

LDAP Implementation HOWTO

2. LDAP authentication using pam_ldap and nss_ldap

This section focuses on how to use LDAP as a NIS substitute for user accounts management. Having a lot of user accounts on several hosts often causes misalignments in the accounts configuration. LDAP can be used to build a centralized authentication system thus avoiding data replication and increasing data consistency.

At the moment the most used method to distribute users account data and other information through a network is the Network Information Service (NIS). Like LDAP, NIS is a distributed service that allows to have a central server where configuration files such as passwd, shadow, groups, services, hosts etc. are kept. The NIS server is queried by NIS clients to retrieve this information.

LDAP can offer the same functionality of NIS, moreover there are several advantages on using LDAP:

- * Information on the LDAP server can be easily used for several purposes. As outlined in this HOWTO, the same users entries on the LDAP database can be used for other applications like phone directories, mail routing, staff databases etc., thus avoiding data replication and inconsistency.
- * LDAP allows complex access control lists to be applied on the database. This allows for a fine grain tuning of permissions on the database entries.
- * A secure transmission channel between the LDAP server and the clients can be implemented through the Secure Socket Layer (SSL).
- * A fault tolerant service can be implemented using slapd replication [1] and DNS round robin queries (this is not covered in this document).
- * Having a single instance of users on the network helps to maintain users on many hosts from a single management point (i.e. you can create and delete accounts in the LDAP server and this changes are available immediately to LDAP clients).

Herein I'll focus on how an LDAP server can be used for authentication and authorization on systems providing the Pluggable Authentication Module (PAM) and the Name Service Switch (NSS) technologies, in particular I'll refer to the Linux operating system even if this instructions can be applied to other operating systems.

The environment proposed consists of an LDAP server where users account data is stored in a convenient format and a set of Unix clients using this information to authenticate and authorize users on resources in a standard Unix fashion.

A secure channel is also required in client/server communications since critical information such as user account data, should not be sent in clear over the network, this channel will be provided by the Secure Socket Layer.

On the client side a caching mechanism, needed for performance issues, can be provided by the Name Service Caching Daemon.

All (almost) the software used to build this system is Open Source.

2.1. The components of the framework

This section outlines the various components that are used to build the authentication system. For each component is given a brief description.

2.1.1. Authentication: PAM and pam_ldap.so

The Pluggable Authentication Module allows integration of various authentication technologies such as standard UNIX, RSA, DCE, LDAP etc. into system services such as login, passwd, rlogin, su, ftp, ssh etc. without changing any of these services.

First implemented by Sun Solaris, PAM is now the standard authentication framework of many Linux distributions, including RedHat and Debian. It provides an API through which authentication requests are mapped into technology specific actions (implemented in the so called pam modules). This mapping is done by PAM configuration files, in which, for each service are basically given the authentication mechanisms to use.

In our case, the pam_ldap module, implemented in the shared library pam_ldap.so, allows user and group authentication using an LDAP service.

Each service that needs an authentication facility, can be configured through the PAM configuration files to use different authentication methods. This means that it is possible, using the PAM configuration files, to write a custom list of requirements that an user must satisfy to obtain access to a resource.

2.1.2. The Name Service Switch and nss_ldap.so

Once an user is authenticated, many applications still need access to user information. This information is traditionally contained in text files (/etc/passwd, /etc/shadow, and /etc/group) but can also be provided by other name services.

As a new name service (such as LDAP) is introduced it can be implemented either in the C library (as it was for NIS and DNS) or in the application that wants to use the new name service.

Anyway, this can be avoided using a common, general purpose, name service API and by demanding to a set of libraries the task of retrieving this information performing technology based operations.

This solution was adopted in the GNU C Library that implements the Name Service Switch, a method originated from the Sun C library that permits to obtain information from various name services through a common API.

NSS uses a common API and a configuration file (/etc/nsswitch.conf) in which the name service providers for every supported database are specified.

The databases currently supported by NSS[2] are:

- * aliases: Mail aliases.
- * ethers: Ethernet numbers.
- * group: Groups of users.
- * hosts: Host names and numbers.
- * netgroup: Network wide list of host and users.
- * network: Network names and numbers.
- * protocols: Network protocols.

- * passwd: User passwords.
- * rpc: Remote procedure call names and numbers.
- * services: Network services.
- * shadow: Shadow user passwords.

Using the `nss_ldap` shared library it is possible to implement the maps above using LDAP, anyway here I'll focus only on the LDAP implementation of `shadow`, `passwd` and `group` database though all the maps above can be implemented. For most of the other maps it is even unadvisable to store them in `ldap`, as they tend not to change too often, so it is not a problem to have them locally as files, and storing them in `ldap` would cause some minor performance loss.

2.1.3. The Lightweight Directory Access Protocol

For our application LDAP is used to provide clients with information about user accounts and user groups. The standard objectclasses that are used to represent users and groups are: `top`, `posixAccount`, `shadowAccount` and `posixGroup`.

Users entries on the database must belong at least [3] to the `top`, `posixAccount` and `shadowAccount` objectclasses. Group entries must belong to the `top` and `posixGroup` objectclasses.

The implementation of `pam_ldap` and `nss_ldap` that we use refers to this objectclasses, that are described in RFC 2307.

Note: Actually LDAP NSS recognize other objectclasses

2.1.4. The Name Service Caching Daemon

The Name Service Caching Daemon (NSCD) is used to cache name service lookups and can improve performance with the services provided by the NSS.

It must be tuned with a large cache for `passwd` entries in order to have acceptable performance on the client side.

It has some disadvantages however, like the introduction of cache inconsistencies, so you would want to be sure you need this before you use it. We have successfully run some systems without it, and personally I think that it isn't really necessary on relatively small systems.

2.1.5. The Secure Socket Layer

<!--

For details on SSL refer to Section 10.

-->

SSL is needed in the communication between the LDAP server and the clients libraries (`pam_ldap.so` and `nss_ldap.so`), since sensible data, such as password entries, needs to be encrypted between the client and the server. SSL also permits the client to uniquely identify the server, thus avoiding to obtain authentication informations from an untrusted source.

Client authentication (the server identifies the client) is not supported in the current implementation of `pam_ldap` and `nss_ldap` modules though it may be useful.

2.2. Building the authentication system

This section describes the steps needed to build the authentication system using the components described in the previous section.

Figure 1. PAM Layout

image:The relationships among the pieces of the authentication system from the PAM point of view / /data/11/PAMlayout.gif (null)

Figure 2. NSS Layout

image:The relationships among the pieces of the authentication system from the NSS perspective / /data/11/NSSlayout.gif (null)

Though this layout may seem quite complex to implement, most of the components are already in place in a Linux system.

2.2.1. Server side

On the server side an LDAP server must be installed and configured. The LDAP server used is OpenLDAP, an open source LDAP toolkit including an LDAP server (slapd), library and utilities.

At the moment OpenLDAP comes with two implementation of LDAP: a V2 implementation (OpenLDAP 1.2.x) and a V3 (OpenLDAP 2.0.x) implementation

The V3 implementation provides native SSL, the V2 doesn't. Anyway it is possible to use an SSL wrapper to add SSL capabilities to the server.

2.2.1.1. Installing and configuring OpenLDAP

You can refer to the LDAP-HOWTO for instruction on installation and configuration of LDAP. Once slapd is properly configured we need to insert some data for the initial creation of the database. Therefore an LDIF (LDAP Data interchange format) file must be created. This is a text file that can be imported in the LDAP database with the command:

```
#ldif2ldbm -i your_file.ldif
```

Note: ldif2ldbm is provided with the OpenLDAP 1.2.x package, if you use OpenLDAP 2.0.x, you should use the ldapadd command (after the server is started).

If you use OpenLDAP 2.0.x (LDAPv3) you can find the standard nis schema in the file etc/openldap/schema/nis.schema, include it in your slapd.conf with the include directive, to have schema enforcement.

Here is an example of a minimal LDIF file. Each entry is separated by a blank line.

```
dn:dc=yourorg, dc=com
objectclass: top
objectclass: organizationalUnit

dn:ou=groups, dc=yourorg, dc=com
objectclass: top
objectclass: organizationalUnit
ou: groups

dn:ou=people, dc=yourorg, dc=com
objectclass: top
objectclass: organizationalUnit
ou: people

dn: cn=Giuseppe LoBiondo, ou=people, dc=yourorg, dc=com
cn: Giuseppe Lo Biondo
sn: Lo Biondo
objectclass: top
objectclass: person
objectclass: posixAccount
objectclass: shadowAccount
```

Note: Note that lines that are too long are continued on the following line started by a tab or a space, this is true too for LDIF format files

Here we defined the base DN for the organization dc=yourorg,dc=com under which are contained two sub organizational units: people and groups. Then is described a user that belongs to the people organizational unit and a group (which the user belongs to) under the groups organizational unit.

Note: Useful tools to convert existing databases into ldif format are provided by PADL and can be found at the address <ftp://ftp.padl.com/pub/MigrationTools.tar.gz>.

The LDIF file must be imported in the server while it is not running since the ldif2ldbm command builds the database directly, bypassing the LDAP server. Once the LDIF file is imported into the database, the server can be started.

2.2.2. Client side

On the client side pam_ldap.so and nss_ldap.so are required and they must be compiled using the Netscape LDAP Library (Mozilla) since it provides the required LDAPS (LDAP over SSL) API. The library is distributed in a binary package under Netscape One license and is not open source (it is public domain anyway).

The package can be extracted, for example, in the directory /usr/local/ldapsdk. Client libraries must also have access to a certificate database containing the LDAP (stunnel) server certificate and the CA certificate of the CA that signed the server certificate (marked as trusted).

The certificate database must be in Netscape format since the Mozilla LDAP API used to compile pam_ldap and nss_ldap uses certificate databases in Netscape format.

To deal with such certificate databases it is convenient to use the certutil utility found in the PKCS#11 package provided by Netscape [4].

The main configuration file for LDAP clients is /etc/ldap.conf.

Note that if you use nss_ldap, you don't strictly need to use pam_ldap.

You can use the pam_unix_auth module instead, since nss_ldap maps all getpw* and getsh* calls into LDAP lookups and pam_unix_auth uses these calls to authenticate users.

2.2.2.1. PAM LDAP Installation and Configuration

To compile and install pam_ldap, do the following:

```
$ ./configure --with-ldap-lib=netscape4 \
              --with-ldap-dir=/usr/local/ldapsdk
$ make
# make install
```

The configure switch --with-ldap-lib tells which LDAP library you are going to use. The switch --with-ldap-dir tells where you have installed your Netscape ldapsdk toolkit. This will install /lib/security/pam_ldap.so.1 and the various symlinks.

PAM has to be properly configured in order to access the new authentication system. PAM configuration files are located in the directory /etc/pam.d and are named after the service for which authentication is provided.

For example this is the PAM configuration file for the login service (in a file named login).

```
##PAM-1.0
auth      required /lib/security/pam_securetty.so
auth      required /lib/security/pam_nologin.so
auth      sufficient /lib/security/pam_ldap.so
auth      required /lib/security/pam_unix_auth.so use_first_pass
account   sufficient /lib/security/pam_ldap.so
account   required /lib/security/pam_unix_acct.so
password  required /lib/security/pam_cracklib.so
password  sufficient /lib/security/pam_ldap.so
password  required /lib/security/pam_unix_passwd.so use_first_pass md5 shadow
session   required /lib/security/pam_unix_session.so
```

Standard PAM configuration files for use with PAM can be found in the `pam_ldap` source distribution, in the directory `pam_ldap-version/pam.d`.

This files can be copied in the `/etc/pam.d` directory. Caution must be given when performing this operation, since if something goes wrong you probably will not be able to login again. It is suggested to make a backup copy of `/etc/pam.d` before installing new files there and to leave an open privileged shell.

Note: In the example `pam.d` directory, a `sshd` file is not present, so unless you create one, you will be unable to login via `ssh`, if it uses `pam` (OpenSSH does use PAM).

2.2.2.2. NSS LDAP installation and configuration

After you've unpacked the sources, check the makefile. For most configurations, it doesn't need to be edited. Anyway, if you want to use SSL you must link against an SSL aware LDAP library, such as the Netscape one.

Assuming that the `ldap sdk` is in `/usr/local/ldap/sdk` you have to modify the Makefile to enable SSL. Look for `NSFLAGS` in `Makefile.linux.mozilla` and uncomment `-DSSL`.

Also check the `LIBS` definition to see if the `ldapssl` library specified in the file is the same that you have installed (`ldap_nss.so` compiles with both `libldapssl40` and `libldapssl30`).

Then you can install the library:

```
$ make -f Makefile.linux.mozilla
# make -f Makefile.linux.mozilla install
#ldconfig
```

this installs `/lib/libnss_ldap.so`, which is the `nss_ldap` library, and a set of example configuration files, `/etc/nsswitch.ldap` and `/etc/ldap.conf`, in case they do not exist already.

Once you have installed it you must edit the NSS configuration file `/etc/nsswitch.conf`. Though LDAP can be used for all these services we use it only for `passwd`, `group` and `shadow` therefore we should have something like:

```
passwd: files ldap
group:  files ldap
shadow: files ldap
```

in the first lines of the configuration file. With this configuration, entries are first looked in the system files and, if no value is returned, the LDAP server is queried.

Note: Beware when using ldap as backup for your dns lookups. If dns cannot resolve the hostname, we're in infinite recursion, because libldap calls gethostbyname(). [from the nsswitch.ldap]

2.2.2.3. NSCD configuration

NSCD is already available in many Linux distributions, anyway it can be found within the GNU C library package.

The NSCD configuration file is /etc/nscd.conf. Each line specifies either an attribute and a value, or an attribute, cachename, and avalue. Fields are separated either by SPACE or TAB characters. cachename can be hosts, passwd, or groups (in our case we won't cache hosts).

```
enable-cache           passwd  yes
positive-time-to-live  passwd  600
negative-time-to-live  passwd  20
suggested-size         passwd  211
keep-hot-count         passwd  20
check-files            passwd  yes
enable-cache           group   yes
positive-time-to-live  group   3600
negative-time-to-live  group   60
suggested-size         group   211
keep-hot-count         group   20
check-files            group   yes
```

Keep in mind that the nscd program caches passwd entries obtained from LDAP. This means that when an user is modified on the ldap server, the nscd cache remains valid. This is avoided when using flat unix files by the check-files directive that invalidates the cache when the corresponding file is modified. Such a mechanism should be generalized, at the moment anyway does not apply to LDAP. A way to avoid possible misalignments between the LDAP server and the cache is to invalidate the cache manually when updating passwd entries with the command:

```
#nscd --invalidate=TABLE
```

Where TABLE can be passwd, groups or hosts.

To avoid confusion when testing, do not use nscd.

Moreover using nss and nscd will produce a lot of open file descriptors, so is easy to run out of available file descriptors on the system (this can hang your system).

You can increase the maximum number of file descriptors in a Linux box (Kernel 2.2.x) with something like:

```
#echo 16384 &#62; /proc/sys/fs/file-max
```

The maximum number of file descriptors suggested for a system depends anyway from the configuration of your system.

2.2.2.4. LDAP client configuration file

The LDAP client configuration file /etc/ldap.conf is read by pam_ldap and nss_ldap as well as other LDAP clients. The following is an example of how it should look like in our environment.

```
#
# $Id: section-pamnss.sgml,v 1.2 2001/03/26 16:57:07 rolek Exp $
# This is the configuration file for the LDAP nameservice
# switch library and the LDAP PAM module.
# PADL Software
# http://www.padl.com
#
# If the host and base aren't here, then the DNS RR
# _ldap._tcp.[defaultdomain]. will be resolved. [defaultdomain]
# will be mapped to a distinguished name and the target host
# will be used as the server.
#
# Your LDAP server. Must be resolvable without using LDAP.
host 192.111.111.111
#
# The distinguished name of the search base.
base dc=yourorg, dc=com
#
# The LDAP version to use (defaults to 2,
# use 3 if you are using OpenLDAP 2.0.x or Netscape Directory Server)
# ldap_version 3
#
# The distinguished name to bind to the server with.
# Optional: default is to bind anonymously.
# binddn cn=manager,dc=padl,dc=com
#
# The credentials to bind with.
# Optional: default is no credential.
#bindpw secret
#
# The port.
# Optional: default is 389. 636 is for ldaps
port 636
#
# The search scope.
#scope sub
#scope one
#scope base
#
# The following options are specific to nss_ldap.
#
# The hashing algorithm your libc uses.
# Optional: default is des
#crypt md5
#crypt sha
#crypt des
#
# The following options are specific to pam_ldap.
#
# Filter to AND with uid=%s
pam_filter objectclass=posixAccount
#
# The user ID attribute (defaults to uid)
pam_login_attribute uid
#
# Search the root DSE for the password policy (works
# with Netscape Directory Server)
#pam_lookup_policy yes
#
# Group to enforce membership of
#
#pam_groupdn cn=PAM,ou=Groups,dc=padl,dc=com
#
# Group member attribute
pam_member_attribute memberuid
# Template login attribute, default template user
# (can be overridden by value of former attribute
```

Note: To avoid problems with the various applications that may read this file it is suggested not to use tabs between parameters and values, only a single space.

The `pam_groupdn` directive is useful when an LDAP server provides authentication information to a pool of clients, but the user should be authorized only on a set of clients. This directive can provide the same functionality of NIS netgroups.

The SSL configuration directives are not documented in the package, but they tell to enable SSL and where the file containing the LDAP server certificate and the CA certificate is stored.

A Netscape certificate database named `cert7.db` is searched in `sslpath`. This file must contain the server certificate and the CA certificate (unless the server certificate is self signed). There are two ways to generate this file: using the Netscape PKCS#11 tools or using the Netscape browser.

With the Netscape browser, after you have started `slapd` and `stunnel` on the server you can use Netscape Navigator to connect to the URL `https://your.ldap.server:636/`, you will be prompted to insert the server certificate in your database. Also the CA certificate (provided by your CA) must be loaded in the database (unless you are using a self signed certificate). At this point you can copy the `$HOME/.netscape/cert7.db` in `sslpath`. It is preferred that you use a scratch account with a default `cert7.db` file since other server certificates, that may be present in your personal certificate database, will be considered by your LDAP client as trusted authentication servers. Once the browser has imported the server certificate it can be used to debug SSL since it will behave like the `pam` and `nss` libraries.

2.3. Starting up

On the server side you have to start `slapd` (the LDAP daemon process) with a command like:

```
# slapd
```

If you use `stunnel`, it has to be started on the LDAPS port 636:

```
# /usr/local/sbin/stunnel -r ldap -d 636 \  
-p /usr/local/ssl/certs/stunnel.pem
```

If you use OpenLDAP 2.0.x, compiled with TLS (OpenSSL), you can start the server using the command

```
# slapd -h "ldap:/// ldaps:///"
```

On the client `nscd` can be started with the a startup script, usually found in many Linux distributions:

```
# /etc/rc.d/init.d/nscd start
```

If PAM and NSS are correctly configured this should be enough.

2.4. Accounts maintenance

At this point account creation and maintenance should be done using LDAP client tools.

Unfortunately these general purpose tools are not intended for Un*xaccounts maintenance. The one that seems to be enough versatile is the LDAPBrowser/Editor (<http://www-unix.mcs.anl.gov/~gawor/ldap>) that allows to set passwords in various formats and can use SSL to connect to the server.

2.5. Known limits

As it is for NIS with a single master server (no slave servers), LDAP without a replication mechanism represents a single point of failure for the authentication system. For authentication purposes it is rather important to implement LDAP replication. The server that comes with OpenLDAP (slapd) provides replication capabilities.

2.6. File permissions

The following are the file permissions that should be applied to some of the files used by the authentication system.

```
-rw-r--r--  root.root  /etc/ldap.conf
-rw-----  root.root  /usr/local/etc/openldap/slapd.conf
-rwxr-xr-x  root.root  /lib/security/pam_ldap.so.1
-rw-r--r--  root.root  /lib/libnss_ldap-2.1.2.so
-rw-r--r--  root.root  /usr/local/ssl/certs/cert7.db
-rw-----  root.root  /usr/local/ssl/certs/stunnel.pem
```

Notes

[1]

A mechanism that permits LDAP database replication between servers.

[2] It is not a case that these are the maps provided by NIS.

[3] An entry can belong to several object classes.

[4] In a tricky way, it is also possible to use the Netscape Communicator certificate database. rate this article:

current rating:

Your rating: