



Heimdal Kerberos: Installation and Configuration

A technical introduction to installing and configuring Heimdal Kerberos.

Tillman Hodgson

Initial Revision: July 31, 2002
Revision date: August 27, 2002

Table of Contents

1	Introduction	3
1.1	Background	3
1.2	Software versions	3
1.2.1	Operating systems	3
1.2.2	Heimdal Kerberos	3
1.2.3	MIT Kerberos	3
1.3	Naming conventions	4
1.4	Copyright	4
1.5	Disclaimer	4
2	The Heimdal Kerberos KDC	5
2.1	What is the KDC?	5
2.2	Resources required	5
2.3	The rc.conf changes	5
2.4	The krb5.conf config file	5
2.5	Creating the database	6
2.6	Starting the KDC and testing it	6
3	Heimdal Kerberos Clients	8
3.1	Resources required	8
3.2	The krb5.conf file	8
3.3	The Kerberos applications	8
3.4	Testing	8
3.5	Adding new principals	9
4	Heimdal Kerberos Servers	10
4.1	Resources required	10
4.2	The keytab file	10
4.3	Setting up the telnet service	11
5	MIT Kerberos Clients	12
5.1	What the difference between MIT and Heimdal clients?	12
5.2	Fixing the broken bits with a travel kit	12
6	MIT Kerberos Servers	13
6.1	The similarities	13
6.2	The keytab file	13

A Background	14
A.1 Kerberos	14
A.1.1 What is Kerberos?	14
A.1.2 Kerberos terms	15
A.1.3 How Kerberos works	16
A.1.4 Weaknesses in Kerberos	17
A.1.5 Securing the KDC	18
B Material not covered	19
B.1 Integrating with NIS or LDAP	19
B.2 Scalability and Fault Tolerance	19
B.3 Time Synchronization	19
B.4 Integrating Kerberos with DNS	19
C References	20
C.1 Books	20
C.2 Web resources	20
C.3 Usenet resources	20
C.4 Mailing lists	20
D Acknowledgements	21

Introduction

1

1.1 Background

This document was prepared by the author after his experience putting together a central authentication system for the `seekingfire.prv` internal domain. It provides basic background information on creating a reasonably secure central authentication mechanism for Heimdal and MIT Kerberos clients using a Heimdal KDC.

1.2 Software versions

When creating this document, the following versions of the software discussed was used:

1.2.1 Operating systems

Two different operating systems were used:

- FreeBSD 4.6-STABLE¹ was used for the KDC and a Kerberized server/workstation
- RedHat Linux 7.3 was used for a Kerberized server/workstation

1.2.2 Heimdal Kerberos

- Heimdal Kerberos 0.4e from the FreeBSD 4.6-STABLE ports tree was used rather than the Heimdal package built into the base system
- The Heimdal Kerberos travel kit for “i386_linux22” linked to from the Heimdal web site was used for RedHat Linux 7.3

1.2.3 MIT Kerberos

The MIT packages that are provided with RedHat Linux 7.3 was used to Kerberized the workstation, namely the RPM's:

¹The -STABLE branch was CVSup'd daily and taken from the August 2002 timeframe.

- `krb5-libs-1.2.4-1`, and
- `krb5-workstation-1.2.4-1`

1.3 Naming conventions

For purposes of this document, our various namespaces will be handled as follows:

- The DNS domain (“zone”) will be `example.prv`²
- The Kerberos realm will be `EXAMPLE.PRIV`³
- If there was an NIS domain name, it would be `example-nis`⁴

Depending on your Heimdal distribution, you might have commands with or without a “5” in them. For example, `kadmin` might be named `k5admin`. Either version of the commands might be used throughout this document, please adjust according to match your installation.

1.4 Copyright

All contents are copyright © 2001 by Tillman Hodgson. Permission for personal use is explicitly granted. Please contact the author to discuss terms if you wish to seek permission to distribute this document or use it in a commercial setting or for commercial purposes.

1.5 Disclaimer

The author of this document is not responsible for any damages incurred due to actions taken based on this document. This document is meant as a concept paper for secure central authentication using Kerberos, and should not be assumed to be a manual on how to securely install and configure Kerberos. If you do not feel comfortable taking responsibility for your own actions, you should stop reading this document and hire a qualified security professional to handle your Kerberos installation and security for you.

²This is an internal DNS domain, obviously, as `.prv` is not a top-level domain.

³Kerberos realms are traditionally identical DNS zones, but in upper-case in order to easily distinguish them.

⁴Traditionally, the NIS domain name is the same as the DNS domain name, but this is not required and tends to cause confusion over which namespace is being discussed.

The Heimdal Kerberos KDC

2

2.1 What is the KDC?

The KDC, or Key Distribution Center, is the centralized authentication service that Kerberos provides — it is the computer that issues Kerberos tickets.

Kerberos operates with a The KDC is considered “trusted” by all other computers in the Kerberos realm, and thus has heightened security concerns.

2.2 Resources required

Running the Kerberos server requires very little CPU power and a small amount of disk. An old PC with some hundreds of megabytes of free disk space should do fine. Most of the disk space will be used for various logs.

In spite of this, a dedicated machine acting only as a KDC is recommended for security reasons.

2.3 The rc.conf changes

Ensure that your `/etc/rc.conf` file contains the correct settings to act as a KDC (you may need to adjust paths to reflect your own system):

```
1  kerberos5_server_enable="YES"
2  kerberos5_server="/usr/libexec/kdc"
3  kadmind5_server_enable="YES"
4  kadmind5_server="/usr/local/libexec/kadmind"
5  kerberos_stash="YES"
```

2.4 The krb5.conf config file

The `/etc/krb5.conf` for our sample realm looks like this:

```

1  [libdefaults]
2      default_realm = SEEKINGFIRE.PRV
3
4  [realms]
5      SEEKINGFIRE.PRV = {
6          kdc = pluto.seekingfire.prv
7      }
8  [domain_realm]
9      .seekingfire.prv = SEEKINGFIRE.PRV

```

2.5 Creating the database

The keys of all the principals are encrypted with a master password. You don't have to remember this password, it will be stored in a file (`/var/heimdal/m-key`, generally). To create the master key, run `kstash` and enter a password.

Once the master key has been created, you can initialize the database using the `kadmin` program with the `-l` option (standing for “local”). this option instructs `kadmin` to modify the database files directly rather than going through the `kadmind` network service. This handles the chicken-and-egg problem of trying to connect to the database before it is created. Once you have the `kadmin>` prompt, use the `init` command to create your realms database.

Lastly, while still in `kadmin`, create your first principal using the `add` command. Stick to the defaults options for the principal for now, you can always change them later with the `modify` command¹.

A sample database creation session is shown below.

```

1  # kstash
2  Master key: xxxxxxxx
3  Verifying password - Master key: xxxxxxxx
4
5  # kadmin -l
6  kadmin> init SEEKINGFIRE.PRV
7  Realm max ticket life [unlimited]:
8  kadmin> add tillman
9  Max ticket life [unlimited]:
10 Max renewable life [unlimited]:
11 Attributes []:
12 Password: xxxxxxxx
13 Verifying password - Password: xxxxxxxx

```

2.6 Starting the KDC and testing it

Rather than rebooting and letting your `rc.conf` changes bring up the KDC, you can start it manually by running the full path to your KDC and putting it in the background. For example, `/usr/libexec.kdc &`. Note that this does not start the `kadmind` service, and so remote `kadmin` sessions will not work. However, at this point you should be able to use `kinit` and `klist` commands to get and list your ticket, illustrated below.

¹You can also use the `?` command at any prompt to see what your options are.

```
1 $ k5init tillman
2 tillman@SEEKINGFIRE.PRV's Password:
3
4 $ k5list
5 Credentials cache: FILE:/tmp/krb5cc_500
6     Principal: tillman@SEEKINGFIRE.PRV
7
8     Issued           Expires           Principal
9 Aug 27 15:37:58 Aug 28 01:37:58 krbtgt/SEEKINGFIRE.PRV@SEEKINGFIRE.PRV
10 Aug 27 15:37:58 Aug 28 01:37:58 krbtgt/SEEKINGFIRE.PRV@SEEKINGFIRE.PRV
11
12 v4-ticket file: /tmp/tkt500
13 k5list: No ticket file (tf_util)
```

Heimdal Kerberos Clients

3

3.1 Resources required

Kerberized clients don't require realistically measurable increased resources over non-Kerberized clients. Size your client resources normally.

As far as Kerberos software goes, you'll need to put:

- A Kerberos configuration file, usually located at `/etc/krb5.conf`
- The core Kerberos applications: `kinit`, `klist`, `kdestroy` and `kpasswd` (these are likely renamed to `k5init`, `k5list`, `k5destroy` and `k5passwd` on systems such as FreeBSD which support both version 5 and version 4 of Kerberos)
- Whatever client applications are needed, such as Kerberos versions of `telnet`, `ftp`, and `rsh`

3.2 The `krb5.conf` file

Simply copy the `/etc/krb5.conf` file from your KDC over to the client.

3.3 The Kerberos applications

Ensure that your operating system has Heimdal installed, with the client applications *before* their non-Kerberos counterparts on a typical users path. You may find that users will be very sensitive to the path order, as the wrong order results in the non-Kerberos version of a client application being run (usually unsuccessfully).

3.4 Testing

Once you have both a server offering Kerberos services and a user who can successfully get and list a ticket, you can test the Kerberized client by attempting to use any service. Typical gotcha's are path issues, not having a ticket, and server-side issues.

3.5 Adding new principals

You can add new principals using the same `add` command that you used to create the first principal.

DRAFT

Heimdal Kerberos Servers

4

4.1 Resources required

Kerberized servers don't require realistically measurable increased resources over non-Kerberized servers. Size your server resources normally.

As far as Kerberos software goes, you'll need to put:

- A Kerberos configuration file, usually located at `/etc/krb5.conf`
- Kerberos application server daemons, such as `telnetd`, `ftp`, etc (this could also include applications with use Kerberos for authentication only, such as PostgreSQL)
- At least one encryption key, stored in a keytab file usually located at `/etc/krb5.keytab`

4.2 The keytab file

This is the major difference between a Kerberized server and a Kerberized workstation — the server must have a keytab file. This file contains the Kerberized servers host key, which allows it and the KDC to verify each others identity. It must be transmitted to the Kerberos server in a secure fashion, as the security can be broken if the key is made public.

Typically, you transfer to the keytab to the Kerberos server using the `kadmin` program. After installing the `/etc/krb5.conf` file, you can use `kadmin` from the Kerberos server. The `add --random-key` will let you add the servers host principal, and the `ext` command will allow you to extract the servers host principal to its own keytab. For example:

```
1 kadmin> add --random-key host/myserver.seekingfire.prv
2 Max ticket life [unlimited]:
3 Max renewable life [unlimited]:
4 Attributes []:
5 kadmin> ext host/myserver.seekingfire.prv
```

You can confirm that the server has its own principal in its keytab by using the `ktutil list` command.

4.3 Setting up the telnet service

Add something like the following entry to your `/etc/inetd.conf` file. The critical bit is that the `-a` (for authentication) type is set to `user`. Restart `inetd`¹ to have the change take effect. Note that the telnet daemon will not work if your keytab does not contain your host principal.

```
1 telnet stream tcp nowait root /usr/local/libexec/telnetd telnetd -a user
```

Note that the Heimdal `telnet` client automatically encrypts the data stream, making it virtually equivalent to the common functionality of `ssh`.

¹Sending `inetd` the `-HUP` signal is a standard way to do this.

5.1 What the difference between MIT and Heimdal clients?

The format of the `/etc/krb5.conf` file is different. It is recommended that you modify the sample config file on your MIT client to suit your Kerberos environment rather than trying to create a config file from scratch.

The client applications are also different, being independently written. The default options are often different¹. They are mostly interoperable, however, and an MIT client can be treated much like a Heimdal one.

5.2 Fixing the broken bits with a travel kit

If any of your Kerberos client applications don't seem to interoperate, try replacing the MIT version of the application with the Heimdal one from a "travel kit". Pre-compiled travel kits for a variety of operating systems are available from <http://meta.cesnet.cz/software/travelkit//index.en.html>.

¹For example, telnet does not encrypt the data stream by default.

6.1 The similarities

MIT Kerberos work just like Heimdal Kerberos servers, you set up the MIT network daemons that you wish to use and modify the `/etc/krb5.conf` file to point to your realm and KDC.

6.2 The keytab file

The only stumbling block is the `/etc/krb5.keytab` file. The `kadmin` protocol used by MIT isn't interoperable with Heimdal, so you can't simply use `kadmin` and extract the host principal. Instead, you use `kadmin` from a Heimdal installation (probably on a different machine) and extract it there. You can then use a secure file transfer tool (like `scp`) to transfer the keytab file to the MIT computer.

When doing this, be careful that you use `kadmin` on a *new* keytab file so that you don't accidentally mix your new host's principal in with the Heimdal host's keytab (and possibly distribute the other host's complete keytab by accident!). If you suspect that has occurred, `ktutil` has a useful `list` sub-command (and it takes a `-k` option to specify an alternate keytab file). If you need to clean up a keytab, use the `remove` sub-command.

Background

A

A.1 Kerberos

A.1.1 What is Kerberos?

Source material

The following description of Kerberos was inspired by the Kerberos FAQ¹.

A non-technical description

The problem that Kerberos attempts to solve is that the Internet is an insecure place. Many of the protocols commonly used on the Internet do not provide any security at all, or provide only “clear text” (unencrypted) authentication of users. This is dangerous because tools to “sniff” passwords off of the network are in common use by systems crackers. Thus, applications which send an unencrypted password over the network are extremely vulnerable. Worse yet, other client/server applications rely on the client program to be “honest” about the identity of the user who is using it. Other applications rely on the client to restrict its activities to those which it is allowed to do, with no other enforcement by the server.

Kerberos was created by MIT as a solution to these network security problems. The Kerberos protocol uses 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 identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.

Kerberos provides only one function — the secure authentication of users on the network. It does not provide authorization functions (what those users are able to perform) or auditing functions.

Kerberos can be described as an identity-verifying proxy, it can also be described as a trusted² third-party authentication system.

¹A copy of the FAQ is posted regularly to the newsgroup comp.protocols.kerberos and is also mirrored on many places on the web, including <http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html>.

²Note that “trusted” has a specific meaning when applied to security. It is not a system that is trustworthy, but simply one that has to be trusted (in other words, a potential failure point).

A technical description

Kerberos is both the name of a network authentication protocol and an adjective to describe programs that implement the program (Kerberos telnet, for example). The current version of the protocol is version 5, described in RFC 1510³. Kerberos was designed to provide strong authentication for client/server applications (such as traditional Internet services like FTP and telnet) by using secret-key cryptography⁴. Several free implementations of this protocol have been developed. The Massachusetts Institute of Technology, where Kerberos was developed, continues to develop their Kerberos package and it is commonly used in the US (as a cryptography product, it is affected by US export regulations). Heimdal Kerberos is another version 5 implementation, and was explicitly developed outside of the US to avoid export regulations (and is thus often included in non-commercial Unix variants).

A.1.2 Kerberos terms

Client Used interchangeably to mean a user or a host. In both senses of the term, the focus is on their ability to obtain a ticket.

Host A computer that can be accessed over a network. This can be either a server or a workstation.

KDC Key Distribution Center. This is the heart of Kerberos as it is the computer that issues Kerberos tickets. The KDC is considered “trusted” by all other computers in the Kerberos realm, and thus has heightened security concerns.

Keytab A file containing one or more Kerberos keys. A host or service uses a keytab file in much the same way as a user uses his/her password. The keys are part of the shared secret that hosts in a Kerberos realm use to prove their identity.

Principal A name used by Kerberos to specify a particular entity (user, host or service) to which a set of credentials may be assigned. The general format is `primary/instance@REALM` and an example would be `joeuser/admin@SEEKINGFIRE.PRIV`. The principal breaks down into three parts:

Primary The first part of a Kerberos principal. In the case of a user, it is the username. In the case of a service, it is the name of the service. This part always exists.

Instance The second part of a Kerberos principal. It gives information that qualifies the primary. The instance may be null (i.e., it may not exist). In the case of a user, the instance is often used to describe the intended use of the corresponding credentials. In the case of a host, the instance is the fully qualified hostname.

realm The logical network served by a single Kerberos database and a set of Key Distribution Centers (which share the database).

Realm A Kerberos realm is an administrative domain in which all computers agree that they are part of the same grouping. A Kerberos realm is usually named by uppercasing the DNS domain name associated with the hosts in the Kerberos grouping⁵. For example, if all the hosts in `example.org` are to be grouped into a single Kerberos realm the realm would likely be called `EXAMPLE.ORG`. It's also possible to have more than one Kerberos realm associated with the same DNS domain name, something that might occur if there are more than one administrative area within the DNS domain (thus requiring

³RFC 1510 is available at <http://www.ietf.org/rfc/rfc1510.txt>.

⁴As opposed to public-key cryptography.

⁵this is specified in recent versions of the Kerberos standard.

more than one Kerberos realms). Usually realms are created to match DNS sub-domains. For example, the DNS domain `example.com` might have the sub-domains `engineering.example.com` and `marketing.example.com`. The corresponding Kerberos realms would normally be named `ENGINEERING.EXAMPLE.COM` and `MARKETING.EXAMPLE.COM`.

Service Any client/server program or host (computer) you can access over the network. Examples of services include `ftp`, `pop` and `krbtgt` (for the TGT). Remote shell type services (such as `telnet` and `rsh`) fall under the service name `host`.

Ticket A temporary set of electronic credentials that verify the identity of a client for a particular service. Tickets are unique to clients and generally are set to expire after a certain period of time.

TGS & TGT Ticket-Granting Service and Ticket-Granting Ticket. When a user first uses the `kinit` command to authenticate themselves within the Kerberos realm they are connecting to the KDC and asking for a TGT (using their password to encrypt the ticket). When the same user then wants to use a Kerberos service (such as the `telnet` daemon on a Kerberos server within the realm), they use the TGT to talk to the Ticket-Granting Service (built into the KDC) which uses their TGT to verify their identity before issuing a ticket for the desired service. Thus, a TGT is a special Kerberos ticket that permits the client to obtain additional Kerberos tickets within the same Kerberos realm from the Ticket-Granting Service.

A.1.3 How Kerberos works

The following description of how Kerberos works was taken from *The Moron's Guide to Kerberos*⁶ and lightly modified.

Both the user and the service are required to have keys registered with the KDC. The user's key is derived from a password that he chooses; the service key is a randomly selected key (since no person is available to type in a password). For the purposes of this explanation, let us imagine that messages are written on paper (instead of being electronic), and are "encrypted" by being locked in a strongbox by means of a key. In this "box world," principals are initialized by making a physical key and registering a copy of the key with the KDC.

1. First the user sends a message to the KDC: "I, J Random User, would like to talk to the server named Foo."
2. When the KDC receives this message, it makes up two copies of a brand new key. This is called the session key. It will be used in the direct exchange between user and service.
3. It puts one of the session keys in Box 1, along with a piece of paper with the name "Foo Server" written on it. It locks this box with the user's key. Why is this piece of paper here? Recall that this box is really just an encrypted message, and that the session key is really just a sequence of random bytes. If Box 1 only contained the session key, then the user wouldn't be able to tell whether the response came back from the AS, or whether the decryption was successful. By putting in "Foo Server," the user (or more precisely, the user's program) will be able to verify both that the box comes from the AS, and that the decryption was successful.

⁶A copy of which can be found at <http://www.isi.edu/~brian/security/kerberos.html>

4. It puts the other session key in a Box 2⁷, along with a piece of paper with the name “J Random User” written on it. It locks this box with the service’s key.
5. It returns both boxes to the user⁸.
6. The user unlocks Box 1 with his key, extracting the session key and the paper with “Foo Server” written on it.
7. The user can’t open Box 2 (since it’s locked with the service’s key). Instead, he puts a piece of paper with the current time written on it in Box 3, and locks it with the session key. He then hands both boxes to the service.
8. The service opens the Box 2 with its own key, extracting the session key and the paper with “J Random User” written on it. It then opens Box 3 with the session key to extract the piece of paper with the current time on it. These items demonstrate the identity of the user. The timestamp is put in Box 3 to prevent someone else from copying Box 2 (remember, these are simply electronic messages) and using it to impersonate the user at a later time. Because clocks don’t always work in perfect synchrony, a small amount of leeway (about five minutes is typical) is given between the timestamp and the current time. In addition, the service maintains a list of recently sent authenticators, to make sure that they aren’t resent in quick order. You may wonder how the service is able to open Box 2, if there isn’t anyone “back there” to type in a password. Well, the service key isn’t derived from a password. Instead, it’s randomly generated, then stored in a special file called a service key file. This file is assumed to be secure, so that no one can copy the file and impersonate the service to a legitimate user.

Another great non-technical description of how Kerberos works and the issues that the design resolves is the document *Designing an Authentication System: a Dialogue in Four Scenes*⁹.

A.1.4 Weaknesses in Kerberos

There were assumptions made during the design of Kerberos that create weaknesses if those assumptions turn out not to be true. Specifically, Kerberos assumes that users won’t make poor choices for passwords and that workstations and servers are reasonably secure (i.e. that physical access to the computer should be considered safe).

If a user selects an easily guessable password then simply intercepting a few encrypted messages gives enough information to an attacker to allow them to run a dictionary attack¹⁰ on the password. Strong passwords are critical.

If the workstations or servers are not reasonable secure, devices like keystroke loggers can compromise the system.

However, the degree to which Kerberos is compromised can be limited, depending on which host is compromised and how the compromise was performed. If an attacker breaks into a host and steals all of the

⁷In Kerberos terms, Box 2 is called the ticket, and Box 3 is called the authenticator. The authenticator typically contains more information than what is listed here. Some of this added information arises from the fact that this is an electronic message (for example, there is a checksum). There may also be an encryption key in the authenticator to provide for privacy in future communications between the user and the service.

⁸Note that in version 4 of the protocol, Box 2 was placed (unnecessarily) in Box 1.

⁹Available at <http://web.mit.edu/kerberos/www/dialogue.html>

¹⁰An attack in which a large “dictionary” of common passwords is tried sequentially in the hope that the correct one is found through brute force

tickets stored on that machine, he can impersonate the users who have tickets stored on that machine until those tickets expire (which is a good reason to keep the expiry time short). If the attacker discovers a user's password (for example, if their password was weak), then the attacker can impersonate that user indefinitely (or until the password is changed). If an attacker breaks into the KDC, then the entire realm is compromised.

Note that version 4 of the Kerberos protocol had some additional weaknesses.

A.1.5 Securing the KDC

The database of keys stored on the KDC is trusted by all computers in the Kerberos realm and should be considered a weak point in the realm's security. If the database is compromised the realm it serves is completely compromised as well. Securing the KDC is a top priority when designing a Kerberos implementation.

Factors to consider in securing the KDC include:

- The KDC should be a stand-alone machine, not offering any services aside from acting as the KDC
- The KDC should only allow logins on the console, and the console should be physically secured
- The KDC should be reliable and backups properly maintained, as downtime will result in all Kerberized services being unavailable (unless slave KDC's are in use)
- The backups should be treated with the same security care, as they contain a copy of the keys database

Material not covered

B

B.1 Integrating with NIS or LDAP

Integrating Kerberos authentication with NIS or LDAP authorization isn't covered in this document.

B.2 Scalability and Fault Tolerance

The issue of using NIS and Kerberos slave servers to distribute the load and provide fault tolerance hasn't been addressed in this document.

B.3 Time Synchronization

Kerberos requires reasonably accurate time synchronization. While this isn't discussed in this document, a proper NTP configuration is important for all hosts participating in the central authentication system. The question of whether it would be more secure to have the NTP master reside on the Kerberos server (where it would be a potentially exploitable service) or on another machine (where it would be more easily spoofed in order to trick the Kerberos master) isn't addressed.

B.4 Integrating Kerberos with DNS

Using Hesoid DNS entries or other Kerberos integration with DNS issues are not addressed.

References

C

C.1 Books

- Hal Stern, Mike Eisler, and Ricardo Labiaga, *Managing NFS and NIS (2nd Edition)*, O'Reilly 2001

C.2 Web resources

- The FreeBSD Handbook,
http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/index.html
- Securing NIS, <http://www.eng.auburn.edu/users/doug/nis.html>
- Kerberos FAQ v2.0, posted regularly to `comp.protocols.kerberos`
- Designing an Authentication System: a Dialogue in Four Scenes,
<http://web.mit.edu/kerberos/www/dialogue.html>
- A Moron's Guide to Kerberos, <http://www.isi.edu/~brian/security/kerberos.html>

C.3 Usenet resources

C.4 Mailing lists

Acknowledgements

D

I'd like to thank the fine folks on the `heimdal-discuss` mailing list for their assistance in educating me in the finer points of Kerberos:

- Simon Wilkinson, for writing the GSSAPI patches for OpenSSH and talking the time to talk about them
- Douglas E. Engert, for helping me understand why I'd want to use GSSAPI
- Johan Danielsson, for helping me work out krb4/5 service issues