Expanding our capabilities: Authentication, Authorization, and LDAP

Up to now, we have talked about using flat files to do

- ☞ Simple authentication
- ☞ Simple authorization

Expanding our capabilities: Authentication, Authorization, and LDAP

We will now look at some ways to go beyond the world of flat files on single machines to provide homogenous authentication and authorization environments for many machines. We will start with LDAP.

Simple authentication

We talked earlier about simple authentication.

In the Linux world, this simple authentication data is stored in `/etc/passwd` and `/etc/shadow`.

In OpenSolaris, this data is also stored in `/etc/passwd` and `/etc/shadow`.

Simple authentication

In FreeBSD, this data is stored in `/etc/passwd` and `/etc/master.passwd`.

In the Windows world, passwords are stored in the registry hive HKLM "SAM" (Security Accounts Manager) database.

YP/NIS: First steps at extending Authentication to cover more area

We have already briefly touched on YP/NIS authentication, which was a simple extension to the traditional flat file /etc/passwd schema.

YP/NIS: First steps at extending Authentication to cover more area

Because of its similarity to flat file access, YP/NIS was an easy fit into authentication schemes. Plugging NIS into the existing `getpwent(3)` scheme was reasonably simple; doing a `ypcat passwd` was both in concept and in implementation very similar to `cat passwd`.

YP/NIS: First steps at extending Authentication to cover more area

However, it wasn't very secure – in its original form it certainly didn't even solve problems such as obscuring the encrypted password.

YP/NIS: First steps at extending Authentication to cover more area

NIS+ did solve some of these problems, but it is complex, and Sun has deprecated NIS+ in favor of LDAP. There are automated tools to aid in this transition. So let's look at LDAP.

LDAP: Extending Authentication

LDAP (lightweight directory access protocol) can be likened to an object-oriented database rather than a relational one. Unlike a database built on a pure relational model, it can support records that have multiple instances of the same field (as, oddly enough, can some "relational" databases that do not strictly follow the original relational model.)

LDAP: Extending Authentication

More interestingly for system and network administration, LDAP can be used for authentication in the Unix/Linux world and in the Microsoft world.

We will talk about how LDAP hooks into authentication via methodologies such as PAM, but first let's look at ldap itself.

posixAccount

RFC2307 proposed a standard for moving from the NIS world to the LDAP world.

posixAccount

For system administrators, the most important part of the standard was probably the objectClass of `posixAccount` (found by default in a schema file called `nis.schema`); that's the bit with a schema that has the attributes that you want.

posixAccount

```
objectclass ( 1.3.6.1.1.1.2.0 NAME 'posixAccount'
        DESC 'Abstraction of an account with POSIX attributes'
        SUP top AUXILIARY
        MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
        MAY ( userPassword $ loginShell $ gecos $ description ) )
```

Example: Bob Betterman account

LDAP records for such POSIX user accounts look like:

```
# bob, my-domain.com
dn: uid=bob,dc=my-domain,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
sn: Betterman
cn: Bob Betterman
uid: bob
uidNumber: 1000
gidNumber: 1000
homeDirectory: /home/bob
userPassword: {crypt}X5/DBrWPOQQaI
loginShell: /bin/bash
```

Example: Ted Williams' acount

Or like:

```
# ted, my-domain.com
dn: uid=ted,dc=my-domain,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
sn: Williams
cn: Ted Williams
uid: ted
uidNumber: 1001
gidNumber: 1001
homeDirectory: /home/ted
userPassword: {crypt}X5/DBrWPOQQaI
loginShell: /bin/bash
```

Adding entries

While various GUI tools exist, you can also use simple command lines to modify the ldap database; for instance, slapadd is useful for initializing a database (as, for that, is slapcat if you have an existing database.)

Prepwork

To do this, we have to install and configure both openldap-servers and openldap-clients. Once we have done that, we can use the configuration file `13-slapd.conf` to help us set up the ldap server (the actual process is called slapd.)

Installing packages

```
[root@localhost ~]# yum install openldap-servers openldap-clients
Loaded plugins: refresh-packagekit
Setting up Install Process
Parsing package install arguments
Resolving Dependencies
--> Running transaction check
---> Package openldap-clients.x86_64 0:2.4.12-1.fc10 set to be updated
---> Package openldap-servers.x86_64 0:2.4.12-1.fc10 set to be updated
--> Finished Dependency Resolution
```

Installing packages

```
Dependencies Resolved

================================================================================
 Package                 Arch          Version               Repository     Size
================================================================================
Installing:
 openldap-clients        x86_64        2.4.12-1.fc10         fedora        291 k
 openldap-servers        x86_64        2.4.12-1.fc10         fedora        2.5 M
```

Installing packages

```
Transaction Summary
================================================================================
Install       2 Package(s)
Update        0 Package(s)
Remove        0 Package(s)

Total download size: 2.8 M
Is this ok [y/N]: y
Downloading Packages:
(1/2): o
(2/2): o
```

Installing packages

```
Running rpm_check_debug
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing     : openldap-servers                                1/2
  Installing     : openldap-clients                                2/2

Installed:
  openldap-clients.x86_64 0:2.4.12-1.fc10
  openldap-servers.x86_64 0:2.4.12-1.fc10

Complete!
```

ldap manipulation commands...

First, let's look at the script `13-slapadd-initialization` to initialize the database.

Second, we'll look at the script `13-ldapadd.sh` to add new users.

ldap manipulation commands...

Now let's run them

```
# bash -x 13-slapadd-initialization.sh
[ lots of output ]
# bash -x 13-ldapadd.sh
[ even more !]
```

ldap manipulation commands...

By looking at the results of `ldapsearch` at the end of the ldapadd script, we can see what's in the database. (`ldapsearch` dumps ldif (RFC 2849) output without having to stop the database; `slapcat` shouldn't really be run on live databases.)

PAM and LDAP

As neat as LDAP is (and it can be used for much more than providing user account information — many people still use LDAP to provide data to programs such as sendmail rather than using newer concepts such as socketmaps), programs such as /bin/passwd and /bin/login don't speak LDAP natively.

PAM and LDAP

To provide an abstraction layer that would permit authentication and authorization information to have many different types of implementations, the PAM (pluggable authentication modules) framework was developed. While it shows its age in some ways by a somewhat obtuse syntax, it's still quite useful.

PAM example 1

A generic PAM `/etc/pam.d/su` file on Fedora 10 looks like this:

```
#%PAM-1.0
auth               sufficient        pam_rootok.so
# Uncomment the following line to implicitly trust users in the "wheel" group.
#auth               sufficient        pam_wheel.so trust use_uid
# Uncomment the following line to require a user to be in the "wheel" group.
#auth               required        pam_wheel.so use_uid
auth               include               system-auth
account               sufficient        pam_succeed_if.so uid = 0 use_uid quiet
account               include               system-auth
password        include               system-auth
```

```
session                 include                 system-auth
session                 optional        pam_xauth.so
```

PAM example 2

We can add a line that makes it possible for any user to su to root without typing a password:

```
#%PAM-1.0
auth                sufficient        pam_rootok.so
# Uncomment the following line to implicitly trust users in the "wheel" group.
#auth                sufficient        pam_wheel.so trust use_uid
# Uncomment the following line to require a user to be in the "wheel" group.
#auth                required        pam_wheel.so use_uid
# Dangerous!
auth                sufficient        pam_permit.so
auth                include                system-auth
account                sufficient        pam_succeed_if.so uid = 0 use_uid quiet
```

```
account                 include                 system-auth
password        include                 system-auth
session                 include                 system-auth
session                 optional        pam_xauth.so
```

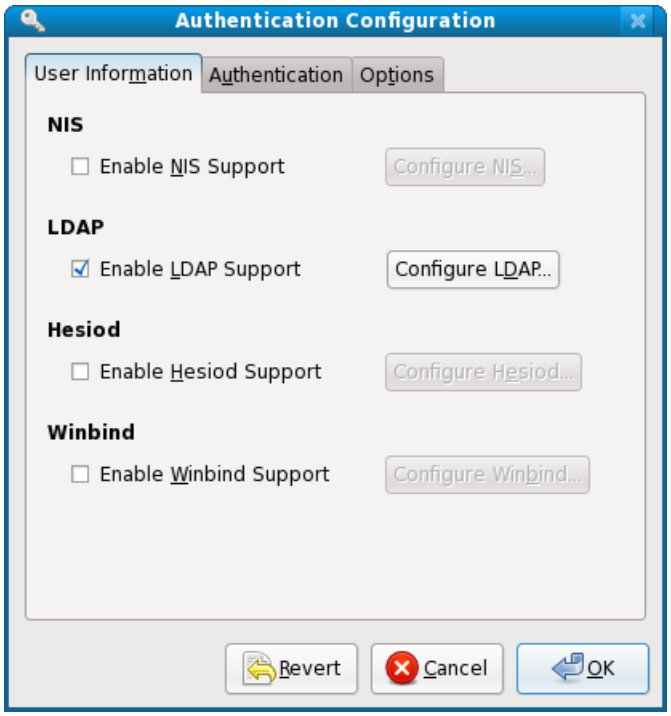
A User Authentication GUI

Fortunately, RedHat has included a nice GUI (it's really a front-end for an older shell script) that lets us change the PAM and NS settings in one go: /usr/bin/system-config-authentication

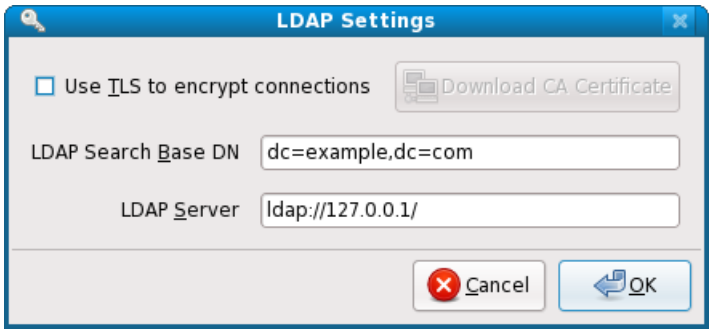
On your Fedora 10 machine, you can find it in the menus via

System → Administration → Authentication

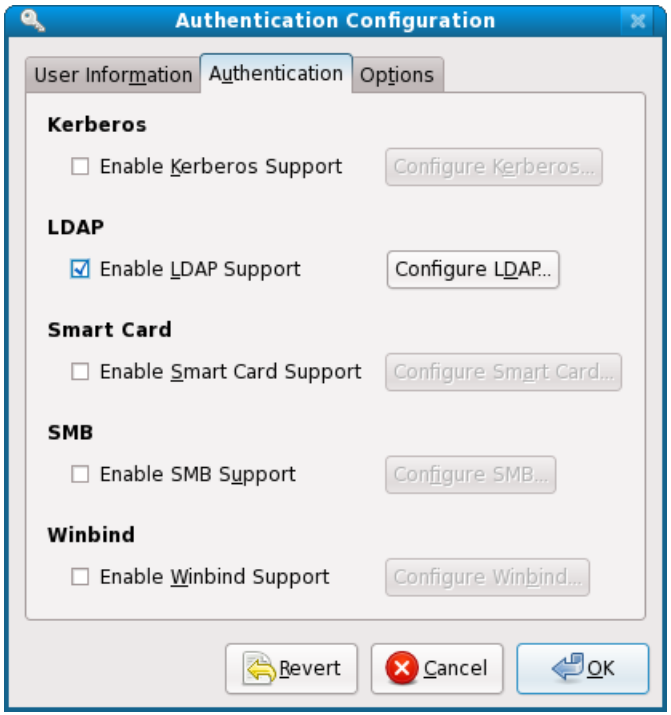
system-config-authentication



system-config-authentication



system-config-authentication



Tying it all together

Once you enable LDAP with the GUI, everything is in place to use your LDAP accounts. Since we are working in a test environment, you probably should enable the "Create Home Directories" under the Options tab in the GUI.