USING SAMBA LIBRARIES OUTSIDE SAMBA

SAMBA XP 2015

Jakub Hrozek
ABOUT ME AND THE TALK

Red Hat developer, working on SSSD full time
I'm not a Samba developer
This talk is not about Samba
WHAT IS THE TALK ABOUT?

This talk is about the libraries Samba provides.
WHAT IS THE TALK ABOUT?

This talk is about...

...what it is like to maintain Samba libraries from outside Samba
...what it is like to use Samba libraries in a large project
...some interesting ways we're using the Samba libraries
...some ways we can extend the Samba libraries
SSSD

System daemon, providing identity information, authentication and access control and for remote accounts

Supports LDAP, Kerberos, Active Directory and FreeIPA

Fairly large project, about 220 kSLOC (sloccount(1))
SSSD AND SAMBA

Heavy user of Samba code
"Standing on the shoulders of Samba"

- talloc
- tevent
- ldb
- tdb
- libndr, libndr-nbt
- libsmbcclient
- cwrap: nss_wrapper, uid_wrapper, socket_wrapper
In Fedora/RHEL, the separate libraries have been maintained by the SSSD maintainer, not the Samba maintainer.

Historical development. Let's blame Simo!

For the most part, it's been a smooth ride, but..
MAINTAINING SAMBA LIBRARIES

As a maintainer, I care about changes
Is this package version safe for a stable Fedora?
Is it a bugfix-only release or are there new features?
USING SAMBA LIBRARIES

SSSD was started by Simo
More developers are hired to work on the SSSD
Documentation is important in getting them up to speed
Result - tevent and talloc tutorials
EXAMPLE - A NEW LIBTALLOC RELEASE

https://talloc.samba.org/ points straight to Doxygen docs
NEWS file has the latest entry from 2007..
git log requires knowing where to look in the Samba tree
diff -Naur is not ideal..
EXAMPLE - CWRAP.ORG LIBRARIES

The libraries were split from Samba and are merged back

Very easy for an outsider to consume

Separate homepage with news and documentation

Development history can be viewed in a separate tree

Several tutorials around the web
THE SAMBA LIBRARIES ROCK

The libraries Samba produces are already great
With a little more polish, they could be awesome
Let's discuss how we can work together
WE COULD DO BETTER!

What's new in a release
git-shortlog or a digest to samba-technical is perfectly fine
Keep the documentation up-to-date
WE COULD DO EVEN BETTER!

Make it easier to track the individual changes
Decouple libraries from the core Samba tree
Make it clear which interfaces are stable
Do we need to rebuild the libldb memberof plugin with each ldb release?
Let the world know about your libraries!
ACCESS CONTROL USING GROUP POLICY OBJECTS
ACCESS CONTROL WITH SSSD

SSSD has several options for access control

- Allow everything
- Allow users that match a filter
- Allow users present in an ACL
- **Use Active Directory GPOs**
USING GPOS FOR ACCESS CONTROL

Group Policy can only be used for access control with SSSD
The rest is not SSSD's domain
But the code is a bit extensible (Sudo?)
CONFIGURING GPO FOR ACCESS CONTROL (IN WINDOWS)

Create-and-link the GPO in the "Group Policy Management Console"
Configure "User Rights assignments" in the "Group Policy Management Editor"
This is all Windows-specific terminology
LOGONRIGHTS TO PAM SERVICES

TRANSLATION

Windows uses LogonRights for access control
Linux uses PAM services
SSSD has default mappings and allows custom mappings in configuration
UNDER THE HOOD

Three-step processing

- LDAP processing determines what GPOs are applicable
- SMB processing fetches the INI GPO policy files
- Enforcement parses the INI GPO policy files and does the policy decision
GPO: LDAP PROCESSING

- Determines which GPOs are applicable to the target computer account
  - Linked to this OU, enabled, enforced
- Retrieves relevant attributes of applicable GPOs
  - File-system paths for example
- Checks if the GPO needs to be fetched from the AD/SMB server
  - Timeout prevents going to SMB too often
GPO: SMB PROCESSING

Fetch the GPO file using libsmbcclient
smbc_getFunctionOpen(), smbc_getFunctionRead(), ..
Stores on disk in INI format
Metadata about the INI file in LDB cache
[Unicode]
Unicode=yes

[Version]
signature="$CHICAGO$
Revision=1

[Privilege Rights]
SeRemoteInteractiveLogonRight = tuser,administrator
GPO: POLICY ENFORCEMENT

Parses the INI files stored previously in GPO cache
Extracts the records corresponding to the Logon Right mapped to the PAM service name
Allows or denies access
EXTENDING LDB WITH AN MDB BACK END
WHY DO THE WORK?

A very frequent complaint about SSSD is "it's slow"
SSSD uses its cache heavily due to its architecture
In some cases (saving large groups), the cache was taking a lot of time
LMDB 101

LMDB is a very fast, compact key-value store
Written by Howard Chu (Symas/OpenLDAP)
The API usage resembles Berkeley DB
See any of Howard Chu's presentations to learn more (FOSDEM, LDAPCon, ...
LMDB BENCHMARKS

Benchmarks comparing different DB engines are available at http://symas.com/mdb/
Pretty graphs!
LMDB wins pretty much all the time
**LEVELDB'S DB_BENCH**

https://github.com/hyc/leveldb/tree/benches

<table>
<thead>
<tr>
<th>Test</th>
<th>LMDB op/s</th>
<th>TDB op/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>fillrandbatch</td>
<td>774503</td>
<td>189630</td>
</tr>
<tr>
<td>fillseqsync</td>
<td>129</td>
<td>41</td>
</tr>
</tbody>
</table>

These are the results where TDB comes the closest..
DATABASE BENCHMARKS - TDBENCH

Attend Ralf Boehme's "tbench, a db microbenchmark, or tdb vs all the rest"
Today, 12:15 PM—13:00 PM Track 1
DATABASE BENCHMARKS - TDBENCH

./bin/tbench -d lmdb -t 10
Testing with 10 processes, loading 10000 rec each, txn_size: 10
starting bench: load
10 processes created 100000 records in 0.296 seconds (337820 rec/s)

./bin/tbench -d tdb -t 10
Testing with 10 processes, loading 10000 rec each, txn_size: 10
starting bench: load
10 processes created 100000 records in 165.263 seconds (605 rec/s)
THE IMPLEMENTATION OF LDB_MDB

Both TDB and LMDB are key-value stores
Much of the work started as s/tdb/mdb/
But lot of the code could be shared
Databases are not one-size fits all
LMDB seems nice for server-side read environments, but some environments might prefer FooDB or BarDB
LevelDB anyone?
RocksDB anyone?
SHARED CODE

Goal: Create the ldb_mdb back end on top of generic keyval utilities

Secondary goal: Reuse generic LDB functions in the modules. Some LDB functions were created later than the tdb backend

Shared code could be used also the tdb back end
LDB_MOD_*.C

Contains the generic logic currently in the TDB back end

- Common module logic
- Tevent wrappers
- Attribute filtering
- ..and more

Already there is more common code than mdb-specific code
LDB_MOD_* AND LDB_MDB STATUS

Implemented & tested

- Tevent-based back end initialization from tdb
- LDB operations (add, del, modify, search) using mdb
- Common logic was split from the operations

In progress

- Indexing
- Special attributes (case sensitiveness, seqnum, ..)
- Permissive control
TESTING LDB_MOD_* AND LDB_MDB

The generic code must be tested to be useful
New code must be compatible with the TDB back end
API tests, not tool tests
Samba, meet cmocka
LDB MODULE BENCHMARKS

We need benchmarks for different workloads.

SSSD is single-writer.

In my tests the LMDB back end was about twice as fast.
SHOW ME THE CODE!

It's a work in progress

https://github.com/jhrozek/samba-ldb-mdb/tree/mdb
THAT'S ALL!

QUESTIONS?

https://github.com/jhrozek/sambaxp2015