Heimdal—an independent implementation of Kerberos 5

Johan Danielsson
Parallelldatorcentrum, KTH
joda@pdc.kth.se

Assar Westerlund
Swedish Institute of Computer Science
assar@sics.se

Abstract

Heimdal is an independently developed and free implementation of the Kerberos 5 protocol, unencumbered by US export restrictions. It is compatible with other implementations and is close to the MIT Kerberos 5 API. It includes versions of common applications such as telnet, ftp, rsh, su, and login. Furthermore, it has some new features not available in other implementations, such as authenticated and encrypted X connections, incremental database propagation, and support for IPv6 and Triple DES[1]. There is also support for operation in firewalled environments.

1 Introduction

Kerberos, developed primarily at the Massachusetts Institute of Technology (MIT), has become a popular security system and is used by many universities and corporations around the world. One big problem has been its legal status. Because of restrictions employed by the United States government, it is not possible to export the source without first going through a lot of bureaucracy and possibly not even then. You can get a copy of the MIT Kerberos 5 implementation from sources outside of the US, but it is unclear how it got there and if you can safely use it. On the other hand, many vendors have some version of Kerberos 5 available to non-US customers, usually only in binary form and without the ability to encrypt user data.

With Kerberos 4 the export dilemma was solved by removing all cryptographic functions and the calls to them, to get something that could be approved of export. To get something that was usable, Eric Young took the exported version, called *Bones*, and replaced the missing pieces, calling the new distribution *eBones*. Since then many different people have been hacking on this, including the authors, adding improvements over time.

[1]When this article is written, work to implement Triple DES in the MIT Kerberos distribution is in progress.

This export trick has not been repeated with Kerberos 5, partly because the MIT code just recently went out of beta, and partly because nobody was willing to do the work.

Since Kerberos 4 is reaching the end of its useful lifetime, we decided that we needed a replacement. A key requirement on a new system was that it could reasonably easy replace an existing Kerberos 4 infrastructure. Therefore compatibility with eBones has been high on the list of things to implement. Support for a large number of systems, including Cray vector machines, was also important for us.

After a short period of discussions about whether and how to go about creating such a beast, we started writing the code that would later become Heimdal. The result is an independent reimplementation of Kerberos 5 (which has proved helpful—we have discovered a number of errors and inconsistencies with the original specification) that is available to people in all parts of the world.

2 What is Kerberos?

Kerberos is a system for authenticating users and services on a network. It is built upon the assumption that the network is "unsafe". For example, data sent over the network can be eavesdropped and altered, and addresses can be faked. Therefore addresses cannot be used for authentication purposes.

Kerberos is a trusted third-party service, meaning that there is a third party (the *Kerberos server*) that is trusted by all the entities on the network (users and services, collectively called *principals*). The principals each share a secret password (or key) with the Kerberos server and this enables principals to verify that the messages from the Kerberos server are authentic. Trusting the Kerberos server, users and services can then authenticate themselves to each other.

For specific details on the protocol refer to other documents on the subject [7, 5, 6, 4].

2.1 The Kerberos server

The central function in a Kerberos environment is performed by the Kerberos server that keeps a copy of the keys of all principals. This function is sensitive, an attacker that obtains a copy of a principal's key can masquerade as that principal. An attacker that obtains a copy of the server's own secret key can masquerade as any principal. Because of this, it is vital to keep the Kerberos server from being compromised.

The server is also a single point of failure. If it is unreachable, no users will be able to authenticate themselves. The solution is to keep several copies of the database, the common way of doing this is with master-slave replication; all changes are performed to the database at the *master* and the database is then periodically copied to one or several *slaves*.

3 Services

Included in the Heimdal distribution are the core Kerberos applications, such as the Kerberos server, applications to obtain and manipulate tickets (*kinit*, *klist*, and *kdestroy*), and administration programs.

Also included are kerberised versions of clients and servers for the some network services: rsh, telnet, ftp, and pop. These programs also support backwards compatibility with Kerberos 4, One Time Password (OTP) [2], and old-style address and password-based authentication.

3.1 X11

Another network service that is quite useful but has some security problems is the X11 protocol [1]. This is a problem because access to an X11 server is mostly binary: either you are allowed all the operations (including listening for key presses) or none at all.

The most commonly used authentication methods are *xhost* and *xauth*.

The problems with xhost are that authentication is done at host granularity and that it trusts IP addresses. This only makes it usable in environments where you trust the other users of multi-users machines and where you have control over the entire networking infrastructure. Needless to say this is not often the case in any but the smallest networks, and specifically not on the global Internet.

Using magic cookies with xauth solves the problems with xhost, but it is not without problems itself. First it is more cumbersome than the magical `xhost +` command. A deeper problem is that it relies on the cookies being secret, which is very difficult to ensure given that they are sent in the clear over the network when initiating an X11 session.

The correct way of solving this problem would be to implement another authentication mechanism that uses Kerberos to authenticate the client and server. This has been specified and implemented in X11R6 [11]. However, these modifications are not well maintained, and have been slow to propagate into the vendor releases of X11. Furthermore they only solve the authentication of the initial connection; having protection against someone modifying the stream and against someone being able to read the traffic are requirements that are not addressed.

Our solution to this problem has been to implement authenticated X11-proxies (see figure 1). The client connects to a local proxy on the same host that forwards the data over an encrypted channel to another proxy on the server host that then talks to the real X11 server. So the first proxy looks like an X11 server to the client and the second one like an X11 client to the server. This is all quite similar to what SSH [10] does.

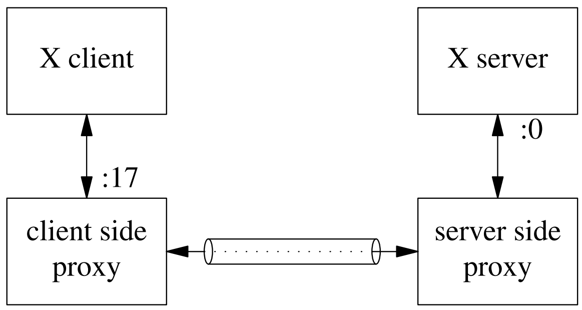

Figure 1: kx authentication

There have been a few problems with this setup. First, not all vendors have compiled in support for using Unix sockets as a transport protocol in their X libraries (we prefer Unix sockets since they give a slightly higher level of security and are more efficient). This is most common with large machines that does not have any graphics console—such as Crays. To get this to work with such machines, we have had to implement support for using TCP connections—with appropriate security precautions. It should also be pointed out that since we bugged them about this, Cray has added support for Unix sockets in their libraries.

Another problem is software that does extra magic when they operate on a local display. One example is the SGI GL-extensions, that work fine over a normal TCP-connection, but when detecting a seemingly local display, insists on talking directly to the X-server. For this particular problem, there is a workaround that involves more magic.

3.2 PAM, afskauthlib, SIA, etc

A practical problem of deploying Kerberos is getting the required support into various system commands (like `login`, and `su`). Included in the Heimdal distribution are generic versions of these programs that should work on most systems. However, some systems' versions of these programs have special features or do weird magic. Furthermore it would be more elegant with a more dynamic way to specify which authentication mechanisms should be used.

As usual, there are several different standards for doing this.

- Pluggable Authentication Modules (PAM): developed by Sun
- Security Integration Architecture (SIA): developed by Digital
- afskauthlib: a hack by SGI

PAM and SIA are general ways of configuring different authentication modules and how these should be used by different programs. The third, however, is a very small but useful hack in the SGI versions of `login` and `xdm` programs that is intended for adding AFS support. We use it to implement Kerberos authentication in these programs. PAM has been incorporated by Sun into Solaris and by an independent group into Linux. Digital Unix is (as far as we know) the only system that uses SIA.

4 Implementation

4.1 Incremental database propagation

As we mentioned before, there are a number of reasons for wanting to keep replicated Kerberos servers in a *realm* (a realm is an administrative domain). The most important is of course the added reliability. Load sharing can also be of a practical importance if there is a very large number of requests. If the realm is spread over a large geographic area, reducing network traffic can be another reason for keeping several servers.

Any system that has a replicated database has to maintain some level of consistency between the different sites, otherwise users get quite confused. It would be possible to do this with some kind of distributed database, but because of the lack of freely available, light-weight, and secure databases, Kerberos systems tend to use some 'proprietary' system. The common way is to propagate the whole database once every certain amount of time (e.g. 1 hour). This is not an optimal solution; the slave database could be out of sync for this amount of time, and the propagation process uses machine resources and network bandwidth, making it impractical to have a very short propagation interval.

We have addressed this problem by implementing incremental propagation that will only send database changes from the master to the slaves. This way, the changes can also be transferred immediately, keeping the slave databases more fresh. All changes to the database receive an increasing *version number* and are written to a log on the master. The propagation daemon tries to keep track of the current version of all the slaves and will send out updates as soon as they are applied to the master database. The slaves also keep track of the last version number they have seen and will request all modifications between this version and the current one when they connect to the master daemon. As a fall-back, if a slave falls too far behind, the whole database will be sent.

4.2 The ASN.1 shop of horror

One of the major hassles with implementing Kerberos 5 turns out to be its use of ASN.1 [3]. ASN.1 is a system for encoding structures so they can be sent over the network to another, possibly architecturely quite different, machine. While this is a sound thing, ASN.1 is overly complex, which is made clear by the many bugs in the encoding you have to be compatible with to successfully inter-operate with other implementations.

We have chosen to do the ASN.1 support with the help of a compiler that takes the Kerberos 5 ASN.1 specification, and produces a set of functions to encode, decode, and otherwise manipulate these structures. There are a few such compilers available, but since we did not find any of them likeable enough, we decided to write our own. It does not have support for the incredibly large number of constructs available in ASN.1, it supports the small subset that is actually used in Kerberos 5, and also has a reasonably small footprint.

4.3 Combating firewalls

Firewalls can be a big problem for Kerberos users. Since Kerberos is not as commonplace as many other services, most firewalls have the special ports used by Kerberos turned off by default, so these will have to be specifically opened if any packets are to get through. Another problem is that Kerberos' default mode is to use UDP as its transport protocol and UDP is commonly regarded by firewall advocates to be an insecure protocol.

To solve some of these problems, we have added support for both TCP and HTTP as additional transport mechanisms. The TCP support is compatible with the soon-to-be updated Kerberos RFC. The HTTP support was added since we discovered that it can be quite difficult to convince firewall administrators to open a new port in their firewall —and very much so if you are just visiting a site. The choice fell on HTTP since, in our experience, most sites do permit access to the WWW, even if it is through a proxy.

4.4 Portability

Heimdal should, in principle, be portable to any system that has:

- an ANSI/ISO C compiler (such as gcc)
- awk
- lex/flex
- yacc/bison
- sh, make, sed, & c:o
- Posix libraries
- Berkeley sockets
- ndbm or Berkeley DB (required by the server)

The build process uses GNU Autoconf to identify the characteristics of the system. More than 50 commonly missing or broken functions are included in a library and conditionally compiled if required on a particular platform.

Heimdal currently builds and runs on most versions of Unix (AIX, NetBSD, OpenBSD, FreeBSD, Digital Unix, HP-UX, Irix, Linux, Solaris, SunOS, Ultrix, Unicos, and UXP/V) and some systems with Unix-like environments (like Windows 95/NT with cygwin32).

5 Compatibility

We believe that Heimdal is protocol compatible with most other implementations of Kerberos. It has been tested with code from MIT, OSF, Cisco, and Microsoft.

6 Future work

Even if it today is possible to penetrate most firewalls, this doesn't always allow you access to Kerberised services. A part of the security of Kerberos requires that the client's network addresses are encoded in the ticket, so the client has to know all possible addresses that it may appear to come from. When you have a firewall between the client and server, the connection will often seem to emanate from the firewall itself. Because of this the client will have to know the address the firewall uses, and add that to the list of addresses put into the ticket. There might be more than one such address. To complicate this, there isn't an easy way for the client to find out which those addresses are (other than via manual configuration).

A possible workaround for this is for the Kerberos server to emit address-less tickets—tickets that are valid from any address. But since the purpose of putting the addresses in the ticket in the first place was to make it harder to steal a ticket, this actually reduces the security. By including the addresses of the firewall, this type of wild card tickets are eliminated, but since *all* tickets issued to clients behind a specific firewall will contain the same addresses, you make the whole firewalled network look like one machine, thus reducing security.

The best way of addressing these problems, as well as other firewall related issues, are still under discussion.

Heimdal is already portable to most Unix like systems, but unfortunately, the world likes to use other operating systems too. We foresee that at least a port to Windows will be done in the future. Other possible systems include the Macintosh and VMS.

Support for public-key cryptography in the initial authentication [8] and for cross-realm operation [9] are other items being considered.

7 Acknowledgments

This work was supported by Parallelldatorcentrum, KTH.

A large number of people have contributed bug-fixes, documentation, and encouragement. They are mentioned in the documentation.

8 Availability

Heimdal is freely available under a BSD-style license. See `http://www.pdc.kth.se/heimdal/`.

9 Mythological background

> Now this Cerberus had three heads of dogs, the tail of a dragon, and on his back the heads of all sorts of snakes.
> — Pseudo-Apollodorus Library 2.5.12

Kerberos is a monster (or a dog, if you prefer the domesticated version), that guards the exit from the Greek underworld Hades (named after its ruler), preventing the spirits from escaping.

Heimdal (also spelt Heimdall) is a god in the Nordic mythology. He is a watchman on the bridge Bifrost that leads to Asgard, the realm of the gods. Asgard is also the location of Valhalla where the men who fall in combat are taken after their death.

Heimdal's duty is to stop any giants from entering Asgard, and he is well suited for this task since he requires less sleep than a bird, and can see for many miles both by night and by day; he can also hear the grass grow, as well as the wool on the sheep's back.

When the world is about to end, Heimdal will blow his Gjallar horn calling the gods to the battle field. In this battle, many of the gods will perish, but a new and better world will rise from the sea.

References

[1] J. Fisher, *Securing X Windows*, U.S. Department of Energy, Computer Incident Advisory Capability document CIAC-2316 (1995)

[2] N. Haller, C. Metz, P. Nesser, M. Straw, *A One-Time Password System*, Network Working Group (1998)

[3] Burton S. Kaliski Jr., *A Layman's Guide to a Subset of ASN.1, BER, and DER*, RSA Laboratories Technical Note (1993)

[4] John Kohl, Clifford Neuman, *The Kerberos Network Authentication Service (V5)*, Network Working Group (1993)

[5] John T. Kohl, B. Clifford Neuman, and Theodore Y. T'so, *The Evolution of the Kerberos Authentication System*, Distributed Open Systems, pages 78-94. IEEE Computer Society Press (1994).

[6] B. Clifford Neuman, and Theodore Ts'o, *Kerberos: An Authentication Service for Computer Networks*, IEEE Communications, 32(9), pages 33-38 (1994)

[7] Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller, *Kerberos: An Authentication Service for Open Network Systems*, Proceedings Winter USENIX Conference, Dallas (1988)

[8] Brian Tung, Clifford Neuman, John Wray, Ari Medvinsky, Matthew Hur, Jonathan Trostle *Public Key Cryptography for Initial Authentication in Kerberos*, Work In Progress, draft-ietf-cat-kerberos-pk-init-06.txt

[9] Brian Tung, Tatyana Ryutov, Clifford Neuman, Gene Tsudik, Bill Sommerfeld, Ari Medvinsky, *Public Key Cryptography for Cross-Realm Authentication in Kerberos*, Work In Progress, draft-ietf-cat-kerberos-pk-cross-04.txt

[10] Tatu Ylönen, *SSH - Secure Login Connections over the Internet*, Sixth USENIX Security Symposium, San Jose (1996)

[11] Tom Yu, *Kerberos Authentication of X Connections*, Proceedings 8th Annual X Technical Conference, Boston (1994)