Deepport.net

Home | Photography | Computing | Web design | Contact me |

JAN 25

# Setting up a Linux server for OS X clients

*This document is fairly long but I have tried to be verbose with my instructions and code so that the process itself should be short and relatively simple.*

The aim of this document is to create a Linux server that will act as an Open Directory equivalent for OS X clients. This includes Kerberos and OpenLDAP support with full client management via Apple's Workgroup Manager. Home directories can be accessed via NFS with Portable Home Directories support for laptops. AFP and other services will authenticate via Kerberos.

Note: This document is still a work in progess!
It is now a fully working solution with some caveats. As such I am publishing it so that anyone else may benefit from my existing notes.

ALL feedback is welcome. Either leave a comment or email me at andrew@deepport.net.

Note 2: OS X 10.5.7 seems to finally correct the filesync issues with Portable Home Directories I had been seeing in some of the earlier 10.5 versions. The release notes mention fixes for PHD syncing to OS X Server 10.4 which also appears to cover using NFS on other platforms.

Contents

Recent Posts

- Driving Force Behind the Australian Internet Filter
- Some thoughts on the cost of Australia's NBN
- OS X Portable Home Directories and syncing flaw with bundles
- iPhone Annoyances
- Does Telstra's Sensis still matter

Categories

- IT
- Opinion

# Issues and Improvements

- Workgroup Manager throws up errors on some tasks though it still works.
- Passwords cannot yet be set via Workgroup Manager.
- Coumputers do not have accounts and therefore cannot have preferences assigned to them.
- Dovecot and Postfix authenticate via Kerberos and/or password (for remote users) but mail delivery doesn't yet work.
- I would like to have an alternative to the OS X Server Software Update service.

# Server Setup

This document is based on Ubuntu 8.10 Server

## Linux Setup and Configuration

### Kerberos

Kerberos references:

http://www.linux.com/base/ldp/howto/Kerberos-Infrastructure-HOWTO/index.html

http://www.centos.org/docs/3/html/rhel-rg-en-3/s1-kerberos-server.html

http://www.centos.org/docs/3/html/rhel-rg-en-3/s1-kerberos-clients.html

Install the Kerberos services by running:

```
aptitude install krb5-admin-server
```

On Ubuntu 8.10 this installed the following:

```
krb5-admin-server
krb5-config
krb5-kdc
krb5-user
libkadm55
```

Edit the `/etc/krb5.conf` file to contain the following (adjusted for your domain as appropriate):

```
[libdefaults]
        default_realm = HOME.DEEPPORT.NET

[realms]
        HOME.DEEPPORT.NET = {
                kdc = zamek.home.deepport.net:88
                admin_server = zamek.home.deepport.net
                default_domain = home.deepport.net
        }

[domain_realm]
        .home.deepport.net = HOME.DEEPPORT.NET
        home.deepport.net = HOME.DEEPPORT.NET
```

Then run:

```
krb5_newrealm
```

*or*

```
kdb5_util create -s
```

Review the contents of `/etc/krb5kdc/kdc.conf` though the default values should be suitable. In particular pay attention to the line `acl_file =` (default: `/etc/krb5kdc/kadm5.acl`).

Edit the `acl_file` defined above (creating the file as necessary) to contain:

```
*/admin@HOME.DEEPPORT.NET     *
```

This means that and account in the HOME.DEEPPORT.NET realm that ends with /admin has full access to administer Kerberos.

Now we need to add an admin user:

```
kadmin.local -q "addprinc admin/admin"
```

And a service principal for kadmin:

```
kadmin.local -q "addprinc -randkey kadmin/zamek.home.deepport.net"
```

Now issue a restart of the Kerberos administrative server so that it uses this principal:

```
/etc/init.d/krb5-admin-server restart
```

And then we can add regular users:

```
kadmin.local -q "addprinc <username>"
```

We need to add a host principal for the server on the KDC:

```
kadmin.local -q "addprinc -randkey host/zamek.home.deepport.net"
```

And get the keys for the server into the `/etc/krb5.keytab` file:

```
kadmin.local -q "ktadd -k /etc/krb5.keytab host/zamek.home.deepport.net"
```

Now create a service principal for slapd and create a keytab that is can access:

```
kadmin.local -q "addprinc -randkey ldap/zamek.home.deepport.net"
kadmin.local -q "ktadd -k /etc/ldap/krb5.keytab.ldap ldap/zamek.home.deepport.
chgrp openldap /etc/ldap/krb5.keytab.ldap
chmod g+r /etc/ldap/krb5.keytab.ldap
```

Now issue a restart of the KDC to make sure everything is running smoothly:

```
/etc/init.d/krb5-kdc restart
```

Finally do test it using kinit and a user created above:

```
kinit <username>
```

## NTP

Kerberos relies heavily on accurate time so install an NTP server.

```
aptitude install ntp
```

The defaults are generally acceptable though you may want to edit `/etc/ntp.conf` and adjust the server line(s) to something geographically closer. For example in Australia you might use:

```
server 0.au.pool.ntp.org
server 1.au.pool.ntp.org
server 2.au.pool.ntp.org
```

Then you will need to restart it via:

```
/etc/init.d/ntp restart
```

## OpenLDAP

Reference: https://help.ubuntu.com/8.10/serverguide/C/openldap-server.html

Install the OpenLDAP server (slapd) and the LDAP utils:

```
aptitude install slapd ldap-utils
```

On Ubuntu 8.10 this installed the following:

```
ldap-utils
libdb4.2
odbcinst1debian1
slapd
unixodbc
libsasl2-modules-gssapi-mit
```

If you are not prompted to setup the LDAP database the run:

```
dpkg-reconfigure slapd
```

Modify the BASE line of /etc/ldap/ldap.conf as appropriate for your domain:

```
BASE    dc=home,dc=deepport,dc=org
```

## Schema Changes

Copy /etc/openldap/schema/apple.schema and /etc/openldap/schema/samba.schema from an OS X computer to /etc/ldap/schema/ on the Linux server.

Some changes need to be made to the apple.schema. The definition of the 'authAuthority' attribute has to be uncommented and moved up in the file to before any reference to it. I moved it and some related lines to just before the 'apple-user' object class and it now looks like this:

```
#
# Authentication authority attribute 1.3.6.1.4.1.63.1000.1.1.2.16.1
#
attributetype (
        1.3.6.1.4.1.63.1000.1.1.2.16.1
        NAME 'authAuthority'
        DESC 'password server authentication authority'
        EQUALITY caseExactIA5Match
        SUBSTR caseExactIA5SubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

#attributetype (
#       1.3.6.1.4.1.63.1000.1.1.2.16.2
#       NAME ( 'authAuthority' 'authAuthority2' )
#       DESC 'password server authentication authority'
#       EQUALITY caseExactMatch
#       SUBSTR caseExactSubstringsMatch
#       SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

The 'container' definition also needs to be uncommented:

```
#
# Container structural object class.
#
objectclass (
        1.2.840.113556.1.3.23
        NAME 'container'
        SUP top
        STRUCTURAL
        MUST ( cn ) )
```

If you have a /etc/ldap/slapd.conf file add the following lines to it:

```
include         /etc/ldap/schema/samba.schema
include         /etc/ldap/schema/apple.schema
```

Otherwise you are using cn=config and will need to create ldif files. While this is more complicated it does have the benefit that the schema is stored in the directory itself and can therefore be replicated or backed up more easily. Also since this is the direction that OpenLDAP is moving it is better to go this way now rather than be forced to do it during an upgrade later.

1. Create /tmp/schema_convert.conf with the following contents:

   ```
   include /etc/ldap/schema/core.schema
   include /etc/ldap/schema/cosine.schema
   include /etc/ldap/schema/inetorgperson.schema
   include /etc/ldap/schema/nis.schema
   include /etc/ldap/schema/samba.schema
   include /etc/ldap/schema/apple.schema
   ```

2. Create the directory /tmp/ldif_output/
3. Run the following command:

   ```
   slaptest -f /tmp/schema_convert.conf -F /tmp/ldif_output/
   ```

4. Copy the two new ldif files to the cn=schema directory:

   ```
   cp /tmp/ldif_output/cn=config/cn=schema/cn={4}samba.ldif /etc/ldap/slapd.c
   cp /tmp/ldif_output/cn=config/cn=schema/cn={5}apple.ldif /etc/ldap/slapd.c
   ```

5. Correct the permissions on the ldif files:

```
chown openldap:openldap /etc/ldap/slapd.d/cn=config/cn=schema/*
chmod 640 /etc/ldap/slapd.d/cn=config/cn=schema/*
```

An `ls -al` on the schema directory should then look something like this:

```
ls -al /etc/ldap/slapd.d/cn=config/cn=schema/
total 80
drwxr-x--- 2 openldap openldap   150 2009-01-26 22:21 .
drwxr-x--- 3 openldap openldap  4096 2009-01-26 14:00 ..
-rw-r----- 1 openldap openldap 15456 2009-01-26 14:00 cn={0}core.ldif
-rw-r----- 1 openldap openldap 11290 2009-01-26 14:00 cn={1}cosine.ldif
-rw-r----- 1 openldap openldap  6420 2009-01-26 14:00 cn={2}nis.ldif
-rw-r----- 1 openldap openldap  2784 2009-01-26 14:00 cn={3}inetorgperson.ldif
-rw-r----- 1 openldap openldap  4233 2009-01-26 22:21 cn={4}samba.ldif
-rw-r----- 1 openldap openldap 28273 2009-01-26 22:21 cn={5}apple.ldif
```

Edit the `/etc/default/slapd` file and uncomment the `export KRB5_KTNAME` line and change it to use the keytab for the ldap service created earlier. If you are going to do some testing of slapd from the command line make sure you export KRB5_KTNAME first so that it is looking for the correct keytab.

```
export KRB5_KTNAME=/etc/ldap/krb5.keytab.ldap
```

## Testing

Stop any running copies of slapd and then run the following command:

```
slapd -d 1 -g openldap -u openldap -F /etc/ldap/slapd.d/
```

If everything is running correctly it should now be sitting waiting for queries. If there were any config errors it will leave you back at a command prompt.

From another terminal run `kinit` to authenticate against kerberos and the run `ldapsearch`. If you get errors like:

```
SASL/GSSAPI authentication started
ldap_sasl_interactive_bind_s: Other (e.g., implementation specific) error (80)
```

and slapd report “Permission denied” it will probably indicate a problem accessing the `/etc/ldap/krb5.keytab.ldap` file. Check the permissions on it are appropriate and that `KRB5_KTNAME` is being exported correctly. If `KRB5_KTNAME` isn’t set correctly it will default to using `/etc/krb5.keytab` which will not be readable by the openldap user by default and denied access to by apparmor.

## SASL mappings

Add the following lines to the end of the `/etc/ldap/slapd.d/cn=config.ldif` file:

```
olcAuthzRegexp: uid=host/([^,]*),cn=.*,cn=gssapi,cn=auth "uid=$1,cn=computers,
olcAuthzRegexp: uid=([^/]*)(/[^,]*|),cn=.*,cn=.*,cn=auth "uid=$1,cn=users,dc=h
olcAuthzRegexp: uid=([^/]*)(/[^,]*|),cn=.*,cn=auth "uid=$1,cn=users,dc=home,dc
```

## Directory Administrator account

Now we need to grant write access to the LDAP directory from a Kerberos account. Create the `diradmin` user:

```
kadmin.local -q "addprinc diradmin/admin"
```

Then edit `/etc/ldap/slapd.d/cn=config/olcDatabase={1}hdb.ldif` and change the lines:

```
olcAccess: {0}to attrs=userPassword,shadowLastChange by dn="cn=admin,dc=home,d
olcAccess: {1}to dn.base="" by * read
olcAccess: {2}to * by dn="cn=admin,dc=home,dc=deepport,dc=net" write by * read
```

to

```
olcAccess: {0}to attrs=userPassword,shadowLastChange by dn="cn=admin,dc=home,d
olcAccess: {1}to dn.base="" by * read
olcAccess: {2}to * by dn="cn=admin,dc=home,dc=deepport,dc=net" write by dn="ui
```

You can now start slapd and everything should be working (try an `ldapsearch` again just to be sure):

```
/etc/init.d/slapd start
```

## LDAP Structure Setup

You will need to import a basic structure into LDAP. Create a file called `OD-structure.ldif` with the following content:

```
dn: cn=users,dc=home,dc=deepport,dc=net
cn: users
objectClass: container

dn: cn=groups,dc=home,dc=deepport,dc=net
cn: groups
objectClass: container

dn: cn=mounts,dc=home,dc=deepport,dc=net
cn: mounts
objectClass: container

dn: cn=accesscontrols,dc=home,dc=deepport,dc=net
cn: accesscontrols
objectClass: container

dn: cn=certificateauthorities,dc=home,dc=deepport,dc=net
cn: certificateauthorities
objectClass: container

dn: cn=computers,dc=home,dc=deepport,dc=net
cn: computers
objectClass: container

dn: cn=computer_groups,dc=home,dc=deepport,dc=net
cn: computer_groups
objectClass: container

dn: cn=computer_lists,dc=home,dc=deepport,dc=net
cn: computer_lists
objectClass: container

dn: cn=config,dc=home,dc=deepport,dc=net
cn: config
objectClass: container

dn: cn=locations,dc=home,dc=deepport,dc=net
cn: locations
objectClass: container

dn: cn=machines,dc=home,dc=deepport,dc=net
cn: machines
objectClass: container

dn: cn=neighborhoods,dc=home,dc=deepport,dc=net
cn: neighborhoods
objectClass: container

dn: cn=people,dc=home,dc=deepport,dc=net
cn: people
objectClass: container

dn: cn=presets_computer_lists,dc=home,dc=deepport,dc=net
cn: presets_computer_lists
objectClass: container

dn: cn=presets_groups,dc=home,dc=deepport,dc=net
cn: presets_groups
objectClass: container

dn: cn=presets_users,dc=home,dc=deepport,dc=net
cn: presets_users
objectClass: container

dn: cn=printers,dc=home,dc=deepport,dc=net
cn: printers
objectClass: container

dn: cn=augments,dc=home,dc=deepport,dc=net
cn: augments
objectClass: container

dn: cn=autoserversetup,dc=home,dc=deepport,dc=net
cn: autoserversetup
objectClass: container

dn: cn=filemakerservers,dc=home,dc=deepport,dc=net
cn: filemakerservers
objectClass: container

dn: cn=resources,dc=home,dc=deepport,dc=net
cn: resources
objectClass: container
```

```
dn: cn=places,dc=home,dc=deepport,dc=net
cn: places
objectClass: container

dn: cn=maps,dc=home,dc=deepport,dc=net
cn: maps
objectClass: container

dn: cn=presets_computers,dc=home,dc=deepport,dc=net
cn: presets_computers
objectClass: container

dn: cn=presets_computer_groups,dc=home,dc=deepport,dc=net
cn: presets_computer_groups
objectClass: container

dn: cn=automountMap,dc=home,dc=deepport,dc=net
cn: automountMap
objectClass: container

dn: ou=macosxodconfig,cn=config,dc=home,dc=deepport,dc=net
ou: macosxodconfig
objectClass: top
objectClass: organizationalUnit

dn: cn=CollabServices,cn=config,dc=home,dc=deepport,dc=net
cn: CollabServices
objectClass: apple-configuration
objectClass: top
```

Then add it to your directory with the following command (making sure you are authenticated using `kinit diradmin/admin`):

```
ldapadd -f OD-structure.ldif
```

## Home Directory Setup

You will need to add a mount point mapping to you directory by hand since this isn't configured through Workgroup Manager but rather Server Admin (which won't work without an OS X Server. Create a file called `mounts.ldif` that looks like this:

```
dn: cn=zamek:/Users,cn=mounts,dc=home,dc=deepport,dc=net
cn: zamek:/Users
mountDirectory: /Network/Servers
mountOption: net
mountType: nfs
objectClass: mount
objectClass: top
```

Then add it to your directory with the following command (making sure you are authenticated using `kinit diradmin/admin`):

```
ldapadd -f mounts.ldif
```

## NFS

Install the NFS server:

```
aptitude install nfs-kernel-server
```

Create a folder for your users OS X home directories (I use `/Users` as this is the Mac convention):

```
mkdir /Users
```

Edit `/etc/exports`:

```
/Users          *.home.deepport.net(rw,sync,insecure,no_subtree_check) 192.168
```

or the following if the users are going to need to be able to write to their home directories as root (e.g. MacPorts needs to be able to write to ~/.macports/ after a sudo):

```
/Users          *.home.deepport.net(rw,sync,insecure,no_subtree_check,no_root_
```

Then run `exportfs -av` to update the exports.

## User Account Creation

After you have configured a OS X client as per the next step you can then use Workgroup Manager on that computer to add accounts when logged in as `diradmin`.

Some manual steps are still required though:

- Create the users home directory on the server.
- Change the ownership on the home directory to the uid and gid listed in workgroup manager.
- Create a Kerberos user matching the LDAP cn and assign a password. Any password entered via Workgroup Manager should be ignored.

> I aim to remove the need for these manual steps if possible over time.
>
> Any feedback or ideas is welcome.

# Mac OS X Client Setup

All the following is performed on the Mac unless otherwise specified.

> In early testing I had an issue that seemed to be corrected by this step but I now believe it isn't needed.

## Domain Name

Run the `hostname` command to confirm that it returns a FQDN like `Andrews-MBP.home.deepport.net` and not just `Andrews-MBP`.

If you do only get the hostname component without the domain then run the following command:

```
hostname Andrews-MBP.home.deepport.net
```

And then edit your `/etc/hostconfig` file to include a line with your hostname as a FQDN:

```
HOSTNAME=Andrews-MBP.home.deepport.net
```

## Kerberos

Copy the contents of the `/etc/krb5.conf` file from the server to `/Library/Preferences/edu.mit.Kerberos` on the Mac (creating the file if necessary).

> I would like to remove the need for this step and have it configured by Directory Utility or have the machine retrieve the settings via LDAP or DNS.

> This may be needed if I can work out a way of getting the OS to bind to the directory but for now it doesn't seem to be required.

Then we can the add the host key for the OS X machine to it's keytab:

```
kinit admin/admin
kadmin -q "ktadd -k /etc/krb5.keytab host/Andrews-MBP.home.deepport.net"
```

## Testing

Test it using kinit and a user you have created on the server (kdestroy will effectively any existing Kerberos user):

```
kdestroy
kinit <username>
```

We should now be able to try a ldapsearch:

```
ldapsearch -h zamek
```

## Directory Access

You will now have to add the directory server to to you OS X client for authentication.

Open `Directory Utility` from the `Utilities` folder under `Applications`.

Click on `Show Advanced Settings` if necessary and click on the lock to unlock changes also if necessary.

Click on the `Services` button and select `LDAPv3` from the list. Then click on the pencil to to configure the LDAP service.

Click on the new button.

Enter the server name and click `Continue`.

Change the template type to `Open Directory Server` and make sure the searchbase is appropriate (in the examples it would be `dc=home,dc=deepport,dc=net`). Click on `Continue` and `OK`.

Pick a name for your configuration (the directory domain name would be appropriate. e.g. home.deepport.net).

Click OK, authenticate as necessary and you're done.

You should now be able to login with a user in the LDAP directory with a Kerberos password and a NFS home directory.

## Post login Kerberos ticket *(optional)*

It is possible to make OS X have a ticket available to you after login (*why isn't this the default Apple?*) so that other software can make use of it (like SSH described below). This requires editing `/etc/authorization` so care is required to make sure you don't lock yourself out of the computer. I suggest making this change via SSH from another computer so that you can revert it in the event you make a mistake.

The following is from 10.5.6 and the location within the file might be slightly different on other version. It should apply equally on 10.4 systems. In both cases below I am replacing the line

```
<string>builtin:authenticate,privileged</string>
```

with

```
<string>builtin:krb5authnoverify,privileged</string>
```

Change line 566 to make the section look like this:

```
                <key>system.login.console</key>
                <dict>
                        <key>class</key>
                        <string>evaluate-mechanisms</string>
                        <key>comment</key>
                        <string>Login mechanism based rule.  Not for general u
                        <key>mechanisms</key>
                        <array>
                                <string>builtin:smartcard-sniffer,privileged</
                                <string>loginwindow:login</string>
                                <string>builtin:reset-password,privileged</str
                                <string>builtin:auto-login,privileged</string>
                                <string>builtin:krb5authnoverify,privileged</string>
                                <string>HomeDirMechanism:login,privileged</str
                                <string>HomeDirMechanism:status</string>
                                <string>MCXMechanism:login</string>
                                <string>loginwindow:success</string>
                                <string>loginwindow:done</string>
                        </array>
                </dict>
```

Change line 796 to make the section look like this:

```
                <key>authenticate</key>
                <dict>
                        <key>class</key>
                        <string>evaluate-mechanisms</string>
                        <key>mechanisms</key>
                        <array>
                                <string>builtin:smartcard-sniffer,privileged</
                                <string>builtin:authenticate</string>
                                <string>builtin:krb5authnoverify,privileged</string>
                        </array>
                </dict>
```

## Testing

After logging in (no reboot required) open a terminal and run `klist`. You should have a ticket listed something like this:

```
Kerberos 5 ticket cache: 'API:Initial default ccache'
Default principal: andrew@HOME.DEEPPORT.NET

Valid Starting     Expires            Service Principal
04/10/09 11:41:29  04/11/09 11:41:21  krbtgt/HOME.DEEPPORT.NET@HOME.DEEPPORT.N
```

# Other Software that can use Kerberos / LDAP

## Netatalk

Note 1: Netatalk does not have SSL support builtin by default. This is due to the longstanding problem with OpenSSL's licensing not being GPL compatible (*will they ever sort this out*). You will need to make some changes to get this working.

Note 2: This requires Post login Kerberos ticket as described in the client config above to work.

### Creating a Netatalk package with SSL support

This is very well described here: How to: Install Netatalk (AFP) on Ubuntu with Encrypted Authentication. Thanks Damon!

So this is gratuitously copied from there since it Just Works™:

```
$ sudo aptitude update
$ mkdir -p ~/src/netatalk
$ cd ~/src/netatalk
$ sudo aptitude install cracklib2-dev libssl-dev
$ apt-get source netatalk
$ sudo apt-get build-dep netatalk
$ cd netatalk-2.0.3
$ sudo DEB_BUILD_OPTIONS=ssl dpkg-buildpackage -us -uc
$ sudo debi
$ echo "netatalk hold" | sudo dpkg --set-selections
```

### Configuring Netatalk

Create a service principal for afpserver:

```
kadmin.local -q "addprinc -randkey afpserver/zamek.home.deepport.net"
kadmin.local -q "ktadd -k /etc/netatalk/krb5.keytab afpserver/zamek.home.deepp
```

Edit the `/etc/netatalk/afpd.conf` file as follow:

```
- -tcp -uamlist uams_gss.so -k5service afpserver -k5realm HOME.DEEPPORT.NET -k
```

These options are:

| | |
|---|---|
| `-tcp` | Use TCP only without AppleTalk. |
| `-uamlist uams_gss.so` | Use GSS (Kerberos). |
| `-k5service afpserver` | Use the afpserver principal created above. |
| `-k5realm HOME.DEEPPORT.NET` | Your Kerberos realm. |
| `-k5keytab /etc/netatalk/krb5.keytab` | The keytab file created above. |
| `-fqdn zamek.home.deepport.net:548` | The FQDN of your server and the port that AFP is running on. |

## Dovecot IMAP server

The basic steps for this come from: http://wiki.dovecot.org/Authentication/Kerberos and http://wiki.dovecot.org/UserIds

Create a service principal for imap:

```
kadmin.local -q "addprinc -randkey imap/zamek.home.deepport.net"
kadmin.local -q "ktadd -k /etc/dovecot/krb5.keytab imap/zamek.home.deepport.ne
```

In it's simplest form edit `/etc/dovecot/dovecot.conf` and find the `auth default` section. Look through the section and edit it to leave this configuration:

```
auth default {
  mechanisms = gssapi
  userdb static {
    args = uid=65534 gid=8 home=/var/mail/%u
  }
}
```

Note: the uid here is for `nobody` and the gid is for `mail` which is appropriate for Ubuntu 8.10 use of the existing `/var/mail/` directory.

Also uncomment and edit the line defining the keytab location:

```
auth_krb5_keytab = /etc/dovecot/krb5.keytab
```

and the mail location:

```
mail_location = maildir:/var/mail/%u
```

# Linux PAM authentication

On your Linux boxes you might also want to look at the following:
`libpam-ldap` to enable lookup of user login details from LDAP.
`libpam-krb5` to enable Kerberos authentication for user logins.

# OpenSSH

This has got to be the easiest software to setup for use with Kerberos authentication!

## Server side

Firstly, in your `/etc/ssh/sshd_config` file on the server uncomment the `GSSAPIAuthentication` line and set the value to `yes`. It should look like the following:

```
# GSSAPI options
GSSAPIAuthentication yes
GSSAPICleanupCredentials yes      # This is the default but I like to be expli
```

Note that the Kerberos lines in `/etc/ssh/sshd_config` refer to using Kerberos to authenticate a supplied password which is not what we really want to achieve here.

Don't forget to restart sshd with `/etc/init.d/ssh restart`.

## Client side

This can be set in a few different ways.

1. On the command line:

   ```
   ssh server -K
   ```

2. System-wide in your `/etc/ssh/ssh_config` file by adding the following:

   ```
   Host *
           GSSAPIAuthentication yes
   ```

3. For all connections for a user in their `~/.ssh/config` file by adding:

   ```
   Host *
           GSSAPIAuthentication yes
   ```

4. On a host by host basis for a user in their `~/.ssh/config` file by adding:

   ```
   Host servername.or.ip.address
           GSSAPIAuthentication yes
   ```

Be careful when testing that you aren't in fact using public key authentication. If you have that already setup I would recommend temporarily renaming your `~/.ssh/id_rsa` or `~/.ssh/id_dsa` file, as ssh will helpfully and silently (without -v being set) fall back to using it.

And that's it! Now if only all the other stuff was this easy.

## Apache

Create a service principal for HTTP and give the apache service (www-data) read access:

```
kadmin.local -q "addprinc -randkey HTTP/zamek.home.deepport.net"
kadmin.local -q "ktadd -k /etc/netatalk/krb5.keytab HTTP/zamek.home.deepport.n
chmod 640 /etc/apache2/krb5.keytab
chgrp www-data /etc/apache2/krb5.keytab
```

Install `libapache2-mod-auth-kerb`:

```
sudo aptitude install libapache2-mod-auth-kerb
```

I chose to put the config in `/etc/apache2/sites-enabled/000-default` file and have it specified for the root directory so that it covers every request to the default site on the server. You will need to decide on what suits you best. The code I used is as follows:

```
        <Directory />
                Options FollowSymLinks
                AllowOverride None
                AuthType                Kerberos
                AuthName                "Kerberos logins only"
                KrbMethodNegotiate      on
                KrbMethodK5Passwd       off
                KrbAuthRealms           HOME.DEEPPORT.NET
                KrbServiceName          HTTP/zamek.home.deepport.net@HOME.DEEP
                Krb5Keytab              /etc/apache2/krb5.keytab
                Require                 valid-user
        </Directory>
```

These options are:

| | |
|---|---|
| `AuthType Kerberos` | Use Kerberos for authentication |
| `KrbMethodNegotiate on` | Enable the negotiate method for authentication |
| `KrbMethodK5Passwd off` | Disable Basic authentication allowing Kerberos tickets only. If you would like to allow your users to also be able to authenticate with a simple username and password dialog box set the to on. |
| `KrbAuthRealms HOME.DEEPPORT.NET` | Your Kerberos realm. |
| `KrbServiceName HTTP/zamek.home.deepport.net@HOME.DEEPPORT.NET` | This is not strictly necessary unless you used a different service principal name to the default HTTP (note the uppercase) but I like to define it for completeness. |
| `Krb5Keytab /etc/apache2/krb5.keytab` | The keytab file created above. |

A simple restart or reload of Apache and it should work.

## Notes

- You must use the same address in the URL as in the service principal. I.e. It would work in the example above with zamek.home.deepport.net but not for zamek. If you have `KrbMethodK5Passwd` set to on then it can fall back to requesting a username and password if a different server name is used.
- Not all browsers support the Negotiate authentication method. The good news is that Safari does, Internet Explorer should and there is a plugin for Firefox.

## Radius

Note: FreeRadius does not have SSL support builtin by default. This is due to the longstanding problem with OpenSSLs licensing not being GPL compatible (*will they ever sort this out*). You will need to make some changes to get this working and I will add the documentation for this here when I do a rebuild.

Create a service principal for freeradius:

```
kadmin.local -q "addprinc -randkey radius/zamek.home.deepport.net"
kadmin.local -q "ktadd -k /etc/freeradius/krb5.keytab.radius radius/zamek.home
chgrp freerad /etc/freeradius/krb5.keytab.radius
chmod g+r /etc/freeradius/krb5.keytab.radius
```

Edit the `/etc/freeradius/modules/krb5` file as follow:

```
krb5 {
        keytab = /etc/freeradius/krb5.keytab.radius
        service_principal = radius/zamek.home.deepport.net
}
```

Radius will need a clear text password to pass to kerberos for authenticating. This will limit the choices available for authentication methods. In particular this rules out any of the CHAP variants which *may* cause some incompatibilities with some clients (most likely Windows, though I haven't tested this at this point in time).

Example `/etc/freeradius/eap.conf`:

```
        eap {
                default_eap_type = peap
                timer_expire     = 60
                ignore_unknown_eap_types = no
                cisco_accounting_username_bug = no
                max_sessions = 2048

#               md5 {
#               }
#               leap {
#               }
                gtc {
                        auth_type = PAP
                }
                tls {
                        certdir = ${confdir}/certs
                        cadir = ${confdir}/certs
                        private_key_password = ThisIsAVerySillyPassword
                        private_key_file = ${certdir}/server.pem
                        certificate_file = ${certdir}/server.pem
                        CA_file = ${cadir}/ca.pem
                        dh_file = ${certdir}/dh
                        random_file = ${certdir}/random
                        cipher_list = "DEFAULT"

                        cache {
                              enable = no
                              lifetime = 24 # hours
                              max_entries = 255
                        }
                }

                ttls {
                        #default_eap_type = md5
                        default_eap_type = gtc
                        copy_request_to_tunnel = no
                        use_tunneled_reply = no
                        virtual_server = "inner-tunnel"
                }

                peap {
                        default_eap_type = gtc
                        copy_request_to_tunnel = no
                        use_tunneled_reply = no
                #       proxy_tunneled_request_as_eap = yes
                #       virtual_server = "inner-tunnel"
                }

#               mschapv2 {
#               }
        }
```

The important lines here are:

Set PEAP as the default time since we can make it send a clear password:

```
                default_eap_type = peap
```

Set GTC to use PAP (which is clear text):

```
                gtc {
                        auth_type = PAP
                }
```

Now set PEAP to use the GTC config defined above:

```
                peap {
                        default_eap_type = gtc
```

These lines in `/etc/freeradius/sites-enabled/inner-tunnel` then cause any PAP authentication requests to make use of kerberos:

```
authenticate {
        Auth-Type PAP {
#               pap
                krb5
        }
```

Tags: Linux, OS X

Posted in IT
28 Comments Posted by Andrew

## 28 Responses to "Setting up a Linux server for OS X clients"

led_belly
says:

June 11, 2009 at 3:14 pm Reply

I am only at the SASL stage of this document but up to this point I can say it? been the best HOWTO Ive come across in a while. Thanks!

Andrew
says:

June 11, 2009 at 7:57 pm Reply

Thanks for the comment. Please let me know if there is anything that needs clearing up, correcting or improving.

frogstar_robot
says:

June 11, 2009 at 11:59 pm Reply

Netatalk has been in active development lately and has been changed to use GNU TLS for SSL. Version 2.0.4 supports this. 2.0.4rc2 is currently in Sid and this feature works well. I haven't had any issues with a backport of the Sid package I built on Lenny. YMMV.

The next release of Debian will support SSL and the DHX uams out of the box for Netatalk.

The CVS for Netatalk has support for the AFP calls Time Machine needs as well.

Andrew
says:

June 12, 2009 at 9:44 am Reply

Thanks for the info.

I am hoping to put together yet another linux box as soon as I get a few more bits of spare hardware and update everything for Ubuntu 9.04. I see that it currently has Netatalk 2.0.4~beta2-5ubuntu1 in it. Do you know at what point it changed to using GNU TLS?

frogstar_rob
says:

December 18, 2009 at 6:43 am Reply

It's been sometime from the late 08 to mid 09 time frame. The netatalk on Sourceforge acquired some new maintainers and contributors. The package now in Sid now supports Time Machine as well. This is not enabled by default but is set by setting the options:tm on a share in AppleVolumes.default.

I'm not sure what the state of it is in Ubuntu but it may not happen until the distro after Lucid. It just depends how much they take out the Sid in question they branch from. I'll note that the Debian source package will probably build just fine. I'm running a ported build of Debian's 2.0.5-2 netatalk on Karmic.

That dance works like this and I'll assume the Debian/ubuntu package building stuff in installed:

Add the Debian src line to your apt sources eg:

deb-src http://ftp.debian.org/debian/ unstable main contrib non-free

Make a directory to build the packages in:

```
mkdir netatalk; cd netatalk
#apt-get update
apt-get source libdb4.8-dev (Netatalk 2.0.5 requires it for cnid
databases)
#apt-get build-dep libdb4.8-dev
cd db-4.8.24
fakeroot dpkg-buildpackage -b
cd ../
dpkg -i libdb4.8_4.8.24-1_i386.deb db4.8-util libdb4.8-dev
apt-get source netatalk
#apt-get build-dep netatalk
cd netatalk-2.0.5
fakeroot dpkg-buildpackage -b
```

You'll then have a netatalk package ../ that is appropriate to the Ubuntu flavor your running. I generally find this procedure vastly preferable to pinning packages from foreign or newer/older distros.

led_belly says:

June 12, 2009 at 6:27 am Reply

Hello Again,

My knowledge of these technologies is limited as this is my first attempt at setting up network authentication. This article has been very helpful except for a few problems I've encountered and questions that I have:

1) The tying together of Kerberos and OpenLDAP is still a little vague to me. Specifically, "Create a Kerberos user matching the LDAP cn and assign a password. Any password entered via Workgroup Manager should be
ignored." How is this performed with LDAP? It looks like I need to add cn=users, cn=mount, etc. to LDAP..

2) Does OpenLDAP require SSL/TLS to be present? I have seen postings in forums saying that some lines be entered into the /etc/default/slapd file… I'm not sure if SASL/Kerberos are talking to LDAP (is this how it works?)

3) Perhaps this article could highlight the importance that realms be entered in capital letters. This messed me up the first few times.

4) Home Directory setup… You have the following for the mount.ldif file:

```
dn: cn=zamek:/Users,cn=mounts,dc=home,dc=deepport,dc=net
cn: zamek:/Users
mountDirectory: /Network/Servers
mountOption: net
mountType: nfs
objectClass: mount
objectClass: top
```

cn=zamek here seems to point back to [realms] in the /etc/krb5.conf file. I did not use this particular design of zamek.home.domain.com. Instead I used kerberos.domain.com for kdc and admin_server entries. Can you clarify what's happening here for me?

This also pertains to my next question…

5) kadmin -q "ktadd -k /etc/krb5.keytab host/kerberos.domain" on the Mac gives an error: kadmin: Operation requires "change-password" privilege while changing host/kerberos.domain.com's key

Thanks so much!

Andrew says:

June 12, 2009 at 9:50 am Reply

This is useful feedback and I'll try to look into it over the weekend.

I should probably also put a table of some of the conventions I've used and important points (such as capitals for the Kerberos realms) at the top of the document.

led_belly says:

June 14, 2009 at 8:28 am Reply

I have realized that zamek is a host on the network:

So for me its:

```
kadmin.local -q "addprinc -randkey host/kalypso.cloud.domain.net"
```

Then run as root on the Mac (so the keytab is writable):

```
kadmin -q "ktadd -k /etc/krb5.keytab host/kalypso.cloud.domain.net"
```

This gets rid of the change-password problem.

Also, I am writing an extended howto on these topics and other related ones that will have verbose explanations, screen shots, program output and possibly a pre-configured virtual machine etc. is it possible to gain permission repost some of the information here on my site? Mostly just the sequence of commands. Full credit will be given… Please let me know. Thanks

Cheers!

Andrew says:

June 14, 2009 at 10:00 pm Reply

By all means repost the info.

I'd mainly ask that you feedback to me any corrections, extra information, etc so that I can continue to improve this document.

I also didn't get a chance to do any more work on it this weekend though I did download the server ISO for Ubuntu 9.04. I hope to set it up in a VM in the next week or so and replicate these steps onto it, correcting and adding to this documentation as I go.

I would love to be able to work towards a functional OS X Server alternative for somewhere around Ubuntu 10.04 too (since it will also be an LTS release). Maybe this document and your extended howto could become the basis for producing this.

led_belly says:

June 15, 2009 at 3:07 am Reply

There seems to be a problem with SASL. Here is some debug info when I try to do an ldapadd:

```
conn=49 op=1 BIND dn="" method=163
conn=49 op=1 RESULT tag=97 err=14 text=SASL(0): successful result: security flags do not match required
conn=49 op=2 BIND dn="" method=163
SASL [conn=49] Error: unable to open Berkeley db /etc/sasldb2: Permission denied
SASL [conn=49] Error: unable to open Berkeley db /etc/sasldb2: Permission denied
SASL [conn=49] Error: unable to open Berkeley db /etc/sasldb2: Permission denied
SASL [conn=49] Error: unable to open Berkeley db /etc/sasldb2: Permission denied
SASL [conn=49] Failure: no secret in database
conn=49 op=2 RESULT tag=97 err=49 text=SASL(-13): user not found:
no secret in database
```

Do I need to use saslpasswd to add passwords for users? If I follow what? happening here, with every new user I would need to add them to kerberos, sasl AND LDAP. Superfluous?

Perhaps, also, the following LDAP entries are required:

cn=gssapi
cn=auth
cn=computers

But with which objects and attributes?

Am I confused or is an ldif file containing all these entries integral and missing from this document?

led_belly
says:

June 17, 2009 at 2:09 pm Reply

http://markmail.org/message/spualh7qhvpxbvv7

led_belly
says:

June 12, 2009 at 7:48 am Reply

Some more information might be helpful… Here is my configuration at this point:

```
kadmin.local: list_principals
K/M@CLOUD.DOMAIN.NET
admin/admin@CLOUD.DOMAIN.NET
admin@CLOUD.DOMAIN.NET
diradmin/admin@CLOUD.DOMAIN.NET
diradmin@CLOUD.DOMAIN.NET
host/kerberos.DOMAIN.net@CLOUD.DOMAIN.NET
jacobc@CLOUD.DOMAIN.NET
kadmin/admin@CLOUD.DOMAIN.NET
kadmin/blake@CLOUD.DOMAIN.NET
kadmin/changepw@CLOUD.DOMAIN.NET
kadmin/history@CLOUD.DOMAIN.NET
kadmin/kerberos.DOMAIN.net@CLOUD.DOMAIN.NET
krbtgt/CLOUD.DOMAIN.NET@CLOUD.DOMAIN.NET
ldap/kerberos.DOMAIN.net@CLOUD.DOMAIN.NET
```

——————-

```
root@blake:~# ldapsearch -H ldap://auth.domain.net/ -b
dc=cloud,dc=domain,dc=net -x
# extended LDIF
#
# LDAPv3
# base with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# cloud.domain.net
dn: dc=cloud,dc=domain,dc=net
objectClass: top
objectClass: dcObject
objectClass: organization
o: domain
dc: cloud

# admin, cloud.domain.net
dn: cn=admin,dc=cloud,dc=domain,dc=net
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
```

```
description: LDAP administrator

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2

———

root@blake:/etc/krb5kdc# cat ../krb5.conf
[libdefaults]
default_realm = CLOUD.DOMAIN.NET

# The following krb5.conf variables are only for MIT Kerberos.
krb4_config = /etc/krb.conf
krb4_realms = /etc/krb.realms
kdc_timesync = 1
ccache_type = 4
forwardable = true
proxiable = true

# The following encryption type specification will be used by MIT
Kerberos
# if uncommented. In general, the defaults in the MIT Kerberos code
are
# correct and overriding these specifications only serves to disable new
# encryption types as they are added, creating interoperability
problems.
#
# Thie only time when you might need to uncomment these lines and
change
# the enctypes is if you have local software that will break on ticket
# caches containing ticket encryption types it doesn't know about (such
as
# old versions of Sun Java).

# default_tgs_enctypes = des3-hmac-sha1
# default_tkt_enctypes = des3-hmac-sha1
# permitted_enctypes = des3-hmac-sha1

# The following libdefaults parameters are only for Heimdal Kerberos.
v4_instance_resolve = false
v4_name_convert = {
host = {
rcmd = host
ftp = ftp
}
plain = {
something = something-else
}
}
fcc-mit-ticketflags = true

[realms]
CLOUD.DOMAIN.NET = {
kdc = kerberos.domain.net:88
admin_server = kerberos.domain.net
default_domain = cloud.domain.net
}

[domain_realm]
.cloud.domain.net = CLOUD.DOMAIN.NET
cloud.domain.net = CLOUD.DOMAIN.NET

[login]
krb4_convert = true
krb4_get_tickets = false
```

led_belly
says:

June 12, 2009 at 12:30 pm Reply

I think what is missing is an ldif file:

```
root@blake:~# ldapadd -x -f domain.ldif -D
“cn=admin,dc=cloud,dc=domain,dc=net” -w secret
adding new entry “ou=users,dc=cloud,dc=domain,dc=net”

adding new entry “ou=groups,dc=cloud,dc=domain,dc=net”

…
```

domain.ldif
———

```
dn: ou=users,dc=cloud,dc=domain,dc=net
objectClass: organizationalUnit
ou: users

dn: ou=groups,dc=cloud,dc=domain,dc=net
objectClass: organizationalUnit
ou: groups
```

THESE ARE JUST TEST VALUES

What would the real values be? Am I off base?

led_belly
says:

June 13, 2009 at 4:53 am Reply

I found this file … still not sure if it? what? req. … the diradmin entry is just an example … it also doesnt create cn=users or cn=groups:

```
dn: uid=diradmin,cn=users,dc=cloud,dc=domain,dc=net
uid: diradmin
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: apple-user
objectClass: extensibleObject
objectClass: organizationalPerson
objectClass: top
objectClass: person
sn: 99
structuralObjectClass: inetOrgPerson
entryUUID: 5355461a-cb28-102b-8468-c17ea9eda937
creatorsName: uid=root,cn=users,dc=barbariangroup,dc=com
createTimestamp: 20070720161633Z
gidNumber: 20
uidNumber: 1000
loginShell: /bin/bash
homeDirectory: /Users/diradmin
cn: Directory Administrator
entryCSN: 20070720161633Z#00000e#00#000000
modifiersName: uid=diradmin,cn=users,dc=cloud,dc=domain,dc=net
modifyTimestamp: 20070720161633Z

# OSX Objects
dn: ou=macosx,dc=cloud,dc=domain,dc=net
ou: macosx
objectClass: top
objectClass: organizationalUnit
description: Holds metadata for OSX Server

# mounts, OSX Object
dn: cn=mounts,ou=macosx,dc=cloud,dc=domain,dc=net
cn: mounts
objectClass: container
objectClass: top

# accesscontrols, OSX Object
dn: cn=accesscontrols,ou=macosx,dc=cloud,dc=domain,dc=net
cn: accesscontrols
objectClass: container
```

```
# certificateauthorities, OSX Object
dn: cn=certificateauthorities,ou=macosx,dc=cloud,dc=domain,dc=net
cn: certificateauthorities
objectClass: container

# computers, OSX Object
dn: cn=computers,ou=macosx,dc=cloud,dc=domain,dc=net
cn: computers
objectClass: container

# computer_groups, OSX Objectd
dn: cn=computer_groups,ou=macosx,dc=cloud,dc=domain,dc=net
cn: computer_groups
objectClass: container

# computer_lists, OSX Object
dn: cn=computer_lists,ou=macosx,dc=cloud,dc=domain,dc=net
cn: computer_lists
objectClass: container

# config, OSX Object
dn: cn=config,ou=macosx,dc=cloud,dc=domain,dc=net
cn: config
objectClass: container

# locations, OSX Object
dn: cn=locations,ou=macosx,dc=cloud,dc=domain,dc=net
cn: locations
objectClass: container

# machines, OSX Object
dn: cn=machines,ou=macosx,dc=cloud,dc=domain,dc=net
cn: machines
objectClass: container

# neighborhoods, OSX Object
dn: cn=neighborhoods,ou=macosx,dc=cloud,dc=domain,dc=net
cn: neighborhoods
objectClass: container

# people, OSX Object
dn: cn=people,ou=macosx,dc=cloud,dc=domain,dc=net
cn: people
objectClass: container

# presets_computer_lists, OSX Object
dn: cn=presets_computer_lists,ou=macosx,dc=cloud,dc=domain,dc=net
cn: presets_computer_lists
objectClass: container

# presets_groups, OSX Object
dn: cn=presets_groups,ou=macosx,dc=cloud,dc=domain,dc=net
cn: presets_groups
objectClass: container

# presets_users, OSX Object
dn: cn=presets_users,ou=macosx,dc=cloud,dc=domain,dc=net
cn: presets_users
objectClass: container

# printers, OSX Object
dn: cn=printers,ou=macosx,dc=cloud,dc=domain,dc=net
cn: printers
objectClass: container

# augments, OSX Object
dn: cn=augments,ou=macosx,dc=cloud,dc=domain,dc=net
cn: augments
objectClass: container

# autoserversetup, OSX Object
dn: cn=autoserversetup,ou=macosx,dc=cloud,dc=domain,dc=net
cn: autoserversetup
objectClass: container

# filemakerservers, OSX Object
```

```
dn: cn=filemakerservers,ou=macosx,dc=cloud,dc=domain,dc=net
cn: filemakerservers
objectClass: container

# resources, OSX Object
dn: cn=resources,ou=macosx,dc=cloud,dc=domain,dc=net
cn: resources
objectClass: container

# places, OSX Object
dn: cn=places,ou=macosx,dc=cloud,dc=domain,dc=net
cn: places
objectClass: container

# maps, OSX Object
dn: cn=maps,ou=macosx,dc=cloud,dc=domain,dc=net
cn: maps
objectClass: container

# presets_computers, OSX Object
dn: cn=presets_computers,ou=macosx,dc=cloud,dc=domain,dc=net
cn: presets_computers
objectClass: container

# presets_computer_groups, OSX Object
dn:
cn=presets_computer_groups,ou=macosx,dc=cloud,dc=domain,dc=net
cn: presets_computer_groups
objectClass: container

# automountMap, OSX Object
dn: cn=automountMap,ou=macosx,dc=cloud,dc=domain,dc=net
cn: automountMap
objectClass: container

# macosxodconfig, config, OSX Object
dn:
ou=macosxodconfig,cn=config,ou=macosx,dc=cloud,dc=domain,dc=net
ou: macosxodconfig
objectClass: top
objectClass: organizationalUnit

# mcx_cache, config, OSX Object
dn: cn=mcx_cache,cn=config,ou=macosx,dc=cloud,dc=domain,dc=net
cn: mcx_cache
objectClass: apple-configuration
objectClass: top

# ldapreplicas, config, OSX Object
dn: cn=ldapreplicas,cn=config,ou=macosx,dc=cloud,dc=domain,dc=net
cn: ldapreplicas
objectClass: apple-configuration
objectClass: top

# passwordserver, config, OSX Object
dn:
cn=passwordserver,cn=config,ou=macosx,dc=cloud,dc=domain,dc=net
cn: passwordserver
objectClass: apple-configuration
objectClass: top

# macosxodpolicy, config, OSX Object
dn:
cn=macosxodpolicy,cn=config,ou=macosx,dc=cloud,dc=domain,dc=net
cn: macosxodpolicy
objectClass: top
objectClass: apple-configuration

# CollabServices, config, OSX Object
dn:
cn=CollabServices,cn=config,ou=macosx,dc=cloud,dc=domain,dc=net
cn: CollabServices
objectClass: apple-configuration
objectClass: top
```

```
# CIFSServer, config, OSX Object
dn: cn=CIFSServer,cn=config,ou=macosx,dc=cloud,dc=domain,dc=net
cn: CIFSServer
objectClass: apple-configuration
objectClass: top

# KerberosKDC, config, OSX Object
dn:
cn=KerberosKDC,cn=config,ou=macosx,dc=cloud,dc=domain,dc=net
cn: KerberosKDC
objectClass: apple-configuration
objectClass: top

# KerberosClient, config, OSX Object
dn:
cn=KerberosClient,cn=config,ou=macosx,dc=cloud,dc=domain,dc=net
cn: KerberosClient
objectClass: apple-configuration
objectClass: top

# Home_Dir_Items, config, OSX Object
dn:
cn=Home_Dir_Items,cn=config,ou=macosx,dc=cloud,dc=domain,dc=net
cn: Home_Dir_Items
objectClass: apple-configuration
objectClass: top

# Group_Dir_Items, config, OSX Object
dn:
cn=Group_Dir_Items,cn=config,ou=macosx,dc=cloud,dc=domain,dc=net
cn: Group_Dir_Items
objectClass: apple-configuration
objectClass: top
```

led_belly says:

June 13, 2009 at 9:04 am Reply

to close this… in the end it looks like all i needed was the diradmin and users:

```
dn: cn=users,dc=cloud,dc=domain,dc=net
objectClass: container
objectClass: top
cn: users

dn: uid=diradmin,cn=users,dc=cloud,dc=domain,dc=net
objectClass: uidObject
objectClass: top
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: apple-user
objectClass: extensibleObject
objectClass: organizationalPerson
objectClass: person
cn: Directory Administrator
gidNumber: 20
homeDirectory: 99
sn: 99
uid: diradmin
uidNumber: 1000
userPassword::
```

Joe M says:

October 9, 2009 at 2:24 am Reply

Great tutorial!
We have an OpenLDAP env at my office and I've spent the better part

of a week doing pretty much what you've documented here.
Snow Leopard server was actually using a Linux OpenLDAP/Kerberos host as it's OD "master".
Problems were that authentication was hit or miss. Some services like Addressbook server worked 100% fine after kerberizing for the Linux KDC, but others
such as iChat and iCal servers refused to authenticate using Kerberos, plaintext passwords were fine. Eventually the server became unstable and would lock up at random times.
I think the kerberos stuff wasn't working right because it wasn't hooked up with LDAP like you have in your example. I'm going to give it one more shot.

The main thing I'd like to have working is 100% kerberos auth with collaboration services and users being able to change their passwords and have in reflected in kerberos and ldap crypt pass for non-kerberized services.
I'll share any new findings if I have any.

Howard F says:

December 11, 2009 at 6:10 pm Reply

Hi,

I hope you could still have a chance or time to try what you did with openldap/kerberos5 here as well on opends(www.opends.org). I believe it would be a very interesting endeavour. Thanks for this article and "may you have more fun and interesting problems to solve". 😀

Kent Watsen says:

February 5, 2010 at 5:37 am Reply

I'm trying to mount Mac home directories off a OpenSolaris box using ZFS-backed directories. I was hoping to first try the solution *without* using Kerboros, but when I followed your instructions, except the Kerboros part, as I'm using an OpenSolaris server and wanted to get things working before turning Kerboros on. I instruct my Mac to use the "Open Directory Server" template, but it will not authenticate when I'm using the LDAP server's rootdn, with the rootpw being either cleartext or encrypted. I'm wondering, is the use of Kerboros somehow required when using the "Open Directory Server" template?

Also, other tutorials I've seen use the "RFC 2307 (Unix)" mapping; does using the "Open Directory Server" template provide any functionality than using the RFC 2307 template?

Thanks,
Kent

Andrew says:

February 9, 2010 at 11:24 am Reply

Unless you have extended your schema you shouldn't use the Open Directory template.

Kerberos shouldn't be needed though. In fact I don't use it at home myself as in the end I got sick of all the nonsense that went with getting OS X to use it outside of an Open Directory or Active Directory environment. At work I have both OD and AD so it "Just Works".

Kent Watsen says:

February 13, 2010 at 2:06 am Reply

Thanks for the reply, Andrew, but I'm confused by your first statement. I'm using the LDAP schema you posted in this article – are you saying that I should *not* use the "Open Directory Server" template?

Note that I'm using Snow Leopard on my desktop. When selecting

"New" in Directory Utility, after entering my server's IP, it takes me to a section called "LDAP Mappings" where it says "Pick a Template" and that is where I select "Open Directory Server" – isn't this what your instructions recommend?

Thanks again,
Kent

Andrew
says:

February 15, 2010 at 12:53 pm Reply

Hi Kent,

So long as you have added the Apple LDAP schema extensions you should be able to use the Open Directory template.

You should also be able to go ahead without Kerberos so long as you have the appropriately encrypted password entry in the LDAP user account. This is what I use at home as I felt the Kerberos integration became too much work for too little real gain. Obviously in a large enterprise environment that wouldn't be the case, but having authentication stored in one place (Kerberos) and everything else in another (LDAP) does make supposedly simple tasks like creating accounts and changing passwords much more of a pain.

Regards,
Andrew

Kent Watsen
says:

February 15, 2010 at 6:12 pm Reply

I'm with you wrt kerboros being overkill for home – I'm just using digest-md5 sasl now…

BTW, I'm getting a "creating a user: 'inetOrgPerson' requires attribute 'sn'" error when trying a add a user thru Workgroup Manager. I see that 'sn' is a MUST attribute in the "person" objectClass. Do I need to create custom mappings after selecting Open Directory Template? You didn't list needing to do any extra fiddling and my LDAP schema is the same as yours, after swapping your base dn for mine. of course. I've searched on this error already, but didn't find any relevant fixes – any ideas?

Thanks again,
Kent

Andrew
says:

February 20, 2010 at 8:52 am Reply

Sorry, no ideas. I haven't had enough to do with this for a while now.

Dave
Abrahams
says:

June 24, 2010 at 12:05 pm Reply

It would be really interesting and useful to have a virtual machine image with all this stuff set up on it.

Andrew
says:

July 31, 2010 at 8:25 am Reply

I agree.

Unfortunately I haven't really been doing enough with this lately and I don't think I'd be able to create a decent setup now. ☹

Jona says:

February 25, 2011 at 5:24 am Reply

Great documentation of the whole process, thanks 😃

Working through this with Debian Squeeze and Snow Leopard right now, and the schema conversion step seems to need the ‘apple_auxillary.schema’ as well, right before ‘apple.schema’.

Fabien says:

June 9, 2011 at 1:56 am Reply

To use the WGM to manage users in he Linux OpenLDAP, from a client binded to the server, do not try to connecter a server. Juste use Cmd+D (WGM: Server > View Directories).

Dodevich says:

September 18, 2011 at 11:06 pm Reply

Thanks.. Very good HOWTO… and apple schema – very nice.

## Leave a Comment

Name (required)

Email (will not be published) (required)

Website URL

SUBMIT COMMENT