SSSD and IdPs

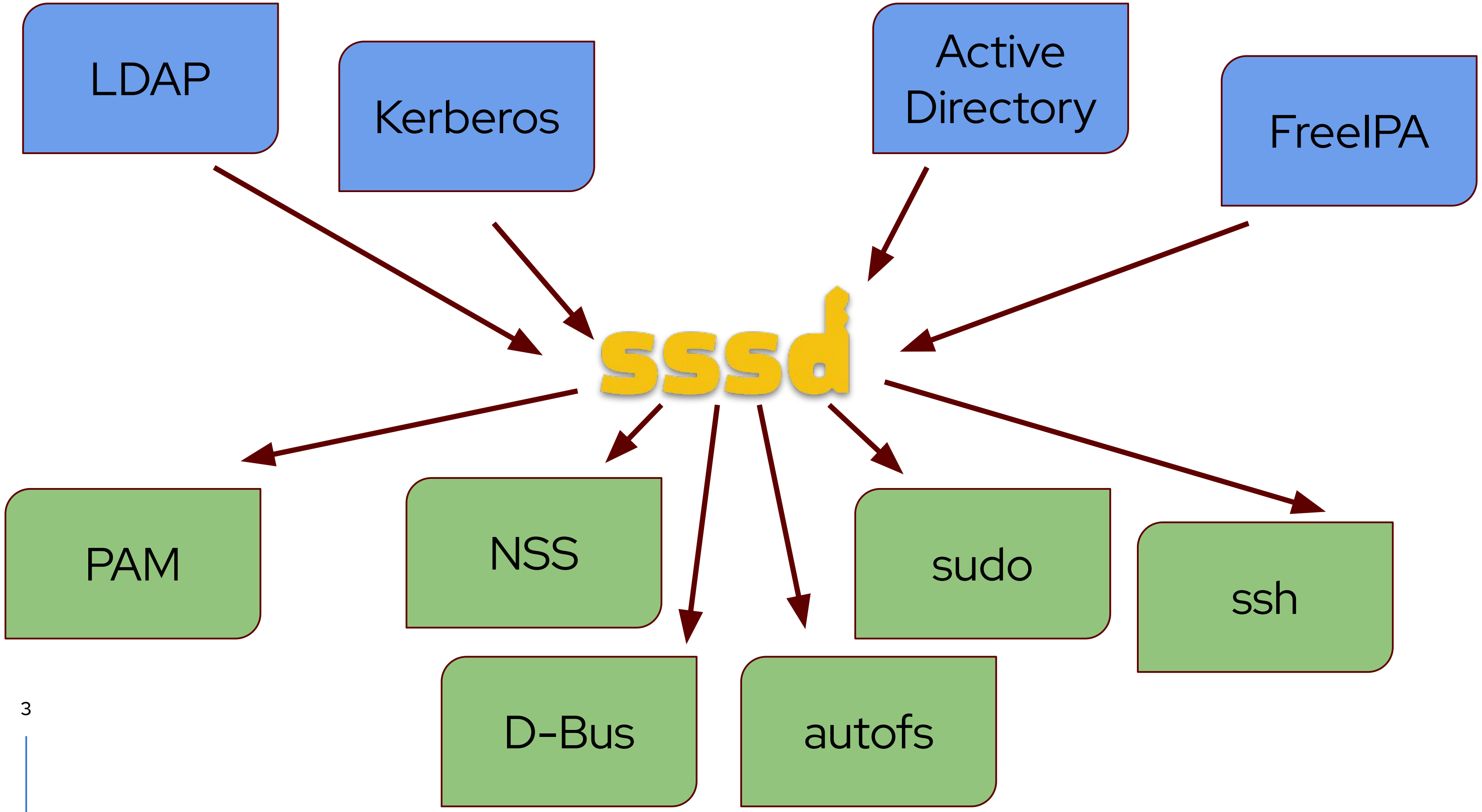
Sumit Bose

Who?

Sumit Bose

- Software Engineer at Red Hat
- Member of the SSSD team
- Maintainer of `realmd` and `adcli`

Where do we come from?


LDAP

Kerberos

Active Directory

FreeIPA

sssd

PAM

NSS

D-Bus

autofs

sudo

ssh

Centralized

Identity

Management

Where do we come from?

LDAP

POSIX schema RFC2307 and the widely used draft RFC2307bis

Kerberos

Platform independent
long tradition in the UNIX/POSIX environment

Well established integration in POSIX environments

Where do we come from?

PAM

Pluggable Authentication Modules

NSS

glibc’s Name Service Switch

It is expected that users can be looked up before authentication.

User Authentication and User and Group Lookups are independent

Why?

- There is already FreeIPAs IdP integration
    - This has many benefits, especially for larger environments
- Many environments, even small ones, need an IdP e.g. for web-based applications
- The integration in SSSD will fill the gap for smaller environments
    - No extra complexity caused by additional products

[FreeIPA IdP integration]

Where do we want to go?

OIDC/OAuth2.0 based
Idendentity Providers (IdP)

Entra ID

Keycloak

Google

Okta

Auth0

Amazon

many more ...

Github

Where do we want to go?

OIDC/OAuth2.0 based
Idendentity Providers (IdP)

- standards only cover authentication/authorization
- web browser based interaction
- user identity token might be returned after authentication
- each provider has it's own REST based user/group lookup APIs
- no common POSIX attribute group or scope
- credentials required

How?

- **Create a new Client in the IdP**
  - ideally each computer has its own IdP client
  - create a random password
  - allow Device Authorization Grant
  - allow user and group lookups
- Think of it as “joining” a domain
- Might be automated in future at least for some IdPs

An **IdP client** is needed for **authentication** and **user and group lookups**

How?

- **Use Device Authorization Grant**
    - "... designed for Internet-connected devices that either lack a browser to perform a user-agent-based authorization or ..."
    - SSSD can trigger the initialization of the authentication
    - User has to finish the authentication in a browser
- Graphical logins (GDM, KDM, ...) might provide minimal browsers in future

RFC 8628: OAuth 2.0 Device Authorization Grant

How?

- **SSSD will do user and group lookups**
  - authentication with IdP client credentials
  - lookups for
    - users and groups
    - groups a user is a member of
    - groupmembers
  - plugin interface for different IdPs
    - Entra ID and Keycloak available
    - no final plugin API yet

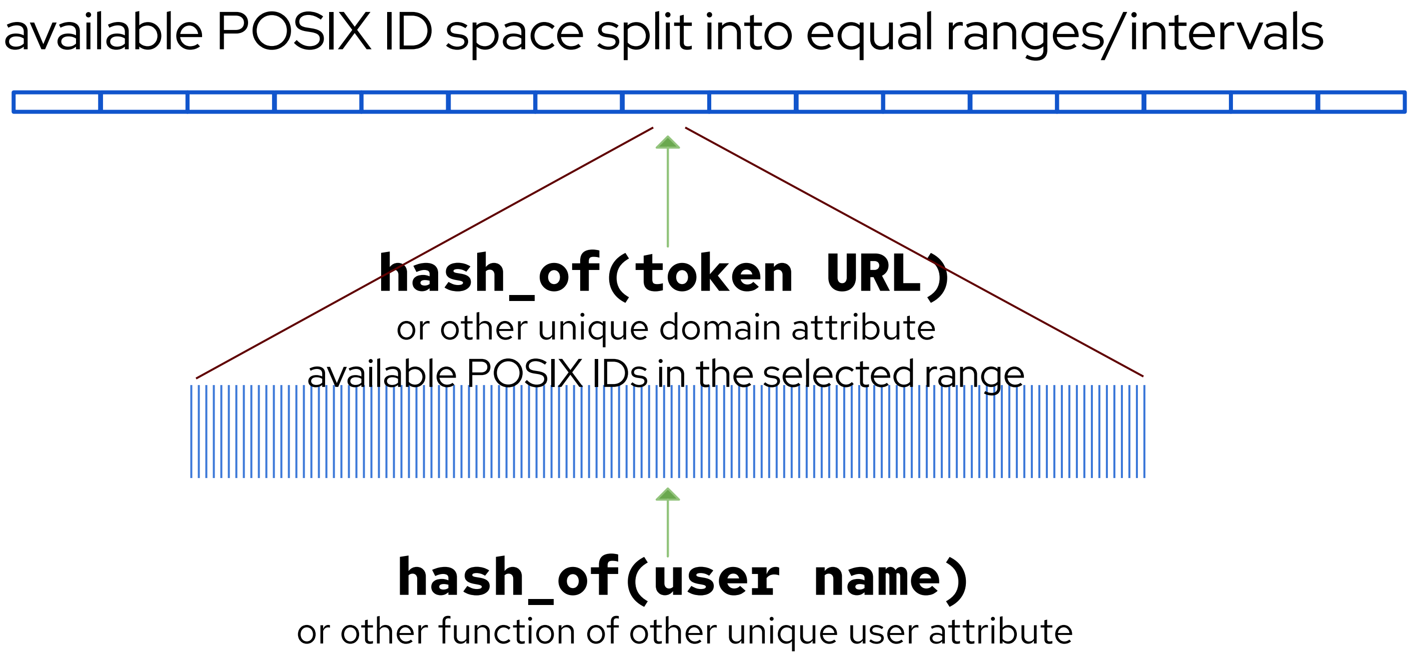
Make
glibc's
NSS interface
happy

How?

- **Generating POSIX attributes:**
  - shell: SSSD's `default_shell` option
  - home directory: SSSD's `fallback_homedir` option
  - POSIX IDs: borrow from SSSD's POSIX ID-mapping
- POSIX attributes in IdP objects might be read in a future version

How?

- **POSIX ID-mapping**


available POSIX ID space split into equal ranges/intervals
hash_of(token URL)
or other unique domain attribute
available POSIX IDs in the selected range
hash_of(user name)
or other function of other unique user attribute

This example is **not invertible,** `getpwnam()` must be called before `getpwuid()`

How does it work?

- Test packages and configuration examples
  - https://copr.fedorainfracloud.org/coprs/sbose/sssd-idp/
- Test environment
  - https://github.com/SSSD/sssd-ci-containers/

Test

Environment

How does it work?

```
[sssd]
config_file_version = 2
services = nss, pam
domains = keycloak

[domain/keycloak]
idp_type = keycloak:https://master.keycloak.test:8443/auth/admin/realms/master/
id_provider = idp
auto_private_groups = true
use_fully_qualified_names = true
debug_level = 9
idp_client_id = myclient
idp_client_secret = ClientSecret123
idp_token_endpoint = https://master.keycloak.test:8443/auth/realms/master/protocol/openid-connect/token
idp_userinfo_endpoint = https://master.keycloak.test:8443/auth/realms/master/protocol/openid-connect/userinfo
idp_device_auth_endpoint = https://master.keycloak.test:8443/auth/realms/master/protocol/openid-connect/auth/device
idp_id_scope = profile
idp_auth_scope = openid profile email

[nss]
debug_level = 9
default_shell = /bin/bash
fallback_homedir = /home/%f
```

Test

Configuration

Thank you!