SLAC-PUB-10466
June 2004

Centralized Authenitcation with Kerberos 5, Part I

Alf Wachsmann
Stanford Linear Accelerator Center

Submitted to Linux Journal

Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94309

Work supported by Department of Energy contract DE–AC03–76SF00515.

Centralized Authentication with Kerberos 5 – Part I

1. Introduction

Account administration in a distributed Unix/Linux environment can become very complicated and messy if done by hand Large sites use special tools to deal with this problem. I will describe how even very small installations like your three computer network at home can take advantage of the very same tools.

The problem in a distributed environment is that password and shadow files need to be changed individually on each machine if an account change occurs. Account changes include: password change, addition/removal of accounts, name change of an account (UID/GID changes are a big problem in any case), additional or removed login privileges to a (group of) computer(s), etc.

In this article, I will show how Kerberos 5 solves the authentication problem in a distributed computing environment. A second article will describe a solution for the authorization problem.

The problem of authenticating users to a computer is mostly solved through passwords, although other methods are available (smart cards, biometrics, etc.). Traditionally, these passwords are stored in /etc/passwd and later in /etc/shadow. Because these files are local to a computer, it is a big problem keeping all these files up to date. Directory services like NIS, NIS+ and LDAP were invented to solve this problem. These services, however, introduce a new problem: they work over the network and, hence, expose passwords, which are only weakly encrypted, on this network.

The authentication protocol implemented by Kerberos combines the advantages of being a networked service and of eliminating the need to communicate passwords between computers altogether.

To do so, Kerberos requires you to run two daemons on a secure server. The Key Distribution Center (KDC) daemon handles all password verification requests and the generation of Kerberos credentials, called Ticket Granting Tickets (TGTs). A second daemon, the Kerberos Administration daemon, allows you to administer (add, delete, modify) accounts remotely without logging into the computer running the Kerberos daemons. It also handles password change requests from users. Note that, with Kerberos, only a password change ever requires transmitting a strongly encrypted password over the network.

The Kerberos KDC grants a temporary credential, a TGT, to the account whenever a user gets authenticated. Typically, these credentials have a lifetime of 10 or 24 hours. This lifetime can be configured and should be no longer than 24 hours in case the TGT gets stolen. A thief can use it only for the remaining TGT lifetime. The credential expiration will cause no issues if you are using Kerberos only for authentication as described in this article. However, if you are using kerberized services, you need to train your users to obtain new credentials after their lifetime expires even though they are still logged in.

Kerberos was invented at MIT. The latest version is Kerberos 5, with its protocol defined in RFC 1510. Today, two Kerberos implementations are freely available: from MIT (U.S. and Canada: http://web.mit.edu/kerberos/dist/; everybody else: http://www.crypto-publish.org/mit-kerberos5/) and Heimdal from KTH in Sweden (http://www.pdc.kth.se/heimdal/). MIT's Kerberos 5 is included in Red Hat Linux, whereas Heimdal is included in Suse's and Debian's Linux distribution. There are Kerberos 5 implementations coming with Microsoft (Windows 2000 and above), Sun's Solaris (SEAM, Solaris 2.6 and above), and Apple's Mac OS X.

I will use MIT's Kerberos distribution throughout this article because it offers simple password quality checking, password aging and password history out of the box.

2. Prerequisites

You have to meet two prerequisites before you can switch authentication over to Kerberos.

The clocks on all computers that will be members of your Kerberos installation need to be synchronized to the clock of the machine running your KDC. The simplest way of doing this is to use the Network Time Protocol (NTP) on all your machines.

The second requirement is harder to meet. All account names, UIDs and GIDs have to be the same on all your computers. This is necessary because each of these accounts will become a new and independent Kerberos account, called a "principal". You will have to go through all your local /etc/passwd files and check whether this requirement is met. If not, you need to consolidate your accounts! If you want to add Windows or Mac OS X clients to your Kerberos installation, you need to look at all accounts on those machines as well.

3. Compiling

If you want to compile the Kerberos distribution yourself, follow these steps for MIT Krb5. If you decide to use the Kerberos package that comes with your Linux distribution, just install the packages and skip these steps.

1. Get the source from one of the URLs above.
   Get the PGP signature of the source package and
   Verify the integrity of the downloaded source with
   ```
   % gpg --verify krb5-1.3.1.targz.asc
   ```
2. Unpack the source with
   ```
   % tar zxvf krb5-1.3.1.tar.gz
   ```
3. and change into the source directory
   ```
   % cd krb5-1.3.1/src
   ```

4. Execute
   ```
   % ./configure --help
   ```
   to find out whether you need to use special configure options for your site. /usr/local/ is the default installation directory. If you need this software in another directory, use a "--prefix=/new/path/to/directory" flag in the next step.
5. In almost all cases, the default should be fine:
   ```
   % ./configure
   ```
6. Compile the package with
   ```
   % make
   ```
7. and check whether everything compiled correctly with
   ```
   % make check
   ```
8. If everything looks ok, install the package with
   ```
   % sudo make install
   ```
   Note: Never compile code as "root"! Use "root" privileges only when necessary   as in these installation steps.
9. You now have MIT Krb5 installed in /usr/local/
10. Some additional directories need to be created by hand and their permissions set:
    ```
    % sudo mkdir -p /usr/local/var/krb5kdc
    % sudo chown root /usr/local/var/krb5kdc
    % sudo chmod 700 /usr/local/var/krb5kdc
    ```

4. Creating your realm

A Kerberos "realm" is an administrative domain which has its own Kerberos database. Each Kerberos realm has its own set of Kerberos servers. The name of your realm can be anything, but should reflect your place in the DNS world. If your new Kerberos realm is for your entire DNS domain "example.com", you should give the same name (with all capital letters - this is a Kerberos convention) to your Kerberos realm: "EXAMPLE.COM". Or, if you are setting up a new realm for your engineering department in example.com, a realm name of "ENG.EXAMPLE.COM" could be chosen.

The first step for creating your own realm is to create a /etc/krb5.conf file that contains all the necessary information about this realm. The krb5.conf file needs to go onto every computer that wants access to your new Kerberos realm. Here is an example file for the realm "EXAMPLE.COM" with the KDC and administration servers running on machine "kdc.example.com":

```
[libdefaults]
        # determines your default realm name
        default_realm = EXAMPLE.COM

[realms]
        EXAMPLE.COM = {
                # specifies where the servers are and on which ports
```

```
                    # they listen (88 and 749 are the standard ports)
                    kdc = kdc.example.com:88
                    admin_server = kdc.example.com:749
            }

        [domain_realm]
            # maps your DNS domain name to your Kerberos realm name
            .example.com = EXAMPLE.COM

        [logging]
            # determines where each service should write its logging info
            kdc = SYSLOG:INFO:DAEMON
            admin_server = SYSLOG:INFO:DAEMON
            default = SYSLOG:INFO:DAEMON
```

The next file, /usr/local/var/krb5kdc/kdc.conf, configures the KDC server. It only needs to be on the computer running the KDC daemon. Every entry has a reasonable default. Creating an empty file should be sufficient for most cases:

```
    % sudo touch /usr/local/var/krb5kdc/kdc.conf
```

The following commands need to be executed on the computer that will become your KDC.

The command

```
    % sudo /usr/local/sbin/kdb5_util create -s
```

creates an initial Kerberos database for the new realm. It will ask you for the database master password for the new realm and stores it in a file (/usr/local/var/krb5kdc/.k5.EXAMPLE.COM).

This command also creates a first set of principals in your Kerberos 5 account database. You can list them by using

```
    % sudo /usr/local/sbin/kadmin.local
```

and then typing "listprincs" (w/o the quotes) at the "kadmin.local:" prompt. This will print the list

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

Note, at this time we are not ready yet to use the remote version of the kadmin tool.

Before you start creating any principals in your new realm, you should define a "policy" that determines how passwords are handled:

```
    kadmin.local:  add_policy -maxlife 180days -minlife 2days -minlength 8 -minclasses 3 -
history 10 default
```

This defines a "default" policy that is used for every principal we create from here on. It determines that the maximum lifetime for passwords is 180 days. The minimum lifetime is 2 days. The minimum password length is 8 characters and these characters have to come from 3 different classes out of these 5 available ones: lower case, upper case, numbers, punctuation, others. A history of the last 10 passwords is kept to prevent reuse. If you want to have passwords checked against a dictionary, add a "dict_file" definition like

```
        [realms]
                EXAMPLE.COM = {
                        dict_file = /usr/share/dict/words
                }
```

to your kdc.conf file which was empty so far.

Now you are ready to create an administration principal for yourself:

```
        kadmin.local: addprinc john/admin
```

Adjust the name to _your_ account name but keep the "/admin"! It will ask twice for a new password for this principal. You can look at the new account with

```
        kadmin.local:  getprinc john/admin
```

which will print something like:

```
        Principal: john/admin@EXAMPLE.COM
        Expiration date: [never]
        Last password change: Wed Dec 24 09:55:17 PST 2003
        Password expiration date: Mon Jun 21 10:55:17 PDT 2004
        Maximum ticket life: 1 day 00:00:00
        Maximum renewable life: 0 days 00:00:00
        Last modified: Wed Dec 24 09:55:17 PST 2003 (root/admin@EXAMPLE.COM)
        Last successful authentication: [never]
        Last failed authentication: [never]
        Failed password attempts: 0
        Number of keys: 2
        Key: vno 1, Triple DES cbc mode with HMAC/sha1, no salt
        Key: vno 1, DES cbc mode with CRC-32, no salt
        Attributes:
        Policy: default
```

Exit the "kadmin.local" program by typing "quit" and start the KDC daemon with

```
        % sudo /usr/local/sbin/krb5kdc
```

Get a Kerberos 5 TGT by typing

```
        % /usr/local/bin/kinit john/admin@EXAMPLE.COM
```

and look at your TGT with

```
        % /usr/local/bin/klist
        Ticket cache: FILE:/tmp/krb5cc_5828
        Default principal: john/admin@EXAMPLE.COM

        Valid starting     Expires            Service principal
        12/23/03 14:15:39  12/24/03 14:15:39  krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

Congratulations! You just did your first successful Kerberos authentication.

You now need to specify which privileges this administration account should have. This is determined by entries in the file /usr/local/var/krb5kdc/kadm5.acl You can give "john/admin" permissions to administer all principals, indicated by the wildcard character "*", by adding the line

```
john/admin@EXAMPLE.COM  *
```

to this file.

Before you can start using the administration daemon (kadmind) over the network, you have to create a keytab file containing the key for one of the kadmin principals that where created when we initialized our realm.

```
kadmin.local:  ktadd -k /usr/local/var/krb5kdc/kadm5.keytab kadmin/changepw
```

Now everything is ready for the Kerberos administration daemon. Start it with

```
% sudo /usr/local/sbin/kadmind
```

This daemon allows you to administer your Kerberos principals remotely, i.e. without logging into your KDC, using the "kadmin" client tool.

If you want your Kerberos daemons to start up automatically at boot time, add them to your KDC's /etc/rc files.

With the Kerberos TGT obtained above, start the remote administration tool

```
% /usr/local/sbin/kadmin
Authenticating as principal john/admin@EXAMPLE.COM with password.
Password for john/admin@EXAMPLE.COM:
```

4.1 Adding new accounts

New accounts still need to be added to your shadow file or password map. However, instead of putting the encrypted password into these places, you have to create a new Kerberos principal and store the password in the KDC.

Use the "kadmin" tool for this purpose

```
% /usr/local/sbin/kadmin
```

and add a principal for a regular users with

```
kadmin:  addprinc john
NOTICE: no policy specified for john@EXAMPLE.COM; assigning "default"
Enter password for principal "john@EXAMPLE.COM":
Re-enter password for principal "john@EXAMPLE.COM":
Principal "john@EXAMPLE.COM" created.
```

The password you have entered during this principal creation process is the one "john" will need to type in order to obtain a Kerberos TGT or just to login to a computer configured to use your Kerberos 5 realm.

You can now either create principals for all your accounts by hand or you can use the technique described in the migration section below.

4.2 Adding Slave KDCs

If you plan to use Kerberos in production at your site, you should plan on using additional slave KDCs to make your installation more fault tolerant. For this, the master KDC needs to have an additional "propagation service" installed that sends updated versions of the KDC database to all slave servers. The slave servers need to have a receiving end for the propagation service installed. See the MIT documentation for how to set this up.

5 Configuring the clients

The easiest way to enable a computer for Kerberos authentication is to use a Pluggable Authentication Module (PAM). Because it uses Kerberos API calls, it needs a working /etc/krb5.conf file. So, the first step is to copy the /etc/krb5.conf file from your KDC (see above) onto each client machine.

Kerberos is not only used to authenticate users. It is also used to authenticate computers, to prevent you from logging into a machine with a hijacked IP address. For this to work, each computer needs its own Kerberos principal with the key (the "password") stored in a file (a "keytab" file). Principals for computers have the special form host/<hostname>.example.com@EXAMPLE.COM.

The first step is to create a new principal for each of your client machines. The following commands are using the computer name "client1" as example. Replace the string "client1" with the hostname of the client computer. Login to every one of your client computers and execute

```
% sudo /usr/local/sbin/kadmin
kadmin: addprinc -randkey host/client1.example.com@EXAMPLE.COM
```

which assignes a random password to the new principal.

Then extract the key into a keytab file with

```
kadmin: ktadd host/client1.example.com@EXAMPLE.COM
```

which will create a file /etc/krb5.keytab. To have write permissions to the /etc/ directory, you need to run the kadmin command with sudo. Just creating a new principal would not have required these special privileges. Watch out for the ownership and file permissions of /etc/krb5.keytab! It has to be readable only by root. Otherwise, the security of this machine is compromised.

There are several PAM modules for Kerberos 5 available and all are called pam_krb5. Most of these will not work any more due to some API changes in MIT Kerberos5 version 1.3. Your best choice right now is to use the PAM module that comes with your Linux distribution.

If you really need to (or want to) compile your own, here is how to get a working version of the module shipped by Red Hat. The following worked for me on a Red Hat 9 system:

```
	% cvs -d :pserver:anoncvs@elvis.redhat.com:/usr/local/CVS login
```
Type in your email address as password.
```
	% cvs -d :pserver:anoncvs@elvis.redhat.com:/usr/local/CVS co pam_krb5
	% cd pam_krb5
```

Your $PATH environment variable has to have the Kerberos distro of your choice _first_ in case you have more than one distribution on your computer.

Example:
```
	% PATH=/usr/local/bin:$PATH
```
if you have installed your own version in /usr/local

Then execute
```
	% ./autogen
```
to have automake and friends create the necessary Makefiles.

If you get an automake error, run this command:
```
	% perl -pi -e "s/AC_CONFIG_HEADER/AM_CONFIG_HEADER/" configure.ac
```
and run again
```
	% ./autogen
```
Then compile and install the package with
```
	% make
	% sudo make install
```

Now add the new PAM module to your system's authentication stack by editing the file /etc/pam.d/system-auth (on Red Hat systems). The entries should look similar to these Red Hat 9 entries:
```
auth        required      /lib/security/$ISA/pam_env.so
auth        sufficient    /lib/security/$ISA/pam_unix.so likeauth nullok
auth        sufficient    /lib/security/$ISA/pam_krb5.so use_first_pass
auth        required      /lib/security/$ISA/pam_deny.so

account     required      /lib/security/$ISA/pam_unix.so
account     [default=bad success=ok user_unknown=ignore service_err=ignore
            system_err=ignore] /lib/security/$ISA/pam_krb5.so

password    required      /lib/security/$ISA/pam_cracklib.so retry=3 type=
password    sufficient    /lib/security/$ISA/pam_unix.so nullok use_authtok md5 shadow
password    sufficient    /lib/security/$ISA/pam_krb5.so use_authtok
password    required      /lib/security/$ISA/pam_deny.so

session     required      /lib/security/$ISA/pam_limits.so
session     required      /lib/security/$ISA/pam_unix.so
session            optional       /lib/security/$ISA/pam_krb5.so
```

These changes will make every program that has the "system-auth" PAM stack in its PAM configuration file (see the other files in /etc/pam.d/) use Kerberos for its authentication.

5.1 Unix clients using a Windows KDC

If you already have a working Windows Active Directory (AD) KDC installation, you can use it as the master KDC for your Linux/Unix machines. In this case, you can skip the entire server installation and only do the above described setup of your clients. Your /etc/krb5.conf file needs to define the Windows KDC instead of a Unix KDC. For more information about how to create and copy a keytab file and this scenario in general, see http://www.microsoft.com/windows2000/techinfo/planning/security/kerbsteps.asp

6 Migration from local passwords or NIS/LDAP to Kerberos

Now that you have a working Kerberos 5 realm and your clients configured, you have to convert all your user accounts. So far, the passwords for your accounts are stored either in machine's local /etc/shadow files or in a NIS/LDAP password map. These passwords are encrypted with a one-way hash function that makes it impossible (or at least impractical for people without a supercomputer to crack them) to just "convert" everything into Kerberos 5 format. A good way to migrate from you current situation to Kerberos, is to use pam_krb5_migrate from http://freshmeat.net/projects/pam_krb5_migrate. This stackable PAM module can be installed on a few computers and every time someone logs on, it creates a new principal for this account in your Kerberos 5 KDC reusing the account's current password.

After everybody has logged on to these special machines, all your users have a corresponding Kerberos 5 principal and you can replace the passwords in your local files or your NIS/LDAP password map with a placeholder, like "krb5". The Kerberos PAM module will authenticate your users from now on. At this point, you can also remove pam_krb5_migrate from the migration systems.

7.1 Unix and Windows

If you have a number of Windows machines in your group, you can use your Unix KDC for them as well. This will only work, however, if your Windows clients don't already belong to Windows AD domain with Kerberos and the account names are the same in Kerberos and Windows. See the above referenced Microsoft document for details.

7.2 Unix and Mac OS X

Using Mac OS X clients in your Kerberos 5 realm is as easy as configuring the names of your Unix KDCs on your Macs. Again, account names have to match.

8 Kerberized Applications

Now that you have Kerberos up and running, you can use services that make use of it. You could install kerberized telnet and ftp but you should really use ssh. You could kerberize your Apache web server and your Mozilla web browser (http://meta.cesnet.cz/software/heimdal/negotiate.en.html). Before Kerberos, you would have to type your password when using these services. With Kerberos, all these applications are using your stored Kerberos credentials and use them internally to authenticate you for the respective service. This is what many mean by "single-sign-on."

9 Further Reading

The MIT Kerberos 5 distribution comes with good documentation about installing, administrating and using Kerberos. The recently published O'Reilly book "Kerberos: The Definitive Guide" by Jason Garman (ISBN 0-596-00403-6) is a good resource for setting up a new realm and getting interoperability with other operating systems to work.