SSSD: FROM AN LDAP CLIENT TO SYSTEM SECURITY SERVICES DEAMON

ABOUT ME AND THE TALK

- I'm a developer working for Red Hat, mostly on SSSD
- Twitter: @JakubHrozek
- Github: https://github.com/jhrozek/fosdem2018
- This talk is about SSSD, but (hopefully) not about the known parts
- About what "SSSD" stands for
  - Simo's Super Secure Deamon?
  - Simo Stephen Sumit Dmitri?
- **System Security Services Deamon**
- This talk is about what the program can do beyond its core business

TALK TOPICS

- What APIs does SSSD have
- Why can SSSD handle your Kerberos tickets
- Why can SSSD handle local users
- Why can SSSD handle smart cards

SSSD APIS

TALKING TO SSSD FROM AN APP

SSSD API USE-CASES

- Many applications implement some sort of an "LDAP driver" or an "LDAP connector"
- Code reuse, DRY
- SSSD is a "domain expert", let someone else do this job
  - The job is not as easy as it might sound
  - Server discovery, affinity, fail over, caching, different schemas

HOW DOES ONE TALK TO SSSD

- API vs. plugin
- SSSD already provided several *plugins* for system APIs
  - NSS - getent passwd $user -> getpwnam(3) -> _nss_sss_getpwnam
  - PAM - su - $user -> pam_authenticate -> pam_sss
  - sudo plugin, automounter plugin, NFS plugin, ...
- Nontrivial to call directly, only through the system API
- APIs managed by SSSD directly are more flexible and/or performant

TALKING TO SSSD DIRECTLY

- D-Bus API
  - Pros: many language bindings, type-safe, signals (notifications), introspection
  - Cons: Requires a system bus, some language bindings not that great
  - Currently used by several applications like ManageIQ, Keycloak, mod_lookup_identity, ...
- C API
  - Pros: no dependencies, quite flexible (e.g. options to bypass cache)
  - Cons: Not considered stable (#define needed ATM), no bindings
  - Currently used by FreeIPA
- Would some other API be more appealing to a project?
  - REST perhaps?
  - ldapi:// ?

DEMO TIME

- D-Bus API example compared to raw Python
- Keep in mind the raw Python script doesn't to caching, failover, service discovery, ...
- Two D-Bus examples
  - https://github.com/jhrozek/fosdem2018/tree/master/dbus-api
  - The OO one is more verbose but more flexible as well
    - all objects are represented with a path regardless of how was the object found
    - signals (notifications)

SSSD-KCM

LET SSSD HANDLE YOUR KRB5 TICKETS

WHERE'S MY TICKET?

- Any successful Kerberos authentication yields a "ticket"
  - KDC initial authentication (kinit) -> TGT
  - service authentication -> service ticket
- A blob that must be stored in a *credential cache*
- Several options
  - FILE, DIR, KEYRING, ...
  - **KCM**

KCM CCACHE TYPE

- Not our idea
- Comes from the Heimdal Kerberos distribution, circa 2005
- The credentials are handled by a daemon
  - All the other credential cache types are "passive"
  - The application (e.g. kinit) is a client, KCM daemon is a server
  - Client talks to the KCM server over a UNIX socket

KCM CCACHE BENEFITS

- Stateful
  - renewals, notifications, cleanup of expired caches or on logout ...
- Credentials are not written to disk
- Better suited for containers
  - UNIX socket can be selectively shared between containers or container and host
  - The KCM deamon is subject to namespacing (root in container vs. root on host)
  - Do people use Kerberos with containers, though?

SSSD-KCM

- Why another implementation
- Why not a standalone project
  - Credentials can be stored in memory or encrypted in SSSD's database
  - (planned) Communication between SSSD-KCM and the SSSD D-Bus API provider
  - Code reuse
- You don't need the rest of SSSD at all
  - systemctl enable sssd-kcm.socket should be enough

STATUS AND FUTURE OF SSSD-KCM

- Default in Fedora since F-27
- Several open bugs, though
- Feature-wise equivalent to other ccache types, more improvements planned for F-29
  - renewals
  - notifications
  - logind-scoped credentials
- Configuration handling needs improvement

MORE RESOURCES

- SSSD upstream design page:
  - https://docs.pagure.org/SSSD.sssd/design_pages/kcm.html
- MIT documentation
  - http://k5wiki.kerberos.org/wiki/Projects/KCM_client

MANAGING LOCAL USERS

SSSD AND /ETC/{PASSWD,GROUP}

MOTIVATION

- Caching
- APIs
- Additional attributes
- Smart cards for local users
- Fully backwards compatible

CURRENT STATUS

- Caching enabled in Fedora since F-26
  - https://fedoraproject.org/wiki/Changes/SSSDCacheForLocalUsers
  - /etc/passwd and group are mirrored into sssd on-disk cache
  - any request triggers putting the user or group entry into mmap-ed cache
  - without nscd or sssd, a request would trigger opening and parsing the files
- Pros: Interoperates easily with other SSSD domains, unlike nscd
- Cons: SSSD is "fatter" than nscd, requires modifications to nsswitch.conf, the sssd memory cache size is not (yet) configurable

FUTURE DEVELOPMENT

- Improve the smart card integration for local users
  - More on this later in this presentation..
- Enable extending the SSSD database with extra attributes
  - Currently the 'sss_override' tool can be used to add certs, but there's no general API
- Extend the D-Bus API to enable user database modifications
- Implement the https://www.freedesktop.org/wiki/Software/AccountsService/ API to get a consistent API for local and remote users
- Hopefully will happen this year...

SMART CARDS WITH SSSD

FOR LOCAL AND REMOTE USERS

DISCLAIMER

- I'm not a Smart Cards expert
- The previous Jakub is
  - https://fosdem.org/2018/schedule/event/smartcards_in_linux/
  - https://www.youtube.com/watch?v=x2mpba45UVc
- Not even expert in this part of SSSD
- Nonetheless, let's illustrate the state

CURRENT STATE

- Traditionally, pam_pkcs11 had been used
  - lot of features, stable
  - also dead upstream..
- SSSD in the meantime gained support primarily for remote users
  - FreeIPA/IDM + AD trusts is the main scenario
    - match or list user(s) against a certificate stored in the directory
    - "local" authentication with the help of the keys on the smart card or Kerbreos PKINIT
  - ~~Usable~~Functional for local users as well already

SMART CARDS FOR LOCAL USERS

- SSSD must be serving users from local files
- The user database must augmented with the certificate
- pam_sss, not pam_unix must be handling authentication
- Several manual configuration changes needed now
  - certificate in sssd database
  - preauth flag file
  - dummy authentication target so that sssd falls back to cert auth
- Should work in a user-friendly manner in F-29

FUTURE DEVELOPMENT

- Usability improvements for SC for local users
- OpenSSL implementation of the helper process for SC auth
- Add more missing features that pam_pkcs11 has

DEMO TIME

- Smart card authentication with SSSD and a local user

QUESTIONS?

THANK YOU FOR LISTENING!