Centralized Authentication with Kerberos 5, Part I

Alf Wachsmann

Abstract

Kerberos can solve your account administration woes.

Account administration in a distributed UNIX/Linux environment can become complicated and messy if done by hand. Large sites use special tools to deal with this problem. In this article, I describe how even small installations, such as your three-computer network at home, can take advantage of the 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 changes, addition/removal of accounts, account name changes (UID/GID changes are a big problem in any case), addition/removal of login privileges to computers and so on. I also explain how the Kerberos distribution solves the authentication problem in a distributed computing environment. In Part II, I will describe a solution for the authorization problem.

The problem of authenticating users to a computer is solved mostly through passwords, although other methods, including smart cards and biometrics, are available. These passwords had been stored in /etc/passwd, but now with shadow passwords, they reside in /etc/shadow. Because these files are local to a computer, it is a big problem keeping them up to date. Directory services such as NIS, NIS+ and LDAP were invented to solve this problem. These services, however, introduce a new problem: they work over the network and expose passwords, which are encrypted only weakly.

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 dæmons on a secure server. The Key Distribution Center (KDC) dæmon handles all password verification requests and the generation of Kerberos credentials, called Ticket Granting Tickets (TGTs). A second dæmon, the Kerberos Administration dæmon, allows you to add, delete and modify accounts remotely without logging in to the computer running the Kerberos dæmons. It also handles password change requests from users. 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 during the process of authenticating the user. 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 is stolen; a thief could use it only for the remaining TGT lifetime. The credential expiration causes 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 current ones expire, even though they still are 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 (see the on-line Resources). MIT's Kerberos 5 is included in Red Hat Linux, whereas Heimdal is included in SuSE's and Debian's Linux distributions. Kerberos 5 implementations also are included in Microsoft Windows (2000 and later), in Sun's Solaris (SEAM, Solaris 2.6 and above) and Apple's Mac OS X. I use MIT's Kerberos distribution throughout this article because it offers simple password quality checking, password aging and password history out of the box.

Prerequisites

You have to meet two prerequisites before you can switch authentication over to Kerberos. First, the clocks on all computers to be included in your Kerberos installation need to be synchronized to the clock of the machine running the 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 becomes a new and independent Kerberos account, called a principal. You 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 the accounts on those machines as well.

If you decide to use the Kerberos package that comes with your Linux distribution, simply install it. If you want to compile the Kerberos distribution yourself, follow the instructions below.

Building and Installing MIT Kerberos

1) Get the source from one of the URLs listed in the on-line Resources. Get the PGP signature of the source package and verify the integrity of the downloaded source with:

```
% gpg --verify krb5-1.3.4.targz.asc
```

2) Unpack the source with:

```
% tar zxvf krb5-1.3.4.tar.gz
```

3) Change into the source directory:

```
% cd krb5-1.3.4/src
```

4) Execute:

```
% ./configure --help
```

This tells you if 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
```

I had a problem with one file in the krb5-1.3.4/src/kadmin/testing/util directory, which can be safely ignored. Restart the compilation with **`% make -i`** in this case.

7) Check whether everything compiled correctly with:

```
% make check
```

8) If everything looks okay, install the package with:

```
% sudo make install
```

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/. 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
```

If you really need or want to compile your own PAM module, here are the steps to get a working version of the module shipped by Red Hat. Get the source (see Resources) and upack it with:

```
% tar zxf pam_krb5-1.3-rc7.tar.gz

% cd pam_krb5-1.3-rc7
```

Your $PATH environment variable has to have the Kerberos distribution of your choice first, in case you have more than one distribution on your computer. For example:

```
% PATH=/usr/local/bin:$PATH
```

if you have installed your own version in /usr/local. Then execute:

```
% ./configure
```

Then compile and install the package with:

```
% make
% sudo make install
```

Creating Your Realm

A Kerberos realm is an administrative domain that has its own Kerberos database. Each Kerberos realm has its own set of Kerberos servers. The name of your realm can be anything, but it should reflect your place in the DNS world. If the 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 your 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 be on 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 needs to be on only the computer running the KDC dæmon. 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 asks 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`** at the kadmin.local: prompt. This prints the list:

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

At this time, we are not ready 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 input defines a default policy used for every principal we create from now on. It determines that the maximum lifetime for passwords is 180 days. The minimum lifetime is two days. The minimum password length is eight characters, and these characters have to come from three different classes out of these five available ones: lowercase, uppercase, numbers, punctuation and others. A history of the last ten passwords is kept to prevent reuse. If you want to have passwords checked against a dictionary, add a dict_file definition such as:

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

to your kdc.conf file.

You now 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 then asks twice for a new password for this principal. You can look at the new account with:

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

which prints 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 dæmon 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 completed your first successful Kerberos authentication.

You now need to specify which privileges this administration account should have, which 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 dæmon (kadmind) over the network, you have to create a keytab file containing the key for one of the kadmin principals 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 dæmon. Start it with:

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

This dæmon allows you to administer your Kerberos principals remotely, without logging in to your KDC, using the kadmin client tool. If you want your Kerberos dæmons 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:
```

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.

Using the kadmin tool:

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

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 needs to enter in order to obtain a Kerberos TGT or to log in to a computer configured to use your Kerberos 5 realm.

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

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.

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) to each client machine.

Kerberos is used not only to authenticate users, it also is used to authenticate computers, to prevent you from logging in to 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 use the computer name client1 as an example. Replace the string client1 with the hostname of the client computer. Log in to every one of your client computers and execute:

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

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

Several PAM modules for Kerberos 5 are available and all are called pam_krb5. Most of these do not work any more due to some API changes in MIT Kerberos 5 version 1.3. Your best choice right now is to use the PAM module that comes with your Linux distribution. See the section above on how to build a PAM module for Kerberos 5 from source.

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 make every program with the system-auth PAM stack in its PAM configuration file (see the other files in /etc/pam.d/) use Kerberos for its authentication.

Interoperation with Non-Linux Clients

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 do only 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 on how to create and copy a keytab file and this scenario in general, see Resources.

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

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.

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 the 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 or to convert everything into Kerberos 5 format. A good way to migrate from your current situation to Kerberos is to use pam_krb5_migrate (see Resources). This stackable PAM module can be installed on a few computers; 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. You then can replace the passwords in your local files or your NIS/LDAP password map with a placeholder, such as krb5. The Kerberos PAM module authenticates your users from now on. At this point, you also can remove pam_krb5_migrate from the migration systems.

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 really should use SSH. You could Kerberize your Apache Web server and your Mozilla Web browser. 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.

Resources for this article: http://www.linuxjournal.com/article/7706.