GNU/Linux Administration - Support #174

Integrated Kerberos OpenLDAP provider on Debian 7

08/19/2013 03:43 PM - Daniel Curtis

Status:	Closed	**Start date:**	08/19/2013
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:	Domain Controller	**Estimated time:**	4.00 hours
Target version:		**Spent time:**	12.00 hours

Description

Introduction

These instructions describe how to set up an OpenLDAP provider server and an MIT Kerberos V master KDC on the same host with Kerberos using LDAP as its back-end database. This combines the excellent security and encryption offered by Kerberos with OpenLDAP's superior replication engine. It is a strategy made possible by a package that extends the OpenLDAP database schema to support Kerberos. It provides a plugin for the Kerberos server to allow it to use an LDAP directory as its primary back-end database.

In this example, OpenLDAP is installed on a host running Debian 7.0 (wheezy). If followed properly, the step-by-step process should produce an OpenLDAP provider server with a new Directory Information Tree (DIT), followed by a Kerberos master server that stores its database in that same DIT. The system relies heavily on timestamps, so reasonably accurate time synchronization among all participating hosts is essential.

But, before the interesting parts can begin, it will first be necessary to install the operating system on a new host called **kls1.example.com**. A DNS server must be available on the network with zone files to which forward and reverse mappings can be added for this host, as well as an alias for it called **kls.example.com**. After the initial installation of the operating system, make sure these packages are installed on the system as well:

```
apt-get install ssh ntp ntpdate nmap
```

Afterwards, edit /etc/ntp.conf so that the machine synchronizes to a common NTP server (preferably a local one) and edit /etc/default/ntpdate to use the same host also. Now the installation of the new server can begin.

1. OpenLDAP install

On the new host, **kls1.example.com**, start by installing these two packages:

```
apt-get install slapd ldap-utils
```

A total of eight packages are installed as a result, including six dependencies:

```
ldap-utils          2.4.23-7          OpenLDAP utilities
libltdl7            2.2.6b-2          A system independent dlopen wrapper for GNU libtool
libperl5.10         5.10.1-16         shared Perl library
libslp1             1.2.1-7.8         OpenSLP libraries
odbcinst            2.2.14p2-1        Helper program for accessing odbc ini files
odbcinst1debian2    2.2.14p2-1        Support library for accessing odbc ini files
slapd               2.4.23-7          OpenLDAP server (slapd)
unixodbc            2.2.14p2-1        ODBC tools libraries
```

During the install process, an administrator password will be requested for slapd. The one used here is ***arietans***:

Administrator password: **arietans**
Confirm password: **arietans**

Run the following command to test if the OpenLDAP server is actually running:

```
nmap -p 389 localhost
```

```
Starting Nmap 5.00 ( http://nmap.org ) at 2011-01-05 01:41 CET
Warning: Hostname localhost resolves to 2 IPs. Using 127.0.0.1.
Interesting ports on localhost (127.0.0.1):
PORT    STATE SERVICE
389/tcp open  ldap

Nmap done: 1 IP address (1 host up) scanned in 0.09 seconds
```

The ldap service is made available on port 389 by the Standalone LDAP Daemon (slapd). Note that the state of the port is open. To be sure, the LDAP v3 technical specification (RFC-3377) does not mention anything about a backend solution in which to store the database; it is only a description of the network protocol itself. The daemon, slapd, the way it stores its data and the various utilities it comes with are all unique to OpenLDAP.

Perform a quick test by generating an LDAP Data Interchange Format (LDIF) dump of the contents of a the database:

```
slapcat
```

```
hdb_db_open: database "dc=example,dc=com": unclean shutdown detected;
attempting recovery.
hdb_db_open: database "dc=example,dc=com": recovery skipped in read-only
mode. Run manual recovery if errors are encountered.
dn: dc=example,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: example.com
dc: example
structuralObjectClass: organization
entryUUID: a7904b74-acae-102f-81f2-1f93a1bd1f6f
creatorsName: cn=admin,dc=example,dc=com
createTimestamp: 20110105002951Z
entryCSN: 20110105002951.593891Z#000000#000#000000
modifiersName: cn=admin,dc=example,dc=com
modifyTimestamp: 20110105002951Z

dn: cn=admin,dc=example,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator
userPassword:: e1NTSEF9cW9aeDFqZlI5Y215MWxoaTU1eWxNVnFIRE9SQ09Rc3E=
structuralObjectClass: organizationalRole
entryUUID: a790e34a-acae-102f-81f3-1f93a1bd1f6f
creatorsName: cn=admin,dc=example,dc=com
createTimestamp: 20110105002951Z
entryCSN: 20110105002951.597818Z#000000#000#000000
modifiersName: cn=admin,dc=example,dc=com
modifyTimestamp: 20110105002951Z
```

2. ldap.conf

Edit /etc/ldap/ldap.conf and use these two lines:

```
BASE    dc=example,dc=com
URI     ldap://kls1.example.com/
```

This configuration file is used to set system-wide defaults for LDAP clients.

3. Kerberos master install

On the new host, **kls1.example.com**, start by installing these three packages:

```
apt-get install krb5-{admin-server,kdc-ldap,user}
```

A total of five packages are installed as a result, including two dependencies:

```
krb5-admin-server       1.8.3+dfsg-4            MIT Kerberos master server (kadmind)
krb5-config             2.2                     Configuration files for Kerberos Version 5
krb5-kdc                1.8.3+dfsg-4            MIT Kerberos key server (KDC)
krb5-kdc-ldap           1.8.3+dfsg-4            MIT Kerberos key server (KDC) LDAP plugin
krb5-user               1.8.3+dfsg-4            Basic programs to authenticate using MIT Kerberos
```

During package installation a few questions are asked regarding the krb5-admin-server package that should be answered as follows:

Default Kerberos version 5 realm: **EXAMPLE.COM**
Kerberos servers for your realm: **kls1.example.com**
Administrative server for your Kerberos realm: **kls.example.com**

Towards the end of the automated configuration sequence for these packages, a problem appears:

```
Setting up krb5-kdc (1.8.3+dfsg-4) ...
krb5kdc: cannot initialize realm EXAMPLE.COM - see log file for details
Setting up krb5-admin-server (1.8.3+dfsg-4) ...
kadmind: No such file or directory while initializing, aborting
```

The issue occurs because at this point the realm, *EXAMPLE.COM*, or rather a database for it, has not yet been created. This will be dealt with soon enough.

4. Boot order

Since the goal here is for Kerberos to use OpenLDAP as its back-end database, both the Kerberos KDC and administrative services must be made to start up after slapd, as well as to stop before it. However, as opposed to Debian 5.0 (lenny), in which this could be done by changing the sequence numbers of the symbolic links in the /etc/rc?.d/ directories, squeeze works with dependency based boot sequencing. Edit two files, /etc/init.d/krb5-kdc and /etc/init.d/krb5-admin-server, and modify them both as follows to achieve the desired effect:

1. Required-Start: $local_fs $remote_fs $network $syslog slapd
2. Required-Stop: $local_fs $remote_fs $network $syslog slapd

These lines are part of the Linux Standard Base (LSB) specification v3.1. Both consist of a hash, followed by a key word, a colon and a number of boot facilities. Of the latter, the first four all start with a dollar sign, because they are predefined virtual facilities. Otherwise, the names of System V boot scripts, found in /etc/init.d/, should be used, although without a dollar sign or possible .sh extension. The addition of "slapd" is an example of this. In exceptional cases, the name of the service can be used instead (see /etc/services).

What the two lines above say is that the service, which is started and stopped by this script, can only be started up after certain other boot facilities have been started (Required-Start), and must be stopped before those same boot facilities have been stopped (Required-Stop).

However, simply modifying the /etc/init.d/ scripts in this manner will not change anything. The actual changes are made when insserv is run. This is what the sysv-rc package uses to (re)order init.d scripts based on their declared dependencies. Use it now to apply the changes that have been made to the two init scripts:

```
insserv /etc/init.d/krb5-kdc
insserv /etc/init.d/krb5-admin-server
```

5. Admin authorization

Create a new file, /etc/krb5kdc/kadm5.acl, and add the following two lines:

```
*/admin *
admin *
```

This is to allow certain principals, **admin** and names ending in /**admin**, to perform any operation.

6. Kerberos schema

Prepair the OpenLDAP server so that it will be able to function as a back-end database for a Kerberos KDC. First, execute the following five commands to convert the Kerberos schema to LDIF format:

```
gunzip -c /usr/share/doc/krb5-kdc-ldap/kerberos.schema.gz > /etc/ldap/schema/kerberos.schema
echo "include /etc/ldap/schema/kerberos.schema" > ~/schema_convert.conf
mkdir ~/ldif_result
slapcat -f ~/schema_convert.conf -F ~/ldif_result -s "cn=kerberos,cn=schema,cn=config"
cp ~/ldif_result/cn\=config/cn\=schema/cn\=\{0\}kerberos.ldif ~/kerberos.ldif
```

Next, edit ~/kerberos.ldif, first by making these two changes to the top of the file:

```
dn: cn=kerberos,cn=schema,cn=config
objectClass: olcSchemaConfig
cn: kerberos
```

Second, still editing ~/kerberos.ldif, remove the following attributes from the end of the file (even though the values will not match precisely):

```
structuralObjectClass: olcSchemaConfig
entryUUID: 8817918a-8dc3-102f-8103-2371e8422500
creatorsName: cn=config
createTimestamp: 20101126161112Z
entryCSN: 20101126161112.176097Z#000000#000#000000
modifiersName: cn=config
modifyTimestamp: 20101126161112Z
```

Now add the Kerberos schema to the cn=config DIT with this command:

```
ldapadd -QY EXTERNAL -H ldapi:/// -f ~/kerberos.ldif
```

```
adding new entry "cn=kerberos,cn=schema,cn=config"
```

Verify that the new schema has been added with this command:

```
ldapsearch -LLLQY EXTERNAL -H ldapi:/// -b cn=config cn={4}kerberos
```

The allocated index number is four, because this schema is not the only one in the cn=config DIT. The curious may feel inclined to check out these other pre-installed schema objects:

```
cn={0}core
cn={1}cosine
cn={2}nis
cn={3}inetorgperson
```

7. cn=config

Besides the addition of the Kerberos schema, a number of general modifications need to be made to the OpenLDAP runtime configuration directory regarding two existing entries. First consider the current state of these objects:

```
ldapsearch -LLLQY EXTERNAL -H ldapi:/// -b cn=config "(|(cn=config)(olcDatabase={1}hdb))"
```

```
dn: cn=config
objectClass: olcGlobal
cn: config
olcArgsFile: /var/run/slapd/slapd.args
olcLogLevel: none
olcPidFile: /var/run/slapd/slapd.pid
olcToolThreads: 1

dn: olcDatabase={1}hdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcHdbConfig
olcDatabase: {1}hdb
olcDbDirectory: /var/lib/ldap
olcSuffix: dc=example,dc=com
olcAccess: {0}to attrs=userPassword,shadowLastChange by self write by anonymou
s auth by dn="cn=admin,dc=example,dc=com" write by * none
olcAccess: {1}to dn.base="" by * read
olcAccess: {2}to * by self write by dn="cn=admin,dc=example,dc=com" write by *
read
olcLastMod: TRUE
olcRootDN: cn=admin,dc=example,dc=com
olcRootPW: {SSHA}kOyNI8zCJbVHz1e/NMW0y3IoOMZ+UjhA
olcDbCheckpoint: 512 30
olcDbConfig: {0}set_cachesize 0 2097152 0
olcDbConfig: {1}set_lk_max_objects 1500
olcDbConfig: {2}set_lk_max_locks 1500
olcDbConfig: {3}set_lk_max_lockers 1500
olcDbIndex: objectClass eq
```

Here is a description of the changes that will be made to the cn=config DIT:

- 1. – For the root object, cn=config, the value for olcLogLevel is currently set to none (the Debian default). This prevents slapd from writing almost anything to the syslog. This is okay for production environments, but less desirable for educational purposes, so it is going to be changed to stats (the value recommended by the OpenLDAP project). This way, slapd will log more about what it does to /var/log/syslog.
- 2.1.x. – For the database definition, olcDatabase={1}hdb,cn=config, all three of the current ACLs, each one being a value of an olcAccess attribute, will be deleted. For clarity, this will be done in reverse order, or else the index numbers would all have to be zero.
- 2.2.1. – The first new olcAccess attribute will determine access to password information stored in the DIT. Anonymous users will have auth access and all other users will have no access. This type of authentication may seem like a security issue, but since only IPC connections (i.e. Unix domain sockets) will be used for this – no network connections – there is nothing to worry about.
- 2.2.2. – The second new olcAccess attribute will determine access to the future ou=krb5 subtree, which will contain the Kerberos database. The future entry for the Kerberos administration server will have write access to it, while the future entry for the Kerberos KDC will have read access.
- 2.2.3. – The third new olcAccess attribute will allow authenticated users to change their own shell settings, the attribute for which is included in the LDAP schema for the posixAccount object class.
- 2.2.4. – The fourth new olcAccess attribute will allow read access to the base of the tree for things like supportedSASLMechanisms. This allows clients to discover which SASL mechanisms an LDAP server supports.
- 2.2.5. – The fifth and last new olcAccess attribute is the default ACL that determines access to everything else in the tree. Authenticated users will have read access and anonymous users will have no access.
- 2.3. – An extra equality (eq) index will be added for User ID (uid) objects. olcDbIndex directives are used to specify the attributes on which slapd should maintain indices to optimize directory searches. This new equality index will speed up searches for entries based on exact matches of the uid attribute. At the moment it is not really necessary to add any extra indices, since the new directory tree contains almost no data. However, it is recommended to add an eq index for uid entries, because these are used for storing user account names and later on, as the DIT grows in size, this index can significantly decrease the time it takes for users to log in. The same eq will also prevent "bdb_equality_candidates" errors...
- 2.4. – An equality index will also be set for Common Name (cn) objects to prevent "bdb_equality_candidates" errors from appearing in the log file, /var/log/syslog. One such error will otherwise appear in the log for every search for an entry of this type.
- 2.5. – Idem for Organizational Unit (ou) objects.
- 2.6. – Idem for Domain Component (dc) objects.
- 2.7. – Idem for the uidNumber attribute.
- 2.8. – Idem for the gidNumber attribute.
- 2.9. – Idem for the memberUid attribute.
- 2.10. – Idem for the uniqueMember attribute.
- 2.11. – An equality, presence and substring index will be set for the krbPrincipalName attribute.

- 2.12. – An equality index will be set for the krbPwdPolicyReference attribute.

To make both of the above changes to the cn=config DIT, first create an LDIF file, called ~/olc-mod1.ldif, with the following contents:

```
# 1.
dn: cn=config
changetype: modify
replace: olcLogLevel
olcLogLevel: stats

# 2.1.1
dn: olcDatabase={1}hdb,cn=config
changetype: modify
delete: olcAccess
olcAccess: {2}to *
  by self write
  by dn="cn=admin,dc=example,dc=com" write
  by * read
-
# 2.1.2
delete: olcAccess
olcAccess: {1}to dn.base=""
  by * read
-
# 2.1.3.
delete: olcAccess
olcAccess: {0}to attrs=userPassword,shadowLastChange
  by self write
  by anonymous auth
  by dn="cn=admin,dc=example,dc=com" write
  by * none
-
# 2.2.1.
add: olcAccess
olcAccess: to attrs=userPassword,shadowLastChange
  by anonymous auth
  by * none
-
# 2.2.2.
add: olcAccess
olcAccess: to dn.subtree="ou=krb5,dc=example,dc=com"
  by dn="cn=adm-srv,ou=krb5,dc=example,dc=com" write
  by dn="cn=kdc-srv,ou=krb5,dc=example,dc=com" read
  by * none
-
# 2.2.3.
add: olcAccess
olcAccess: to attrs=loginShell
  by self write
  by users read
  by * none
-
# 2.2.4.
add: olcAccess
olcAccess: to dn.base=""
  by * read
-
# 2.2.5.
add: olcAccess
olcAccess: to *
  by users read
  by * none
-
# 2.3.
add: olcDbIndex
olcDbIndex: uid eq
```

```
-
# 2.4.
add: olcDbIndex
olcDbIndex: cn eq
-
# 2.5.
add: olcDbIndex
olcDbIndex: ou eq
-
# 2.6.
add: olcDbIndex
olcDbIndex: dc eq
-
# 2.7.
add: olcDbIndex
olcDbIndex: uidNumber eq
-
# 2.8.
add: olcDbIndex
olcDbIndex: gidNumber eq
-
# 2.9.
add: olcDbIndex
olcDbIndex: memberUid eq
-
# 2.10.
add: olcDbIndex
olcDbIndex: uniqueMember eq
-
# 2.11.
add: olcDbIndex
olcDbIndex: krbPrincipalName eq,pres,sub
-
# 2.12.
add: olcDbIndex
olcDbIndex: krbPwdPolicyReference eq
```

After the file has been saved, apply all of the above changes with this command:

```
ldapmodify -QY EXTERNAL -H ldapi:/// -f ~/olc-mod1.ldif
```

modifying entry "cn=config"

modifying entry "olcDatabase={1}hdb,cn=config"

Rerun the previous ldapsearch command to verify that all of the changes have been made successfully. Examination will show that the values for all of the olcAccess attributes will automatically have been given index numbers.

During the initial setup, an administrative account, cn=admin, was created that will soon be a security risk. The aim here is to create a secure service with all credentials stored exclusively in the Kerberos database, so, now that all references to it have been removed from the cn=config DIT, this account can be deleted.

```
ldapdelete -xh localhost -D cn=admin,dc=example,dc=com -w arietans cn=admin,dc=example,dc=com
```

This does not actually remove administrative access to the DIT, because this name and its matching password still exist as attributes of the database definition in the slapd runtime configuration. The difference is that now they cannot be used from anywhere else – only on this server. If it ever becomes necessary, or desirable, for the admin account to also have full read/write access to the DIT from elsewhere on the network, the ACLs above should be altered to once again include write access for the admin account. However, that should be done at some later point, because in this example the DN for the admin account will soon be changed.

Perform a quick test by generating an LDIF dump of the contents of a the database:

```
slapcat
```

```
hdb_db_open: database "dc=example,dc=com": unclean shutdown detected;
attempting recovery.
hdb_db_open: database "dc=example,dc=com": recovery skipped in read-only
mode. Run manual recovery if errors are encountered.
dn: dc=example,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: example.com
dc: example
structuralObjectClass: organization
entryUUID: a7904b74-acae-102f-81f2-1f93a1bd1f6f
creatorsName: cn=admin,dc=example,dc=com
createTimestamp: 20110105002951Z
entryCSN: 20110105002951.593891Z#000000#000#000000
modifiersName: cn=admin,dc=example,dc=com
modifyTimestamp: 20110105002951Z
```

The results should look like the above, showing that only the organization object is left.

8. krb5.conf

Edit the Kerberos realm configuration file, /etc/krb5.conf. This file is initially created by the Debian installer and contains information about the realms of a number of organizations, but none of that is necessary in this case. Instead, replace its contents with this:

```
[libdefaults]
    default_realm = EXAMPLE.COM
    forwardable = true
    proxiable = true

[realms]
    EXAMPLE.COM = {
        kdc = kls1.example.com
        admin_server = kls.example.com
        database_module = openldap_ldapconf
    }

[domain_realm]
    .example.com = EXAMPLE.COM
    example.com = EXAMPLE.COM

[dbdefaults]
    ldap_kerberos_container_dn = ou=krb5,dc=example,dc=com

[dbmodules]
    openldap_ldapconf = {
        db_library = kldap
        ldap_kdc_dn = cn=kdc-srv,ou=krb5,dc=example,dc=com
        ldap_kadmind_dn = cn=adm-srv,ou=krb5,dc=example,dc=com
        ldap_service_password_file = /etc/krb5kdc/service.keyfile
        ldap_conns_per_server = 5
    }

[logging]
    kdc = FILE:/var/log/krb5/kdc.log
    admin_server = FILE:/var/log/krb5/kadmin.log
    default = FILE:/var/log/krb5/kadmin.log
```

See this section for a more detailed explanation of this file.

Note the absence of the ldap_servers statement from the list of openldap_ldapconf options. This is possible because the Kerberos daemons use IPC by default to connect to the LDAP back-end, while in Debian squeeze the default configuration for slapd now includes listening for IPC connections.

After /etc/krb5.conf has been saved, create the Kerberos log directory:

```
mkdir /var/log/krb5
```

To prevent the log files from growing too large, create a pair of logrotate configuration files. First, edit /etc/logrotate.d/krb5-kdc and give it the following contents:

```
/var/log/krb5/kdc.log {
    daily
    missingok
    rotate 7
    compress
    delaycompress
    notifempty
    postrotate
        /etc/init.d/krb5-kdc restart > /dev/null
    endscript
}
```

Then edit /etc/logrotate.d/krb5-kadmin and give it this slightly different configuration:

```
/var/log/krb5/kadmin.log {
    daily
    missingok
    rotate 7
    compress
    delaycompress
    notifempty
    postrotate
        /etc/init.d/krb5-admin-server restart > /dev/null
    endscript
}
```

9. Realm subtree

Create the entries mentioned in the /etc/krb5.conf file that will support the new Kerberos server. First create a file, ~/krb5.ldif, with the following contents:

```
dn: ou=krb5,dc=example,dc=com
ou: krb5
objectClass: organizationalUnit

dn: cn=kdc-srv,ou=krb5,dc=example,dc=com
cn: kdc-srv
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: Default bind DN for the Kerberos KDC server
userPassword: gabonica

dn: cn=adm-srv,ou=krb5,dc=example,dc=com
cn: adm-srv
objectClass: simpleSecurityObject
objectClass: organizationalRole
description: Default bind DN for the Kerberos Administration server
userPassword: nasicornis
```

Although the last two of these entries require only simple authentication (clear-text passwords), this is not an issue because, as opposed to using the network, only IPC will be used during the authentication of these accounts. After saving ~/krb5.ldif, add the new objects to the DIT with this command:

```
ldapadd -xWD cn=admin,dc=example,dc=com -f ~/krb5.ldif
```

Enter LDAP Password: **arietans**
adding new entry "ou=krb5,dc=example,dc=com"

adding new entry "cn=kdc-srv,ou=krb5,dc=example,dc=com"

adding new entry "cn=adm-srv,ou=krb5,dc=example,dc=com"

10. Realm creation

To create the new realm, as opposed to the krb5_newrealm command, use kdb5_ldap_util:

```
kdb5_ldap_util -D cn=admin,dc=example,dc=com -H ldap://kls1.example.com create -r EXAMPLE.COM -s
```

Password for "cn=admin,dc=example,dc=com": **arietans**
Initializing database for realm 'EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key: **parviocula**
Re-enter KDC database master key to verify: **parviocula**

Use the slapcat command to verify the creation of the new database.

Passwords are needed for the two Kerberos services to access the Kerberos database in LDAP DIT. These passwords will be stashed in a file called /etc/krb5kdc/service.keyfile. First, create the stash for the KDC service, represented by the **cn=kdc-srv** object and referred to in /etc/krb5.conf with the ldap_kdc_dn option:

```
kdb5_ldap_util -D cn=admin,dc=example,dc=com stashsrvpw -f /etc/krb5kdc/service.keyfile cn=kdc-srv
,ou=krb5,dc=example,dc=com
```

Password for "cn=admin,dc=example,dc=com": **arietans**
Password for "cn=kdc-srv,ou=krb5,dc=example,dc=com": **gabonica**
Re-enter password for "cn=kdc-srv,ou=krb5,dc=example,dc=com": **gabonica**

Then create a stash for the Kerberos administration server, represented by the **cn=adm-srv** object and referred to in /etc/krb5.conf with the ldap_kadmind_dn option:

```
kdb5_ldap_util -D cn=admin,dc=example,dc=com stashsrvpw -f /etc/krb5kdc/service.keyfile cn=adm-srv
,ou=krb5,dc=example,dc=com
```

Password for "cn=admin,dc=example,dc=com": **arietans**
Password for "cn=adm-srv,ou=krb5,dc=example,dc=com": **nasicornis**
Re-enter password for "cn=adm-srv,ou=krb5,dc=example,dc=com": **nasicornis**

11. Admin user

Start up the local administrative interface for the new Kerberos database and create a principal for the admin user:

```
kadmin.local
```

Authenticating as principal root/admin@EXAMPLE.COM with password.
kadmin.local: **addprinc admin**
WARNING: no policy specified for admin@EXAMPLE.COM; defaulting to
no policy
Enter password for principal "admin@EXAMPLE.COM": **ammodytes**
Re-enter password for principal "admin@EXAMPLE.COM": **ammodytes**
Principal "admin@EXAMPLE.COM" created.
kadmin.local:

12. Ticket lifetime

Still in the local Kerberos administrative interface, use it now to allow more flexible lifetime and renewal time frames for the ticket granting ticket (TGT) service. The commands and their responses should look like this:

kadmin.local: **modprinc -maxlife "1 day" -maxrenewlife "90 day"** krbtgt/EXAMPLE.COM@EXAMPLE.COM
Principal "krbtgt/EXAMPLE.COM@EXAMPLE.COM" modified.
kadmin.local: **q**

The values entered above are instead of 10 hours and 1 day respectively, which are the default values. However, every organization should use values that best suit their own security needs. As for kadmin.local, this is a fail-safe version of the kadmin tool that can only be used on the KDC as root and requires no password to modify the database directly. The kadmin tool, on the other hand, can be used from anywhere on the network.

Now edit /etc/krb5kdc/kdc.conf and modify these two lines to reflect the changes made above:

max_life = 1d 0h 0m 0s
max_renewable_life = 90d 0h 0m 0s

13. Kerberos server start

Start the Kerberos admin and KDC servers for the first time:

```
/etc/init.d/krb5-admin-server start ; /etc/init.d/krb5-kdc start
```

If there are no errors, run the following command to verify that the new MIT Kerberos V master server is indeed available on the network:

```
nmap -sU -sT -p U:88,464,T:464,749 localhost
```

Starting Nmap 5.00 (http://nmap.org) at 2011-01-05 01:49 CET
Warning: Hostname localhost resolves to 2 IPs. Using 127.0.0.1.
Interesting ports on localhost (127.0.0.1):
PORT STATE SERVICE
464/tcp open kpasswd5
749/tcp open kerberos-adm
88/udp open|filtered kerberos-sec
464/udp open|filtered kpasswd5

Nmap done: 1 IP address (1 host up) scanned in 3.14 seconds

Note that the state of all of these ports should be open.

14. Service princ & keytab

Use **kadmin** to create a Kerberos principal for the LDAP service and a matching keytab file by issuing a few commands:

```
kadmin -p admin
```

Authenticating as principal admin with password.
Password for admin@EXAMPLE.COM: **ammodytes**
kadmin: **addprinc -randkey ldap/kls1.example.com**
WARNING: no policy specified for ldap/kls1.example.com@EXAMPLE.COM; defaulting to no policy
Principal "ldap/kls1.example.com@EXAMPLE.COM" created.
kadmin: **ktadd ldap/kls1.example.com**
Entry for principal ldap/kls1.example.com with kvno 2, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal ldap/kls1.example.com with kvno 2, encryption type ArcFour with HMAC/md5 added to keytab

WRFILE:/etc/krb5.keytab.
Entry for principal ldap/kls1.example.com with kvno 2, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal ldap/kls1.example.com with kvno 2, encryption type DES cbc mode with CRC-32 added to keytab WRFILE:/etc/krb5.keytab.
kadmin: **q**

The -randkey switch is used to avoid having to use a password. To list the keys in /etc/krb5.keytab, use the klist -ke command.
A host (or service) principal and a keytab file should be created for and saved on all of the various client machines that are part of a Kerberos realm.

15. Slapd kerberization

To kerberize slapd, start by installing this package:

```
apt-get install libsasl2-modules-gssapi-mit
```

Only one package is installed as a result:

libsasl2-modules-gssapi-mit 2.1.23.dfsg1-6 Cyrus SASL - pluggable authentication modules (GSSAPI)

GSSAPI stands for Generic Security Service API. Defined in RFC-2743, it provides a common interface for accessing different security services, one of the most popular of which is Kerberos V. The way that GSSAPI services can be used for SASL authentication and the establishment of a security layer is described in RFC-2222 (Simple Authentication and Security Layer).

Change the permissions and ownership of the Kerberos keytab file to make sure that slapd can access it:

```
chmod 640 /etc/krb5.keytab
chown root.openldap /etc/krb5.keytab
```

Edit /etc/default/slapd and uncomment a line near the end of the file that will export the location of the Kerberos system keytab file as a variable:

export KRB5_KTNAME=/etc/krb5.keytab

Edit /etc/ldap/ldap.conf and add the following line to the end of the file to specify the authentication mechanism:

SASL_MECH GSSAPI

Now to configure the link that maps GSSAPI-format user names to LDAP names. Once users have been authenticated by Kerberos and have valid Kerberos tickets, the SASL layer redirects them to the GSSAPI mechanism. This results in distinguished names that consist of four parts:

- uid=user_name
- cn=realm_name
- cn=gssapi
- cn=auth

For instance, **uid=admin,cn=example.com,cn=gssapi,cn=auth** would be the GSSAPI-format name of the Kerberos admin account. The key is to match and replace these names with ones that can be found in the DIT. To do this, a couple of organizational units will be necessary. These will contain user and group objects. Create a file, ~/people-groups.ldif, with the following contents:

dn: ou=people,dc=example,dc=com
objectClass: organizationalUnit
ou: people

dn: ou=groups,dc=example,dc=com
objectClass: organizationalUnit
ou: groups

Add this information to the DIT with the ldapadd command:

```
ldapadd -xWD cn=admin,dc=example,dc=com -f ~/people-groups.ldif
```

Enter LDAP Password: **arietans**
adding new entry "ou=people,dc=example,dc=com"

adding new entry "ou=groups,dc=example,dc=com"

The slapd runtime configuration directory will also have to be modified again. Two objects need to be altered that at the moment look like this:

```
ldapsearch -LLLQY EXTERNAL -H ldapi:/// -b cn=config "(|(cn=config)(olcDatabase={1}hdb))"
```

dn: cn=config
objectClass: olcGlobal
cn: config
olcArgsFile: /var/run/slapd/slapd.args
olcPidFile: /var/run/slapd/slapd.pid
olcToolThreads: 1
olcLogLevel: stats

dn: olcDatabase={1}hdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcHdbConfig
olcDatabase: {1}hdb
olcDbDirectory: /var/lib/ldap
olcSuffix: dc=example,dc=com
olcLastMod: TRUE
olcRootDN: cn=admin,dc=example,dc=com
olcRootPW: {SSHA}qoZx1jfR9cmy1lhi55ylMVqHDORCOQsq
olcDbCheckpoint: 512 30
olcDbConfig: {0}set_cachesize 0 2097152 0
olcDbConfig: {1}set_lk_max_objects 1500
olcDbConfig: {2}set_lk_max_locks 1500
olcDbConfig: {3}set_lk_max_lockers 1500
olcDbIndex: objectClass eq
olcDbIndex: uid eq
olcDbIndex: cn eq
olcDbIndex: ou eq
olcDbIndex: dc eq
olcDbIndex: krbPrincipalName eq,pres,sub
olcAccess: {0}to attrs=userPassword,shadowLastChange by anonymous auth by * no
ne
olcAccess: {1}to dn.subtree="ou=krb5,dc=example,dc=com" by dn="cn=adm-srv,ou=k
rb5,dc=example,dc=com" write by dn="cn=kdc-srv,ou=krb5,dc=example,dc=com" rea
d by * none
olcAccess: {2}to attrs=loginShell by self write by users read by * none
olcAccess: {3}to dn.base="" by * read
olcAccess: {4}to * by users read by * none

Here is a description of the changes that will be made to the configuration directory to complete the kerberization of the slapd provider server:

- 1.1. – For the root object, cn=config, an olcAuthzRegexp directive will be added to match the GSSAPI-format admin account and replace it with a DN that will match the admin account in the DIT.
- 1.2. – An olcSaslRealm attribute will be added to specify the SASL realm, in this case a Kerberos realm.
- 2.1. – For the database definition, olcDatabase={1}hdb,cn=config, the name of the administrative user will be changed to match the DN to which it will be translated after it encounters the olcAuthzRegexp.
- 2.2. – Delete the password for the administrative user, so that from this point on it will only be possible to use this account after first authenticating to Kerberos.

To make the above changes to the cn=config DIT, create an LDIF file, called ~/olc-mod2.ldif, with the following contents:

```
# 1.1.
dn: cn=config
changetype: modify
add: olcAuthzRegexp
olcAuthzRegexp: uid=([^,]+),cn=example.com,cn=gssapi,cn=auth
  uid=$1,ou=people,dc=example,dc=com
-
# 1.2.
add: olcSaslRealm
olcSaslRealm: EXAMPLE.COM

# 2.1.
dn: olcDatabase={1}hdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: uid=admin,ou=people,dc=example,dc=com
-
# 2.2.
delete: olcRootPW
```

The **olcAuthzRegexp** attribute shown above works as follows. Its value consists of two parts: a regular expression (search pattern) that matches one or more of the GSSAPI-format user names from the Kerberos database, and a replacement string that changes the name to a format compatible with the DIT, usually (but not necessarily) with the intention to match an existing LDAP entry. The text matched by the symbols between the parentheses in the search pattern substitutes the $1 in the replacement string.

Apply changes in the LDIF file with this command:

```
ldapmodify -QY EXTERNAL -H ldapi:/// -f ~/olc-mod2.ldif
```

modifying entry "cn=config"

modifying entry "olcDatabase={1}hdb,cn=config"

Rerun the previous ldapsearch command to verify that all of the changes have been made successfully.

Because of the new software that has been added and the changes that have also been made outside of the slapd configuration directory, it must be restarted before they will all have the desired effect:

```
/etc/init.d/slapd restart
```

Stopping OpenLDAP: slapd.
Starting OpenLDAP: slapd.

16. Authentication test

Run some tests. First try a simple unauthenticated (-x) LDAP query:

```
ldapsearch -xLLL ou=people
```

No such object (32)

This does not work, because the default ACL has been set to allow "access to * by none." In other words, no anonymous access is allowed. The previously used SIMPLE bind for the admin user, using either its old or new name, does not work either:

```
ldapsearch -xWLLL -D cn=admin,dc=example,dc=com ou=people
```

Enter LDAP Password: **arietans**
ldap_bind: Invalid credentials (49)

```
ldapsearch -xWLLL -D uid=admin,ou=people,dc=example,dc=com ou=people
```

Enter LDAP Password: **arietans**
ldap_bind: Invalid credentials (49)

That is because its plain-text password – the olcRootPW attribute – no longer exists. However, the authenticated version of this query, which is its default operational mode, will also give an error:

```
ldapsearch -LLL ou=people
```

SASL/GSSAPI authentication started
ldap_sasl_interactive_bind_s: Local error (-2)
additional info: SASL: generic failure: GSSAPI Error:
Unspecified GSS failure. Minor code may provide more information
(Credentials cache file '/tmp/krb5cc_0' not found)

The solution is to first acquire a Kerberos ticket for the admin user (password **ammodytes**):

```
kinit admin
```

Password for admin@EXAMPLE.COM: **ammodytes**

A verification of the ticket should show a success:

```
klist
```

Ticket cache: FILE:/tmp/krb5cc_0
Default principal: admin@EXAMPLE.COM

Valid starting Expires Service principal
01/05/11 02:14:31 01/06/11 02:14:29 krbtgt/EXAMPLE.COM@EXAMPLE.COM

Now the previously attempted authenticated query should work:

```
ldapsearch -LLL ou=people
```

SASL/GSSAPI authentication started
SASL username: admin@EXAMPLE.COM
SASL SSF: 56
SASL data security layer installed.
dn: ou=people,dc=example,dc=com
objectClass: organizationalUnit
ou: people

Also, under these circumstances the ldapwhoami command behaves more like whoami:

```
whoami
```

root

```
ldapwhoami
```

SASL/GSSAPI authentication started

SASL username: admin@EXAMPLE.COM
SASL SSF: 56
SASL data security layer installed.
dn:uid=admin,ou=people,dc=example,dc=com

As stated before, authentication is necessary for all LDAP commands, but thanks to Kerberos' single-sign-on technology, this is no longer in any way inconvenient.

17. Adding a new user

The last thing to do is to test the system by adding a new user. Each account must consist of a Kerberos principal, a matching user ID object in **ou=people**, and a matching common name object in **ou=groups**. Start by using kadmin to create an account for a user ccolumbus with password NewWorld:

```
kadmin -p admin
```

Authenticating as principal admin with password.
Password for admin@EXAMPLE.COM: **ammodytes**
kadmin: **addprinc ccolumbus**
WARNING: no policy specified for ccolumbus@EXAMPLE.COM; defaulting
to no policy
Enter password for principal "ccolumbus@EXAMPLE.COM": NewWorld
Re-enter password for principal "ccolumbus@EXAMPLE.COM": NewWorld
Principal "ccolumbus@EXAMPLE.COM" created.
kadmin: *q

Having completed that, the matching LDAP user objects are next. First, create a file, called ~/ccolumbus.ldif, with the following contents:

dn: cn=ccolumbus,ou=groups,dc=example,dc=com
cn: ccolumbus
gidNumber: 20001
objectClass: top
objectClass: posixGroup

dn: uid=ccolumbus,ou=people,dc=example,dc=com
uid: ccolumbus
uidNumber: 20001
gidNumber: 20001
cn: Christopher
sn: Columbus
objectClass: top
objectClass: person
objectClass: posixAccount
objectClass: shadowAccount
loginShell: /bin/bash
homeDirectory: /home/ccolumbus
userPassword: {CRYPT}*

The value for the userPassword attribute is an invalid hashed version of a password. The password cannot simply be omitted, because it is a required attribute.

Then apply this change with the ldapadd command:

```
ldapadd -Qf ~/ccolumbus.ldif
```

adding new entry "cn=ccolumbus,ou=groups,dc=example,dc=com"

adding new entry "uid=ccolumbus,ou=people,dc=example,dc=com"

Now test if the new account works. First destroy the admin user's Kerberos ticket (not strictly necessary) and authenticate as the new

user:

```
kdestroy
kinit ccolumbus
```

Password for ccolumbus@EXAMPLE.COM: **NewWorld**

Display the name of the Kerberos principal and tickets held in the credentials cache to confirm that authentication was successful:

```
klist
Ticket cache: FILE:/tmp/krb5cc_0
```

Default principal: ccolumbus@EXAMPLE.COM

Valid starting Expires Service principal
01/05/11 02:51:20 01/06/11 02:51:16 krbtgt/EXAMPLE.COM@EXAMPLE.COM

Finally, confirm that the new user has the correct LDAP identity:

```
ldapwhoami
```

SASL/GSSAPI authentication started
SASL username: ccolumbus@EXAMPLE.COM
SASL SSF: 56
SASL data security layer installed.
dn:uid=ccolumbus,ou=people,dc=example,dc=com

This account is now ready to be used for logging into suitably configured clients.

18. See also

- Integrated Kerberos-OpenLDAP consumer on Debian squeeze
- Integrated Kerberos-OpenLDAP client on Debian squeeze

19. Resources

- http://www.rjsystems.nl/en/2100-d6-kerberos-openldap-provider.php
- http://www.rjsystems.nl/en/2100-d6-kerberos-openldap-client.php
- https://wiki.debian.org/LDAP/Kerberos

Related issues:

Related to GNU/Linux Administration - Feature #175: Kerberizing phpLDAPAdmin ...	**Closed**	**08/19/2013**

History

#1 - 08/19/2013 03:52 PM - Daniel Curtis

- Description updated

I had a very difficult time setting this server up, but by sticking to each step in this tutorial I was able to get a Kerberos/OpenLDAP integrated authentication and user information server set up. The problem I kept encountering was when I was setting client servers libnss-ldap to **ldaps://lxc-auth2/** which caused a BIND failure because TLS was already being used. I did not need to do this because the clients automatically use StartTLS on the normal LDAP port (389).

I should run tcpdump to verify that no sensitive information is sent over the wire.

#2 - 08/20/2013 02:59 PM - Daniel Curtis

- Tracker changed from Bug to Support

#3 - 02/16/2015 02:31 PM - Daniel Curtis

- Project changed from 22 to GNU/Linux Administration

- Description updated

- Category set to Domain Controller