

Appendix B. About the Kerberos Network Authentication Service V5

In this appendix we provide an introduction to the Kerberos Network Authentication Service V5, discuss the important terms and components, and describe the authentication process.

B.1 Introduction to Kerberos V5

B.1.1 Background

Kerberos V5 is a network authentication protocol designed to serve as a trusted third-party authentication service. It is a single-sign-on system, meaning that a user only has to type his password once, and the Kerberos V5 programs do the authenticating (and, optionally, encrypting) for the user as connections to other machines are made.

Kerberos was developed at MIT in 1987 and has matured into a stable product with widespread operating system and application support. Microsoft has based the authentication in Windows 2000 on Kerberos V5. Kerberos continues to see active development, with new releases occurring approximately twice per year. Kerberos V4 has been in use at Fermilab as part of AFS, and both Kerberos V4 and V5 are widely used at other laboratories and universities.

A machine on which Kerberos has been installed and which enforces the Kerberos authentication is referred to as a *strengthened* or *Kerberized* machine. Kerberos has been built into each of a suite of network programs, including **telnet**, **FTP**, **rsh**, **rcp**, **rlogin** and **ssh**. It can be built into other programs as well. The Kerberized version of a program is also referred to as *strengthened* or *Kerberized*, and requires individual authentication for use.

B.1.2 About Kerberos Authentication

Kerberos verifies the identity of a user or a network service (users and services are collectively called *principals*) on an unprotected network using conventional cryptography in the form of a shared secret key. The shared secret key technology allows a client and server (e.g., a principal and a strengthened machine) to mutually establish their identity across an insecure network connection without exposing passwords. They can also assure integrity and/or privacy of their communications with cryptographic methods.

B.1.3 How Secure is Kerberos?

Password Centralization

In Kerberos V5, the password-checking (authentication) happens in a central place for all the machines in the strengthened realm, not on the end systems. End systems need not store any information which can be used to try to guess a password, and they are not involved in password maintenance or quality control. Let's compare this to standard UNIX and (nonKerberized) ssh:

- For standard UNIX passwords, each end system has to store information sufficient to check the password, which is therefore also sufficient to try to *guess* the password. Another problem is that password changes must be repeated on each system or NIS cluster of systems, and quality, aging and reuse prevention are hard to ensure.
- The problems with UNIX passwords are also present with ssh, and ssh presents a couple of additional problems:
 - RSA keys¹ can give access to various accounts, and there's no way to know with certainty who possesses which keys. In the event of a compromise of a private key, there's no mechanism for locating every host on which the corresponding public key appears. The private keys are protected by passphrases which (a) are often no better than a very short password, (b) are sometimes typed in the clear, and (c) are sometimes completely lacking.
 - It is difficult to determine where a password/account resides. Consequently it is much more difficult to control access in a thorough way.
- The problems with UNIX passwords are also present with ssh. There is in addition the issue of RSA keys². RSA keys can give access to various accounts, and there's no way to know with certainty who possesses which keys. In the event of a compromise of a private key, there's no mechanism for locating every host on which the corresponding public key appears. The private keys are protected by passphrases which (a) are often no better than a very short password, (b) are sometimes typed in the clear, and (c) are sometimes completely lacking.

1. RSA is an authentication method supported by ssh; it is based on public-key cryptography in which encryption and decryption are done using separate keys, and it is not possible to derive the decryption key from the encryption key. The idea is that each user creates a public/private key pair for authentication purposes. The server knows the public key, and only the user knows the private key.

2. RSA is an authentication method supported by ssh; it is based on public-key cryptography in which encryption and decryption are done using separate keys, and it is not possible to derive the decryption key from the encryption key. The idea is that each user creates a public/private key pair for authentication purposes. The server knows the public key, and only the user knows the private key.

Password Compromise

As noted in section A.2 *Goals of Strong Authentication at Fermilab*, it is impossible to entirely prevent the transmission of clear text passwords, but Kerberos V5 removes the most common opportunities as well as most of the necessity for typing a password. Our implementation of Kerberos allows an unencrypted mode of access in order to accommodate users who have no specialized software of any sort available. This was a requirement we had to meet. The down side is, it means that all users must pay attention to whether their connection is encrypted or not whenever they need to type their Kerberos password.



It is possible to issue your Kerberos password over an unencrypted connection, but this is a violation of common sense and FNAL policy! Please see Appendix : *Encrypted vs. Unencrypted Connections* for instructions on how to avoid doing this.

In the event a Kerberos password is stolen by eavesdropping, it's not impossible for the thief to use it, but there is one serious obstacle: Because a system configured according to our rules will not accept *any* password, correct or incorrect, for a network login (described in section B.4 *The Authentication Process*), the thief must first get onto a system in order to use the stolen password. If the thief installs Kerberos software on his or her own system in order to use the password, we have a record of exactly when and where the password was used.

Furthermore, once into a Fermilab system as a normal user, gaining root access is not necessarily any harder than on other systems, but doing so does not let the perpetrator harvest a password file to crack more passwords, nor exploit any “.rhosts” trusts that may exist. Some valid Kerberos credentials of other users could get stolen, but those are strictly time-limited in value and do not contain information which can be used to guess another password.

B.2 Keys, Tickets and the KDC

Kerberos authentication is implemented primarily via a service called the *key distribution center (KDC)*.¹ The KDC shares a permanent secret key with each principal (user and service).² Most KDC implementations store the principals in a database; therefore the term “Kerberos database” is sometimes applied to the KDC. The KDC implements the Authentication Service (AS) and the

1. A Kerberos strengthened realm has one primary KDC, and may have one or more secondary KDCs. We refer to them here collectively as “the KDC”. Authentication is still possible if the primary KDC is not reachable, but certain administrative tasks are not (e.g., changing passwords, creating new principals).

2. For a user, this shared secret key is a hash of the user's password; for a service, the key is a random bit string.

Ticket-Granting Service (TGS) for all the machines in the realm. To understand what these do, you first need to know what session keys, tickets and credentials are:

Session Key

A session key is a temporary secret encryption key, generated at random by the KDC to be shared between two principals (usually a user and a service). Its validity is limited to the lifetime of an accompanying ticket. The session key is used to authenticate the two principals to each other, possibly multiple times during the ticket lifetime. Its purpose is to limit the use of the permanent key (which for a user is derived from the password) over the network. If encryption or integrity protection of bulk data is required, yet another key is negotiated by the two principals, called a *subkey* or a *sub-session key*.

Ticket Kerberos uses encrypted records called *tickets* to authenticate to Kerberized services¹. Tickets generally contain the session key, the user and service ids and the client's IP address. Some of the information is encrypted with the service's permanent key, known only to the service and the KDC. A ticket is accompanied by an extra copy of the session key encrypted under the user's key. The ability of both user and service to correctly decrypt the relevant parts of the ticket establishes knowledge of the correct keys and therefore establishes authentication for the service.

Credential The combination of the ticket and the session key is called a *credential*.

The Authentication Service (AS) issues secret session keys and credentials based on a user password or encryption key. It can issue both Ticket-Granting Tickets (TGTs) and individual service tickets. A TGT is a ticket that authenticates a user process to the Ticket-Granting Service (TGS) portion of the KDC. The Ticket-Granting Service (transparently) issues tickets to clients for individual Kerberized services.

B.3 Fermi vs. Standard MIT Kerberos

The Computing Division at Fermilab has taken the MIT Kerberos V5 product and modified it to provide additional features. (Some of these in turn have been incorporated into MIT's releases.) The "Fermi Kerberos" is packaged as

1. Technically, both a ticket and a record called an *authenticator* are required. An authenticator is generated and sent by the user process any time a ticket gets used. It contains, among other things, a timestamp and optionally a sequence number, all encrypted with the session key in the ticket. This proves to the service that the client knows the session key, and hence is the legitimate holder of the ticket, and that this is not an adversary's replay of a previously used ticket/authenticator.

the UPS/UPD product **kerberos** for Fermilab-supported UNIX systems and, recently, also in RPM format for FRHL. It is available in the central product repository, KITS¹. The most important features that have been added include:

- 1) CRYPTOCARD logins through telnet and FTP.
- 2) The tools to do authentication of users' cron jobs.
- 3) Flexible fallback to a non-Kerberized client if you default to encryption "on" but connect to a non-Kerberos server.
- 4) An FTP client that plays nicely with emacs' efs mode.

Users whose operating systems are not supported at Fermilab, or who don't use UPS/UPD for other reasons, have been installing Kerberos V5 from non-Fermilab sources.

B.4 The Authentication Process

When a user logs in to a strengthened machine, or runs **kinit** (described in section 9.2.1 *Obtaining Tickets (Authenticating to Kerberos)*), the Kerberos program transmits some short "behind-the-scenes" messages. First it sends a message, encrypted with (but not containing) your password, to the KDC. This message also contains a timestamp, to confirm that you gave the right password very recently. The KDC attempts to decrypt the message with its copy of your password. If it can do so, and if the timestamp is recent, the KDC believes you know the password, and that you are who you say you are. This portion of the exchange is called *preauthentication* (error messages generated in this portion of the exchange use this word).

Now that the KDC believes you are who you say you are, it makes a Ticket Granting Ticket (TGT), which is sent back to you (also encrypted), and which contains an encryption key for future ticket requests. This gets written in your credential cache, and is the first entry listed when you run **klist** (described in section 9.2.2 *Viewing Tickets*).

When you connect over the network from one Kerberized host to another, your client application obtains a service ticket for the destination (or re-uses a valid one from a credential cache) and presents it, together with an authenticator (see third footnote in section B.2) it constructs fresh for each access, to the target host. The application can optionally forward a TGT to the target host, enabling access from that host to others.

1. There are a few related products that get installed automatically by UPD when kerberos is installed.



Kerberized hosts at Fermilab running AFS are configured to obtain AFS tokens automatically at login via the **aklog** program, provided that a Kerberos ticket has been forwarded to the system. If not, the **kinit** command obtains both a Kerberos ticket and an AFS token. The **aklog** program authenticates to a cell or directory in AFS.