

Using an LDAP Directory for Samba Authentication

Presented by developerWorks, your source for great tutorials

ibm.com/developerWorks

Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

1. Tutorial overview	2
2. Gathering and building the required software	5
3. Configuring OpenLDAP	9
4. Configuring Samba	16
5. User management	21
6. Summary and resources	24

Section 1. Tutorial overview

The road ahead

This tutorial details the steps required to use an LDAP (Lightweight Directory Access Protocol) directory for storing Samba user account information typically stored in the `smbpasswd` file. The procedures outlined here are based on the current stable releases of [Samba](#) and [OpenLDAP](#), which at the time of this writing are 2.2.4 and 2.0.23 respectively. If you're building a system up from scratch, it is advisable to always use the most recent versions of both Samba and OpenLDAP unless you have a specific need or reason not to. This is especially true for Samba as a great deal of development has been taking place recently on the program's LDAP and Windows domain integration. The current CVS HEAD branch of Samba (the bleeding edge development tree, which will eventually be Samba 3.0) will make integration with LDAP even easier, as administrators will not have to worry about synchronizing two users accounts (the UNIX `smbpasswd` account and the Windows domain account) as they do now; Samba 3.0 will do away with the current need for an existing `smbpasswd` account.

Upon completion of this tutorial you should have have a clear understanding of:

- How to collect, build and/or install the required software.
- How to configure the OpenLDAP server for use.
- How to configure Samba, and add users, groups, and machine accounts to the OpenLDAP server.
- And how to maintain (delete and modify) account information.

This tutorial *does not* detail the mechanics of configuring Samba to function as a PDC (Primary Domain Controller). This information is available in another tutorial on IBM eServer Developer Domain titled [Using Samba as a PDC](#).

Assumptions

This tutorial assumes the following:

- A functional, fully configured Linux/UNIX system. This tutorial is based on [Red Hat's](#) 7.3 release. It has also been tested on Red Hat 7.2 and [Gentoo Linux](#) (a "metadistribution" for advanced users). In theory, the instructions given should work on any *NIX operating system; pathnames and conventions will vary across distributions.
- A functional, configured Samba PDC installation. In addition to the IBM eServer Developer Domain tutorial mentioned on the preceeding panel, F. Hagethorn has created an excellent set of scripts to help you quickly create a PDC using Samba. He calls his product VASC (Very Advanced Samba Configuration) and you can find it [here](#).

- A basic understanding of the concepts and terms used by LDAP (for example, distinguished name, organizational units, objectclass, etc). Even a brief introduction to the basic concepts behind correctly implementing an LDAP server could easily consume 50 tightly written pages. Adam Tauno Williams has prepared a good introduction to LDAP structures and conventions; it's available for download in PDF format. See the [Summary and resources](#) on page 24 section for details.
 - A basic understanding of common administrative terms and procedures such as file permissions, creating user accounts, moving and copying files, etc.
 - And while it may seem obvious to some, you need enough computer to handle the load of file sharing and directory lookups (assuming both Samba and OpenLDAP are run on the same machine -- which is a relatively common configuration for a small to medium-sized business). This tutorial was written and tested on an IBM eServer [xSeries model 220](#) with a three disk ServeRAID array, dual P3 1.2 GHz processors, and 1GB of RAM. As configured this machine could easily serve 250-350 users, depending on the size of the files being served and the number of concurrent users.
-

Why an LDAP directory?

In a traditional Samba installation, user account and password information is stored in `/usr/local/samba/private/smbpasswd` (or, depending on configuration, sometimes in `/etc/smbpasswd`). Due to incompatibilities between the LM/NT password hashes and the `smbpasswd` hashes, a user must first exist in the server's `/etc/passwd` file, then be manually added to the `smbpasswd` file. This fine for a small number of users, but when the user-base grows large, the disadvantages mount. For example:

- All authentication lookups must be performed sequentially. Given that there are approximately two lookups per domain logon (one for a normal session connection and another when mapping a network drive or printer), this is a performance bottleneck for large sites. What is needed is an indexed approach as is used in databases.
- Administrators who want to replicate a `smbpasswd` file to more than one Samba server are forced to use external tools such as `rsync` and `ssh`, or write custom, in-house scripts.
- The structure of the `smbpasswd` file lacks fields to store attributes such as home directory, password expiration time/date plus other password details (for example, can/can't change), and user/group RIDs (Relative Identifiers).

Using an LDAP directory service alleviates these and other limitations, plus allows for a central source for all user administration.

There are two things a Samba/LDAP installation *cannot* do "out of the box":

- Retrieve user account information from an Windows 2000 Active Directory server.
- Alleviate the need for `/etc/passwd`.

Both issues will be resolved with the release of Samba 3.0 tentatively scheduled to

appear late summer 2002.

About the author

Over the last several years, Tom Syroid has brought his skills and hands-on experience to the work of technical writing; he is the author of *Outlook 2000 in a Nutshell* (O'Reilly) and *OpenLinux Secrets* (Hungry Minds). He has also written numerous short articles on Linux/UNIX system administration and security issues. Tom is proficient in the nuances of Samba, Apache, Microsoft Office (97, 2000, and XP), SSH, DNS/BIND, and various other Open Source products/technologies. On the operating system front, Tom has configured and administered systems running all variants of Windows (95, 98, ME, NT, 2000, XP), most Linux distributions (Redhat, OpenLinux, Mandrake, Slackware, TurboLinux, SuSE, and Yellow Dog Linux), and AIX (4.3.x, and 5L). Tom's hardware capabilities are equally eclectic; he has experience in spec'ing, building, and maintaining all shapes and sizes of PCs, as well as considerable background with IBM's xSeries product line and the RS/6000 product line.

Comments and feedback are welcome. You can reach Tom at tom@syroidmanor.com

Section 2. Gathering and building the required software

The keyword is choice...

If there's one thing that truly distinguishes the Open Source software movement, it's the concept of choice. There is choice in distributions, which in turn brings choices in bundled software and a variety of filesystem layouts. There are choices in software package managers and in software distribution -- will that be source code, or a precompiled binary? And while all this freedom of choice is grand and glorious in its own right, it also makes it very hard to provide detailed instructions on how to build something as complicated as an LDAP server that functions as a Samba PDC for 300 workstations, half of which use roaming profiles.

To simplify and focus on the task at hand, as previously noted this tutorial is based on Red Hat's 7.3 release. Why Red Hat? Because it's readily available, Red Hat's filesystem layout and *init* script conventions are relatively well-known to Linux administrators, and it just happened to be the box on my shelf of distributions. The bottom line is -- whether you use Red Hat or SuSE or Gentoo or Slackware, the base process is similar. To keep things on track regarding the topic of this tutorial, however, we had to start somewhere, and that somewhere is a clean Red Hat installation, some RPM packages, and a handful of Perl scripts.

Required packages

If you intend on following along with the instructions presented in this tutorial, you'll need to download or have pre-installed the following packages:

- The latest stable Samba release (www.samba.org). As of this writing, this is 2.2.4.
- The latest stable OpenLDAP release (www.openldap.org), currently 2.0.23.
- There are literally hundreds of scripts floating around the Internet written by admins and programmers to alleviate the manual process of populating an LDAP directory with user account information. *IDEALX* has a package of Perl scripts that allow an administrator to create the required Samba objectclasses, and manage the directory. The folks at *PADL* have published a collection of scripts that allow the administrator to migrate a variety of account information from common UNIX configuration files such as `/etc/passwd`, `/etc/group`, NIS details, etc. A warning is in order regarding Samba-to-LDAP migration scripts: a set of scripts that work for one installation may *not* work for another. Open Source applications have just enough variance across revisions, Linux distributions are notorious for adopting their own unique filesystem layout, and every administrator likes to do things his or her own way. All this adds up to the aforementioned irregularities. This tutorial uses the scripts from *IDEALX* as they seem to be relatively refined. If you don't get the results detailed here, try another set. Whatever you do, ensure you test your choice of scripts on a non-production system.

- Begin by fetching the latest samba-2.2.X-X.src.rpm from Red Hat (ftp.redhat.com or a local mirror).
- Install the RPM: `rpm -ivh samba-2.2.X-X.src.rpm`.
- Edit the `/usr/src/redhat/SPECS/samba.spec` file and add your desired options. Using the example from the previous panel: `--with-smbmount --with-pam --with-ldapsam`.
- Now build the RPMS: `cd /usr/src/redhat/SPECS/ && rpm -ba samba.spec`. This will create three RPM files, `samba-common-X.X.X-X`, `samba-client-X.X.X-X`, and `samba-X.X.X-X`.

One caveat when dealing with RPMs: If you're installing an RPM from a source other than Red Hat (for example, the RPMs available from the Samba site), *always* uninstall any existing Samba packages. All too frequently I've seen people end up with two distinct copies of Samba on their system because the third-party RPM they installed put the binaries in a different location than Red Hat would have (i.e., the new installation didn't overwrite the previous one). This caveat also applies to those installing from source -- *uninstall any existing RPMs first!*

To uninstall the Samba RPMs from a Red Hat system:

```
[root@pdc root]# rpm -qa | grep samba
samba-common-2.2.3a-3
samba-2.2.3a-3
samba-client-2.2.3a-3
[root@pdc root]# rpm -e samba
```

Installing OpenLDAP

Installing OpenLDAP is a snap; if you did a standard Server installation, you're already two-thirds of the way there.

Here's a "Go Figure" for you... performing a default Server setup under Red Hat 7.2 and 7.3 installs the `openldap-version.rpm`, and the `openldap-clients-version.rpm`, but neglects to install the `openldap-servers-version.rpm`. So the first thing you should do is check to ensure the folks at Red Hat have not changed their installation policy since this was written:

```
[root@pdc root]# rpm -qa | grep samba
```

Scan the output produced. You should have three RPMs installed: a common (just `openldap`), a client, and a server component. Chances are you are going to have to fish the server RPM off the installation CDs and install it. Without it you'll be missing some key configuration files, a directory of schemas, and the actual `ldap` daemon.

Installing the IDEALX Perl scripts

The final pieces of the installation puzzle are a set of scripts maintained by [IDEALX](#). While not absolutely necessary (you could always manually insert records using the OpenLDAP provided utilities), they do make life easier when it comes time to configure and populate the server. There are actually many, many scripts and programs out there that interface with LDAP and Samba, so if the IDEALX contribution is not to your liking, do a search on [Google](#) and find something that is.

The IDEALX scripts are available [here](#) in both RPM and TGZ format. Download them to your hard drive and extract them to a directory of your choice. For the purposes of this tutorial, we've placed the scripts in `/usr/local/sbin`.

Now with all the requisite software in place, we're ready to start configuring.

Section 3. Configuring OpenLDAP

The LDAP configuration process

With the server software installed, the next step is configuring OpenLDAP to serve as a user authentication source for Samba. This section tackles creating the required LDAP configuration files, adding a Samba-specific schema, populating the LDAP directory with some basic entries, and configuring LDAP to use PAM (Pluggable Authentication Modules) for user authentication.

There are numerous ways to layout your LDAP tree. For the purposes of this tutorial, we'll create a DN (Distinguished Name) tree as follows:

Note that the shadowAccount object is not required for UNIX systems that do not use shadow passwords; for such scenarios posixAccount object layout (again, from nis.schema) is sufficient.

Step 1: Copy samba.schema

The first step is to copy the samba.schema to the `/etc/openldap/schema`.

If you compiled Samba from source, this file is located in the `[base-src-dir, typically, /usr/local/src/samba]/example/LDAP/` directory. If you used a modified RPM for your installation, the file is *probably* (layouts vary across distributions) under `/usr/share/doc/samba-version/examples/LDAP/`.

TIP: Some older Samba packages (pre-2.2.x) had an attribute type of `displayName` listed in `samba.schema`. This entry duplicates an identical entry found in the `inetorgperson.schema`. In such cases, edit `samba.schema` and comment out the `displayName` attributetype by placing a hash mark (#) in front of each line of the associated block.

Step 2: Edit/create `slapd.conf`

The next step is to create or edit `/etc/openldap/slapd.conf`; some OpenLDAP installations create a base `slapd.conf` file to use as a starting point, some do not.

An explanation of the pertinent entries from the listing below is discussed in the next panel.

```
# /etc/openldap/slapd.conf
# last modified, 4/20/02 by TMS
```

```

include /etc/openldap/schema/core.schema
include /etc/openldap/schema/cosine.schema
include /etc/openldap/schema/inetorgperson.schema
include /etc/openldap/schema/nis.schema
include /etc/openldap/schema/redhat/rfc822-MailMember.schema
include /etc/openldap/schema/redhat/autofs.schema
include /etc/openldap/schema/redhat/kerberosobject.schema
include /etc/openldap/schema/samba.schema

pidfile //var/run/slapd.pid
argsfile //var/run/slapd.args

#Sample Access Control
# Allow read access of root DSE
# Allow self write access
# Allow authenticated users read access
# Allow anonymous users to authenticate
access to dn="" by * read
access to *
    by self write
    by users read
    by anonymous auth
# if no access controls are present, the default is:
# Allow read by all
# rootdn can always write

#####
##### ldbm database definitions#####
#####
database ldbm
suffix "dc=syroidmanor,dc=com"
rootdn "cn=Manager,dc=syroidmanor,dc=com"
rootpw secret

# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd/tools. Mode 700 recommended.
directory /var/lib/ldap

# Indices to maintain
index primaryGroupID eq
index rid eq
index uid eq
index uidNumber eq
index gidNumber eq
index cn pres,sub,eq
index objectClass eq
index default sub
# ends

```

Dissecting slapd.conf

Note the following regarding the slapd.conf shown on the preceding panel:

- The filename is slapd.conf, *not* sldap.conf as one might suspect.
- The purpose of slapd.conf is to control/configure the OpenLDAP server daemon.
- The include statements tell the LDAP server which schemas to enable for object storage; ensure you have samba.schema listed.

- Some schemas have dependencies. Many people do not realize this and end up pulling their hair out trying to troubleshoot why their LDAP server won't start. Dependencies are always listed at the start of a schema file (schema files are just text; to quickly investigate a schema file type: `less nameofschema.schema`). For example, `samba.schema` is dependent on the `uid` attribute in `cosine.schema` and the `displayName` attribute from `inetorgperson.schema`; `core.schema`, on the other hand, is a "top level" schema and has no dependencies.
- Pay attention to the comment regarding access control -- if no access controls are present all users can read only; the `rootdn` (`cn=Manager`) can always read/write.
- Entries are case-sensitive. "`cn=Manager`" is distinct from "`cn=manager`". This fact has caused many an administrator days of head-scratching trying to figure out why they can't access their directory.
- The `rootpw` entry is just what it seems -- the password the `rootdn` or "Manager" must supply to write to the directory. Obviously "secret" is not a good choice. Good passwords mix upper and lower case, punctuation, and are at least 8 characters in length. Furthermore, given that the root password is stored in plain text, make sure you carefully guard access to the file by ensuring the it is not "world readable". Most installations correctly configure the file, but it doesn't hurt to double-check. Under Red Hat, the OpenLDAP server is run as user/group `ldap.ldap`.
- The `database` entry determines the format the directory data is stored in. This can be one of `ldbm`, `shell`, or `passwd`, depending on which backend is serving the directory. Options also exist to use a relational database such as MySQL or DB2 to store directory objects. See `man slapd.conf` for details (remember, the `man` command is your friend!). Typically, support for RDBMs requires a recompilation of OpenLDAP.

Next we'll create the `/etc/openldap/ldap.conf` file.

Step 3: Create `ldap.conf`

In contrast to `/etc/openldap/slapd.conf`, the `/etc/openldap/ldap.conf` is relatively simple. `ldap.conf` is used by OpenLDAP clients and libraries. *Caution:* In some LDAP configurations there are *two* `ldap.conf` files present on the server: `/etc/ldap.conf` (used by PAM) and `/etc/openldap/ldap.conf` (used by LDAP clients and libraries); don't confuse the two.

```
# /etc/openldap/ldap.conf
# LDAP defaults for clients and libraries
# At a minimum, HOST and BASE need to be set
# See man ldap.conf for further settings and options

HOST 127.0.0.1
BASE dc=syroidmanor, dc=com

# ends
```

Other available options for `ldap.conf` include search timeouts, maximum returned search size, the port to connect to (if non-default), and several security properties for connecting via Cyrus SASL. Note that users can each specify unique settings by

creating `.ldaprc` and placing it in their home directory. Administrators can also override this option by forcing LDAP to read only the `/etc/openldap/ldap.conf` file.

Step 4: Start the server

Finally, start your OpenLDAP server. On a Red Hat based system, type `/etc/init.d/ldap start`. The server should start with error or protest. If it doesn't:

- Verify you have the correct schemas loaded and all dependencies are met.
 - Verify `/var/lib/ldap` exists and is owned by the user/group designated to run the server (under Red Hat this is `ldap.ldap`).
 - Double-check all your configuration files for typos, the correct "dn" entries, etc.
 - Start reading the LDAP documentation (`man ldap`, `info slapd`, and www.openldap.org are three good starting points).
-

Initial entries

With the LDAP server running, it's time to populate the directory with some initial entries. There are two ways to proceed:

- One, use the `smbldap-tools` scripts we downloaded and installed earlier in the tutorial.
- Two, create a text file (`base.ldif`) and add entries using the `ldapadd` command.

We'll start with the `smbldap-tools` approach...

Using `smbldap-populate.pl`

Based on the installation instructions provided in Section 2, first "cd" to `/usr/local/sbin`. There you'll find a file called `mkntpwd.tar.gz`. Do the following to uncompress, build, and compile the program:

```
[root@thor sbin]# tar xvzf mkntpwd.tar.gz
[root@thor sbin]# cd mkntpwd
[root@thor sbin]# make
[root@thor sbin]# make install
# Note: this will place the built executable in /sbin;
# I prefer all my smbldap files together in one place,
# so the following further steps are required.
[root@thor sbin]# cd .. && rm -rf ./mkntpwd (remove the directory
so the file can be copied back)
```

```
[root@thor sbin]# mv /sbin/mkntpwd /usr/local/sbin
```

Now open the file `/usr/local/sbin/smbldap_conf.pm` in your favorite text editor and start editing. You'll find instructions roughly one page down (`## Configuration Start here`) indicating exactly which entries require changing. If you've departed from the organization units discussed earlier in this section, you'll also have to change the `$usersdn`, `$computersdn`, and `$groupsdn` entries. Also:

- Under `$binddn` enter `dn=Manager` *not* "manager" (if you've been following the examples used in this tutorial; the key is to ensure the case is matched across all LDAP and SMLDAP tools configuration files).
- Don't forget to escape ("\\") any backslashes used in the `$_userSmbHome/$_userProfile` section.
- Ensure the path to `smbpasswd` is correct (my entry reads `/usr/local/samba/bin/smbpasswd`).
- Check all entries pertaining to UID, GID, etc. to make sure the entries used match your user/group numbering conventions.

Finally, execute `smbldap-populate.pl`. If your configuration is correct, the script will spew off a bunch of "adding entry..." material and your directory will be populated with a base set of objects. If you get any "credential" or "cannot bind" errors, you've got a typo somewhere in `smbldap_conf.pm` -- go back and recheck your entries.

Manually populating using an LDIF file

For those admins who like to do things manually, OpenLDAP can certainly accommodate you. The second method of populating an LDAP directory is with an *LDIF* file. An LDIF file is a text file, with multiple entries conforming to a specific format. Below are two sample directory object entries.

```
dn: dc=syroidmanor,dc=com
objectClass: domain
dc: syroidmanor
```

```
dn: ou=Users,dc=syroidmanor,dc=com
objectClass: top
objectClass: organizationalUnit
ou: Users
description: System Users
```

Create a file called `base.ldif` using the above format and create your desired objects. Save the file in a directory of your choosing, then execute the following command:

```
[root@thor root]# ldapadd -x -h localhost -D "cn=Manager,dc=syroidmanor,dc=com"
-f /root/base.ldif -W
```

You'll be prompted for the "Manager's" password ("secret" in the examples given), and if there are no syntactical errors in the file, the directory will be populated with its

contents.

Type `man ldapadd` for an explanation of what the above option switches do. A sample `base.ldif` file is available in the [Summary and resources](#) on page 24 section.

Configuring LDAP for PAM authentication

If you've elected to use PAM (keep in mind the earlier warning -- PAM *does not* play nice with Samba and encrypted passwords) we need to configure LDAP to use PAM for user authentication (which is a separate issue from using LDAP to authenticate your Samba users). Fortunately, under Red Hat this process is straightforward due to a bundled program: `/usr/sbin/authconfig`.

As root, from the command line or terminal window, type `authconfig` and select the following options/checkboxes:

- Cache Information
- Use LDAP
- DO NOT select Use TLS -- getting TLS up and running is a painful process unless you know your secure authentication protocols inside-out
- Your Server is 127.0.0.1 (unless your LDAP server is physically located on a separate system)
- Base DN: `dc=syroidmanor,dc=com`
- Use Shadow Passwords
- Use MD5 Passwords
- Use LDAP Authentication
- Again, Server: 127.0.0.1
- Base DN: `dc=syroidmanor,dc=com`

The Cache Information options implies the use of the `nscd` (Name Service Caching Daemon) service; it caches frequently-used LDAP requests. Selecting OK on the final dialog automatically starts the `nscd` daemon. The last piece of the puzzle is the `/etc/ldap.conf` file.

Editing `/etc/ldap.conf`

As noted earlier in this section, LDAP uses a second `ldap.conf` file when PAM authentication is used. Fire up your favorite text editor and add the following to `/etc/ldap.conf`:

```
# /etc/ldap.conf
# LDAP authentication configuration
# last modified, 4/15/02, TMS
```

```
host 127.0.0.1
# your LDAP server
base dc=syroidmanor,dc=com
# your DN search base
# point the nss modules to the correct search base
nss_base_passwd    dc=syroidmanor,dc=com?sub
nss_base_shadow    dc=syroidmanor,dc=com?sub
nss_base_group     ou=Groups,dc=syroidmanor,dc=com?one
ssl no
pam_passwd md5
# ends
```

In the above example, the "?sub" query option is used because of the organization of our directory tree; we've chosen to separate out "ou=Computers" and "ou=Users". For more details on the ?sub and ?one options, do an Internet search on RFC2307. There's also a PDF available explaining the concepts and implementation behind PAM and NSS; see the [Summary and resources](#) on page 24 section for more information.

Testing

Finally, the moment of truth has arrived -- does everything work?

Switch to the directory containing the SMLDAP Perl scripts; in the examples given, /usr/local/sbin. Using the smbldap-tools, create a new user enter the following:

```
[root@thor sbin]# ./smbldap-useradd.pl -m testuser2
adding new entry "uid=testuser2,ou=Users,dc=syroidmanor,dc=com"
[root@thor sbin]# ./smbldap-passwd.pl testuser2
Changing password for testuser2
New password :
Retype new password :
all authentication tokens updated successfully
[root@thor sbin]#
```

Now try to login to the system from another computer using the account just created. If everything is configured correctly, you should be correctly authenticated:

```
[tom@phaedrus tom]$ ssh testuser2@thor
testuser2@thor's password:
Last login: Thu Apr 25 11:28:24 2002 from 192.168.1.100
[testuser2@thor testuser2]$ id
uid=1000(testuser2) gid=100(users) groups=100(users)
```

Now let's switch gears and look at the Samba end of the equation.

Section 4. Configuring Samba

Configuration

With the LDAP portion of the configuraton complete, the only remaining task is to tell Samba to look to the LDAP directory service for user authentication rather than its own password file, `smbpasswd`. This is a relatively simple procedure, especially given that we already have a correctly configured and functioning Samba PDC to work with as a starting point. First, some background notes:

- Samba stores all its configuration information in one file, `smb.conf`. The location of this file varies depending on installation options; the default directory is `/usr/local/samba/lib`.
- The domain our server is providing authentication for is called `SYROIDMANOR` (the "workgroup =" option in `smb.conf`).
- The server's NETBIOS name is `THOR`.
- The server is configured as the "master browser" of the network.
- The server is configured to provide "roaming" profiles; they're stored under `/home/samba/profiles`.
- When a user logs in, any scripts located in the `/home/netlogon` directory will be run.

The next step is to add the required LDAP-specific options. These are all placed under the `[global]` section of `smb.conf`.

LDAP server specifics

The following configuration snippet shows the LDAP-specific options required to direct Samba to the directory service for user authentication:

```
;LDAP-specific settings
  ldap admin dn = "cn=Manager,dc=syroidmanor,dc=com"
  ldap server = localhost
  ldap port = 389
  ldap ssl = no
  ldap suffix = "ou=Users,dc=syroidmanor,dc=com"
```

- The first line tells Samba who the administrator of the LDAP directory is; this is who Samba will connect as when adding, deleting, or modifying user accounts. Note that this entry must match (exactly -- case counts!) the `rootdn` entry from your `slapd.conf` file.
- The next line specifies the name of the computer hosting the LDAP directory. In the scenario presented in this tutorial, both Samba and OpenLDAP are hosted on the same system. If your network topology has LDAP residing on a separate system, replace `localhost` with the computer's name. A static IP could also be used as an

alternative.

- The third line is the port the LDAP server is configured to "listen" on. Port 389 is the standard unencrypted port; 636 is the standard encrypted port. If the next line is set to `ldap ssl = on`, Samba will automatically attempt to contact the directory server on port 636.
- The `ldap ssl` option determines whether or not to encrypt communications between the PDC and the LDAP server. If you're using SSL encryption, change the `ldap ssl = no` to `yes` (or "on"). We leave encryption off as both services are based on the same server.
- The `ldap suffix` is the base DN to use when searching the directory. If eliminated, LDAP will start all searches from the top of the tree (i.e., `dc=syroidmanor,dc=com`). Setting this option to a common search point like "Users" speeds lookup times.

There's one further option that can be added to the above section: `ldap root passwd =`. Leaving administrative passwords lying around in configuration files is never a good idea if other options exist (they don't in the case of `/etc/openldap/slapd.conf`). In this case we do have a choice -- we can hide it as a *secret* in a *tdb* database.

Administrative secrets

The purpose of this whole exercise is to store all user authentication information in a central repository, namely an LDAP directory. However the Samba daemon needs to act on behalf of the LDAP administrator to change the information stored there. This means Samba must know the administrative password. Instead of placing this password in a file someone could potentially break in to, we're going to encrypt it and store it in a *tdb* database that only Samba can access. The command to accomplish this is:

```
smbpasswd -w yoursecretpassword
```

This will create a file called `secrets.tdb`, which in the case of a default installation, is stored in the directory `/usr/local/samba/private`. Note that the password you enter above *must* match the "secret" password contained in `/etc/openldap/slapd.conf`.

Script updates

The last changes we need to make to `smb.conf` are the two lines that allow users to change their password from a client, and for machine accounts to be automatically added when required. Below are the respective options lines:

```
;password sync
```

```
passwd program = /usr/local/sbin/smbldap-passwd.pl -o %u
passwd chat = *New*password* %n\n *Retype*new*password*
%n\n *passwd:*all*authentication*tokens*updated*
unix password sync = Yes
```

```
;automatically add trust accounts
add user script = /usr/local/sbin/smbldap-useradd.pl
-m -d /dev/null -g computers -s /bin/false
```

As you can see, we simply substitute the `/usr/local/samba/bin/smbpasswd` command with the Perl scripts installed earlier in the tutorial. The reason for the change is because `smbpasswd` does not know how to converse with the LDAP server; the replacement Perl scripts do. On the next panel is the completed `smb.conf` listing.

The completed `smb.conf`

```
# /usr/local/samba/lib/smb.conf
# samba configuration file
# last updated: 4/19/2002 by tms

[global]

    ;basic server settings
    workgroup = SYROIDMANOR
    netbios name = THOR
    server string = Samba-LDAP PDC running %v
    socket options = TCP_NODELAY IPTOS_LOWDELAY SO_SNDBUF=8192 SO_RCVBUF=8192

    ;PDC and master browser settings
    os level = 64
    preferred master = yes
    local master = yes
    domain master = yes
    wins support = yes

    ;security and logging settings
    security = user
    encrypt passwords = yes
    log file = /var/log/samba/log.%m
    log level = 2
    max log size = 50
    hosts allow = 127.0.0.1 192.168.1.0/255.255.255.0

    ;password sync
    passwd program = /usr/local/sbin/smbldap-passwd.pl -o %u
    passwd chat = *New*password* %n\n *Retype*new*password* %n\n
    *passwd:*all*authentication*tokens*updated*
    unix password sync = Yes

    ;LDAP-specific settings
    ldap admin dn = "cn=Manager,dc=syroidmanor,dc=com"
    ldap server = localhost
    ldap port = 389
    ldap ssl = no
    ldap suffix = "ou=Users,dc=syroidmanor,dc=com"

    ;user profiles and home directory
```

```
logon home = \\%L\%U\  
logon drive = H:  
logon path = \\%L\profiles\%U  
logon script = netlogon.bat  
  
;automatically add trust accounts  
add user script = /usr/local/sbin/smbldap-useradd.pl  
-m -d /dev/null -g computers -s /bin/false  
  
# ===== shares =====  
  
[homes]  
comment = Home Directories  
valid users = %S  
browseable = no  
writeable = yes  
create mask = 0664  
directory mask = 0775  
  
[profiles]  
path = /home/samba/profiles  
writeable = yes  
browseable = no  
create mask = 0600  
directory mask = 0700  
  
[netlogon]  
comment = Network Logon Service  
path = /home/netlogon  
read only = yes  
browseable = no  
write list = tom
```

Throwing the switch...

The moment of truth has arrived... it's time to test your handiwork. At the console (or in a console window), and as root, type (on a Red Hat system; some distributions use a different layout for their initialization scripts):

```
/etc/init.d/smb start [or if Samba is already running, restart]
```

With Samba it's always easy to know if you've misspelled an option or parameter -- the daemons will start and quit again without any error warnings. That's why it's always a good idea to follow a Samba start|restart with command `ps -ef | grep smb`. You should see at least one *smbd* and one *nmbd* daemon running.

If Samba failed to start:

- Type **testparm** (or `/usr/local/samba/bin/testparm`, depending on your Samba installation) and examine the output. If you have a syntax error, testparm will indicate the line number it occurs on.
- Check your network connection (**ifconfig -a** as root); Samba needs to see an active network interface before it will run.
- Check your Samba logs for any insights. Again, log location depends on build-time

configuration; on a default compile, you'll find them under
`/usr/local/samba/var`.

Next up we'll look at account management using IDEALX's scripting utilities.

Section 5. User management

Managing Samba users

Before we begin the actual business of adding users to our LDAP directory, it wouldn't hurt to do a quick review of Samba's rules pertaining to user/machine accounts, and how those rules shape Samba's behavior when it comes to accessing an LDAP store.

- **RULE ONE:** Before you can add a user to the `smbpasswd` password file, that user *must* have an UNIX/Linux account (typically stored in `/etc/passwd`) on the system hosting the Samba service. If you try to add a user account with the `/usr/local/samba/bin/smbpasswd` utility, you will be informed of this fact. If you try to add a user account with `smbldap-useradd.pl` script (which we're about to discuss), you *will not* be warned of this fact.
- **RULE TWO:** If you try to add a user account (and by user, I mean either a user account or machine account) using the above-mentioned Perl script, you will create a POSIX account in the LDAP directory for said user. This user will then be able to log into any POSIX-based machine that authenticates from this LDAP server. This user will *not* be able to access any Samba shares hosted by the Samba server and authenticated by the LDAP server. This is because the user exists in the `posixAccount` branch of the LDAP tree, but not in the `sambaAccount` branch.
- **RULE THREE:** (which is a pairing of rule one and two above) If you want a user to have *both* a POSIX account and a Samba account on the LDAP server, make sure they have an existing system account on the Samba server first, then add them to the LDAP directory using `smbldap-useradd.pl`.

LDAP's account structures

Most of our discussion of LDAP account structures has, up to now, been pretty much theoretical. To add some dimension to the difference between a POSIX account and a Samba account under LDAP, and to cement the account creation details discussed in the previous panel, let's look at an example. The LDIF (LDAP Directory Information File) output shown below is for a user created with the following command:

```
/usr/local/sbin/smbldap-useradd.pl -m -P tom ("-m", create user directory and profile based on templates contained in /etc/skel; "-P" prompt for a password after the user is added). To produce the actual LDIF output:  
/usr/local/sbin/smbldap-usershow.pl tom
```

```
[root@thor sbin]# /usr/local/sbin/smbldap-usershow.pl tom  
dn: uid=tom,ou=Users,dc=syroidmanor,dc=com  
objectClass: top  
objectClass: account  
objectClass: posixAccount  
cn: tom  
uid: tom  
uidNumber: 500
```

```
gidNumber: 100
homeDirectory: /hometom
loginShell: /bin/bash
gecos: User
description: User
userPassword:: e1NTSEF9bWxBL1RHZFN0TkREEWlGTndZOF1CWUVUdWp3MGgrbTc=
```

Next let's add another user to the same user entry and do the following:

```
root@thor root # useradd -p test tom2
root@thor root # /usr/local/samba/bin/smbpasswd tom2
New SMB password: typesecretpassword
Retype new SMB password: typesecretpassword
User added
All authentication tokens updated
root@thor root #
```

Now, if we type `/usr/local/sbin/smbldap-usershow.pl tom` we get the following output:

```
dn: uid=tom,ou=Users,dc=syroidmanor,dc=com
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: sambaAccount
cn: tom
uid: tom
uidNumber: 500
gidNumber: 100
homeDirectory: /hometom
loginShell: /bin/bash
gecos: User
description: User
userPassword:: e1NTSEF9bWxBL1RHZFN0TkREEWlGTndZOF1CWUVUdWp3MGgrbTc=
lmPassword: 552902031BEDE9EFAAD3B435B51404EE
pwdCanChange: 0
pwdMustChange: 2147483647
ntPassword: 878D8014606CDA29677A44EFA1353FC7
pwdLastSet: 1010179230
rid: 2000
```

As you can see, the LDIF for user `tom` now has *both* `posixAccount` details and `sambaAccount` information.

Sidebar: uid's, gid's, and RIDs

You've probably seen reference to the term *RID* either in the Samba documentation or perhaps in the output from a LDAP account. So what's a RID? UNIX operating systems (including derivatives like Linux) uniquely identify a user via an integer *uid* (User ID) and groups by an integer *gid* (Group ID). This information can be accessed by typing `id` when logged in as a given user.

Current Microsoft operating systems uniquely identify users and groups by a value known as an RID, which is an integer typically expressed in hexadecimal. Under UNIX,

users and groups exist in independent name spaces. On a Microsoft operating system, users and groups exist in a single name space.

Samba maps UNIX uids and gids to RIDs using the following formula:

```
rid = 2 (uid) + 1000  
rid = 2 (gid) + 1001
```

So if user `tom`, working on a Red Hat system, had a uid of 500 and a gid of 500, his mapped RIDs on a Microsoft domain would be 2000 and 2001 respectively.

Section 6. Summary and resources

Summary

So there you have it -- a fully configured OpenLDAP server that provides user authentication for a Samba PDC. Or at least that's what you're supposed to end up with. Teasing aside, there will no doubt be circumstances where the material contained within this tutorial does not work for some people for whatever reason. Remember back in [Gathering and building the required software](#) on page 5 where we talked about choice in Open Source software? Well, choice is a double-edged sword at times. Choice also adds a layer of complexity whereby multiple paths do not always lead to the same destination. So we tinker and fiddle, and eventually find a solution -- which, in the end, just may be better than the original. And then, in the spirit of the Open Source model, we contribute those solutions back to the community.

Due to the scope of the topics in this tutorial, certain background material could not be covered. For example, we began the tutorial under the assumption you already had a functional Samba PDC up and running; we also took as a given that you had a basic understanding of the terms and concepts behind LDAP. If you found yourself missing pieces of the LDAP/Samba puzzle, have a glance through the resources listed in the next two panels. They contain some excellent background material and insights that will surely benefit even the seasoned administrator.

I trust you've found the material presented here useful and beneficial to your particular implementation. As always, comments, missing bits, and errata are appreciated. Either fill in the Feedback form on the last panel, or drop me a line directly (ibm-feedback@syroidmanor.com).

Further resources: LDAP and OpenLDAP

The following resources will get you started down the road to understanding and implementing LDAP directory services:

- The definitive source for all things pertaining to the OpenLDAP Project be found at www.openldap.org. Not only will you find the current OpenLDAP release here (which at the time of this writing was 2.0.23), but there's also oodles of good documentation available. In particular, check out the [OpenLDAP 2.0 Administrator's Guide](#).
- There is a TON of material on LDAP configuration and implementation on the mighty World Wide Web. For an extensive list, open your favorite browser, go to Google.com and type in "LDAP and OpenLDAP" as a search phrase. The real trick is sorting the wheat from the chaff. Two resources I've found particularly useful are Meer and Biondo's [LDAP Implementation HOWTO](#) and the [YoLinux LDAP Tutorial](#). The YoLinux Tutorial also has a good page of [OpenLDAP Links](#). Another good resource is Luiz Ernesto Pinheiro Malere's [LDAP Linux HOWTO](#). Personally, I find that no one HOWTO provides me all the pieces I need, so it's a matter of harvesting the best tidbits from a variety of sources.

- IBM's [Redbooks](#) are always a good source of both high-level and platform specific information. A simple search on the Redbook site will net you [over 40 hits using the term "ldap"](#). I can personally recommend the titles *Understanding LDAP* (SG24-4986-00) and *LDAP Implementation Cookbook* (SG24-5110-00).
 - Adam Tauno Williams has produced two good tutorials detailing the hows and whys of [PAM and NSS authentication](#) and [LDAP V3](#). Both are available as PDF files.
 - [PADL Software](#) maintains and distributes the nameservice modules `nss_ldap` and `pam_ldap` discussed in this tutorial.
-

Further resources: Samba

Like LDAP, Samba resources are aplenty on the Web:

- The official Samba site is www.samba.org. Here you'll find the latest downloads of the latest stable source, RPMs and TGZ files for a variety of platforms and distributions, and the latest development offerings via CVS. There's also a good deal of [documentation available on the Samba site](#). In particular, grab the Samba-HOWTO-Collection; it's available in [PDF](#) and [HTML](#). The [Samba Mailing Lists](#) are also a great source of experience and information.
- Don't forget to browse through the documentation that ships with Samba itself. Given a base installation directory of `/usr/local/samba` you'll find the complete text of O'Reilly's *Using Samba* title under `../swat/using_samba` and several useful HTML documents under `../swat/help`.
- The definitive guide to configuring Samba to play the role of a Windows domain controlled is, of course, the [Samba as a PDC](#) tutorial I wrote that is currently hosted on the [IBM eServer Developer Domain](#).
- [IDEALX.org](#) is the place to grab the Samba/LDAP Perl scripts used in this tutorial. There's also a Samba PDC/LDAP HOWTO available from the same location; it is incomplete and confusing in spots, however, and not up-to-date regarding Samba's current capabilities.
- The [PADL Software site](#), in addition to hosting the aforementioned `nss_ldap` and `pam_ldap` PAM modules, also has a [set of migration scripts](#) you may be interesting in checking out. Like the IDEALX scripts, they're written in Perl but rather than being targeted at Samba user management the PADL scripts are designed to migrate UNIX configuration files such as `/etc/hosts`, `/etc/group`, `/etc/networks`, and `/etc/passwd`.
- Finally, Ignacio Coupeau maintains an [Samba-PDC LDAP V3 HOWTO](#). It too is rather incomplete in spots, but it will point you in the general direction. It also has some good material on configuring Samba to authenticate using SSL and TLS. Note that the HEAD patches Coupeau lists are already integrated into the Samba HEAD CVS tree.

Good luck, and have fun!

Feedback

Please send us your feedback on this tutorial. We look forward to hearing from you!

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The open source Toot-O-Matic tool is an XSLT stylesheet and several XSLT extension functions that convert an XML file into a number of HTML pages, a zip file, JPEG heading graphics, and two PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML. (It also saves our production team a great deal of time and effort.)

You can get the source code for the Toot-O-Matic at www6.software.ibm.com/dl/devworks/dw-tootomatic-p. The tutorial [Building tutorials with the Toot-O-Matic](#) demonstrates how to use the Toot-O-Matic to create your own tutorials. developerWorks also hosts a forum devoted to the Toot-O-Matic; it's available at www-105.ibm.com/developerworks/xml_df.nsf/AllViewTemplate?OpenForm&RestrictToCategory=11. We'd love to know what you think about the tool.