# Linux LDAP authentication

Riccardo Murri <riccardo.murri@uzh.ch>

04/11/2014

# What is LDAP?

LDAP is a distributed database, optimized for reading and searching.

These days, LDAP is mostly used for distributing *authentication* information, i.e., as a replacement for `/etc/passwd`, NIS/YP, etc.

# Example LDIF entry

```
# ldapsearch -x -H ldaps://idldapmaster01.uzh.ch \
  -W -D cn=idSysRoHPC2HPC,ou=Admin,ou=id,dc=uzh,dc=ch \
  -b "ou=People,ou=HPC,ou=id,dc=uzh,dc=ch" "(uid=rmurri)"
dn: uid=rmurri,ou=People,ou=HPC,ou=id,dc=uzh,dc=ch
objectClass: idAccount
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
objectClass: top
userPasswordHash: {crypt}DjY4deC6nz9F
uid: rmurri
cn: rmurri
homeDirectory: /home/user/rmurri
uidNumber: 6909
gidNumber: 1001
uzhuuid: u1042765
userPassword:: e2NyeXB0fUs0bG1vNnNHMnM5Q1U=
loginShell: /bin/bash
gecos: Riccardo Murri
mail: riccardo.murri@uzh.ch
```

*See also:* RFC 2307 for details.

# Three options for LDAP auth in Linux

1. libnss-ldap/pam-ldap: Library-only implementation by PADL software
2. nss-pam-ldapd: More recent implementation by A. De Jong, offloading LDAP connection to a shared service.
3. SSSD: Local authentication service with pluggable backends and caching features.

# libnss-ldap / libpam-ldap

**Pros:**

- very well known
- available for every distro

**Cons:**

- LDAP client library (and its dependencies) is loaded into *every* running process
- potentially one LDAP connection per process and one LDAP search per `get*ent()` call
- requires `nscd` otherwise you're spamming the server
- login and other lookup operations can hang if network is not available

*See also:* https://wiki.debian.org/LDAP/NSS

# Example libnss-ldap configuration

```
# cat /etc/ldap.conf
uri ldap://192.168.160.35 ldap://192.168.160.36
ldap_version 3
rootbinddn cn=root,dc=gc3,dc=uzh,dc=ch
pam_password md5
base dc=gc3,dc=uzh,dc=ch
nss_base_passwd        ou=People,dc=gc3,dc=uzh,dc=ch
nss_base_shadow        ou=People,dc=gc3,dc=uzh,dc=ch
nss_base_group         ou=Group,dc=gc3,dc=uzh,dc=ch
nss_initgroups_ignoreusers root,daemon,bin,...,postfix
```

# nss-pam-ldapd

Tries to address some shortcomings of `libnss-ldap`.

- offloads all LDAP connection and searching to a separate daemon called `nslcd`
- NSS clients talk to `nslcd` over local UNIX domain sockets

**Pros:**

- simple and lightweight NSS client code
- **per host connection pooling**
- in case of network failures, `nslcd` fails quickly
- **flexible attribute mapping features**

**Cons:**

- code changes quickly: some features you need may not be available in the distro you're using
- superceded by SSSD: will likely remain a niche system

# Example pam-nss-ldap configuration

```
# cat /etc/nslcd.conf
uri ldap://192.168.160.35 ldap://192.168.160.36
ldap_version 3
uid nslcd
gid nslcd
base dc=gc3,dc=uzh,dc=ch
binddn cn=root,dc=gc3,dc=uzh,dc=ch
bindpw ⋆⋆⋆⋆⋆⋆⋆⋆
ssl on
tls_reqcert demand
tls_cacertfile /etc/ssl/certs/ca-certificates.crt
nss_initgroups_ignoreusers root,daemon,bin,...,postfix
```

# SSSD, I

The *System Security Services Daemon* is developed by Red Hat and available in Fedora, RHEL6, but also Debian and Ubuntu.

Same architecture/idea as*nss-pam-ldapd*: separate client code into a daemon, and let NSS/PAM talk to it via a local UNIX-domain socket.

Pluggable backends: SSSD starts one daemon per *authentication domain*, so there is actually a family of processes:

```
# ps fauxwww | fgrep sss
root     Ss   Oct29   2:19 /usr/sbin/sssd -f -D
root     S    Oct29   8:21  \_ /usr/lib64/sssd/sssd_be -d 0 -debug-to-files -domain ldap1.ft
root     S    Oct29   0:28  \_ /usr/lib64/sssd/sssd_nss -d 0 -debug-to-files
root     S    Oct29   0:22  \_ /usr/lib64/sssd/sssd_pam -d 0 -debug-to-files
```

## SSSD, II

**Pros:**

- simple and lightweight NSS client code
- **per host connection pooling**
- **Not just for LDAP:** can do Active Directory, and Kerberos too
- **Can aggregate data from various sources**
- Can also retrieve and cache SSH *known host* entries

**Cons:**

- Configuration is more complex
- Documentation is extensive and there currently is no concise overview
- Actively developed, some features you need may not be available in the distro you're using
- Own caching daemon distinct from `nscd`

# Example SSSD configuration, I

```
[sssd]
config_file_version = 2

# what services should SSSD interface to
services = nss, pam

# index of auth backends
domains = LDAP

[nss]
filter_users = root,ldap,...,nscd

[pam]
reconnection_retries = 3
debug_level = 8
```

# Example SSSD configuration, II

```
# "Schroedinger" cluster users
[domain/ldap1.ften.es.hpcn.uzh.ch]
id_provider = ldap
auth_provider = ldap
chpass_provider = ldap
access_provider = ldap
enumerate = true
cache_credentials = true

ldap_access_filter = uidNumber>1000

ldap_uri = ldap://ldap1.ften.es.hpcn.uzh.ch
ldap_tls_reqcert = never

ldap_schema = rfc2307
ldap_search_base = dc=unizh,dc=ch
ldap_user_search_base = ou=People,ou=Matterhorn,ou=zi,dc=unizh,dc=ch
ldap_group_search_base = ou=Groups,ou=Matterhorn,ou=zi,dc=unizh,dc=ch

ldap_default_bind_dn = cn=matterhorn,...,ou=zi,dc=unizh,dc=ch
ldap_default_authtok = ********
```

# Further reading

RFC 2307, https://www.ietf.org/rfc/rfc2307.txt standard LDAP schema for UNIX user/group information

RFC 2307bis, https://tools.ietf.org/html/draft-howard-rfc2307bis-02 draft update to RFC 2307

https://wiki.debian.org/LDAP/NSS Debian's guide to configuring `libnss-ldap` and `libpam-ldap` (works almost verbatim in any Linux distro)

http://arthurdejong.org/nss-pam-ldapd/ home page of the `nss-pam-ldapd` project

https://fedorahosted.org/sssd/ home page of the SSSD project

http://preview.tinyurl.com/rhel6-sssd-guide Red Hat's guide to installing and configuring SSSD

# Appendix

# LDAP schemas

Entries in an LDAP database are sets of key/value pairs. (Keys need not be unique; equivalently: a key can map to multiple values.)

An *LDAP schema* specifies the names of allowed keys, and the type of corresponding values.

*Each entry* declares a set of schemas it conforms to; every attribute in an LDAP entry must be defined in some schema.

# X.500/LDAP Directories

Entries are organized into a *tree structure* (DIT). (So LDAP queries return subtrees, as opposed to flat sets of rows as in a RDBMS query.)

Each entry is uniquely identified by a “Distinguished Name” (DN). The DN of an entry is formed by appending a one or more attribute values to the parent entry’s DN.

LDAP accesses might result in *referrals*, which redirect the client to access another entry at a remote server.