About This Guide

Below are instructions for setting up a Nucore development environment to test LDAP authentication. Here are this guide's assumptions:

- You're working on Mac OS X Snow Leopard.
- You don't have a LDAP server to test against.
  - If you do, you might want to skip to the "Enabling LDAP On Nucore" section

Setting Up A Test LDAP Server

Install OpenLDAP

The easiest way to get OpenLDAP up and running on your Mac is via MacPorts.

1. Download MacPorts
2. Install MacPorts
3. Open a terminal and run **`sudo port install openldap`**

Installation will take some time, but once it's complete you should have a runnable LDAP server.

Configure The LDAP Server

MacPorts installs OpenLDAP's configuration file to ***/opt/local/etc/openldap/slapd.conf***. You should replace the stock version with one that is preconfigured for authentication.

Start The LDAP Server

```
sudo /opt/local/libexec/slapd -d -1
```

Seed The LDAP Directory

For testing you'll need at least one organization and one user in the LDAP directory. First you need to create the organization:

```
ldapadd -x -D "cn=admin,dc=example,dc=com" -w secret -f org.ldif
```

org.ldif needs to be a LDAP data interchange formatted file. See this one for a quick start and/or as an example. Second you need to create a user:

```
ldapadd -x -D "cn=admin,dc=example,dc=com" -w secret -f usr.ldif
```

usr.ldif also needs to be a LDAP data interchange formatted file. See this one for a quick start and/or as an example.

Search The LDAP Directory

Now that the server is running and you have some seed data you should be able to run a command line search to make sure it is working properly:

```
ldapsearch -x -D "cn=cgreen,dc=example,dc=com" -w secret -
b "dc=example,dc=com" "cn=*"
```

You should get a result that looks similar to the following:

```
# extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> with scope subtree
# filter: cn=*
# requesting: ALL
#
```

```
# cgreen, example.com
dn: cn=cgreen,dc=example,dc=com
cn: Chico Green
cn: cgreen
givenName: Chico
sn: Green
mail: cgreen@example.com
userPassword:: bm90Z3VtcA==
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Enabling LDAP Authentication On Nucore

Nucore uses Devise for authentication, and the ldap_authenticatable Devise module for LDAP authentication. When LDAP is enabled on Nucore Devise will try to authenticate users against both the LDAP server and the local users table. Whichever works first wins. If neither works login is denied.

To configure Nucore to use LDAP you need to update ***initializers/devise.rb*** in the ***config*** directory of Nucore's **`Rails.root.`** Specifically, at the bottom of the file you need to uncomment the following:

```
# LDAP server configuration, if any
config.ldap_host = 'localhost'
config.ldap_port = 389
config.ldap_base_dn = 'dc=example,dc=com'
config.ldap_login_attribute = 'cn'
```

These settings will work for the test LDAP server described in the previous sections of this document. You'll need to change `config.ldap_host` if you are using an existing LDAP server or if the test LDAP server is not on the same computer as Nucore.

Configuring The LDAP dn

LDAP authenticates users with a distinguished name (dn). The dn is typically composed of multiple LDIF attributes. One of the attributes of the dn will specify the user's login. The additional attributes of the dn, for Nucore, are expected to be consistent across all users.

LDAP users allowed to access Nucore must have a password-less record in Nucore's users table. The record's username attribute must correspond to the login part of the user's LDAP dn. The LDIF attribute identifying the login must be the value of `config.ldap_login_attribute`. The additional LDIF attributes that make up the remainder of the LDAP dn must be the value of `config.ldap_base_dn`.

Testing Nucore LDAP Authentication

Make sure you have a password-less user in Nucore that corresponds to the LDAP user we created previously:

From Nucore's **`Rails.root...`**

```
script/console
> User.create!(:username => 'cgreen', :first_name
=> 'Chico', :last_name => 'Green', :email => 'cgreen@example.com')
```

Now fire up Nucore and try logging in with username 'cgreen' and password 'secret'. You should login successfully. Now, check the standard output of the LDAP server and you should see the authentication query made by Nucore.

References

- Setting up OpenLDAP on OS X Leopard
- Building an Address Book with OpenLDAP
- OpenLDAP Password Protection, security and Authentication
- OpenLDAP Administrator's Guide

Samples

org.ldif

```
dn:   dc=example,dc=com
objectClass:    top
objectClass:    dcObject
```

```
objectClass:    organization
dc:     example
o:      Table XI
```

usr.ldif

```
dn: cn=cgreen,dc=example,dc=com
cn: Chico Green
gn: Chico
sn: Green
mail: cgreen@example.com
userPassword: secret
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

slapd.conf

```
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include         /opt/local/etc/openldap/schema/core.schema
include         /opt/local/etc/openldap/schema/cosine.schema
include         /opt/local/etc/openldap/schema/inetorgperson.schema

# Define global ACLs to disable default read access.

# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral  ldap://root.openldap.org

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

# Load dynamic backend modules:
# modulepath    /opt/local/libexec/openldap
# moduleload    back_bdb.la
# moduleload    back_hdb.la
# moduleload    back_ldap.la

# Sample security restrictions
#    Require integrity protection (prevent hijacking)
#    Require 112-bit (3DES or better) encryption for updates
#    Require 63-bit encryption for simple bind
# security ssf=1 update_ssf=112 simple_bind=64

# Sample access control policy:
#     Root DSE: allow anyone to read it
#     Subschema (sub)entry DSE: allow anyone to read it
#     Other DSEs:
#           Allow self write access
#           Allow authenticated users read access
#           Allow anonymous users to authenticate
#     Directives needed to implement policy:
# access to dn.base="" by * read
# access to dn.base="cn=Subschema" by * read
# access to *
#     by self write
#     by users read
#     by anonymous auth
#
# if no access controls are present, the default policy
# allows anyone and everyone to read anything but restricts
# updates to rootdn.  (e.g., "access to * by * read")
#
# rootdn can always read and write EVERYTHING!

#######################################################################
#
# BDB database definitions
#######################################################################
#

database   bdb
suffix          "dc=example,dc=com"
rootdn          "cn=admin,dc=example,dc=com"
# Cleartext passwords, especially for the rootdn, should
# be avoid.  See slappasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw          secret

access to attr=userPassword
       by dn="cn=admin,dc=example,dc=com" write
       by self write
       by * auth
access to *
       by dn="cn=admin,dc=example,dc=com"  write
       by dn="cn=cgreen,dc=example,dc=com" read
       by users read
       by self write
       by * auth

# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory  /opt/local/var/openldap-data
# Indices to maintain
index objectClass     eq
```