

Network Security with Kerberos

by Boris Loza, PhD, CISSP

In Greek mythology, Kerberos (Cerberus in the Latin spelling) is the three-headed dog that guarded the entrance to Hades. In the computer world, Kerberos is a network authentication protocol. It's designed to provide strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server have used Kerberos to prove their identities, they can also encrypt all of their communications.

Massachusetts Institute of Technology created Kerberos in the mid 1980s as part of the Athena project. Today Kerberos version 5 (Kerberos V5) is the latest implementation and is available as a product from many different vendors for various computing platforms. In this article, we'll show you how to use Kerberos V5 in the Solaris environment as a solution to your network security problems.

How Kerberos works

When you log on to a workstation running Kerberos, you provide your username and password. The workstation sends the Key Distribution Center (KDC) on the Kerberos Server a message consisting of your username and the current time encrypted with your password. The KDC looks up your username, determines your password, and attempts to decrypt the encrypted time. If the server can decrypt the current time, it then creates a ticket-granting ticket, encrypts it with your password, and sends it back to you. The user can then contact the KDC to obtain tickets for any network service within the Kerberos realm. The Kerberos realm is the domain of servers and users who are known to the KDC.

As you can see from this scenario, the user's password is never transmitted on the network in clear text. Kerberos will give you credentials only if you have an entry in the Kerberos server's database. This database includes a Kerberos principal (an identifying string, which is often just your user name) and your Kerberos password. Every Kerberos user must have an entry in this database.

There are two types of KDC: master and slave. Each KDC contains a copy of the Kerberos database. The master KDC contains the master copy of the database, which it propagates to the slave KDC(s) at regular intervals. All database changes (such as password changes) are made on the master KDC(s).

Obtaining and installing Kerberos

Although Kerberos is freely available, only US and Canadian citizens can legally download it from MIT. To do so, go to <http://web.mit.edu/network/kerberos-form.html> and fill out the request form. After downloading the latest release, unpack it with a `tar`. You'll get source, documentation, and digital signature files all in `.tar.gz` format. As root, `gunzip` and `untar` each of the above `.tar.gz` files and switch into the `<where-you-install>/krb-5.1.1.1/src` directory. To create the Makefile, run `configure` with appropriate options. For example,

```
./configure --with-cc=gcc prefix=/usr/local/kerbero
```

These options tell `configure` to use the GNU `gcc` compiler and install all binaries and man pages under the `/usr/local/kerberos` directory. For many other options, check the Kerberos V5 Installation Guide that comes with the distribution.

After `configure` is finished, make all binaries by typing `make` and install the utilities by typing `make install`.

Configuring the primary KDC

The machine that you choose as your Kerberos server is the key to your network security. Therefore, it must be as secure as possible. Make sure that:

- The server is physically secure and all the latest OS patches have been applied.
- There are no user accounts on the machine except for the Kerberos administrator.
- All unnecessary network services are commented out in the `/etc/inetd.conf` file.
- You have one more physically and computationally secure machine to serve as alternative server (as a slave KDC), in case there's a hardware problem or a network outage.

For detailed instructions on how to secure a Solaris server, you can check "Securing Solaris: Step-by-Step." Ref. No. SG110A from SANS.

For the sake of simplicity, let's assume that you have the following configuration:

- Domain name—`tree.com`
- Primary KDC—`birch.tree.com`
- Slave KDC—`pine.tree.com`
- Kerberos Client—`oak.tree.com`

Configuring the master KDC

Suppose you've already installed Kerberos on `birch.tree.com` following the instructions from the section "Obtaining and installing Kerberos." The next step is to modify the configuration files `krb5.conf` shown in **Listing A** and `kdc.conf` shown in **Listing B** that come with the distribution (`<where-you-install>/krb5-1.1.1/src/config-files`) to reflect the correct information, such as the hostnames and realm name for your realm. The entries you need to modify are shown in color. Note that the realm names are case sensitive. Place these files into the appropriate directories. For the `krb5.conf`, it's the `/etc` directory. For the `kdc.conf`, it's `/usr/local/var/krb5kdc`. Note that you have to manually create the `/usr/local/var/krb5kdc` directory.

Listing A: Kerberos KDC configuration file `/etc/krb5.conf`

```
[libdefaults]
  default_realm = TREE.COM
  default_tkt_enctypes = des-cbc-crc
  default_tgs_enctypes = des-cbc-crc
[realms]
TREE.COM = {
  kdc = birch.tree.com:88
  kdc = pine.tree.com:88
  admin_server = birch.tree.com:749
}[domain_realm]
.tree.com = TREE.COM
[kdc]
  profile = /usr/local/var/krb5kdc/kdc.conf
[logging]
  kdc = FILE:/var/log/kdc.log
  default = FILE:/var/log/kdc.log
```

Listing B: Kerberos KDC configuration file `usr/local/var/krb5kdc/kdc.conf`

```
[kdcdefaults]
  kdc_ports = 88
[realms]
TREE.COM = {
  database_name = /usr/local/var/krb5kdc/principal
  admin_keytab = FILE:/usr/local/var/krb5kdc/kadm5.keytab
  acl_file = /usr/local/var/krb5kdc/kadm5.acl
  key_stash_file = /usr/local/var/krb5kdc/.k5.TREE.COM
  kadmind_port = 749
  max_life = 10h 0m 0s
  max_renewable_life = 7d 0h 0m 0s
  master_key_type = des-cbc-crc
  supported_enctypes = des-cbc-crc:normal
}
```

For an explanation of each entry in the configuration files, consult the `krb5.conf(5)` and `kdc.conf(5)` man pages and the Kerberos V5 System's Administrators Guide (comes with a

distribution). The convention is to make a realm the same as your domain name, in upper-case letters.

Now we're ready to create a Kerberos database and the stash file using the `kdb5_util` utility. The stash file is a local copy of the master key that resides in encrypted form on the Master KDC's local disk. Use this file to authenticate the KDC to itself automatically before starting the Kerberos daemons during the server boot. If the stash file (`/usr/local/var/krb5kdc/.k5.TREE.COM` in the `kdc.conf` file in **Listing B**) were compromised, it would allow unrestricted access to the Kerberos database. Make sure that this file is readable only by root and it isn't a part of a backup:

```
#/usr/local/kerberos/sbin/kdb5_util create -s
```

This file prompts you for the master key for the Kerberos database. This key can be any string. Choose a key that's very difficult to guess.

Next you need to create an Access Control List (acl) file:

```
#cat /usr/local/var/krb5kdc/kadm5.acl
*/admin@TREE.COM *
```

This gives the principal `*/admin@TREE.COM` permission to change all of the database permissions on any principal permissions.

The next step is to set up the database entry for at least one Kerberos administrator. Run `kadmin.local` and use `addprinc`, as shown in **Listing C**.

Listing C: *Setting up the Kerberos administrator*

```
#/usr/local/kerberos/sbin/kadmin.local
Authenticating as principal root/admin@TREE.COM with password.
kadmin.local: addprinc admin/admin
WARNING: no policy specified for admin/admin@TREE.COM;
=>defaulting to no policy
Enter password for principal "admin/admin@TREE.COM":
Re-enter password for principal "admin/admin@TREE.COM":
Principal "admin/admin@TREE.COM" created.
```

To get information about this user, you can use `getprinc` with `kadmin.local`. Type `?` inside of `kadmin.local` to see a list of all available requests.

Create the `keytab` file on the server. `Kadmind` uses this to determine what access it should give to administrators. You need to create the keytab entries for the principals `kadmin/admin`

and `kadmin/changepw`. (These principals are placed in the Kerberos database automatically when you create it.) Now, stay in `kadmin.local` and run:

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

Edit the `/etc/services` file to include the services shown in **Listing D**.

Listing D: Lines added to our `/etc/services` file for Kerberos

```
kerberos 88/udp kdc # Kerberos V5 KDC  
kerberos 88/tcp kdc # Kerberos V5 KDC  
klogin 543/tcp # Kerberos authenticated rlogin  
kshell 544/tcp krcmd # and remote shell  
kerberos-adm 749/tcp # Kerberos V5 admin/chpwd  
kerberos-adm 749/udp # Kerberos V5 admin/chpwd  
krb5_prop 754/tcp # Kerberos V5 slave propagation  
eklogin 2105/tcp # Kerberos auth. and encrypted rlogin
```

Start the master KDC and the Kerberos administration daemons by typing:

```
#!/usr/local/kerberos/sbin/krb5kdc  
#!/usr/local/kerberos/sbin/kadmind
```

If you'd like to start these servers automatically when the Kerberos Server is rebooted, you have to create a startup file in the `/etc/init.d` directory and link it with the corresponding files in the `/etc/rc3.d`. Now we're ready to go to pine.tree.com (our second KDC) and proceed with a Slave KDC installation.

Configuring the slave KDC

Setting up a slave KDC is relatively simple. You need to install Kerberos on the slave KDC (pine.tree.com) and edit the configuration files `krb5.conf` and `kdc.conf` the same as for the primary KDC.

Both slave and master KDCs need host principals in the Kerberos database. You can run `kadmin` on pine.tree.com to administer the Kerberos database remotely by using the following:

```
#!/usr/local/kerberos/sbin/kadmin -p admin/admin  
kadmin: addprinc -randkey host/birch.tree.com  
kadmin: addprinc -randkey host/pine.tree.com
```

Add the service principal to the server's keytab so it can provide the Kerberized service:

```
kadmin: ktadd host/birch.tree.com
kadmin: ktadd host/pine.tree.com
```

In the Kerberos database directory (`/usr/local/var/krb5kdc`) on both your master and slave, you need to create a file called `kpropd.acl` and place in it all of the host principals for your KDCs:

```
#cat /usr/local/var/krb5kdc/kpropd.acl
host/birch.tree.com
host/pine.tree.com
```

On the slave KDC (`pine.tree.com`), add an entry for `kpropd` in `inetd.conf`:

```
krb5_prop stream tcp nowait root
=>/usr/local/kerberos/sbin/kpropd kpropd
```

On the master KDC (`birch.tree.com`), dump the database into a file using `kdb5_util`:

```
#!/usr/krb5/sbin/kdb5_util dump
=>/usr/local/var/krb5kdc/slave_datatrans
```

Now you can run `kprop` on the master to propagate the database to the slave:

```
#!/usr/local/kerberos/sbin/kprop -f
=>/usr/local/var/krb5kdc/slave_datatrans
=>pine.tree.com
```

It's a good idea to setup a `crontab` job to propagate the database at regular intervals.

The final step in configuring the slave KDC is to run the KDC daemon:

```
#!/usr/local/kerberos/sbin/krb5kdc
```

You're all set. Now let's connect `oak.tree.com` as a client machine.

Connecting a client

The Kerberos client is a machine that offers kerberized services such as `telnet`, `ftp`, `r*` services (`rlogin`, `rcp`, `rsh`), or even NFS and NIS+ network facilities. These services are modified to work with Kerberos.

First you have to install Kerberos on the client the same as you did on all KDCs. Edit the `/etc/krb5.conf` file. You must add the appropriate Kerberos services into the `/etc/services` file as described in the section "Configuring the master KDC." Comment out

any lines for `telnet` and `ftp` services in the `/etc/inet/inetd.conf` file. You'll use the kerberized versions that come with the Kerberos distribution. Add the lines shown in **Listing D** into the `/etc/inet/inetd.conf` file on `oak.tree.com`.

Listing D: Add the following lines to `/etc/inet/inetd.conf`

```
klogin stream tcp nowait root /usr/local/kerberos/sbin/klogind klogind -k -c
eklogin stream tcp nowait root /usr/local/kerberos/sbin/klogind klogind -k -c -e
kshell stream tcp nowait root /usr/local/kerberos/sbin/kshd kshd -k -c -A
ftp stream tcp nowait root /usr/local/kerberos/sbin/ftpd ftpd -a
telnet stream tcp nowait root /usr/local/kerberos/sbin/telnetd telnetd -a valid
```

Don't forget to kill the `inetd` process with the `-HUP` option. To allow `oak.tree.com` to be a Kerberos client, you have to create a `/etc/krb5.keytab` file. Inside the `kadmin`, add the host as a principal:

```
#/usr/local/kerberos/sbin/kadmin -p admin/admin
kadmin: addprinc -randkey host/oak.tree.com
```

You'll see a message that the principal `host/oak.tree.com` is created. Now, add its `keytab` entry in the `oak`'s `/etc/krb5.keytab` file:

```
kadmin: ktadd oak.tree.com
```

This process securely shares a secret key to be used for communication between `oak` and the KDC server. You need to repeat this process for every host in your realm.

The next step is to create a user principal. You only need to create a user principal for this host:

```
kadmin: addprinc anna
```

For this principal you can choose the same password as a user's UNIX password.

Now you're ready to test it. Try to `telnet`, `rlogin` or `ftp` to your client as a Kerberos user and watch how the authentication works.

Because Kerberos uses `gethostbyname()` to determine the fully qualified domain name, make sure that it works correctly on your system. To test this, run:

```
#/krb-5.1.1.1/src/tests/resolve/resolve <hostname>
```

If it doesn't bring you a fully qualified hostname (e.g., `oak.tree.com`), the KDC will deny authorization. To fix it, add a long name entry into the `/etc/hosts` file (e.g., `192.168.10.1 oak.tree.com oak`). Also make sure that you use kerberized services and utilities. Put the `/usr/local/kerberos/bin` and the `/usr/local/kerberos/sbin` first in your `PATH`.

For more information about Kerberos administration and usage, refer to the Kerberos V5 System Administrator's Guide and the Kerberos V5 UNIX User's Guides that come with the distribution.

Sun's implementation of Kerberos

Before Solaris 8, Sun used to ship a basic set of Kerberos V4 utilities. The RPC that comes with this also supports Kerberos V4. Sun has also announced a Sun Enterprise Authentication Mechanism (SEAM) that's based on the Kerberos V5 standard (currently only for Solaris 2.6 and 7). This is a component of the Solaris Easy Access Server. Sun developed this product for integration with Microsoft's Windows 2000 networks.

SEAM software supports a Java technology-based administrative tool for easy access and configuration. It also enables users to load authentication information in batch mode, which is particularly useful if your enterprise loses or gains large numbers of users each year. For more information about SEAM, check <http://www.sun.com/software/solaris/ds/ds-seamss/>.

Limitations

Even though Kerberos is a strong security solution, it also has several limitations. First, Kerberos makes no provisions for host security; it assumes that it's running on trusted hosts with an untrusted network. If your host security is compromised, then Kerberos is compromised as well.

Second, Kerberos stores all passwords encrypted with a single key. This means that in the event the Kerberos Server is compromised, all user passwords must be changed.

Third, Kerberos uses a principal's password (encryption key) as the fundamental proof of identity. If an attacker steals a user's Kerberos password, then the attacker can impersonate that user with impunity.

Finally, Kerberos doesn't work well in a multi-user environment. It keeps tickets in the `/tmp` directory. It's possible that another user sharing the same workstation can steal the user's ticket. Stolen tickets can then be used to obtain fraudulent service.

Conclusion

In this article, we've shown you a basic Kerberos configuration and administration. Kerberos is very complex and robust. In spite of these shortcomings, Kerberos is an excellent solution to a difficult problem like network security. We hope that you will find Kerberos as useful as we did for your network environment.