Kerberos Infrastructure HOWTO

V. Alex Brennen

<vab@cryptnet.net>

2004−05−29

Revision History

Revision 2.0.0	2004−05−28	Revised by: VAB
Conversion to DocBook XML. General Content Updates, including incorporation of Technical and Metadata/Markup Reviews.		
Revision 1.0.3	2003−04−01	Revised by: VAB
Minor Updates, Minor Corrections, Additional links added.		
Revision 1.0.2	2002−09−13	Revised by: VAB
Minor Updates, Minor Corrections, Added 8.6, Additional links added.		
Revision 1.0.1	2002−07−15	Revised by: VAB
Minor Updates, Fixes.		
Revision 1.0.0	2002−06−13	Revised by: VAB
Initial Release.		

This document describes the design and configuration of a Kerberos infrastructure for handling authentication with GNU/Linux. It details steps for a best practices method of setting up servers, Kerberos software, conversion of legacy systems, and answers frequently asked questions.

Table of Contents

Table of Contents

1. About this Document

1.1. General Information

This document lives at http://cryptnet.net/fdp/admin/kerby−infra/en/kerby−infra.html

---

1.2. Translations

This document is currently only available in the following languages:

- [en] English

If you know of a translation or would like to translate it to another language please let me know so that I can distribute or link to the translated versions.

1.3. Credits and Contributors

- V. Alex Brennen (VAB) `<vab (at) cryptnet.net>` (Principal Author)
- Nickolai Zeldovich `<kolya (at) zepa.net>` (Technical suggestions, technical corrections)

1.4. Feedback

Please send any additions, comments, corrections and criticisms to the following email address: `<vab@cryptnet.net>`.

---

2. An Overview of a Kerberos Infrastructure

2.1. An Introduction to Kerberos

Kerberos is a system of authentication developed at MIT as part of the Athena project. Kerberos uses encryption technology and a trusted third party, an arbitrator, to perform secure authentication on an open network. Specifically, Kerberos uses cryptographic tickets in order to avoid transmitting plain text passwords over the wire. Kerberos was based upon the Needham−Schroeder protocol.

There are two versions of Kerberos currently in use, version 4 and version 5. Kerberos versions 1 through 3 were internal development versions and never released. Kerberos version 4 has a number of known weaknesses and should no long be used. This document deals only with Kerberos 5. Kerberos 5 is defined in RFC1510.

The term Kerberos Infrastructure refers to the software, servers, and client configurations that will allow an administrator to use the Kerberos protocol to perform authentication on their network. Specifically, Kerberos Infrastructure consists of the Kerberos software itself, secured redundant authentication servers, a centralized account and password store, and systems configured to authenticate through the Kerberos protocol. This document will take you through the steps to install, configure, and deploy such an infrastructure.

2.2. The Benefits of Kerberos

For individuals unfamiliar with the Kerberos protocol, the benefits of deploying it in their network may not be clear. However, all administrators are familiar with the problems Kerberos was designed to mitigate. Those problems include, password sniffing, password filename/database stealing, and the high level of effort necessary to maintain a large number of account databases.

A properly deployed Kerberos Infrastructure will help you address these problems. It will make your enterprise more secure. Use of Kerberos will prevent plaintext passwords from being transmitted over the network. The Kerberos system will also centralize your username and password information which will make it easier to maintain and manage this data. Finally, Kerberos will also prevent you from having to store password information locally on a machine, whether it is a workstation or server, thereby reducing the likelihood that a single machine compromise will result in additional compromises.

To summarize, in a large enterprise, the benefits of Kerberos will translate into reduced administration costs through easier account and password management and through improved network security. In a smaller environment, scalable authentication infrastructure and improved network security are the clear benefits.

2.3. How Kerberos Works

Kerberos is an authentication protocol which uses a shared secret and a trusted third party arbitrator in order to validate the identity of clients. In Kerberos, clients may be users, servers, or pieces of software. The trusted third party arbitrator is a server known as a Key Distribution Center (KDC) which runs the Kerberos daemons. The shared secret is the users password transformed into a cryptographic key. In the case of servers or software systems, a random key is generated.

In Kerberos, users are known as principals. The KDC has a database of principals and their secret keys which is uses to perform authentication. In Kerberos knowledge of the secret key is considered sufficient for proof of identity. Since knowledge of a secret key translates into proof of identity in Kerberos, the Kerberos server can be trusted to authenticate any client to any other client. Authentication is Kerberos is done with out sending any clear text passwords across the wire. Below I'll explain how the Kerberos protocol maps to the GNU/Linux Kerberos software.

The KDC runs two important Kerberos daemons. These daemons are kadmind and krb5kdc. While GNU/Linux daemon naming conventions suggests that processes which have names starting with "k" are Kernel related or Kernel space processes, this is not true in the case of krb5kdc and kadmind. These two daemons are run as root in user space.

kadmind is the administrative daemon for the Kerberos server. kadmind is used by a program named kadmin to maintain the database of principals and policy configuration. If you choose to disallow any remote logins via ssh on your Kerberos hardware, kadmin will allow you to remotely administer the Kerberos components of the server.

krb5kdc is the workhorse of the Kerberos server. It is responsible for performing the role of the trusted third party arbitrator in Kerberos authentication. When a user wants to authenticate himself to a system or service, the user requests a ticket from the KDC. A ticket is a datagram consisting of the client's identity, a session key, a timestamp, and some other information. The datagram is encrypted with the server's secret key.

In detail that process works as follows, first the request for authentication is sent to the krb5kdc daemon. When the daemon received this request, it looks up the client, the principal, trying to authenticate in the principal database. It reads the clients secret key from this database and encrypts a special ticket called a Ticket Granting Ticket (TGT) which it then sends back the client. The client receives this encrypted TGT which contains a session key. If the client knows the password (the secret key stored in the principal database) and can successfully decrypt the TGT, it can present the ticket encrypted with the enclosed session key to a Ticket Granting Service (TGS). The TGS will then issue a subsequent ticket which will provide the client with the authentication they need to use a specific system or service.

Through the use of encrypted tickets which can only be decrypted if the client knows the secret key, secure authentication takes place. Time stamp information is included in the tickets to prevent replay attacks. A replay attack would be the fraudulent representation of a previously issued ticket in order to gain unauthorized access.

2.4. Compromise of Kerberos Infrastructure

The primary way in which an attacker will attempt to compromise a Kerberos Infrastructure would be to attack the Kerberos servers. If an attacker can gain root access to a KDC, the attacker will have access to the database of encrypted passwords of the principals. The attacker will also have access to the Kerberos software and configuration files both of which they can then modify to make the system perform authentications which should not be successful.

Other methods of attacking Kerberos infrastructure include replay attacks and password−guessing attacks. A replay attack would involve intercepting or otherwise acquiring a Kerberos ticket and then fraudulently representing that ticket in an attempt to gain authentication. Password guessing in a Kerberos system could be done by intercepting Kerberos tickets from the network and then making a brute force attempt to decrypt the intercepted tickets.

An attacker may exploit outdated software in the infrastructure. For example, there are many problems with Kerberos version 4. Most importantly, Kerberos version 4 has a basic protocol weakness in the encryption used. The design of Kerberos 4 included the use of DES in standard mode which allows attackers to intercept and modify the ciphertext of Kerberos tickets in an undetectable way. In order to prevent these attacks, Kerberos 5 has been modified to use triple DES in Cipher Block Chaining (CBC) mode.

When discussing the strength of Kerberos 4, it is also important to note that many implementations of Kerberos version 4 have buffer overflow vulnerabilities. While the reference implementations of version 5 fixed the buffer overflow weaknesses in version 4, distributions of Kerberos 5 usually ship with software to provide backward compatibility to support legacy Kerberos 4 applications. Some of the backward compatibility code for version 4 support in Kerberos version 5 releases is still believed to be vulnerable to buffer overflow attacks.

Therefore, due to the protocol weaknesses in version 4 and the potential for buffer overflow attacks with version 4 implementations and version 4 backward compatibility code, it is best not to support or use Kerberos version 4.

In summary, from this description on how a Kerberos infrastructure can be compromised, we realize that we must give great priority to the security of the Kerberos servers themselves, run up to date Kerberos software, and remain vigilant in picking good passwords and in setting good password policy.

3. Installing and Configuration

3.1. General Machine Configuration Overview

This section of the document describes the installation and configuration of the machines and software which will function as the KDCs. You may want to make some adjustments to the configurations suggested, but there are a few key points presented that are very important to remember when configuring your KDCs. So, if you do decide on an alternative configuration strategy make sure you understand the material presented here.

The machines will run the Kerberos daemon and store password and policy data. Therefore, it's very important to the security of the network that these servers remain secure. We should take every possible measure to prevent these servers from being compromised. Pay particular attention to the security advice given in this section.

The key points of that security advice are that you should dedicate hardware to providing Kerberos KDC service. You should physically secure that hardware, and you should harden GNU/Linux as much as possible on that hardware. If a KDC is compromised, your entire Kerberos infrastructure is compromised.

3.2. Hardware

Kerberos service does not place a great demand on hardware and the Kerberos services have a capability for redundancy, therefore server hardware can be minimal. For the Kerberos servers which I've deployed I've used uniprocessor PIII machines with two hardware RAID 1 drives. These machines are meant to handle between forty and one hundred thousand authentications per day. While servers may be deployed with redundant NIC cards, having both cards active simultaneously should be avoided. Kerberos includes the IP of the KDC in tickets, therefore difficulties authenticating may occur if the KDC is contacted on multiple interfaces by a client during an authentication session.

It is important to note that Kerberos service should be run on dedicated hardware. Dedicating a machine to Kerberos means that only the Kerberos administrator will need to log in on those machines. It also means that no other services, except perhaps SSH, will be run on the machines. Since all of your users passwords are stored on the Kerberos servers, it is a good idea to limit access as much as possible to the hardware. Along with dedicating servers to Kerberos, you should also physically secure your servers as much as possible. For Kerberos servers, this may include locking the servers in a cabinet and having a dedicated terminal attached to them.

In order to take advantage of Kerberos' built in capability for redundancy, you must have at least two machines running as KDCs. Kerberos is designed to be deployed with one primary master server, and one or more secondary slave servers. You may have as many secondary servers as you would like.

3.3. GNU/Linux Installation

The servers we're installing GNU/Linux on will be dedicated to the task of performing Kerberos service, therefore we can take some extra steps to secure them.

First, we'll only install software that is absolutely necessary for Kerberos service. This includes the base operating system and the Kerberos packages. We should not install X or any GUI applications. SSH is optional. SSH may be installed if you wish to be able to administer the servers remotely. However, the servers would be significantly more secure if you only provide login access to them through an attached terminal.

In Fedora Core based GNU/Linux, the packages required to provide Kerberos service are:

```
krb5-server
krb5-libs
```

Documentation and development libraries should not be installed on the KDC, since we do not want to use this machine for anything but the performance of KDC service.

The next step will be to make sure that no ports are open that do not need to be and that any necessary security patches that are needed are applied. The methodology to determining what security patches need to be applied is depended up what package management software is installed. To determine what ports the machine is currently listening on, the netstat command can be used. For example, on a machine which only ssh running we should see the following:

```
bash$ netstat -an | grep -i listen | less
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
```

Finally, we'll configure the server to restrict access only to servers that need to talk to it for authentication. This should be done by editing `/etc/hosts.allow` and `/etc/hosts.deny` as well as with `iptables`.

3.4. Choosing A Realm

Realm names are case sensitive and must be unique on your network. It is a standard best practice to use your second level domain name in all uppercase letters as your realm name. If you are setting up Kerberos for only a subnet rather than your entire network, you should use the trailing domain name of the subnet.

When determining your realm topology, you should take the overall structure of your organization into account. If your organization has one or more remote offices or independent sub groups, they may be best included under a separate realm. Kerberos realm topology should mirror system management topology rather than physical network topology.

Finally, legacy systems should also be taken into account. For example, legacy Kerberos deployments or existing network topology grouping which you wish to preserve (i.e. Windows NT domains).

If you are installing Kerberos on a network which already has Kerberos deployed in the overall network or in a subnet, you must avoid a realm name collision. The most common occurrence of deploying Kerberos on a network with a preexisting Kerberos installation occurs when working with a network that includes an IBM SP cluster. The best solution is to create a realm specifically for the SP cluster at third or high domain name level and then use a second level domain name for your primary Kerberos realm.

In this document, we'll use an example to help illustrate the design and configuration of an infrastructure. For our example, we'll use a mythical university which was founded to educate people with, and perform research in the area of, free content – Gnu University in Dublin, Ireland. The Gnu University Dublin example will include two Kerberos servers used to authenticate students and faculty. The TLD for the university is gnud.ie, therefore we'll use the Kerberos realm of GNUD.IE.

3.5. Kerberos Software Configuration

Now, you'll need to configure Kerberos, create an administrator, determine a policy, and initialize the Kerberos principal database.

The first step is to edit the `/etc/krb5.conf` configuration file. In this file you'll need to set your realm, expand on the realm definition by specifying the Kerberos servers, and finally setting the domain realm. For our example, this is done as follows:

```
default_realm = GNUD.IE

[realms]
 GNUD.IE = {
  kdc = kerberos1.gnud.ie:88
  kdc = kerberos2.gnud.ie:88
  admin_server = kerberos1.gnud.ie:749
  default_domain = gnud.ie
 }

[domain_realm]
 .gnud.ie = GNUD.IE
 gnud.ie = GNUD.IE
```

To initialize and create the Kerberos database, you must run the follow command:

```
{Kerberos1}bash# /usr/Kerberos/sbin/kdb5_util create -s
```

The *−s* flag tell the KDC to create a stash file to authenticate itself. You may also use a *−r* flag to specify a realm. Specifying a realm for the new database is only necessary if you have more than one realm defined in your `krb5.conf` file.

Kerberos will then ask you to set the master password for your Kerberos database. It is very important that you do not forget this password. You will not be able to administrate your server if you do not remember the master password.

Next on the KDC you must edit the acl file to grant administrative access. Typically, this file is located at `/var/Kerberos/krb5kdc/kadm5.acl`. If necessary, specify the acl file location in your `kdc.conf` file. The location of your `kdc.conf` file is specified in your `/etc/krb5.conf` file and defaults to `/var/Kerberos/krb5kdc/kdc.conf`. For our GNU University Dublin example, we'll modify the acl file to include the following contents:

```
*/admin@GNUD.IE     *
```

The meaning of those acl contents are that any account which ends with a /admin in the *GNUD.IE* realm is granted full access privileges.

Now that we've set up access for our administrative user, we need to create that administrative user. You can do this with the kadmin.local command from a root shell on the KDC, using the addprinc sub command. The standard is to name the administrative account *admin*. For the Gnu University Dublin Kerberos Administrator, the following command would accomplish this:

```
{Kerberos1}bash# /usr/Kerberos/sbin/kadmin.local -q "addprinc admin/admin"
```

The daemons that must run on the server are krb5kdc and kadmin. If necessary, krb524 may also be run to provide backward compatibility to Kerberos 4 clients. However, before starting krb524 remember our security warning about Kerberos V4 and be sure that you really need to provide that functionality. On the KDCs krb5kdc and kadmin should be configured to start automatically by turning them on with the chkconfig command.

```
{Kerberos1}bash# /sbin/chkconfig krb5kdc on
{Kerberos1}bash# /sbin/chkconfig kadmin on
```

Finally, we can start them up manually, with the following command:

```
{Kerberos1}bash# /etc/rc.d/init.d/krb5kdc start
{Kerberos1}bash# /etc/rc.d/init.d/kadmin start
```

and we have a working KDC.

3.6. Principal Creation

You can create the first user principal in Kerberos with the following command:

```
{Kerberos1}bash# kadmin.local
{Kerberos1}kadmin.local: addprinc <username>
```

A script could be written to create principals in bulk if you have a large number of account which you will be supporting with Kerberos.

4. Time Synchronization

4.1. The Importance of Time Synchronization

Since the security of Kerberos authentication is in part based upon the time stamps of tickets, it is critical to have accurately set clocks on Kerberos servers. As we mentioned in the introduction to Kerberos, a short lifetime for tickets is used to prevent attackers from performing successful brute force attacks or replay attacks.

If you allow your clocks of your servers to drift, you will make your network vulnerable to such attacks. Since clock synchronization is so important in the security of the Kerberos protocol, if clocks are not synchronized with in a reasonable window Kerberos will report fatal errors and refuse to function. Clients attempting to authenticate from a machine with an inaccurate clock will be failed by the KDC in authentication attempts due to the time difference with the KDC's clock.

4.2. Introduction to NTP

The Network Time Protocol (NTP) is available for the time synchronization of servers. A number of public NTP servers are available for synchronization. NTP is capable of synchronizing client clocks within milliseconds on LANs, and tens of milliseconds over WANs. NTP servers are divided by stratum. Primary NTP servers are classified as stratum 1. These public primary servers should not be used for synchronization of client machines due to the relatively small number of them. Public stratum 2 servers are available for client machine synchronization and synchronize their own clocks with the public stratum 1 servers. For our Kerberos servers, we'll set up NTP to query against three stratum two servers. A maintained list of public stratum two servers can be found here.

4.3. NTP Installation and Configuration

In order to turn NTP on GNU/Linux, you must install the NTP package and edit the configuration filename. By default, the NTP configuration filename is `/etc/ntp.conf`. The default values in the configuration are acceptable. All that needs to be done is to add the servers we wish to use to synchronize our clocks. Authentication is not necessary, but authentication can be enabled from additional security. If you're using NTP servers on your LAN, you may wish to use authentication. Here is a sample `ntp.conf` filename from the Gnu University Dublin.

Finally we put a cron job in place to do the actual synchronization:

```
30 * * * * /usr/sbin/ntpdate -s
```

If our systems are behind a firewall we may need to provide use *−su* rather than just *−s*. The *−u* argument will direct ntpdate to use an unprivileged port for its outgoing connection to the stratum 2 time servers.

5. Kerberos Server Replication

5.1. Description of Replication

Kerberos was designed to allow for a Master/Slave replication cluster. While a Kerberos cluster can consist of any number of hosts, it is recommended that you have at least two. A master which serves as the primary server and at least one slave which is available as a backup to the master. The master and slave servers may be thought of as Primary and Secondary servers respectively.

Kerberos stores all of its information, both account and policy data, in application databases. The Kerberos software distribution includes software for replicating, or copying, this data to other servers.

Kerberos client applications are designed to attempt authentication against secondary servers if the primary master is down. Therefore you do not need to do any extra work during a system failure to fail over your Kerberos authentication service to the backup server. However, the administrative features of Kerberos do not provide for automatic failover.

In the event that your primary server fails, kadmind will be unavailable. Therefore, administrative functions will be unavailable until the primary server is restored or replace. Specifically, principal management, key creation, and key changes, cannot be done during a primary server failure.

5.2. Implementation

Server replication is handled by the kprop command. kprop must be run on the primary master KDC. It should be run in a scheduled cron job to keep the principal database in sync across all servers.

The first step in setting up replication, is to set up ACLs for kpropd. The kpropd acl filename is by default located at `/var/Kerberos/krb5kdc/kpropd.acl`. In our example, it would have the following contents:

```
host/kerberos1.gnud.ie@GNUD.IE
host/kerberos2.gnud.ie@GNUD.IE
```

The `kpropd.acl` file should only exist on the slave Kerberos server. In Fedora derived GNU/Linux, kadmin will not run on a Kerberos server on which `/var/Kerberos/krb5kdc/kpropd.acl` exists.

Next you'll need to create host keys for your master and slave Kerberos servers:

```
{Kerberos1}bash# kadmin.local
{Kerberos1}kadmin.local:  addprinc -randkey host/kerberos1.gnud.ie
{Kerberos1}kadmin.local:  addprinc -randkey host/kerberos2.gnud.ie
```

The next step is to extract these keys to the keytab file. The keytab file is a keyring which contains the cryptographic keys needed to authenticate with the KDC. Extraction of keys to the keytab is done with the ktadd sub command:

```
{Kerberos1}kadmin.local:  ktadd host/kerberos1.gnud.ie
{Kerberos1}kadmin.local:  ktadd host/kerberos2.gnud.ie
```

Finally, we want to copy the keytab over to the slave server so that it has the keys it needs available to authenticate.

```
{Kerberos2}bash# scp root@kerberos1.gnud.ie:/etc/krb5.keytab /etc
```

Here is a crontab entry from the master Kerberos server used to synchronize principal databases every fifteen minutes:

```
15 * * * * /usr/local/bin/krb5prop.sh
```

Here are the contents of the `krb5prop.sh` script:

```
#!/bin/sh

/usr/Kerberos/sbin/kdb5_util dump /var/Kerberos/krb5kdc/slave_datatrans

/usr/Kerberos/sbin/kprop -f /var/Kerberos/krb5kdc/slave_datatrans kerberos2.gnud.ie > /dev/null
```

Initially running this command by hand, you should see something similar to the following:

```
{Kerberos1}bash# /usr/Kerberos/sbin/kdb5_util dump /var/Kerberos/krb5kdc/slave_datatrans
{Kerberos1}bash# /usr/Kerberos/sbin/kprop -d -f /var/Kerberos/krb5kdc/slave_datatrans kerberos2.gnud
3234 bytes sent.
Database propagation to kerberos2.gnud.ie: SUCCEEDED
{Kerberos1}bash#
```

The slave server will now synchronize its principal database with the master server.

5.3. Maintenance

With these cron jobs in place principal propagation should be sufficiently automated as to require no maintenance. At the time of a primary KDC failure, there is no need for human intervention unless the failure will last for an extended period of time.

6. Client Configuration

6.1. General GNU/Linux Client Configuration

GNU/Linux distributions of Kerberos include a client package which contains all of the software and configuration files needed for setting up a GNU/Linux machine to be able to perform Kerberos authentications against a KDC. In Fedora derived GNU/Linux, this package is *krb5−workstation*. In order for your system to be capable of Kerberos authentication, including by authentication by kerberized applications, you must configure Kerberos on the system.

Configuration involves editing the `/etc/krb5.conf` file. In this file, you must specify your realm, KDC's, administrative server, logging, default domain, and KDC information. You must also modify the `kdc.conf` file, which you are allowed to specify a location for in the `krb5.conf` file. The default location is `/var/Kerberos/krb5kdc/kdc.conf`. The `kdc.conf` file contains information about the encryption algorithm policy of the realm.

The configuration information for the system on which you wish to perform Kerberos authentications is the same information which was placed in the `/etc/krb5.conf` filename on the KDC. Here are example `krb5.conf` and `kdc.conf` configuration files from a client for the Gnu University Dublin example.

Now, you can test Kerberos authentication using the kinit command:

```
bash$ kinit <username>
```

If your authentication fails, the best place to look for a description of the cause are the system log files on the client and the KDC log file on the KDC which authentication was performed against. When trouble shooting authentication issues, it can be very helpful to have a terminal windows open to the KDC running a *tail −f* on the KDC log. In our example `krb5.conf`, the location of the KDC log was `/var/log/Kerberos/krb5kdc.log`.

6.2. PAM

PAM, or Pluggable Authentication Module, technology which is shipped with many distributions of GNU/Linux is capable of integration with Kerberos through the pam_krb5 module. In order to use Kerberos authentication with PAM you must install the pam_krb5 module and modify the pam configuration files.

The pam_krb5 module comes with sample configuration filenames which are located in `/usr/share/doc/pam_krb5−1.55/pam.d`. The basic change that these configuration files make to allow PAM controlled services to authenticate against Kerberos is similar to the following:

```
auth       required     /lib/security/pam_krb5.so use_first_pass
```

6.3. Apache Web Server

Kerberos can be used as an authentication mechanism for the Apache Web Server. The mod_auth_kerb application is an apache module which provides that functionality. Using that module, you will be able to set Kerberos as an authentication type for access control stanzas in the `httpd.conf` file. Be aware that while Kerberos is being used, this is a less than ideal authentication mechanism because tickets are stored on the web server rather than on the client machine. However, if your goals is to implement a single sign−on solution or consolidate accounts, there is value here. mod_auth_kerb is capable of supporting Kerberos 4, however that is not covered in this howto because of the known weaknesses with version 4 of the protocol.

The mod_auth_kerb website can be found at http://modauthkerb.sourceforge.net/. It is important to use the HTTPS protocol when accessing a site which uses mod_auth_kerb, since mod_auth_kerb uses the base auth mechanism. Base auth uses base64 encoding which can easily be translated back to plaintext. Therefore it's important that the authentication exchange is SSL encrypted to ensure that the user name and password are protected when they are transmitted to the webserver.

To compile apache with the mod_auth_krb module, you must take the following steps:

```
bash$ export 'LIBS=-L/usr/Kerberos/lib -lkrb5 -lcrypto -lcom_err'
bash$ export 'CFLAGS=-DKRB5 -DKRB_DEF_REALM=\\\"GNUD.IE\\\"'
bash$ export 'INCLUDES=-I/usr/Kerberos/include'
bash$ mkdir apache_x.x.x/src/modules/kerberos
bash$ cp mod_auth_kerb-x.x.x.c apache_x.x.x/src/modules/kerberos
bash$ ./configure --prefix=/home/httpd --add-module=src/modules/Kerberos/mod_auth_kerb.c
bash$ make
bash$ make install
```

You should test apache to make sure that it works. Once you have a known working copy of SSL enabled apache on the machine you can modify the `httpd.conf` filename to provide Kerberos authentication for a directory:

Here is an example stanza from the mod_auth_kerb apache modules which enables Kerberos 5 authentication for a directory:

```
<Directory "/home/httpd/htdocs/content">
   AllowOverride None
   AuthType KerberosV5
   AuthName "Kerberos Login"
   KrbAuthRealm GNUD.IE
   require valid-user
</Directory>
```

6.4. Microsoft Windows

Due to a flawed implementation of the Kerberos standard by Microsoft, there is limited compatibility between standard MIT Kerberos and Microsoft's version. Microsoft has published a document which describes the limited ways in which Microsoft's broken version of Kerberos is able to interoperate with standard Kerberos. That document is available here.

7. Programming With Kerberos

7.1. The Kerberos API

Kerberos development libraries will allow you to Kerberos enable, or kerberize, any application. There are two primary libraries, a general usage Kerberos library used for basic authentication and a Kerberos administration library used to perform administrative functions such as operations on principals. In Fedora derived GNU/Linux, the rpm `krb5-devel` contains the development libraries and documentation. An API specification for these libraries can be found in the Kerberos documentation included with most Kerberos distributions. In Fedora derived GNU/Linux, it installs in:
`/usr/share/doc/krb5-devel-1.2.2/api`

The documentation is in the LaTeX format. In order to view it you must generate dvi filenames from it which can then be viewed with xdiv. This can all be done with the following commands:

```
bash$ cd /usr/share/doc/krb5-devel-x.x.x/api/
bash$ su
bash# make
bash# (^d)
bash$ xdvi library.dvi
```

A. Relevant Sources for More Information

A.1. Links to related documents

- Kerberos V5 Installation Guide
- Kerberos V5 UNIX User's Guide
- Kerberos V5 System Administrator's Guide
- Upgrading to Kerberos V5 from Kerberos V4
- Kerberos FAQ
- Designing an Authentication System: a Dialog in Four Scenes
- How To Kerberize Your Site
- The Moron's Guide to Kerberos
- AFS FAQ
- The Kerberos 5 API
- The Kerberos 5 Admin API

A.2. Related web sites

- MIT Kerberos Website
- The NTP Distribution Website
- List of Public Stratum 2 NTP Servers
- OpenAFS Website
- Heimdal Kerberos Website
- The Crypto Publishing Project (Unrestricted source for Kerberos source code)
- SESAME (Secure European System for Applications in a Multi−vendor Environment)

A.3. Related RFCs

- RFC2744: Generic Security Service API Version 2: C−bindings
- RFC2743: Generic Security Service Application Program Interface, Version 2 Update 1
- RFC2712: Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)
- RFC2078: Generic Security Service Application Program Interface, Version 2
- RFC1964: The Kerberos Version 5 GSS−API Mechanism
- RFC1510: The Kerberos Network Authentication Service (V5)
- RFC1509: Generic Security Service API : C−bindings
- RFC1508: Generic Security Service Application Program Interface
- RFC1411: Telnet Authentication: Kerberos Version 4
- RFC1305: Network Time Protocol (Version 3) Specification, Implementation and Analysis
- RFC1119: Network Time Protocol (Version 2) Specification and Implementation
- RFC1059: Network Time Protocol (Version 1) Specification and Implementation
- RFC958: Network Time Protocol (NTP)

A.4. Other references

- [Applied Cryptography] Second Edition, Bruce Schneier [ISBN: 0−471−11709−9]

A.5. Additional resources

- The Kerberos Authentication System Mailing List
- The Kerberos Authentication System Mailing List Archives
- comp.protocols.kerberos UseNet Newsgroup

A.6. Companies which provide specialist Kerberos consulting

- Cybersafe, Ltd.
- e−TechServices.com, Inc. IBM Business Partner

Glossary of Terms

ASN.1

Abstract Syntax Notation One. ASN.1 is a notation used describe messages. It describes them as a sequence of components. The described components may be sequences also. ASN.1 is used to describe the internals of Kerberos datagrams. Unless you are a software developer, you do not need to gain an understanding of ASN.1.

Authenticator

A record containing information that can be shown to have been recently generated using the session key known only by the client and server. (Definition taken from RFC1510)

Credentials

A ticket for the server and a session key which is used to authenticate the principal.

Cross−Realm Authentication

Kerberos has the ability for a KDC is one realm to authenticate a principal in another realm if a secret is shared between the KDCs of both realms. This inter−realm authentication is called cross−realm authentication.

Data Encryption Standard [DES]

An algorithm used for encrypted which was the official algorithm of the United Sates Government. It was developed by IBM with assistance from the NSA. The algorithm is a sixteen round block cipher which uses a 64bit block and a 56bit key.

Forwardable Ticket

A ticket granted by the KDC which allows the user to request additional tickets with different IP addresses. In effect, a TGT which allows the authenticated principal to request tickets valid on other additional machines.

Generic Security Services Application Programming Interface [GSS−API]

A set of C language bindings which provide security services to its callers. The API may be implemented on top of various cryptographic systems. Kerberos is one example of such a system.

Key Distribution Center [KDC]

The machine and software which perform the role of the trusted arbitrator in the Kerberos protocol.

Kerberos

An authentication protocol in which a trusted third party, an arbitrator, is relied upon to perform the authentication of clients on a TCP/IP network. The protocol was designed in a way that encrypted tickets are transmitted over the network rather than traditional plaintext passwords providing for secure network authentication.

Kerberize

(v.) The act of modifying a system, service, or piece of software to make use of the Kerberos protocol to perform authentication. (adj. kerberized) A system, service, or piece of software which supports authentication through Kerberos.

Network Time Protocol [NTP]

A protocol used to synchronizes clocks of hosts and routers on the Internet.

Postdatable ticket

In Kerberos 5, a ticket which is invalid initially and which becomes valid at some time in the future. Normal Kerberos tickets are only valid from the time they are requested until the time that they expire.

Preauthentication

Additional authentication which takes place before a KDC grants a TGT to a principal. An example of such authentication may be the satisfaction of a biometrics system.

Principal

A user or server for which a secret key is stored in the KDC database.

Proxiable Ticket

In Kerberos 5, a ticket which allows you to request a TGT for alternative IP addresses.

Realm
: The scope of a Kerberos deployment. Specifically, the organization domain for which the KDC is trusted to authenticate principals.

Renewable Ticket
: In Kerberos 5, a ticket which allows the principal a maximum renewable lifetime in addition to the standard ticket lifetime. Renewable tickets may be used to acquire additional tickets from the KDC as long as the ticket is valid. Renewed tickets can be requested up to the maximum renewable lifetime of the original renewable ticket.

Salt
: A seed value used in the encryption of a plaintext password to expand the number of possible resulting ciphertexts from a given plaintext. The use of a salt value is a defensive measure used to protect encrypted passwords against dictionary attacks.

Stash File
: A disk store of secret keys.

Ticket
: A data message consiting of the client's identity, a session key, a timestamp, and other information all encrypted with the server's secret key. It is used to perform authentication.

Ticket Granting Service [TGS]
: A service which is capable and authorized in the issuing of tickets to clients after they have acquire a Ticket Granting Ticket (TGT).

Ticket Granting Ticket [TGT]
: A ticket which contains a session key to be used in communication between the client and the KDC.

Transitive Cross−Realm Authentication
: In Kerberos 5, the ability to chain trust together between realms building in effect a trust path so that a principal in realm X that wishes to authenticate a principal in realm Z does not need the KDC for realm X to share a secret with realm Z if both realm X and realm Z share a secret with realm Y. Realm Y can be used as a "hop" in a trust path.

Triple DES
: A variant of DES in which data is encrypted three times with standard DES using two different keys.