ISTITUTO NAZIONALE DI FISICA NUCLEARE
Sezione di Milano

INFN/TC-00/15
8 Settembre 2000

LDAP as a Network Information Service

Giuseppe Lo Biondo
Giuseppe.LoBiondo@mi.infn.it

INFN, Sezione di Milano,
Via Celoria, 12 I-20133 Milano, Italy

Abstract

This document 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.

PACS:89.80 (C.S. and Technology)

Published by **SIS-Pubblicazioni**
Laboratori Nazionali di Frascati

1 Copyright Notice

2 Conventions used in this document

Commands, file names and configuration files are in `fixed width` fonts. Commands that can be executed by any user are prefixed with the dollar sign ($). Commands that must be executed under the root account are prefixed with the hash sign (#).

3 Introduction

The *Lightweight Directory Access Protocol* [1] (LDAP) seems to address many applications where information centrality, retrieving and maintenance is critical. LDAP, in fact, can be used to access data stored in a central server in a fast, standard and uniform way.

Tough the representation of data used by LDAP may not be very structured, like in a relational database, this is not a concern of many administrative tasks, like user accounts management where the entries of the database are simple user records.

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` 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. For example 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 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).

An LDAP service can then be used by computer pools to retrieve user accounts information for authentication and authorization purposes, and, at the same time, provide other services.

Herein we will 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 Un*x 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*.

4 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.

4.1 Authentication: PAM and pam_ldap.so

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

[1] A mechanism that permits LDAP database replication between servers.

First implemented by Sun Solaris, PAM is 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 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 gain access to a resource.

4.2 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`).

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 wishes to use the new nameservice.

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 using technology based actions.

This solution was adopted in the GNU C Library that implements the *Name Service Switch*, a method originated form 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 are specified the name service providers for every supported database.

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.

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

- 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 nss_ldap it is possible to implement the maps above using LDAP, anyway this document is focused only on the LDAP implementation of shadow, passwd and group database tough all the maps above can be implemented.

4.3 The Lightweight Directory Access Protocol

The "Lightweight Directory Access Protocol" is an open-standard protocol for accessing information services. The protocol runs over Internet transport protocols, such as TCP, and can be used to access stand-alone directory servers or X.500 directories. [3]

Information on LDAP is represented in a hierarchical way, in the so named *Directory Information Tree* (DIT).

The representation of information on LDAP is based on the *entry*, relationships among entries are hierarchical and built on a special entry attribute, the *objectclass*. This attribute is used to establish what the entry represents, a naming schema defines relationships among objectclasses. Each entry is uniquely identified by the entry *Distinguished Name* (DN).

Objectclasses are described in a *schema*, if an object belongs to an objectclass it *must* and *could* have some attributes on the basis of the objectclass definition.

Applications use objectclasses to know how entries are represented on the server, since the client and the server share the same objectclass schema. There are several standard objectclass, anyway, it is simple to build a custom objectclass if standardization is not a concern.

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*.

[3] From the OpenLDAP FAQ.

Users entries on the database must belong at least [4] 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 [5] that we use refers to this objectclasses that are described in RFC 2307 [2] (see appendix B for RFC2307 objectclasses definitions).

4.4 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 gain acceptable performance on the client side.

4.5 The Secure Socket Layer

The *Secure Socket Layer* (SSL) is an application layer protocol that provides a secure transmission channel between parties. It is based on public key cryptography systems (RSA) and on X.509 certificates.

The SSL protocol provides:

- data encryption: Client/server session is encrypted.
- server authentication: Client can verify the server identity.
- message integrity: Data is not modified during transmission.
- client authentication: Server can verify the client identity.

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. SLL 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 tough it may be useful.

[4] An entry can belong to several objectclasses.

[5] Actually LDAP NSS recognize other objectclasses, but this is outside the purpose of this document.

5 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 shows the relationships among the pieces of the authentication system from the PAM point of view.

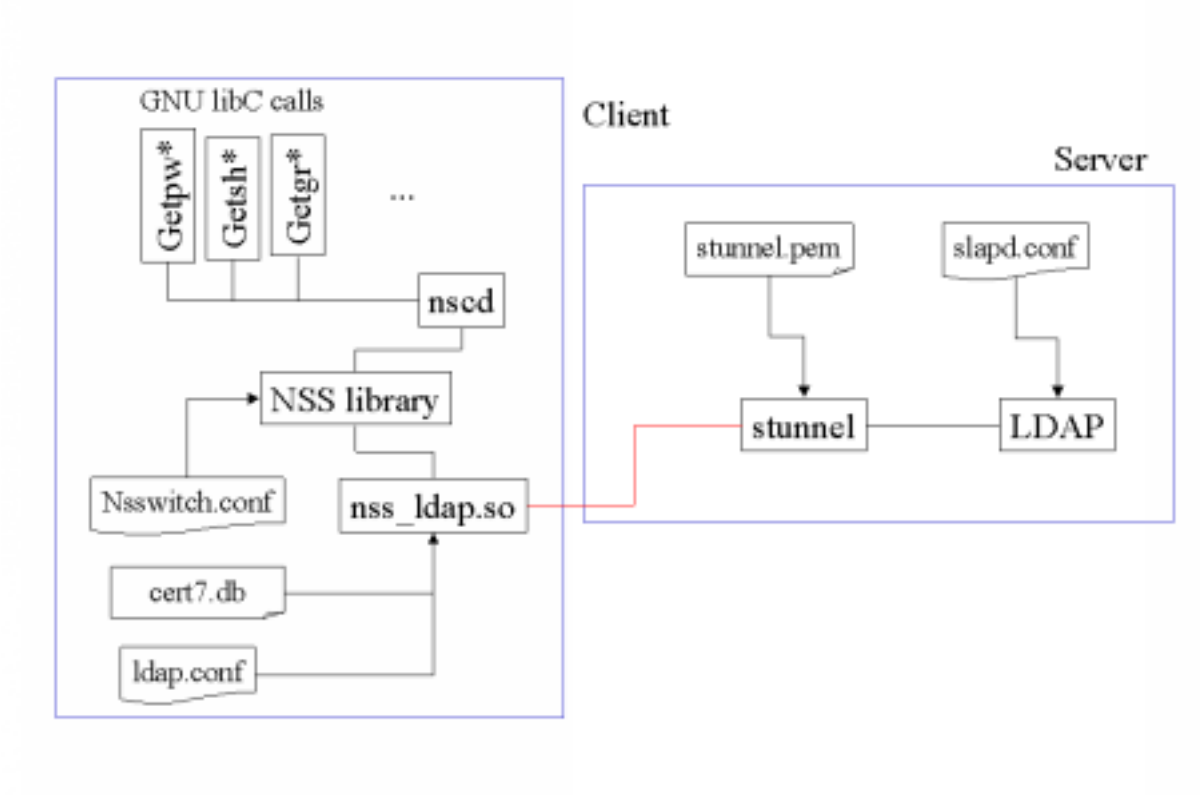

Figure 1: PAM Layout

Figure 2 shows the system from the NSS perspective.

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

5.1 Server side

On the server side an LDAP server must be installed and configured. The LDAP server used is OpenLDAP-1.2.11[6] an open source LDAP (v2) toolkit including an LDAP server (slapd), library and utilities.

[6] http://www.OpenLDAP.org

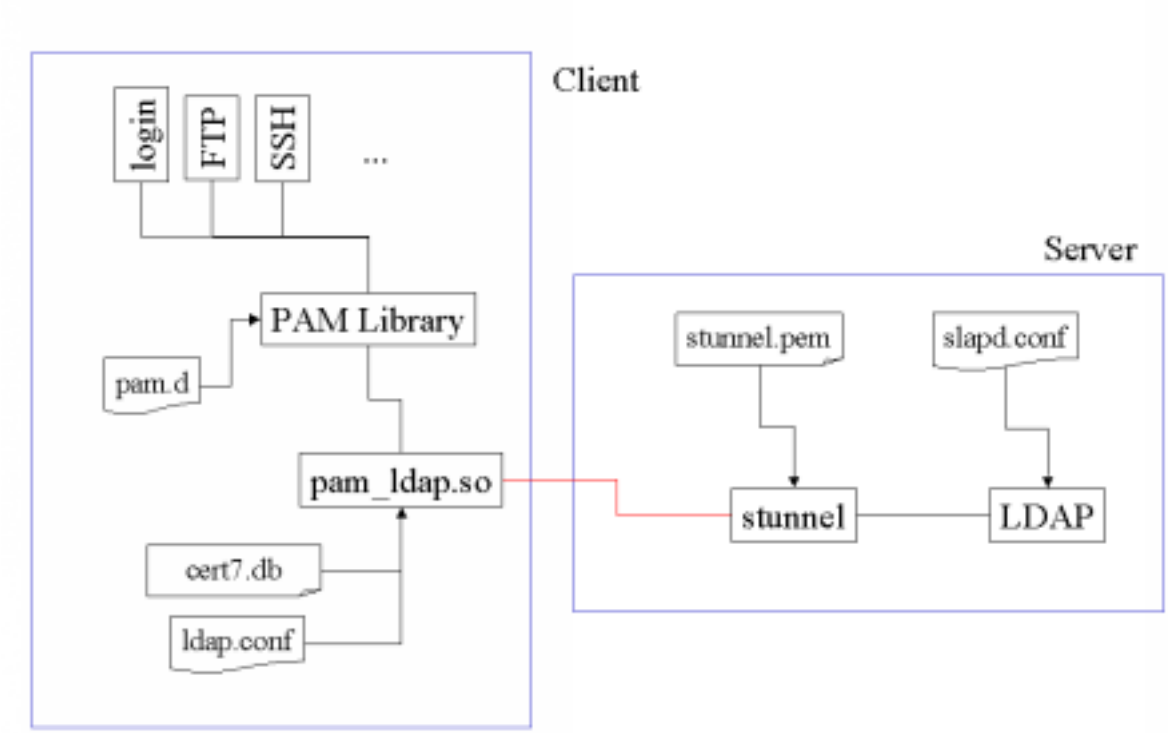

Figure 2: NSS Layout

On the server the OpenSSL library [7] and stunnel [8], an SSL wrapper are also required to provide SSL to the LDAP server.

5.1.1 Installing and configuring OpenLDAP

The installation is quite simple, once downloaded and untarred the sources, you most likely have to type the following commands:

```
$ ./configure
$ make depend
$ make
# make install
```

By default everything will be installed under the directory `/usr/local`.

[7] http://www.OpenSSL.org
[8] http:/www.stunnel.org

The LDAP server, named `slapd` is in `/usr/local/libexec` and the main configuration file of `slapd` is `/usr/local/etc/openldap/slapd.conf`

At this point the `slapd.conf` file has to be customized for our needs.

The first section of the configuration file, doesn't need to be configured. Anyway, one thing that is useful is to add the `lastmod` directive, this enables automatic updating of entries creation and modification times in the entry attributes `createtimestamp` and `modifytimestamp`. It also updates, with the DN of the creator/modifier, the attributes `creatorsname` and `modifiersname`.

```
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
lastmod on
include         /usr/local/etc/openldap/slapd.at.conf
include         /usr/local/etc/openldap/slapd.oc.conf
schemacheck     off
pidfile         /usr/local/var/slapd.pid
argsfile        /usr/local/var/slapd.args
```

Next comes the database configuration, the database type must be `ldbm`, that is an embedded database built with the Berkeley DB. The `suffix` directive defines the root *Distinguished Name* of the LDAP server. This is the base DN of all the entries in the LDAP server.

The `rootdn` directive defines the Distinguished Name of the privileged user of the database. The `rootpw` directive defines the encrypted password for the `rootdn`. A simple way to generate an encrypted password that can be added in the configuration file is by the following Perl command:

```
$ perl -e 'print "{crypt}".crypt('passwd','salt')."\n";'
```

Where `salt` is a two-characters string and `passwd` the password that you have chosen. This generates an Unix crypt password, anyway you should consider to put at least an MD5 password here. The encryption algorithm used for the password is specified in a prefix tag inside curly brackets.

The `directory` directive points to the directory were the database will be actually stored.

Indexes are special index files built to speed up searches on the database for particular attributes.

For our application is crucial a fast search on the: cn, uid, uidnumber, gidnumber, gecos and objectclass attributes.

```
#############################################
# ldbm database definitions
#############################################
database       ldbm
suffix         "ou=Sezione di Milano,
                o=Istituto Nazionale di Fisica Nucleare, c=IT"
rootdn         "cn=Manager,ou=Sezione di Milano,
                o=Istituto Nazionale di Fisica Nucleare, c=IT"
rootpw          {crypt}a_OOAbS2vPWRYq
# rootpw        {MD5}bmvE5J3Ud+vJjvQEbAZ7Xw==
# clear text passwords, especially for the rootdn, should
# be avoided.  See slapd.conf(5) for details.
directory      /usr/tmp
index          default pres,eq
index          cn,uid,uidnumber,gidnumber,gecos,objectclass
```

In the last part of the configuration file are defined access control lists.

```
#
# ACLS
#
defaultaccess read
access to attr objectClass
        by * search

access  to attr=loginshell,userpassword
        by self write
        by dn= "cn=Manager,ou=Sezione di Milano,
          o=Istituto Nazionale di Fisica Nucleare,c=IT" write
        by domain=ldapclient1.mi.infn.it read
        by domain=ldapclient2.mi.infn.it read
        by * none

access  to attr=uidnumber,gidnumber,uid,gecos
```

```
        by dn="cn=Manager,ou=Sezione di Milano,
          o=Istituto Nazionale di Fisica Nucleare,c=IT" write
        by domain=ldapclient1.mi.infn.it read
        by domain=ldapclient2.mi.infn.it read
        by * none

access  to attr = shadowwarning, shadowflag, shadowmin,
        shadowmax, shadowinactive, shadowlastchange,
        shadowexpire
        by dn="cn=Manager,ou=Sezione di Milano,
        o=Istituto Nazionale di Fisica Nucleare,c=IT" write
        by self read
        by domain = ldapclient1.mi.infn.it read
        by domain = ldapclient2.mi.infn.it read
        by * none
```

A special attention must be given to this part of the configuration, since we don't want to make publicly available user data (especially passwd records). [9]

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
```
[10]

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

```
dn:ou=Sezione di Milano,
    o=Istituto Nazionale di Fisica Nucleare,C=it
objectclass: top
objectclass: organizationalUnit
ou: Sezione di Milano

dn:ou=groups, ou=Sezione di Milano,
    o=Istituto Nazionale di Fisica Nucleare,c=it
objectclass: top
```

[9] Note that lines that are too long are continued on the following line started by a tab or a space, this in true too for LDIF format files.

[10] ldif2ldbm is provided with the OpenLDAP package

```
objectclass: organizationalUnit
ou: groups

dn:ou=people, ou=Sezione di Milano,
    o=Istituto Nazionale di Fisica Nucleare,c=it
objectclass: top
objectclass: organizationalUnit
ou: people

dn: cn=Giuseppe LoBiondo, ou=people, ou=Sezione di Milano,
    o=Istituto Nazionale di Fisica Nucleare,C=it
cn: Giuseppe Lo Biondo
sn: Lo Biondo
objectclass: top
objectclass: person
objectclass: posixAccount
objectclass: shadowAccount
uid:giuseppe
userpassword:{crypt}$1$ss2ii(0$gbs*do&@=)eksd
uidnumber:104
gidnumber:100
gecos:Giuseppe Lo Biondo
loginShell:/bin/zsh
homeDirectory: /home/giuseppe
shadowLastChange:10877
shadowMin: 0
shadowMax: 999999
shadowWarning: 7
shadowInactive: -1
shadowExpire: -1
shadowFlag: 0

dn: cn=mygroup, ou=groups, ou=Sezione di Milano,
    o=Istituto Nazionale di Fisica Nucleare,c=it
objectclass: top
objectclass: posixGroup
cn: mygroup
```

```
gidnumber: 100
memberuid: giuseppe
memberuid: anotheruser
```

Here we defined the organization unit “ou=Sezione di Milano,o=Istituto Nazionale di Fisica Nucleare, C=it” 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 users belongs to) under the groups organizational unit. [11].

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

5.1.2 Installing OpenSSL

OpenSSL, is an open source implementation of the SSL protocol that provides the SSL library and a set of cryptography tools. OpenSSL is needed to compile stunnel.

To install OpenSSL you have to type the following commands:

```
$ ./config
$ make
$ make test
# make install
```

everything will be installed in `/usr/local/ssl`.

5.1.3 Installing stunnel

Since OpenLDAP 1.2.x is a V2 implementation of LDAP it doesn’t provide SSL/TSL by itself. This means that an SSL wrapper is needed (if you have an SSL aware LDAP server, you don’t need to use stunnel). If OpenSSL is correctly installed the only command needed to compile and install stunnel are:

```
$ ./configure
$ make
# make install
```

[11] 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

stunnel uses a server certificate for SSL, this can be a self signed certificate, or, better, a certificate signed by your own Certification Authority (the client has to trust the CA too).

By default the server certificate is in contained in the file:

```
/usr/local/ssl/certs/stunnel.pem
```

If having a Certification Authority is not a concern, a self signed certificate can be produced using the tools provided by the OpenSSL suite.

In the stunnel directory type the following commands:

```
$ openssl req -new -x509 -days 365 -nodes -config stunnel.cnf \
            -out stunnel.pem -keyout stunnel.pem
$ openssl gendh 512 >> stunnel.pem
```

This will produce a self signed certificate, valid for a year, in the file `stunnel.pem`.

5.2 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 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 stunnel 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 certificate databases it is convenient to use the `certutil` utility found in the PKCS#11 package provided by Netscape [12].

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

5.2.1 PAM LDAP Installation and Configuration

To compile and install pam_ldap, do the following

[12] In a tricky way, it is also possible to use the Netscape Communicator certificate database.

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

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.

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.

5.2.2 NSS LDAP installation and configuration

Assuming that the ldap sdk is in `/usr/local/ldapsdk` 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
```

Once you have installed it you must edit the NSS configuration file `/etc/nsswitch.conf`. Tough LDAP can be used for all the 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.

5.2.3 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 a value. 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 filedescriptors, so is easy to run out of available filedescriptors on the system (this can hang your system).

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

```
# echo 16384 > /proc/sys/fs/file-max
```

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

5.2.4 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: ldap.conf,v 2.10 2000/01/14 23:29:47 lukeh 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 ou=Sezione di Milano,o=Istituto Nazionale di Fisica Nucleare,C=it
#
# The LDAP version to use (defaults to 2)
# 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
# in user's entry)
#pam_login_attribute userPrincipalName
#pam_template_login_attribute uid
#pam_template_login nobody
#
# Hash password locally; required for University of
# Michigan LDAP server, and works with Netscape
# Directory Server if you're using the UNIX-Crypt
# hash mechanism and not using the NT Synchronization
# service.
pam_crypt local
```

```
#
# SSL Configuration
ssl yes
sslpath /usr/local/ssl/certs
#
```

To avoid problems with the various applications that may read this file it is suggested that you don't 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 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.

6 Starting up

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

```
# /usr/local/libexec/slapd
```

Then stunnel have to be started on the LDAPS port 636:

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

For debugging you can start `stunnel` in foreground with the following syntax:

```
# /usr/local/sbin/stunnel  -r ldap  -d 636 \
-D 7 -f -p /usr/local/ssl/certs/stunnel.pem
```

On the client nscd can be started with the RedHat startup script:

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

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

7 Accounts maintenance

At this point account creation and maintenance should be done using LDAP client tools (see appendix A for LDAP clients).

Unfortunately these general purpose tools are not intended for Un*x accounts maintenance. The one that seems to be enough versatile is the *LDAP Browser/Editor* that allows to set passwords in various formats and can use SSL to connect to the server.

8 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 cames with OpenLDAP (`slapd`) provides replication capabilities.

A 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
```

B Public domain software

The following list gives pointers to open source resources needed to implement the infrastructure described in this document. For each package is reported the version used for the tests.

- Linux
  - Description: An open source Un*x operating system.
  - URL: http://www.kernel.org
  - Version: 2.2.16
- OpenLDAP
  - Description: An open source implementation of LDAP derived from the University of Michigan code.
  - URL: http://www.OpenLDAP.org
  - Version: 1.2.11
- OpenSSL
  - Description: An open source implementation of SSL derived from the SSLeay code.
  - URL: http://www.OpenSSL.org
  - Version: 0.9.5a
- Linux-PAM

  - Description: Pluggable Authentication Modules for Linux
  - URL: http://www.kernel.org/pub/linux/libs/pam/
  - Version: 0.72

- stunnel
  - Description: A general purpose SSL tunnel.
  - URL: http://www.stunnel.org
  - Version: 3.8p4

- nss_ldap and pam_ldap libraries
  - Description: PAM LDAP module and NSS LDAP library implementations, the core of the authentication system.
  - URL: ftp://ftp.padl.com/pub/
  - Version: nss_ldap 113
  - Version: pam_ldap 72

- Netscape LDAPSDK and PKCS#11 tools
  - Description: The Netscape LDAPSDK provides the required LDAPS library needed to compile SSL aware pam and nss libraries. PKCS#11 package instead provides utilities for Netscape certificate databases. This is not Open Source and is under Netscape One license.
  - URL: http://www.iplanet.com/downloads/developer/index.html
  - Version: LDAPSDK for C 4.1
  - Version: PKCS #11 Utility Package 1.0.6

- nscd
  - Description:The Name Service Caching Daemon.
  - URL: http://www.gnu.org/software/libc/
  - Version: 2.1.3

- LDAP Browser/Editor
  - Description: A powerful Java LDAP Browser/Editor.

  - URL: http://www-unix.mcs.anl.gov/~gawor/ldap/
  - Version: 2.8 beta II

- GQ
  - Description: A GTK LDAP Browser/Editor.
  - URL: http://biot.com/gq
  - Version: 0.2.3

C RFC 2307 Objectclasses

The following are objectclasses as defined by the RFC 2307 [2].

```
nisSchema.2.0
NAME ”posixAccount”
SUP top
AUXILIARY
DESC ”Abstraction of an account with POSIX attributes”
MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
MAY ( userPassword $ loginShell $ gecos $ description )
```

```
nisSchema.2.1
NAME ”shadowAccount”
SUP top
AUXILIARY
DESC ”Additional attributes for shadow passwords”
MUST uid
MAY ( userPassword $ shadowLastChange $ shadowMin
shadowMax $ shadowWarning $ shadowInactive $
shadowExpire $ shadowFlag $ description )
```

```
nisSchema.2.2
NAME ”posixGroup”
SUP top
STRUCTURAL
DESC ”Abstraction of a group of accounts”
MUST ( cn $ gidNumber )
MAY ( userPassword $ memberUid $ description )
```

nisSchema.2.3
NAME ”ipService”
SUP top
STRUCTURAL
DESC ”Abstraction an Internet Protocol service.
Maps an IP port and protocol (such as tcp or udp)
to one or more names; the distinguished value of
the cn attribute denotes the service”s canonical
name”
MUST (cn $ ipServicePort $ ipServiceProtocol)
MAY (description)

nisSchema.2.4
NAME ”ipProtocol”
SUP top
STRUCTURAL
DESC ”Abstraction of an IP protocol. Maps a protocol number
to one or more names. The distinguished value of the cn
attribute denotes the protocol”s canonical name”
MUST (cn $ ipProtocolNumber $ description)
MAY (description)

nisSchema.2.5
NAME ”oncRpc”
SUP top
STRUCTURAL
DESC ”Abstraction of an Open Network Computing (ONC)
RFC1057 Remote Procedure Call (RPC) binding.
This class maps an ONC RPC number to a name.
The distinguished value of the cn attribute denotes
the RPC service”s canonical name”
MUST (cn $ oncRpcNumber $ description)
MAY description)

nisSchema.2.6
NAME ”ipHost”
SUP top
AUXILIARY
DESC ”Abstraction of a host, an IP device. The distinguished
value of the cn attribute denotes the host”s canonical
name. Device SHOULD be used as a structural class”
MUST (cn $ ipHostNumber)
MAY (l $ description $ manager)

nisSchema.2.7 NAME ”ipNetwork”
SUP top
STRUCTURAL
DESC ”Abstraction of a network. The distinguished value of
the cn attribute denotes the networks canonical name”
MUST (cn $ ipNetworkNumber)
MAY (ipNetmaskNumber $ l $ description $ manager)

nisSchema.2.8
NAME ”nisNetgroup”
SUP top
STRUCTURAL
DESC ”Abstraction of a netgroup. May refer to other netgroups”
MUST cn
MAY (nisNetgroupTriple $ memberNisNetgroup $ description)

nisSchema.2.09
NAME ”nisMap”
SUP top
STRUCTURAL
DESC ”A generic abstraction of a NIS map”
MUST nisMapName
MAY description

nisSchema.2.10
NAME ”nisObject”
SUP top
STRUCTURAL
DESC ”An entry in a NIS map”
MUST (cn $ nisMapEntry $ nisMapName)
MAY description

nisSchema.2.11
NAME ”ieee802Device”
SUP top
AUXILIARY
DESC ”A device with a MAC address; device SHOULD be
used as a structural class”
MAY macAddress

nisSchema.2.12
NAME ”bootableDevice”
SUP top
AUXILIARY
DESC ”A device with boot parameters; device SHOULD be
used as a structural class”
MAY (bootFile $ bootParameter)

References

[1] W. Yeong - Performance Systems International, T. Howes - University of Michigan, S. Kille - ISODE Consortium, Network Working Group, Request for Comments: 1777, *Lightweight Directory Access Protocol*, March 1995.

[2] L. Howard - Network Working Group, Request for Comments: 2307, *An Approach for Using LDAP as a Network Information Service*, March 1998.

[3] University of Michigan, *The SLAPD and SLURPD Administrator's Guide*, 30 April 1996.

[4] Vipin Samara, Charlie Lai - SunSoft Inc, *Making Login Services Independent of Authentication Technologies*, March 1996

[5] Andrew G. Morgan, *The Linux-PAM System Administrators' Guide*, 11 August 1999.

[6] Heinz Johner, Larry Brown, Franz Stefan Hinner, Wolfgang Reis, Johan Westman, *Understanding LDAP - IBM Redbook*, June 1998.

Contents

1 Copyright Notice **2**

2 Conventions used in this document **2**

3 Introduction **2**

4 The components of the framework **3**
- 4.1 Authentication: PAM and pam_ldap.so . . . 3
- 4.2 The Name Service Switch and nss_ldap.so . . . 4
- 4.3 The Lightweight Directory Access Protocol . . . 5
- 4.4 The Name Service Caching Daemon . . . 6
- 4.5 The Secure Socket Layer . . . 6

5 Building the authentication system **7**
- 5.1 Server side . . . 7
  - 5.1.1 Installing and configuring OpenLDAP . . . 8
  - 5.1.2 Installing OpenSSL . . . 13
  - 5.1.3 Installing stunnel . . . 13
- 5.2 Client side . . . 14
  - 5.2.1 PAM LDAP Installation and Configuration . . . 14
  - 5.2.2 NSS LDAP installation and configuration . . . 16
  - 5.2.3 NSCD configuration . . . 16
  - 5.2.4 LDAP client configuration file . . . 17

6 Starting up **20**

7 Accounts maintenance **21**

8 Known limits **21**

A File permissions **22**

B Public domain software **22**

C RFC 2307 Objectclasses **24**

List of Figures

1 PAM Layout 7
2 NSS Layout 8