



Using LDAP to Manage Unix Accounts

Jeff Machols

User management is one of the most tedious tasks in a systems administrator's job. There have been some attempts to centralize user management with NIS and NIS+. NIS fizzled out because of its security holes, and NIS+ is not very straightforward to configure. So, what's the best way to centralize user management in an environment? The answer is looking more and more like LDAP.



LDAP (Lightweight Directory Access Protocol) is quickly emerging as the standard in hierarchical data, such as user and group data. LDAP servers are designed for an "update seldom, access often" scenario. One of the roadblocks LDAP has faced in gaining popularity as a centralized user management system is the effort to get client machines to securely authenticate users. In the past, this required writing custom PAM modules or trying to configure existing ones. However, as major Unix vendors are realizing the potential of LDAP, they are including clients in the operating system.

These built-in clients also contain the PAM libraries for authentication with an LDAP server. These client-side applications are included in the Solaris 8 and 9 distributions, as well as AIX 5L. HP-UX has a free software depot called ldapux that can be found at software.hp.com, and Linux has an RPM called nss_ldap. These clients include built-in libraries, so fears about writing C programs to authenticate or having holes in your security can be put to rest.

Server Considerations

Several LDAP servers are available on the market. Currently, the two predominant servers are OpenLDAP and Sun One Directory Server (formerly iPlanet). If Solaris is your OS for the LDAP server, Sun One Directory Server is the way to go. It is currently bundled with Solaris 9, and version 5.1 is free up to 200,000 entries (each distinguished name in the server is considered an entry). Sun One Directory Server is also available on HP-UX, AIX, and Linux. OpenLDAP is free and can be compiled on most flavors of Unix, but configuration and compilation take a little more effort.

The LDAP community has created an RFC (RFC 2307) to define a schema for Unix to use LDAP as an NIS provider. The schema defines all the previous maps that were available for NIS, so it is aptly named the `nis.schema`. This schema comes loaded in the standard installation of Sun One Directory Server. If you are using OpenLDAP or another server, be sure the schema is loaded into your server. As part of the `nis.schema`, the following services can be centralized with LDAP: password, shadow password, groups, and netgroups. Other services, such as DNS, are also available with LDAP, but that is beyond the scope of this article.

The default communications for the LDAP server and clients are clear ASN1 strings. All information is sent in clear text. This is the major problem with NIS, so be sure you don't make this mistake with your LDAP implementation. I recommend using the default communication method during the installation and initial test. Once you have tested the server and the clients, and are comfortable with the configuration, switch to SSL.

When configuring your clients, you will need to specify a search base. This is the point in the directory at which the client starts looking for NIS entries. This functionality allows you to separate Unix user and group accounts from other parts of the directory. For example, you may want to set up a group with a distinguished name (DN) of `"ou=unix nis, dc=mydomain, dc=com"`. You can segregate this more if desired, but all Unix accounts would be under this organizational unit; this DN would be your clients' search base.

Segregating your NIS environment inside the LDAP server will give you two advantages. The first advantage is speed. When you specify the start DN, your clients will not have to search through the entire directory to find the accounts, which will help performance. The second benefit is administrative flexibility. You can give account administrators access to a specific area of the server, instead of to your entire directory. You can have different search bases for different Unix clients. If you do this, note that user accounts will need to be replicated in both search bases to have access to the different clients.

Adding NIS Entries in the LDAP Server

To add a Unix group into LDAP, you will need to create an entry of object class `posixGroup`. The attribute `gidNumber` corresponds to the Unix GID number for the group. Since the local and LDAP groups are treated the same, it is important to keep all the `gidNumbers` in LDAP as well as the local GIDs unique. The Full Name, or `cn` attribute, is the name of the group. The `memberUid` attribute correlates to the users who are part of that group. Each user will be an additional attribute in the group's entry. In the example below, users `jdjoe` and `scarter` are part of the `admins` group:

```
dn: cn=admins, ou=unix nis, dc=mydomain, dc=com
gidNumber: 900
memberUid: jdjoe
memberUid: scarter
objectClass: top
```

```
objectClass: posixgroup
cn: admins
```

Here are the instructions for adding entries. If you are unsure how to add an entry into an LDAP server, copy the above entry information into a file on the LDAP host and run the following command:

```
$ ldapadd -D "directory_manager_dn" -w "directory_manager_password" -f filename
```

You can assign users to groups in two ways; the first way is described above. Each posixGroup entry will have a list of users. The second method is to assign gidNumbers for each group in the posixAccount entry (described below). These methods can be intermixed but, for consistency, I recommend choosing one method. When you create a new user, you will use the object class posixAccount and shadowAccount along with the standard person object classes. The uid attribute is the user login name, and the uidNumber is the user's Unix uid. The gidNumber attribute lists the groups to which the user belongs. When you specify the password, there are multiple ways to store the ciphered password. For authentication on Unix, you must use the crypt method. This is accomplished by using the **{CRYPT}** tag in front of the cipher password:

```
dn: uid=jdoe, ou=unix nis, dc=mydomain, dc=com
givenName: John
sn: Doe
userPassword: {CRYPT}QGxOG7iX3lbLU
loginShell: /usr/bin/ksh
uidNumber: 343
gidNumber: 900
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
objectClass: posixAccount
objectClass: shadowaccount
uid: jdoe
cn: John Doe
homeDirectory: /export/home/jdoe
```

Along with specifying the password, you can also set password options such as length, minimum characters, expiration, etc. All of the attributes in the NIS schema can be found in RFC2307. With replication and the correct setup, your LDAP environment will be reliable, but there are still some users you should keep out of LDAP. The most obvious is root; this user should always be kept local. I suggest keeping application users local, so even if the network goes down, your applications will still be able to run (assuming they don't need the network).

LDAP supports the use of netgroups, so you can control user access to individual servers. The object class is called NisNetGroup and uses the same "triple" notation as NIS. Each entry has three fields: host, user, and domain. If you leave a field blank, it allows complete access. In the entry below, jdoe is in the appuser netgroup for all servers, all domains. The user scarter is in the appuser netgroup only on the server mars, and all users are in the appuser netgroup on the server pluto:

```
dn: cn=appuser, ou=unix nis, dc=mydomain, dc=com
    nisNetgroupTriple: ( , jdoe , )
    nisNetgroupTriple: ( mars , scarter , )
    nisNetgroupTriple: ( pluto , , )
cn: appuser
objectClass: top
objectClass: nisnetgroup
```

The **passwd** command can be used to change the password in the LDAP server. The only change is an extra switch -- the **-r** option. To change the password for user jdoe:

```
$ passwd -r ldap jdoe
```

Configuring the Unix Client

Unfortunately, the client setup is different for each version of Unix. There is an effort on the Apache Directory project to document the steps for configuring each client. As each configuration is tested, the documentation will be posted on the Directory Project site. I will use Solaris 9 as an example for the rest of the article.

The Solaris 9 clients will run a daemon called `/usr/lib/ldap/ldap_cachemgr`, which will handle all communication between the client and the LDAP servers. The clients use a configuration profile stored on the servers and periodically update themselves against the profile. This allows you to make a configuration change once that will be populated out to clients automatically. To do this, you need to create an organizational unit for the profile to reside in. Add the following OU:

```
dn: ou=profile, dc=mydomain, dc=com
ObjectClass: top
ObjectClass: OrganizationalUnit
ou: profile
```

Next, create the profile. This is accomplished by using the **ldapclient** command built into the OS. This command will create LDIF output that will be added as an entry into the server. The following example will create a profile that will configure the clients to use multiple servers (for replication and failover) and map where in the directory the NIS information will be stored. This command can be run on any Solaris 9 machine regardless of whether it is a server or a client:

```
$ /usr/sbin/ldapclient genprofile \
-a defaultSearchBase="ou=unix nis,dc=mydomain,dc=com" \
-a serviceSearchDescriptor="passwd: ou=unix nis,dc=mydomain,dc=com" \
-a serviceSearchDescriptor="group: ou=unix nis,dc=mydomain,dc=com" \
-a serviceSearchDescriptor="shadow: ou=unix nis,dc=mydomain,dc=com" \
-a serviceSearchDescriptor="netgroup: ou=unix nis,dc=mydomain,dc=com" \
-a authenticationMethod=simple \
-a credentialLevel=proxy \
-a "defaultServerList=192.168.0.155 192.168.0.156 192.168.10.100" >
profile.ldif
```

The profile.ldif file will contain the entry information. Normally, you should not have to do anything to this file, but you can make changes before adding the entry, if necessary. When you are happy with the profile, add it to the server:

```
$ ldapadd -D "directory_manager_dn" -w "directory_manager_password" \
-f profile.ldif
```

Once the profile entry has been added, set up each of the clients to use it. On the client machine, run the **ldapclient** command with the **init** option. This command will configure the `ldap_cachemgr`; it will also copy the `/etc/nsswitch.ldap` to `/etc/nsswitch.conf` and start the client in the background:

```
$ ldapclient -v init -a proxyDN=" directory_manager_dn" \
-w "directory_manager_password" ldapserver_IP_address
```

If the client does not start or you encounter other problems, you can add debug flags to get more information. Just add option **-d 6** to the command for verbose output:

```
$ /usr/lib/ldap/ldap_cachemgr -d 6
```

After `ldap_cachemgr` is up and running, you can test the connection to the server. It will be easier to test if you have at least one entry for each NIS component you are using. To get a list of entries the client finds, run the **ldaplist** command. You should see all the entries in your search base:

```
$ ldaplist
```

```
dn: cn=admins, ou=unix nis, dc=mydomain,dc=com
dn: uid=jdoe,ou=unix nis,dc=mydomain,dc=com
dn: cn=appuser, ou=unix nis, dc=mydomain,dc=com
```

Once you get the client talking to the LDAP server, you can begin configuring the OS for user authentication. The first step is to add LDAP as a service in the `/etc/nsswitch.conf` file. The following `nsswitch.conf` file will support user authentication, groups, and netgroups in LDAP. If you are not using netgroups, replace the **passwd: compat** entry with **passwd: files ldap** and delete the **passwd_compat** entry:

```
#
# /etc/nsswitch.conf
#
passwd: compat
passwd_compat: ldap
group:      files ldap
#
# All other services are unchanged
#
netgroup:   ldap
```

If you are not using netgroups, you don't need to change the `/etc/passwd` file. If you are using netgroups, you will need to add the name of the netgroups with access. You will also need to deny other net users. The following snippet can be inserted at the end of the `/etc/passwd` file to allow users in the netgroup `appusers` to log onto the server:

```
+@appusers:x:::::
-:x:::::
```

After the entry is added, run the **pwconv** command to update the `/etc/shadow` file.

You will need to add the LDAP PAM library to the `/etc/pam.conf` in order to authenticate. The library should already exist in `/usr/lib/security`; it will be called `pam_ldap.so.1`:

```
#
# Authentication management
#
# login service (explicit because of pam_dial_auth)
#
login    auth required          pam_authtok_get.so.1
login    auth required          pam_dhkeys.so.1
login    auth required          pam_dial_auth.so.1
login    auth sufficient        pam_unix_auth.so.1
login    auth required          pam_ldap.so.1
#
other    auth required          pam_authtok_get.so.1
```

```

other    auth required          pam_dhkeys.so.1
other    auth sufficient       pam_unix_auth.so.1
other    auth required        pam_ldap.so.1
#
passwd  auth sufficient       pam_passwd_auth.so.1
passwd  auth required        pam_ldap.so.1
other   account requisite   pam_roles.so.1
other   account required    pam_projects.so.1
other   account required    pam_unix_account.so.1
#
# Default definition for Session management
# Used when service name is not explicitly mentioned for session
#
other   session required    pam_unix_session.so.1
#
other   password required   pam_dhkeys.so.1
other   password required   pam_authtok_get.so.1
other   password required   pam_authtok_check.so.1
other   password sufficient  pam_authtok_store.so.1
other   password required   pam_ldap.so.1

```

Conclusion

Besides GUIs supplied by the LDAP server, command-line clients also exist. These LDAP clients allow you to add, delete, and modify LDAP entries via the command line and LDIF files. This is useful for scripting or creating custom application to access LDAP.

With the built-in clients, secure connections, and the ability to build GUIs around the server, LDAP has become a viable solution for user management and authentication. Not only does it make it easier to administer users, but LDAP also allows much of the work to be moved to a help desk or level 2 systems administrator support.

Jeff Machols is the Manager of the Unix Administration team for a financial institution and the co-founder of the Apache Directory Project. He can be contacted at: jmachols@apache.org.

Copyright © 2001 Sys Admin, [Sys Admin's Privacy Policy](#). Comments about the Web site:
webmaster@sysadminmag.com