GNU/Linux Administration - Feature #163

Installing Kerberos 5 on Debian

08/08/2013 03:21 PM - Daniel Curtis

Status:	Closed	**Start date:**	08/08/2013
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:	Domain Controller	**Estimated time:**	2.00 hours
Target version:		**Spent time:**	2.00 hours

Description

Server Installation

A Kerberos server installation basically consists of just two packages

- the KDC (Key Distribution Center), which takes care of handling authentication requests and issuing Kerberos tickets.
- kadmind (Kerberos master server), which allows remote administration access to the Kerberos database and carrying out of administrative tasks.

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

Here are the Debconf answers for reference. The listing here includes all questions; some were asked in Kerberos 1.6 packages and some are asked only in Kerberos 1.7 and newer, and their order has changed a little as well. In any case, it's no problem — just answer the subset of questions you are asked:

- Default Kerberos version 5 realm? **EXAMPLE.COM** #(Your domain name in uppercase - a standard for naming Kerberos realms)
- Does DNS contain pointers to your realm's Kerberos Servers? **No** #(No DNS configuration is required for our setup)
- Add locations of default Kerberos servers to /etc/krb5.conf? **Yes** #(Adding entries to krb.conf instead of DNS)
- Create the Kerberos KDC configuration automatically? **Yes**
- Kerberos4 compatibility mode to use: **none** #(No krb4 compatibility needed in our setup)
- Run a Kerberos V5 to Kerberos V4 ticket conversion daemon? **No**
- Should the data be purged as well as the package files? **No**
- Run the Kerberos V5 administration daemon (kadmind)? **Yes**
- Kerberos servers for your realm: **krb1.example.com** # (Make sure your DNS resolves krb1.example.com to the NETWORK IP of the server, NOT 127.0.0.1!).
- Administrative server for your Kerberos realm: **krb1.example.com** #(Make sure your DNS resolves krb1.example.com to the NETWORK IP of the server, NOT 127.0.0.1!)
- Create the Kerberos KDC configuration automatically? **Yes** #New question added in Kerberos 1.7 and newer packages

As soon as the installation is done, the Kerberos admin server (kadmind) and the KDC will start. Kadmind will fail, initially, as there are no Kerberos realms created.

Initial configuration

To create the Kerberos realm, invoke Debian-specific command krb5_newrealm:

```
sudo krb5_newrealm
```

The command will first ask for the master password, and then proceed with creating the realm using the name we defined earlier in the Debconf step (EXAMPLE.COM). Make sure to pick a good password, and possibly write it down for later reference as it is hardly ever used, yet very important.

After the realm is created, we will need to adjust the Kerberos config file, /etc/krb5.conf. That file also needs to be the same on all Kerberos servers and clients belonging to the same realm.
/etc/krb5.conf is split into sections; you should search for section [domain_realm] and append your definition:

```
    .example.com = EXAMPLE.COM
    example.com = EXAMPLE.COM
```

At the bottom of the file, you should add the logging section:

```
[logging]
kdc = FILE:/var/log/kerberos/krb5kdc.log
admin_server = FILE:/var/log/kerberos/kadmin.log
default = FILE:/var/log/kerberos/krb5lib.log
```

To create the logging directory and set up permissions, run:

```
sudo mkdir /var/log/kerberos
sudo touch /var/log/kerberos/{krb5kdc,kadmin,krb5lib}.log
sudo chmod -R 750  /var/log/kerberos
```

Don't forget to also restart the log monitoring command — the tail program needs to pick up the new log files from the kerberos/ directory.

It is also a good idea to disallow "weak enctypes" in Kerberos. These are the cryptographic algorithms that are no longer considered secure (or at least robust), and currently include all single-DES enctypes. The option called "allow_weak_crypto" was added in Krb 1.7, defaulting to true. In Kerberos 1.8 and later, weak enctypes are disabled by default and no manual setting is necessary.
So, if you are using Kerberos 1.7, search for the section called "[libdefaults]" in /etc/krb5.conf (usually at the top of the file) and append the following definition:

```
allow_weak_crypto = false
```

Finally, to apply all changes, run:

```
sudo service krb5-admin-server restart
sudo service krb5-kdc restart
```

Initial test

It is already the time to test the installation. We assume that both the admin server and the KDC can be restarted with no errors (which should be no problem to determine if you're monitoring the log files as advised).

As the first test, we will run command kadmin.local on the server.

Start kadmin.local, then type listprincs. That command should print out the list of principals (user, host and service accounts) in the database. The whole session should look like this:

```
sudo kadmin.local

Authenticating as principal root/admin@EXAMPLE.COM with password.

kadmin.local:  listprincs

K/M@EXAMPLE.COM
kadmin/admin@EXAMPLE.COM
kadmin/changepw@EXAMPLE.COM
kadmin/history@EXAMPLE.COM
kadmin/krb1.EXAMPLE.COM@EXAMPLE.COM
krbtgt/EXAMPLE.COM@EXAMPLE.COM

kadmin.local:  quit
```

The kadmin command ordinarily requires principal name and password before letting anyone access the administrative interface. However, kadmin.local is a variant of the command that must be run locally on the same machine as the KDC, and with administrator privileges. It is then able to open the Kerberos database file directly (taking advantage of Unix file permissions), without requiring extra privileges and without using the kadmind (Kerberos master server) daemon.

Principal Names

In the test step above, you might have noticed principal names similar to kadmin/admin@EXAMPLE.COM. The general naming syntax for principals is SPEC@+REALM+, where *SPEC* by convention consists of components separated by "/", and the default REALM can be omitted.

In the case of user names, the first component identifies the user name, and the second component, if present, identifies user role. For regular users, there will usually be one principal with no special role, named simply *USERNAME*. But when administrative or other roles are required, there will be no need to condense them all to one "root" prinicpal — each user can simply be given conveniently named additional principals with special privileges, such as *USERNAME*/+admin+.

In the case of service names, the components will be used to identify service and hostname, such as *host*/+krb1.example.com+ or *ldap*/+krb1.example.com+.
"host" is somewhat of a misnomer from today's perspective — it has nothing to do with host per-se, but is actually the service name for all remote shell protocols, such as rsh, rlogin and ssh).

Access rights

Let's take a look at the /etc/krb5kdc/kadm5.acl file; it defines user access rights for the Kerberos database. For regular users with no special privileges, no action will be required. For admin users, we will want to grant all privileges. To do this, make sure the following line is present in the file and enabled (that is, without the comment '#' character at the beginning):

```
*/admin *
```

(While the above syntax might remind you of shell globbing, unfortunately it does not work that way at all. The only matching character supported is the asterisk ("*"), it does not match multiple components, and it can only be used in form of "component1/*" or "*/component2".)

Make sure to restart the admin server to apply /etc/krb5kdc/kadm5.acl changes:

```
sudo service krb5-admin-server restart
```

Kerberos policies

Kerberos "policies" offer an elegant way to sort principals into a kind of categories and to automatically apply corresponding defaults onto newly created principals.

Let's create four basic policies: for *admins*, *hosts*, *services* and *users*. In this example, each policy will define minimum password strength (measured in number of character classes present in the password, from 1 to 5), but a few other options can be set — run addpol (the supported abbreviation of add_policy) if you're curious.

```
sudo kadmin.local
```

    Authenticating as principal root/admin@EXAMPLE.COM with password.

```
kadmin.local:  add_policy -minlength 8 -minclasses 3 admin
kadmin.local:  add_policy -minlength 8 -minclasses 4 host
kadmin.local:  add_policy -minlength 8 -minclasses 4 service
kadmin.local:  add_policy -minlength 8 -minclasses 2 user
kadmin.local:  quit
```

Creating first privileged principal

As you might have noticed, the kadmin.local command identified us as the principal root/admin. Still, that principal does not actually exist in the database so we might as well create it now. Once the principal is actually there, we'll be able to connect to the administrative server using kadmin from any machine within the Kerberos realm, and not just by using kadmin.local on the Kerberos server.

Creating a principal based on your regular identity (such as *USERNAME*/+admin+) is preferred over creating one called root/admin, so feel free to subsitute "root" in the following transcript (and the rest of the guide) with your username:

```
sudo kadmin.local
```

```
Authenticating as principal root/admin@EXAMPLE.COM with password.
```

```
kadmin.local:  addprinc -policy admin root/admin
```

```
Enter password for principal "root/admin@EXAMPLE.COM": PASSWORD
Re-enter password for principal "root/admin@EXAMPLE.COM": PASSWORD
Principal "root/admin@EXAMPLE.COM" created.
```

```
kadmin.local:  quit
```

Kadmin test

Now that the root/admin principal exists in the Kerberos database, we should be able to use kadmin just as we used kadmin.local. The only exception, of course, is that kadmin will prompt for a password to connect to the Kerberos admin server.
Double-check that all the permissions are granted to admin roles in the /etc/krb5kdc/kadm5.acl, and that the admin server has been restarted to read the new configuration; then proceed to test kadmin connection:

```
sudo kadmin -p root/admin
```

```
Authenticating as principal root/admin@EXAMPLE.COM with password.

Password for root/admin@EXAMPLE.COM: PASSWORD
```

```
kadmin:  listprincs
```

```
K/M@EXAMPLE.COM
root/admin@EXAMPLE.COM
kadmin/admin@EXAMPLE.COM
kadmin/changepw@EXAMPLE.COM
kadmin/history@EXAMPLE.COM
kadmin/krb1.EXAMPLE.COM@EXAMPLE.COM
krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

```
kadmin:  quit
```

Creating Unprivileged Principal

Let's add a principal that will correspond to your regular, unprivileged user account. In our example, the username will be called "user1". We've essentially performed this procedure for the root/admin principal above, but we'll repeat it here for your regular user account, using a different policy, replacing user1 with your username:

```
sudo kadmin -p root/admin
```

```
Authenticating as principal root/admin@EXAMPLE.COM with password.

Password for root/admin@EXAMPLE.COM: PASSWORD
```

```
kadmin:  addprinc -policy user user1
```

```
Enter password for principal "user1@EXAMPLE.COM": PASSWORD
Re-enter password for principal "user1@EXAMPLE.COM": PASSWORD
Principal "user1@EXAMPLE.COM" created.
```

```
kadmin:  quit
```

Obtaining Kerberos ticket

As hinted in the introduction, each user is expected to type in the password once, to obtain the initial TGT (Ticket-granting Ticket).

Obtained tickets are saved to a so-called ticket cache, which is most commonly a file named /tmp/krb5cc_*, stored on the user's workstation.

Let's run the klist command to inspect our ticket cache (run this command under your regular, non-privileged username). As one might guess, since we did not obtain any tickets yet, the cache will be empty:

```
klist -5f
```

```
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_0)
```

Let's use kinit now to obtain the ticket, and then re-inspect the ticket cache. If the command seemingly "hangs" and does nothing, wait a few seconds — DNS misconfiguration may cause a delay.

```
kinit
```

```
Password for user1@EXAMPLE.COM: PASSWORD
```

```
klist -5f
```

```
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: user1@EXAMPLE.COM

Valid starting     Expires            Service principal
11/22/06 22:30:36  11/23/06 08:30:33  krbtgt/EXAMPLE.COM@EXAMPLE.COM
renew until 11/23/06 22:30:34, Flags: FPRIA
```

If you remember the story from the beginning, you will recognize the "krbtgt" to be the Ticket-granting Ticket.

klist switch -f produced the "renew until..." line. The meanings of each flag letter are not important at this stage, but long-term it is useful to get into the habit of using -f, and the flags descriptions can be looked up in the klist(1) manpage.

All great. Let's run kdestroy to terminate the ticket now:

```
kdestroy
```

Troubleshooting

If there are any problems, check the log files first:

```
cd /var/log; sudo tail -F daemon.log sulog user.log auth.log debug kern.log syslog dmesg messages kerberos/{krb5kdc,kadmin,krb5lib}.log
```

Related issues:

Related to GNU/Linux Administration - Feature #164: Centralized User Authenti...	**Closed**	**08/09/2013**
Related to GNU/Linux Administration - Feature #162: Installing OpenLDAP with ...	**Closed**	**08/08/2013**
Related to GNU/Linux Administration - Support #167: Backing Up and Restoring ...	**Closed**	**08/12/2013**

History

#1 - 08/08/2013 03:26 PM - Daniel Curtis

- Description updated

#2 - 08/08/2013 03:27 PM - Daniel Curtis

- Description updated

#3 - 08/08/2013 03:29 PM - Daniel Curtis

- Description updated

#4 - 02/16/2015 02:25 PM - Daniel Curtis

- Project changed from 22 to GNU/Linux Administration

- Category set to Domain Controller