Accessing domain users with SSSD interface

Pavel Březina, Jakub Hrozek, Jan Pazdziora

Audience

This talk’s audience are developers or architects who work on software that needs data (mostly users/groups) from centralized user stores based on LDAP.

Application integration with user DB

- Most applications start with some local DB
  - Fine for development and testing

- In production a centralized database of users and groups is used
  - Most enterprises already run one for workstation authentication
  - The application needs to support that

Use-cases

- Authenticate against a central server
- Display custom properties of a user
  - show AD user's phone number or department code
  - list members of groups
- Fetch related attributes from the server
  - SSH keys, certificate
- Find an entry by value
  - Find a user by SSH key or by certificate

Direct support in the application

- Use a native LDAP library to read entries
- But what about
  - authentication to the LDAP server
  - server failover
  - server discovery
  - support for different servers (schema, access control, ...)
- SSSD has the functionality to solve these

What is SSSD

- system daemon, able to connect to remote servers like LDAP, IPA, AD and serve identity and policy data (HBAC, GPO, sudo)
- OS-level APIs since the beginning
  - primarily not targeted at application support
- Recently a D-Bus API was added to SSSD

Our proposal

application - consumes objects

- SSSD API - provides objects
  - SSSD - reads attributes from the server,
    - publishes objects
      - remote server

Application changes

- Changes to the application code are not as big when using the SSSD D-Bus API than coding up LDAP support correctly from scratch...
- ...and the application gets better functionality at the same time

The SSSD D-Bus API

- discoverable (by introspection)
  - D-Feet from Gnome can display the interface
- type-safe
- fine-grained access-control
- language bindings for (almost) any language

What is exposed on the bus

- Users and groups objects
- By default exposing standard POSIX attributes
  - user: name, uid, gid, gecos, home, shell
  - group: name, gid
- Additional attributes can be added
  - certificate, avatar, phone number, …

D-Bus 101, with SSSD as an example

- The SSSD is connected to the system bus
  - as opposed to session bus
- Applications register well-known names
  - `org.freedesktop.sssd.infopipe`

SSSD interfaces on the bus

- Each path on the bus exports interfaces
  - Interface to query users or groups
    - `org.freedesktop.sssd.infopipe.Users`
  - The interfaces define methods
    - `FindByName, FindByID, ...`
  - Users and groups are represented as objects
    - methods return object paths to represent users or groups
    - these implement other interfaces in turn

Allowing access to the SSSD interface

- the service is called `InfoPipe` in SSSD
- By default only root is permitted to talk to it
- To permit other users:
  - `vi /etc/sssd/sssd.conf`
  - locate or add the `[ifp]` section
  - add usernames or UIDs to the `allowed_uids` list
  - `systemctl restart sssd`

Example - permitting access to IFP

```
[ifp]
allowed_uids = 996, jhrozek
```

Example: Retrieve user's real name

Python:

```
>>> bus = dbus.SystemBus()
>>> users_obj = bus.get_object('org.freedesktop.sssd.infopipe',
                               '/org/freedesktop/sssd/infopipe/Users')
>>> users_iface = dbus.Interface(users_obj, "org.freedesktop.sssd.infopipe.Users")
>>> jhrozek_path = users_iface.FindByName('jhrozek')
>>> print jhrozek_path
/org/freedesktop/sssd/infopipe/Users/redhat_2ecom/10327
>>> jhrozek_obj = bus.get_object('org.freedesktop.sssd.infopipe', jhrozek_path)
>>> jhrozek_iface = dbus.Interface(jhrozek_proxy, 'org.freedesktop.DBus.Properties')
>>> jhrozek_iface.Get('org.freedesktop.sssd.infopipe.Users.User', 'gecos')
dbus.String(u'Jakub Hrozek', variant_level=1)
```

Retrieving custom attributes

- By default, only the standard POSIX set of attributes can be retrieved from the cache
- Additional attributes can be configured in sssd.conf
- Two places to edit in sssd.conf
  - tell SSSD what extra attributes to fetch from server
  - allow these attributes to be served via the D-Bus interface

Example - location from RH LDAP

- Add the extra attribute in the domain section
  - `ldap_user_extra_attrs = location:rhatOfficeLocation`
  - The format is cache_attr:LDAP attr
  - multiple domains with different schemas can be served this way
- Permit the D-Bus responder to serve this attribute
  - `[ifp]`
  - `allowed_uids = 996, jhrozek`
  - `user_attributes = +location`

Retrieving the location

Python:

```
# Retrieving the object is similar to earlier examples
>>> jhrozek_iface = dbus.Interface(jhrozek_proxy, 'org.freedesktop.DBus.Properties')
>>> jhrozek_iface.Get('org.freedesktop.sssd.infopipe.Users.User', 'extraAttributes')
dbus.Dictionary({dbus.String(u'location'): dbus.Array([dbus.String(u'RH - Stockholm')],
signature=dbus.Signature('s')) }, signature=dbus.Signature('sas'), variant_level=1)
>>> extra_attrs = jhrozek_iface.Get('org.freedesktop.sssd.infopipe.Users.User',
                                    'extraAttributes')
>>> print str(extra_attrs['location'][0])
RH - Stockholm
```

D-Bus bindings

- Python, Java (not in RHEL), Ruby
- Several bindings available for C:
  - libdbus - low-level, portable, bit awkward to use
  - dbus-glib - convenient to use, but some people don’t like glib
  - sd-dbus - systemd implementation of the dbus protocol
    - nicer API than libdbus, but quite recent code

Future work

- SSSD at the moment exposes only POSIX users
  - normally services and apps don’t carry attributes that system users have (ID, shell, homedir)
  - SSSD also only exposes domain users
  - local user support TBD
- No authentication API yet except for PAM
- D-Bus signal support is being added
- Polkit support for access control

Apache module `mod_lookup_identity`

- Real life SSSD D-Bus client
- Retrieves attributes and group membership for authenticated user, to be consumed by Web application
- Populates `REMOTE_USER_EMAIL`, `REMOTE_USER_GROUP_1`, ...
  - Production use in Satellite 5.7, 6.0, CFME 5.3
- Support for `LookupUserByCertificate` added in RHEL 7.2
  - calls `org.freedesktop.sssd.infopipe.Users.FindByCertificate`
  - Integration with client cert authentication and smart cards