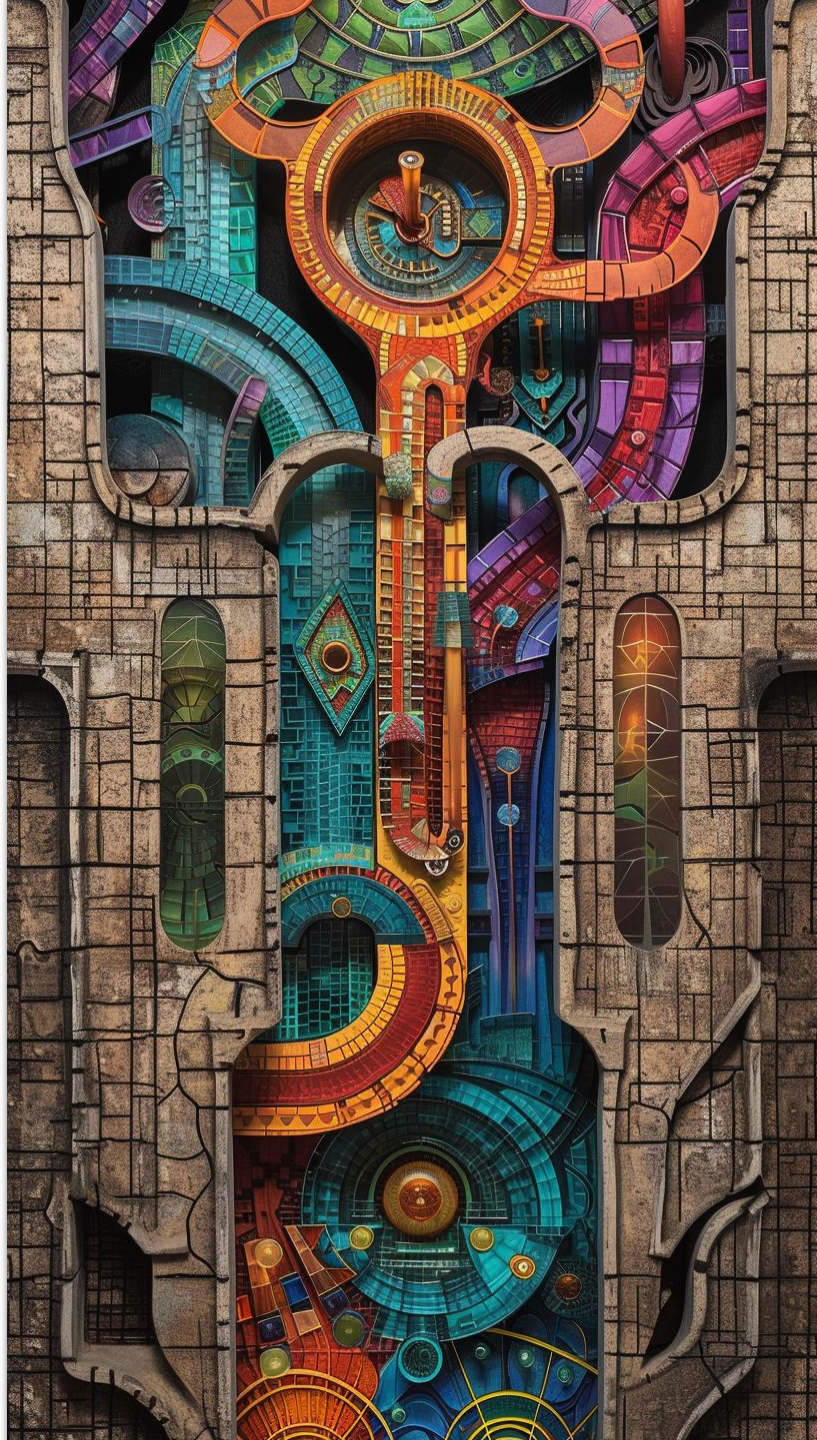
FOSDEM 2024

Brussels / 3 & 4 February 2024

POSIX identities out of OAuth2 identity providers

How to redesign SSSD and Samba

Alexander Bokovoy || Andreas Schneider || Sumit Bose



Who are we?

Alexander Bokovoy

- Software engineer at Red Hat
- Focus on identity management and authentication in Red Hat Enterprise Linux and Fedora Project
  - FreeIPA, SSSD, Samba, MIT Kerberos
- Samba Team member, FreeIPA core developer

Andreas Schneider

- Software engineer at Red Hat
- Samba maintainer for Red Hat Enterprise Linux and Fedora Project
  - Samba, libssh, cmocka, ...
- Samba Team member

Sumit Bose

- Software engineer at Red Hat
- SSSD core developer

What is this talk about?

- POSIX identities: past and present
- FreeIPA, SSSD, Samba, and MIT Kerberos
- Future

POSIX identities

- POSIX identities
  - Stable user/group information (UID and GID values) is used to run processes in environments, compatible with POSIX standards
  - File system access is arbitrated with IDs, not user/group names. Names resolved to IDs by the operating system components
  - POSIX identity metadata: what shell to run at login, where to find default home directory

- Focus: traditional workstations and servers in enterprise environments
  - Users have the same UID/GID values on all machines they can login to
  - Data stored locally under different user/group IDs belong to different users

```
[root@0526d529abab /]# id sssd
id: ‘sssd’: no such user
[root@0526d529abab /]# id nobody
uid=65534(nobody) gid=65534(nobody) groups=65534(nobody)
[root@0526d529abab /]# # nobody is more real than SSSD
```

POSIX ID needs and their coverage by OAuth2 IdPs

- POSIX users
  - User name
  - UID and (primary) GID numbers (32-bit)
  - [may be] Description (`gecos`)
  - home directory
  - Shell

- POSIX groups
  - Group name
  - GID number (32-bit)
  - [may be] Description
  - List of group members

OIDC Connect default claims (excerpt from OIDC Connect specification)

Member	Type	Description
sub	string	Subject - Identifier for the End-User at the Issuer.
name	string	End-User's full name in displayable form including all name parts, possibly including titles and suffixes, ordered according to the End-User's locale and preferences.
given_name	string	Given name(s) or first name(s) of the End-User. Note that in some cultures, people can have multiple given names; all can be present, with the names being separated by space characters.
family_name	string	Surname(s) or last name(s) of the End-User. Note that in some cultures, people can have multiple family names or no family name; all can be present, with the names being separated by space characters.
middle_name	string	Middle name(s) of the End-User. Note that in some cultures, people can have multiple middle names; all can be present, with the names being separated by space characters. Also note that in some cultures, middle names are not used.
nickname	string	Casual name of the End-User that may or may not be the same as the `given_name`. For instance, a `nickname` value of `Mike` might be returned alongside a `given_name` value of `Michael`.
preferred_username	string	Shorthand name by which the End-User wishes to be referred to at the RP, such as `janedoe` or `j.doe`. This value MAY be any valid JSON string including special characters such as `@`, `/`, or whitespace. The RP MUST NOT rely upon this value being unique, as discussed in **Section 5.7**.

Authenticated access

- User information is needed before user session is established
  - SSH server or console login process needs to know POSIX identity and user metadata before login

- OAuth2 IdP requires client identification and user consent to get access to user information
  - OAuth2 client identification ~ host enrollment into enterprise domain
  - OAuth2 client credentials need to be guarded on the host side if anything non-trivial is exposed through their permissions
    - TPM integration is needed

Authenticated access (2)

- Host enrollment
  - Simplest case = create OIDC client creds for this host
    - Users can do so with public IdPs, an enrollment tool can handle the details on behalf of a user
    - Protect OIDC client creds locally with systemd-creds or similar interface (binding to TPM)
  - Advanced case: Azure AD allows host enrollment with a special endpoint
    - Authenticate against a Broker application endpoint on user's behalf
      - Windows does it with pre-authorized (private) Windows OIDC client creds
    - Client then register by exchanging cryptographically signed data with a DRS service
    - Expects integration with TPM and derivation of tokens based on the primary resource token's possession

Draft Azure AD join code for Samba from David Mulder (Samba Team, SUSE): https://gitlab.com/samba-team/samba/-/merge_requests/3394

Enrolled and (dangerous)

- Enrolled host is really an OIDC client
  - Define IdP claims to POSIX ID metadata mapping
  - Process data to retrieve or generate POSIX information
  - Perform authorization against IdP to delegate authentication on login
- Online only
  - IdP is not available offline
  - Easy: offline login as a PAM stack option
  - No access token delegation to user single sign-on into web applications

Generate POSIX information

- IdPs have no POSIX information
  - Algorithmic mapping
    - Have N non-overlapping ID ranges defined by (startID, sizeID) for each range
    - num = hash(unique_identifier_of_ID_server) % N
    - offset = f(unique_attribute_value) % sizeID[num]
    - POSIX-ID = startID[num] + offset
  - Hash is a configurable message digest function with configurable seed
  - f is configurable function depending on unique attribute/claims from the object properties

Fully qualified names

- username@idp.suffix
  - Generate ID range off the idp.suffix
  - Generate ID offset in the range by username value
- Works for multiple IdPs
- Stable ID mapping on multiple workstations without additional requirements from IdP
- No support for username aliases (ID collisions)

Generate POSIX information

- IdPs do have POSIX information
  - No IdP provided one so far, green field
- Solution: use IdP-integrated OAuth2 application
  - Provide ID ranges
  - Provide user POSIX ID metadata
  - Enforce data consistency
  - Provide access control extensions
- Local system configuration
  - Store mapping locally, allow admins to adjust
  - Pull system configuration from the OAuth2 application
    - Self-provisioning in large environments

POSIX OAuth2 application

- Host enrollment mechanism
  - Same enrollment process for all IdPs
  - Same integration mechanism for different enterprise domain systems
- POSIX ID self-management for users (read/write for specific data)
  - Customizable by admin and users
- May implement algorithmic mapping

Can we trust federation?

- Federation is common
  - User authentication is delegated to other OAuth2 IdP (Google, Azure, Github, Gitlab, etc.)
  - Some claims from the federated IdP response used to fill in user claims in our IdP
  - There is no way to know origin of the claims in a response

- What should we trust for POSIX needs?
  - Multiple IdPs run POSIX OAuth2 app
    - Use Identity chaining to communicate between them and coordinate POSIX ID mapping in trusted environments

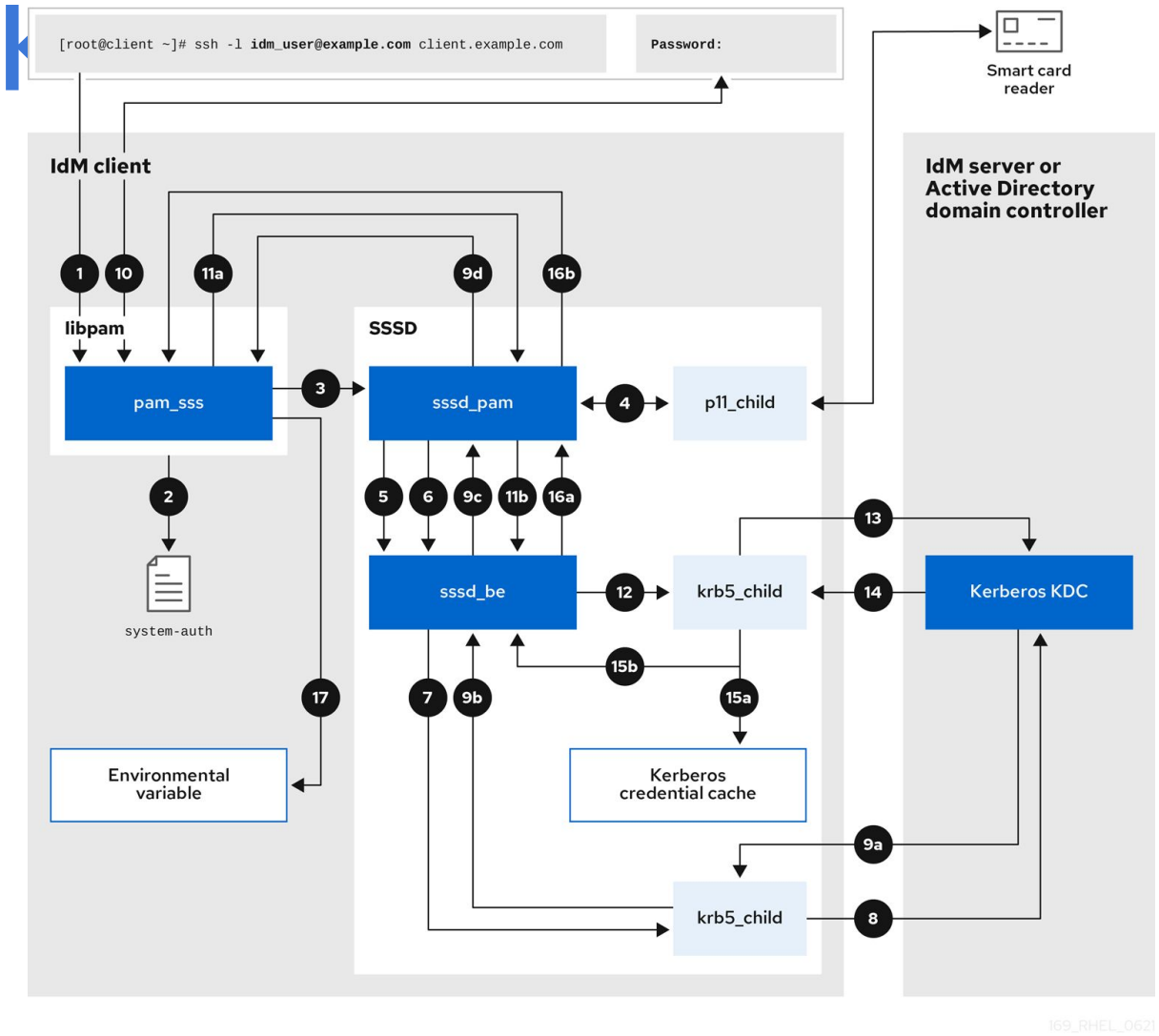
draft-ietf-oauth-identity-chaining is promising

- Requires explicit cross-domain trust agreement, unrealistic for public IdPs
- Similar to S4U extensions and constrained delegation in Kerberos

Authentication at login time

- IdP authentication
  - Typically browser based
  - Needs a browser before login
    - Or a device authorization flow
    - See Iker's 'Passwordless authentication in the GUI' talk
  - Login fact should be reusable in the session
    - SUDO reauthentication should not be constant
    - Local SSH access should be seamless
    - Browsers should be able to sign-on seamlessly

Authentication with



Detailed description is in RHEL IdM guide 'Configuring and managing Identity Management': 8.3. Data flow when authenticating as a user with SSSD in IdM

Use OAuth2 behind Kerberos authentication

- Done already with FreeIPA
  - KDC authenticates user through OAuth2 device authorization grant flow against IdP
  - Issues Kerberos ticket with 'idp' authentication indicator
  - PAM module pam_sss_gss can check authentication indicator to limit Kerberos ticket use for PAM authentication and authorization
    - Gives SUDO authentication
  - Web browsers can already use Kerberos tickets for single sign-on
  - Use of Kerberos for VPN, SSH, network file systems' access

- Downsides:
  - Requires FreeIPA deployment

Use OAuth2 behind Kerberos authentication

- Run KDC locally
  - Use a pre-defined realm
  - Run KDC in a namespace only accessible by selected apps
  - Use KDC proxy, IAKerb extension, and GSS proxy
- Re-use existing code from FreeIPA and SSSD
  - MIT Kerberos pre-authentication plugins by SSSD to authenticate against IdPs
  - No changes to the rest of the system
  - Drop-in configuration for the pre-defined realm
  - [demo]
- Interoperate
  - Use OAuth2 IdP to allow users to trust each other machines for access
    - Manage cross-realm trust on their behalf per-machine automatically
  - Use IAKerb Kerberos extension to let Kerberos propagate without direct access to the KDC

- Crazy?
  - Microsoft will use local KDC in Windows 11 to solve the NTLM problem
  - See Microsoft's talk "The Evolution of Windows Authentication":
  - We have to work on that anyway

Access control

- Who can login to that account on that machine
- Who is authorized to use these PAM services
- Who can raise privileges to run SUDO
- Can we move SSSD decisions to **that** OAuth2 application?
  - Expose an endpoint to handle POSIX-friendly abstractions
  - Let the endpoint to map those to OAuth2-friendly world
    - Zanzibar-like system as a backend? SpiceDB?

Flood of changes?

- Common
  - A library to handle algorithmic POSIX ID mapping for OAuth2-provided data
    - Handle SIDs and POSIX ID ranges together
- SSSD
  - Identity: Identity provider to talk to OAuth2 IdP
  - Authentication: no change if local KDC adopted
  - Access control: access provider to talk to OAuth2 IdP
- Samba
  - Make MIT Kerberos KDC fully supported
  - Make idmap modules handle multiple ID ranges and manage them automatically
  - Make Samba to support being enrolled to multiple “domains” properly
  - Add OAuth2 idmap support
  - Integrate OAuth2 data source to DCE RPC
- Kerberos implementations
  - Heimdal Kerberos
    - Enable dynamic pre-authentication plugins
  - MIT Kerberos
    - Pluggable PAC modules
  - GSSAPI
    - Add API to help dealing with passwordless authentication mechanisms in Kerberos
- Enrollment tools
  - Integrate OAuth2 enrollment and configuration generators
- Graphical environments
  - Login enhancements

Thank you!


sssd
SAMBA
TEAM
FreeIPA
Open Source Identity Management Solution