/debugging, interactive use on the commandline
  - Batch jobs (including long-running jobs)

The problem (cntd.)

- Obstacles when using non-web-OIDC
  - Obtaining the Access Tokens (ATs)
  - ATs are (sometimes) long strings
  - Lifetime (~10min) requires frequent renewal
    - Lack of batch jobs use-cases
  - aud?
- Current solution:
  - Copy-paste from web applications

OpenID Connect (OIDC) Recap

- Authentication protocol on top of OAuth2
- Three different tokens:
  - Access Token (AT): for authorization with a service / resource (OAuth2 and OIDC)
  - Refresh Token (RT): to obtain additional access tokens (OAuth2 and OIDC)
  - ID Token: contains user information (only OIDC)
- Several flows:
  - Authorization code flow (standard web flow, e.g. “Sign in with Google”)
  - Implicit flow (mainly for SPAs)
  - Refresh flow (to obtain additional ATs from a RT)
  - Device flow (for input constrained devices (e.g. TVs)
  - Dynamic client registration flow
  - ...

Requirements for a solution

- Provide an easy to use interface so that
  - Applications can easily obtain access tokens (ATs)
  - Users can obtain ATs on the command line
    - syntax that allows easy integration into other program calls
  - Integration with major linux distributions (and MacOs)
- Provide access tokens (ATs) to
  - applications
    - interfaces for Applications that need ATs
  - commandline
- Hide complexity:
  - refresh token, ID token, client registration, client secret handling, how many users per client?
  - Handle / hide OIDC specific communication
- Security
  - Strong cryptography, careful implementation, security audits

Our solution oidc-agent

- Design based on ssh-agent:
- ssh-agent => **oidc-agent**: Daemon, used to
  - Communicate with OpenID Provider
  - Interface to clients
  - Store secrets in (encrypted) memory
- ssh-keygen => **oidc-gen**:
  - Register OIDC client and initialise configuration
  - Store encrypted data (RefreshToken, client_secret, ...)
  - oidc-keychain – enables using oidc-agent between sessions
- ssh-add => **oidc-add**:
  - Prompt for password and load secrets into agent
- ssh => **oidc-token**:
  - Obtain and return an access token

oidc-agent - Advanced features

- Agent is integrated with Xsession to autostart at startup
- Agent can make use of ssh-askpass to load configurations, when necessary
- Agent forwarding:
  - ssh to remote hosts
  - private material (i.e. refresh token) never leaves the local machine
- Support for debian-, redhat- and mac-os based systems
- Works with many OIDC providers:
  - IAM, EGI-Checkin, eduTeams/satosa, Keycloak, Human-Brain, B2Access, ...
- Supports multiple profiles (user accounts in different OIDC providers)
- Supports „aud“

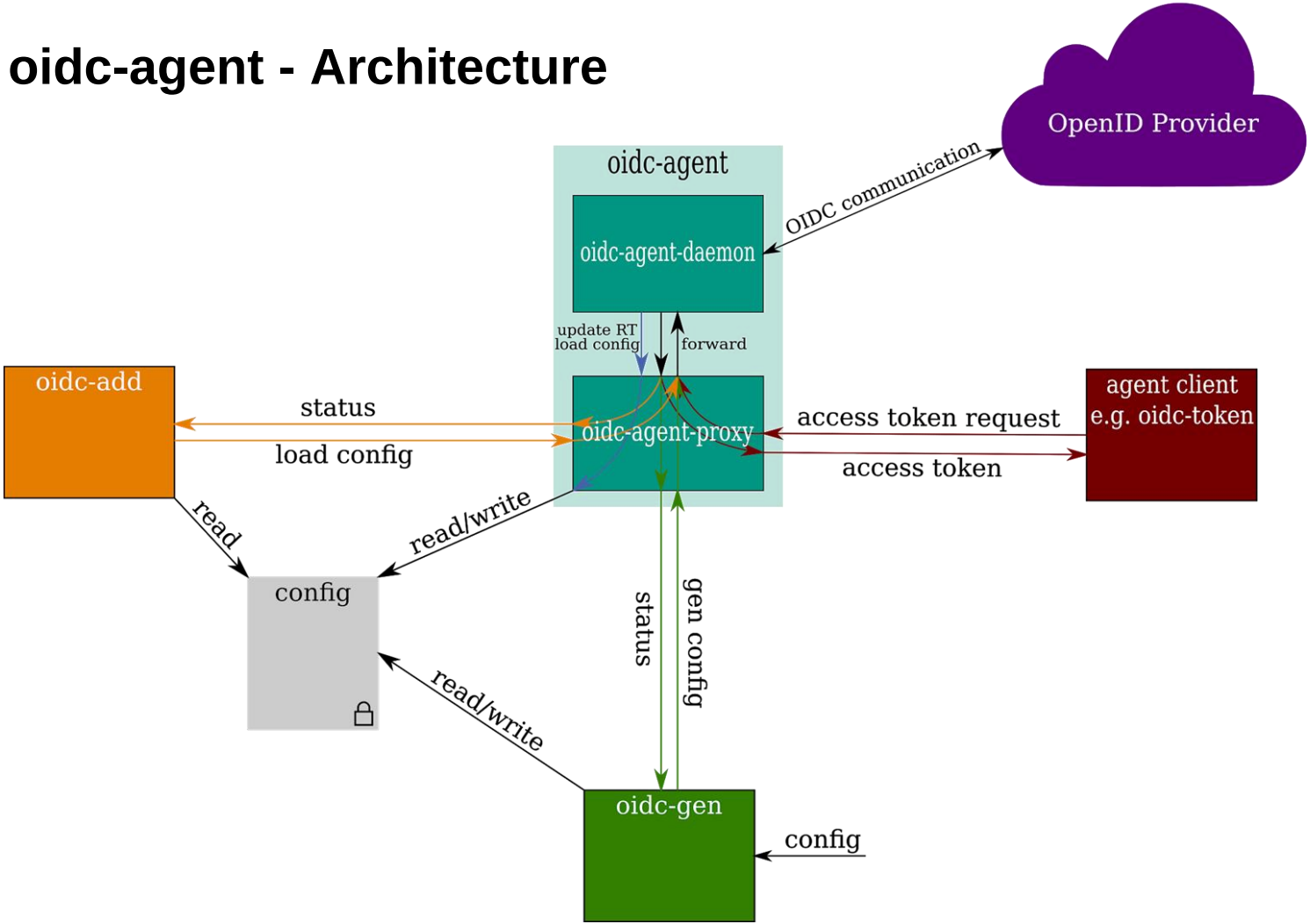
oidc-agent - Client registration

- Client registration is rather complex, and depends on OIDC Provider
- Many options for client registration:
  - Dynamic client registration (default)
    - oidc-agent self-registers himself as a new client
    - depending on the provider, this may require an initial token
    - not supported by all providers
    - **`IAM, HBP (protected)`**
  - Public client (fallback)
    - oidc-agent comes as a public client for all providers that support it
      - pre-registered by us with the correct configuration parameters and approved by the provider
    - not supported by all providers
    - **`KIT, Google, IAM, HBP, Elixir, EGI`**
  - Manual client registration (fallback of fallback)
    - user registers a client manually (e.g. through a web interface)
    - user has to specify the correct configuration parameters
    - **`eduTEAMS, HDF, B2Access`**

oidc-agent - Security highlights

- All sensitive information on disk is encrypted (using libsodium)
- Everything (sensitive) in RAM is obfuscated (dynamic random passwords)
- Privilege separation supported
  - Enforcement via seccomp available (off by default)
  - Separate processes for disk- and network access
- Keep the user from stupid moves
  - unlike ssh we do allow them (if you RTFM)

oidc-agent - Architecture



oidc-agent - usage

- Generating a new account configuration:
  - General: $ oidc-gen [<shortname>]
  - IAM
    - $ oidc-gen iam
  - EGI-Checkin
    - $ oidc-gen --pub egi
  - Google
    - $ oidc-gen --pub google
  - Any other provider:
    - https://indigo-dc.gitbooks.io/oidc-agent/provider.html#how-to-get-an-account-configuration-with

oidc-agent - usage

- Obtaining access tokens from the agent:
  - $ oidc-token <shortname>
    - Prints the AT for the requested account configuration
  - $ curl -H “Authorization: Bearer `oidc-token <shortname>`” https://….
    - Integrate the oidc-token call into another command, e.g. curl call
  - $ export OIDC_AT=`oidc-token <shortname>`
    - Put the AT into an environment variable
  - $ eval `oidc-token -oOIDC_AT –c <shortname>`
    - Put the AT into the OIDC_AT environment variable
  - There is also additional information available (expiration time, issuer url, ...)

oidc-agent - usage

- Applications that need an AT can obtain an AT directly from the agent:
  - User does not have to pass anything (in some cases the shortname)
  - Applications can assume valid ATs
    - While content of environment variable would expire at some point
  - API interfaces available for **c, go, and python**
- Applications that use oidc-agent:
  - wattson
  - feudalSSH
  - unicore command line client

oidc-agent - Compatibility

- oidc-agent was tested to work with the following providers:
  - Eudat / B2Access (Unity)
  - eduTEAMS
  - EGI-Checkin
  - Elixir
  - Google
  - Helmholtz Data Federation
  - Human Brain Project
  - IAM (Indigo, Deep, Extreme, WLCG)
  - KIT
- **It’s easy to get oidc-agent to work with any OpenID Provider. In case of problems, please contact us: oidc-agent-contact@lists.kit.edu**

oidc-agent - Summary

- https://github.com/indigo-dc/oidc-agent
  - *Star* the project, if you like it!
- MIT License
- Documentation: https://indigo-dc.gitbooks.io/oidc-agent/
- Available for:
  - Debian/Ubuntu: http://repo.data.kit.edu/
  - CentOS7: https://github.com/indigo-dc/oidc-agent/releases
  - MacOS: https://indigo-dc.gitbooks.io/oidc-agent/macos.html#installation
  - Windows: Might come, if there is interest. Let us know.
- Subscribe to updates: https://www.lists.kit.edu/sympa/subscribe/oidc-agent-user