

# Appendix A. Implementation Details of Strong Authentication at Fermilab

In this appendix we discuss the concept of strong authentication and the features and environment as implemented at Fermilab.

## A.1 What is “Strong Authentication”?

### A.1.1 Definition

A succinct definition of strong authentication was given by Tardo and Alagappan<sup>1</sup>:

“Techniques that permit entities to provide evidence that they know a particular secret without revealing the secret.”

In more practical terms, it is a system of verifying workstation user and network server identities on an unprotected network in which the parties must demonstrate knowledge of a “secret” rather than transmit a password. Typically the verification is done via a trusted third-party authentication service using conventional cryptography. Strong authentication avoids relying on authentication by the host operating system or basing trust on host addresses. It does not require that the network be safe from eavesdropping, or from injection of hostile packets or alteration/deletion of packets<sup>2</sup>.

## A.2 Goals of Strong Authentication at Fermilab

Fermilab must demonstrate to the DOE that it is implementing a computer security system that exercises tight control over who uses the lab’s computers and network (which are owned by the government). The Computing Division has been charged with implementing Strong Authentication to meet Fermilab’s obligation.

---

1. J.J. Tardo and K. Alagappan, “SPX: Global Authentication Using Public Key Certificates.” In *Proc IEEE Symp. Research in Security and Privacy*. IEEE CS Press, 1991.

2. The Kerberos authentication process can fail if too many packets are altered or deleted (e.g., all of them in one or both directions, until the client gives up).

A primary goal of this effort is to essentially eliminate the transmission of clear text reusable passwords over the network and their storage on local systems. It is impossible to entirely prevent the transmission of clear text passwords, but we are implementing a solution that removes the most common opportunities as well as most of the necessity for typing a password.

Other important goals for us include:

- Providing a single sign-on environment for users
- Providing access to users who have no specialized software (this necessitates an unencrypted mode of access)
- Integrating existing accounts
- Centralizing account maintenance
- Consistently enforcing password policies such as length, quality and lifetime

## A.3 The Authentication Model Implemented at Fermilab

The strong authentication service implemented at Fermilab is the Kerberos Network Authentication Service V5. We describe many of its features in Chapter : *About the Kerberos Network Authentication Service V5*. In this section we describe the model more generally.

### A.3.1 The Realms

The model employed at Fermilab divides the computing environment into three *realms*:

#### The strengthened realm

The strengthened realm consists of all systems (whether on- or off-site) that require strong authentication for access from the network. On a strengthened system, all traditional means of access that use weak authentication, such as **telnet**, **rlogin**, **FTP**, and so on, are replaced with strengthened versions of these programs. Means of access over the network that do not involve passwords are allowed. Weak authentication (standard security) is allowed for local access only, i.e., via the console or locally attached display.

The production realm at Fermilab is called FNAL.GOV<sup>1</sup>.

#### The trusted realm

Other sites which implement strong authentication, and which meet certain criteria, may be recognized by the strengthened realm at Fermilab as a “trusted” realm. Trusted realms provide levels of security and authentication equivalent to our own. Trust relations (cross-authentication) between the trusted realm and the strengthened realm allow access without further authentication (i.e., the authentication takes place only when user accesses either realm individually).

#### The untrusted realm

---

1. As long as the PILOT.FNAL.GOV realm is operating in parallel, the information in this manual applies equally to both realms.

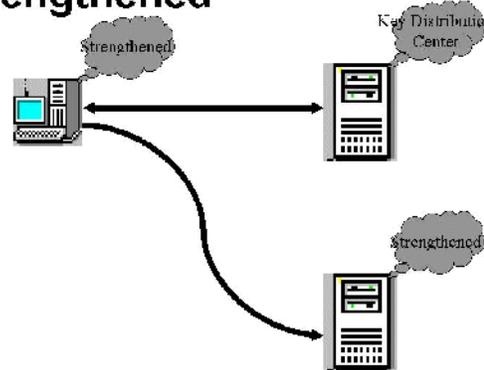
The untrusted realm consists of those systems that do not require strong authentication and that permit traditional means of access. These systems typically expose clear-text passwords on the network.

### A.3.2 Relationships between the Realms

The figures below illustrate the relationships between these realms. (The Key Distribution Center, or KDC, shown on these figures is described in Chapter : *About the Kerberos Network Authentication Service V5.*)

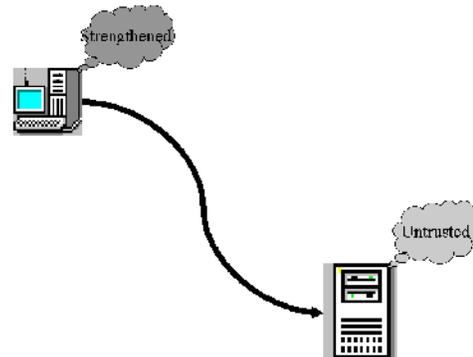
Direct connections between machines in the strengthened realm are allowed

#### Strengthened to strengthened



(the Key Distribution Center is involved in providing credentials to the client's machine which can be passed along to access the other strengthened machine).

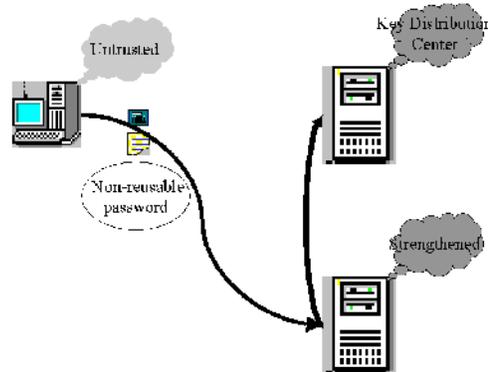
#### Strengthened to untrusted



Direct connections *from* the strengthened *to* the untrusted realm are allowed.

One-time passwords are used for direct connections from the untrusted to the

## Untrusted to strengthened

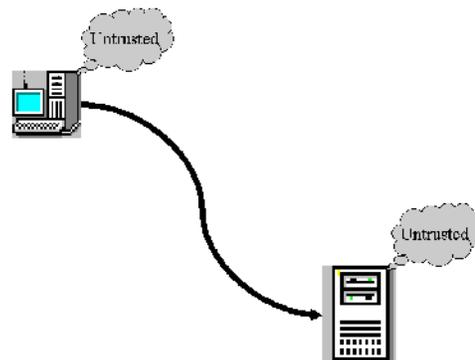


strengthened realm at Fermilab. Strengthened machines are configured to respond in *portal mode* when requests for access come from machines in the untrusted realm. In portal mode, the strengthened machine acts as a secure gateway into the strengthened realm, requiring a single-use password for authentication. This avoids transmission of reusable clear-text passwords over a potentially unprotected network.

Different programs exist for generating non-reusable passwords, and at Fermilab we currently support CRYPTOCARD (described in Appendix : *Using your CRYPTOCARD*). No special hardware or software is required on the untrusted system.

For connections between untrusted machines, strong authentication is not

## Untrusted to untrusted



involved. The standard network programs are used in the normal way.

## A.4 Features of Strong Authentication at Fermilab

The strong authentication model implemented at Fermilab:



- improves authentication and access control
- is adaptable to new computer security threats and changes in system security requirements and to new styles of computing
- is integrated with AFS (I.e., if your machine is part of the strengthened realm and it runs AFS, then when you log on and get Kerberos authentication, you also automatically get an AFS token.)
- is robust and stable
- can be readily deployed to collaborating universities and laboratories, including those outside the United States
- accommodates all the supported UNIX operating systems, as well as Windows and Macintosh systems<sup>1</sup>
- is capable of establishing trust relationships with other institutions where similar strong authentication systems are in place, allowing each user to have a single identity (userid) encompassing multiple institutions
- provides meaningful improvements in security and authentication for the Run II experiments, and is incorporated into the Run II software infrastructure
- provides access for users and systems from outside the strengthened realm via the portal function, without the installation of special hardware or software on the users' desktops (this allows access via systems that do not or can not have strong authentication directly installed, e.g., a public terminal, a "dumb" X terminal, or a PalmPilot)

---

1. Certain systems, such as embedded systems or specialized on-line systems may not be capable of participating directly in strong authentication. These systems may be accommodated by alternate access.



# 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 keys<sup>1</sup> 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 keys<sup>2</sup>. 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)*.<sup>1</sup> The KDC shares a permanent secret key with each principal (user and service).<sup>2</sup> 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 services<sup>1</sup>. 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, KITS<sup>1</sup>. 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.

# Appendix C. More about Choosing a Principal Name

In this appendix, we present information for users who have pre-existing account names and/or an email address at Fermilab, and for whom the guidelines in Chapter 3: *Kerberos Principals and Passwords* are not straightforward to follow.

## C.1 Guidelines for Choosing a Kerberos Principal

In Chapter 3: *Kerberos Principals and Passwords*, we provided the following guidelines for choosing a Kerberos principal and system login ids:

- New principals should be chosen to be eight or fewer characters, and may include a variety of characters. Please use only lowercase letters (and optionally any numbers 0 through 9). **Do not use the characters “at” (@), forward slash (/), or period (.) in principal names.**
- (New users) Choose one login id (account name) common to all systems at Fermilab that you use, and use this id also as your Kerberos principal name.

If you have pre-existing accounts which make the above guidelines hard to follow, here are further guidelines:

- 1) If your existing primary system login name (UNIX and/or NT) is eight or fewer characters, then use this login name for your Kerberos principal. Notes:
  - If your email address and your primary login name do not match, choose the login name as your principal, not your email address. The Computing Division will reserve this login name for you as an email address name. You may continue to use your existing email address on the mail server for a limited time (not yet specified); please transition to the new one. Separate forwards for the two will not be supported.
  - If your primary login name has ever been used as an email address by an individual besides yourself, you must choose a different name for your Kerberos principal. In fact you will need to relinquish the old login name on each system as it becomes Kerberized.
- 2) If your primary login name is longer than eight characters, then you can choose between the following two options:



- Choose a new name that is eight characters or less, and use it both as your principal and as a new, common login name for all systems. In this case you will have to move or rename your current accounts and files.
- Go ahead and use the long login name as your principal, but be aware that you will very likely have difficulty using some UNIX resources, and the problems may be hard to diagnose. For example, Solaris currently does not accept login names longer than eight characters.

## C.2 If your Principal and Login Name do not Match

If your principal does not match your login name, then you need to be aware of the following:

- When connecting over the network (**ssh**, **rlogin**, **rsh**, **telnet**, etc.) you'll always have to give the **-l <login\_name>** option (or **login\_name@host:...** for **rcp**), and there will have to be a **.k5login** file in your home directory that lists your principal (see section 9.3.1 *The .k5login File*).
- If using a CRYPTOCARD (described in Chapter 5: *Using your CRYPTOCARD*), you must initially log in to a system on which your login id matches your principal name. If there are no systems for which this is true, you will not be able to log in with the CRYPTOCARD. The portal mode login code assumes that the login name and principal match. For connecting from the initial machine to a second machine with a different login id, see the above note.

## Appendix D. Transition from Pilot to the FNAL.GOV Realm (User Information)

The production realm FNAL.GOV was launched on May 10, 2001. We are now transitioning to this realm from PILOT.FNAL.GOV. We expect the transition phase to end before October 31, 2001. In this appendix, we discuss issues that users need to be aware of.



The information in this chapter assumes that UNIX/Linux machines have the Fermi Kerberos product installed, and Windows systems use WRQ.

### D.1 When did you Get a Principal?

#### D.1.1 Users who started in the Pilot Program (before 5/10/01)

Everyone who had a pilot realm principal before May 10, 2001 has had their principal, password (including expiration date) and CRYPTOCard key replicated in the production realm.

You don't need to do anything differently until any of the Kerberized machines you use migrates to the FNAL.GOV realm. However, we recommend that you complete a couple of preparatory tasks:

- If your pilot realm password is near expiration, then your password for the production realm is, too. Go ahead and change your password on each realm.
- Wherever you have a `$HOME/.k5login` file (described in section Appendix : *The .k5login File*), be sure both your principals are in it. (If those are the only two principals listed, you don't need this file at all.)

#### D.1.2 Users Starting After Production Realm Launched (after 5/10/01)

Principals created after May 10, 2001 have been issued in either FNAL.GOV alone, or in both realms if requested.

You can run `kinit [<yourFNALprincipal>]` on a machine with any version of the Fermi kerberos product as long as `krb5conf` is `v1_0` or newer. To check on a UPS system, run `ups list krb5conf` and see what's current. If a current instance is listed in more than one database, run `echo $PRODUCTS` to see which database is listed first.

## Users with Production Realm Principal Only

If have only a FNAL.GOV principal, you can freely access machines in the FNAL.GOV realm, of course. In addition, if you get initial credentials in FNAL.GOV on one machine, you can use them to log onto a target host in the pilot realm if and only if one of the following conditions hold:

- There's a `$HOME/.k5login` file on the target host that lists your FNAL.GOV principal
- The target host has Fermi kerberos v1\_2 or later and krb5conf v1\_3 or later, and there is no `$HOME/.k5login` file.

## D.2 Introduction to the Dual-Realm Environment

### D.2.1 The Relationship between the Realms

Trust between the two realms is two-way and transparent. While a Kerberized machine is configured to have a default realm<sup>1</sup>, it may still recognize and honor credentials issued for another realm. In addition to the trust relationship, this just requires and that the client be listed in the `.k5login` file. As long as a machine recognizes both realms and recognizes your principal, it doesn't matter which realm your principal is in, you may freely access Kerberized resources on that machine.

### D.2.2 About the Fermi Kerberos Versions

Fermi kerberos v1\_2 is the first version of the software that knows about both realms. Versions prior to v1\_2 only recognize PILOT.FNAL.GOV. A machine which runs Fermi kerberos v1\_2 (or later) and krb5conf v1\_0 (or later, preferably v1\_3 or later) can recognize principals and credentials in both realms, and may have either the pilot or the production realm as its default realm. If you must access some machines with PILOT.FNAL.GOV as the default realm and others with FNAL.GOV, it is easiest if all the machines run these recent product versions.

### D.2.3 Password Issues

Whatever your Kerberos password for the PILOT.FNAL.GOV realm was at 1:00 p.m. on Thursday, May 10, 2001, that became your Kerberos password in the FNAL.GOV realm . It will expire (or did expire!) on the same date as your pilot realm password. After May 10, password changes in one realm are not reflected in the other; you must change each separately.

---

1. To determine a machine's default realm, see section

If your password has only a short time left until expiration in the pilot realm, you might be required to change it before you have begun using it in the production realm. If so, log in to a pilot node which knows about both realms (i.e., which runs Fermi kerberos v1\_2 and krb5conf v1\_0, or later versions) then enter:

```
% kpasswd [yourprincipal]@FNAL.GOV
```

to change your password in the production realm.

## D.3 Accessing Systems in the Pilot and Production Realms

### D.3.1 Systems Running kerberos v1\_2 and krb5conf v1\_0 (or later versions)

For systems running kerberos v1\_2 and krb5conf v1\_0 (or later versions), you can freely access multiple machines without worrying about which realm each uses as a default, and which principal you're using (many users have a principal in each realm). You don't need a `$HOME/.k5login` file, but if you create one on a machine, include both your principals in it (see section ?? for information on the `.k5login` file).

You may authenticate as either *yourprincipal@FNAL.GOV* or *yourprincipal@PILOT.FNAL.GOV* and you will have normal access. From that system, you should have normal access to any other system which also recognizes both realms.



Note that if you have credentials existing in one realm, and you start a new session and run **kinit** (with no principal argument), you will get credentials in the same realm by default, regardless of the machine's default realm. If you want credentials for a different realm, do one of the following:

- provide a principal argument (including the realm part) with **kinit**
- run **kdestroy** before **kinit**

### D.3.2 Other Configurations

If the machine runs a version of krb5conf v1\_0 or greater and any version of kerberos, then you can **kinit** under either principal, if you have both. You can connect to such a machine from an FNAL.GOV-only host with your FNAL.GOV principal, if one of the following is true:

- the target host's `.k5login` file includes your FNAL.GOV principal
- the target host runs kerberos v1\_2 or later and krb5conf v1\_3 or later.

### D.3.3 CRYPTOCARD Logins

We remind you that your account name on each machine to which you want to login via CRYPTOCARD must match your principal name. CRYPTOCARDS issued before May 10, 2001 13:00 are configured differently than those issued after that date and time.

#### Issue date before 5/10/01 13:00

Assuming that your principal name in both realms is the same (it should be), you should be able to use your CRYPTOCARD to log into a machine with either realm as its default. It will authenticate you to the default realm of the machine.

Note that your CRYPTOCARD challenge sequence will be out of sync in the production realm with respect to the pilot realm, so you may have to manually enter a challenge into your card the first time you do a portal login to a production realm host. In fact, every time you alternate realms with your CRYPTOCARD, you may need to resynchronize it with the realm's KDC. Since you probably don't want to bother keying in a challenge to your CRYPTOCARD very often, it is best to pick only machines in one realm or the other for all your CRYPTOCARD logins, then use your Kerberos ticket to log in from there to any hosts you need to use in the other realm.

In the case of FTP, the realm with which your CRYPTOCARD is synchronized must match the one to which the target machine would map its own hostname (in the `[domain_realm]` section of `krb5.conf` or DNS). This is generally the same as the machine's default realm, but a sufficiently deranged configuration could make it different, and thus make FTP fail.

#### Issue date after 5/10/01 13:00

CRYPTOCARDS issued after this date are valid for login only to a machine in the realm for which the CRYPTOCARD was initialized. Pick only machines in the proper realm for all your CRYPTOCARD logins, then use your Kerberos ticket to log in from there to any hosts you need to use which are in the other realm.

### D.3.4 A Tricky Issue - Expired Credentials

Say your Kerberos tickets expire, and you want to type `kinit` to get new ones. Be aware that you still have a credential cache (with expired credentials) from your previous authentication (unless your `/tmp` has been cleared, you've removed the credential file manually, or you've run `kdestroy`). `kinit` will assume you want to authenticate as the same principal that credential cache belongs to unless (a) you list a different principal on the `kinit` command line, or (b) you issue the `kdestroy` command before

running **kinit**. In the case of (b), the machine's default realm is taken as the realm in which to authenticate, unless you specify your principal in the other realm on the command line.

## D.4 WRQ® Reflection Issues

### D.4.1 Add FNAL.GOV to Configuration

A registry update file has been created to set up the definitions for the FNAL.GOV realm for the **WRQ®** Kerberos Manager v7.0. It will add the realm definition for FNAL.GOV, update the admin servers for both FNAL.GOV and PILOT.FNAL.GOV, add the new KDCs for FNAL.GOV and PILOT.FNAL.GOV, and change the default realm for the machine to FNAL.GOV.

To apply the update, go to **NETWORK NEIGHBORHOOD**, find `Pckits`, and execute the registry export file:

```
\\pckits\WRQ\FNAL.GOV.reg
```

After applying the registry update, open the **Kerberos Manager** and, under the **CREDENTIALS** menu, create a **NEW PRINCIPAL PROFILE...** for yourself in the production FNAL.GOV realm. Note that you can keep principal profiles defined in both realms.

Please note that this update is NOT valid for the v8.0 Kerberos Manager.

If you don't have access to `\\Pckits`, you can perform the configuration by hand. Follow the instructions in section 19.4 *Configuring WRQ® Reflection Kerberos Manager v9.0.0* under step (2).

### D.4.2 Change Password Before Logging In via WRQ®

This note applies only to principals that were initially created in the PILOT.FNAL.GOV realm<sup>1</sup>. Your FNAL.GOV realm password won't work with the current **WRQ®** software until you change the password at least one time in that realm after May 10, '01. If you changed your initial FNAL.GOV password before that date, CHANGE IT AGAIN! You'll probably have to log in on a UNIX system to change the password. After this, you can change your password again from **WRQ®** if you like.

## D.5 Finding out which Hosts have Migrated

The list of machines in the pilot realm was inserted into DNS (domain name server) as explicit records on May 10, 2001 and those that change their default realm to FNAL.GOV are being deleted from the list as their administrators

---

1. This may also apply to Windows 95 users whose principals are in FNAL.GOV only.

report in. If your client software doesn't check in DNS for host-to-realm information but you need to know, look for a `txt` record belonging to `_kerberos.<hostname>`. For example, run `dig` or `nslookup`:

## dig

```
% dig _kerberos.cdfsga.fnal.gov txt
```

This has several lines of output, one of which gives the realm:

```
;; ANSWER SECTION:
_kerberos.cdfsga.fnal.gov.      21600    TXT      PILOT.FNAL.GOV
```

Hosts without their own explicit listing default to the listing for the parent DNS domain, e.g.,:

```
% dig _kerberos.fnal.gov txt
```

```
...
;; ANSWER SECTION:
_kerberos.fnal.gov.           21600    TXT      FNAL.GOV
```

## nslookup

```
% nslookup -q=txt _kerberos.<host>.fnal.gov. | grep kerberos
```

If the host is part of the PILOT realm, you should see the response:

```
_kerberos.<host>.fnal.gov text = "PILOT.FNAL.GOV"
```

## Appendix E. Transition from Pilot to FNAL.GOV Realm (Sysadmin Information)

The production realm FNAL.GOV was launched on May 10, 2001. We are now transitioning to this realm from PILOT.FNAL.GOV. We expect the transition phase to end before October 31, 2001. In this appendix, we describe the things the administrator of a Kerberized system needs to know and do in order to complete the migration of the system from PILOT.FNAL.GOV to FNAL.GOV.



The information in this chapter currently assumes that UNIX/Linux machines have the Fermi Kerberos product installed and Windows systems run WRQ® Reflection software.

### E.1 Migrating your UNIX/Linux System's Configuration to FNAL.GOV using UPD

As of June 5, Kerberos v1\_3a is current in fnkits, as is krb5conf v1\_4. Here are the steps for migrating to the production realm FNAL.GOV from the pilot realm. Use the latest version of Fermi kerberos available.

#### E.1.1 Quick Overview of Steps

First, to make your host aware of both realms and accessible by clients in either realm, issue the following commands on your system (using latest version of Fermi kerberos):

- 1) `setup upd`
- 2) `upd install kerberos v1_x -G "-c"`
- 3) `ksu root`<sup>1</sup>
- 4) `ups install kerberos v1_x`
- 5) `ups add-new-realm kerberos`

---

1. In the root directory there must be a `.k5login` file with your principal listed for this to work.

The actions taken by **ups add-new-realm kerberos** are listed in section E.1.3 *The ups add-new-realm kerberos Action*. Then when you're ready, change your default realm over to FNAL.GOV. To do so, issue the command:

1) **ups change-realm kerberos**

2) and send *nightwatch@fnal.gov* the full name of the host on which you've changed the default realm to FNAL.GOV.



For machines on which Fermi kerberos is installed, but without host and FTP service keys, and hence no keytab file, this “change-realm” command will appear to fail when it can't find `/etc/krb5.keytab`. Just ignore the “ABORT: keytab-convert failed” error, since the job is already done for you at that point. Or, you can just edit the `krb5.conf` file manually instead of running this command.

## E.1.2 The Steps with More Detailed Information

- 1) Install the latest Fermi kerberos that is “current” in KITS. This will bring along the new `krb5conf` product.
- 2) After installation is complete, run the separate action **ups add-new-realm kerberos**. This will make your host aware of both realms and accessible by clients in either realm. Your system's default realm IS NOT CHANGED by this action. Details on what is changed are in section E.1.3 *The ups add-new-realm kerberos Action*.
- 3) You may want to run in this configuration for a while. You should be able to authenticate as *yourprincipal@FNAL.GOV* or as *yourprincipal@PILOT.FNAL.GOV* and have normal access to any other system on which steps 1 and 2 have also been performed.
- 4) When you're ready to migrate your default realm to FNAL.GOV, execute **ups change-realm kerberos**. All this does is to change the default realm of the system to FNAL.GOV in `/etc/krb5.conf`, and to make sure the AFS servers are noted in that file as belonging to the same realm<sup>1</sup>.
- 5) Send an email to *nightwatch@fnal.gov* listing the host(s) for which you've changed the default realm to FNAL.GOV.

---

1. Actually, the AFS servers aren't using Kerberos V5 at all yet, but for the Kerberos-to-AFS ticket/token translation, your host has to think it's in the same realm as the AFS servers.

### E.1.3 The ups add-new-realm kerberos Action

The following describes what is affected by the **ups add-new-realm kerberos** action. Note that as of kerberos version v1\_3a this action is also available as two separate actions: migrate-keytabs and migrate-k5login.

- 1) The status of the krb5conf product is double-checked.
- 2) The new keytab-convert program is run on `/etc/krb5.keytab` and any cron keytabs found in `/var/adm/krb5`. If any key from the new realm is already present in the file, the program exits. Otherwise, the program duplicates every key in the old realm in the new realm. Before and after output of **klist -k**, in order, look like this<sup>1</sup>:

```
KVNO Principal
```

```
-----  
3 ftp/gungnir.fnal.gov@PILOT.FNAL.GOV  
7 host/gungnir.fnal.gov@PILOT.FNAL.GOV
```

and

```
KVNO Principal
```

```
-----  
3 ftp/gungnir.fnal.gov@PILOT.FNAL.GOV  
7 host/gungnir.fnal.gov@PILOT.FNAL.GOV  
3 ftp/gungnir.fnal.gov@FNAL.GOV  
7 host/gungnir.fnal.gov@FNAL.GOV
```

- 3) For every home directory found in `/etc/passwd` or the NIS `passwd` map, if there is already a `.k5login` present, it is updated analogously to the keytab files. If the production realm (FNAL.GOV) is already mentioned, nothing happens. Otherwise, for every pilot realm principal mentioned in the file, the corresponding production realm principal is added. If the file is not readable and writable by the user in whose home directory it exists (perhaps because of AFS protection), it is skipped.

A couple of notes:

For host and FTP principals originally created in the pilot realm before May 10, this action puts the right FNAL.GOV realm key into the keytab file.

For projects (discussed in section ) you have to run the keytab-convert program separately on the file defined as the project's keytab.

---

1. The KVNO (key version number) changes every time the key for a principal is changed. You may want to change a key to protect data. It is useful to know that, if a key is changed on a server while users are holding tickets issued on the old KVNO, the server can pick the right key to verify them. You can wait till all the current tickets expire, say 26 hours, then purge the old keys. A program to change keys may be available soon.

## E.2 Migrating your UNIX/Linux System's Configuration to FNAL.GOV (no UPD)

### E.2.1 Fermi Kerberos

Without UPS/UPD, but with Fermi Kerberos:

```
for i in /etc/krb5.conf /var/adm/krb5/?????????*; do
    /usr/krb5/bin/keytab-convert -v -o PILOT.FNAL.GOV -n FNAL.GOV $i
done
```

Then edit the `/etc/krb5.conf` `default_realm` under `[libdefaults]` and `[domain_realm]` info, as shown in Chapter 17: *The Kerberos Configuration File: krb5.conf*.

### E.2.2 Non-Fermi Kerberos

With non-Fermi Kerberos and no UPD, request new passwords from `compdiv@fnal.gov` for host and ftp principals in FNAL.GOV realm only. Then run the following:

(`fqdn` means your host's fully qualified domain name, e.g., `myhost@fnal.gov`, or `myhost@myuniv.edu`)

```
for i in host ftp; do
    /usr/krb5/sbin/kadmin -r FNAL.GOV -p $i/$fqdn -q "ktadd $i/$fqdn"
    : (Supply password when requested)
done
```

Then edit the `/etc/krb5.conf` `default_realm` under `[libdefaults]` and `[domain_realm]` info, as shown in Chapter 17: *The Kerberos Configuration File: krb5.conf*.

## E.3 Migrating your WRQ® (for Windows) Configuration to FNAL.GOV

Navigate to **START > PROGRAMS > REFLECTION > UTILITIES > KERBEROS MANAGER** to open the **Reflection Kerberos Manager** application. Pull down the **CONFIGURATION > CONFIGURE REALMS...** menu, make sure the *Configuration* tab is selected.

- 1) Press **ADD**, and in the pop-up window, type `FNAL.GOV` for Realm, and `krb-fnal-1.fnal.gov` for KDC host.
- 2) Highlight the `FNAL.GOV` realm and click **PROPERTIES**.
- 3) With the *KDC* tab selected, press **ADD**, and in the pop-up window, type in the KDC host `krb-fnal-2.fnal.gov`. Click **OK** to add it to the *KDC list*. Repeat for the remaining FNAL.GOV realm KDCs (listed in section E.5 *KDC List*).



- 4) In the **KADMIN SERVER** box, replace the default value with `krb-fnal-admin.fnal.gov`.
- 5) Click the *Realm Defaults* tab and change the **PRE-AUTHENTICATION** from None to Encrypted timestamp. Click **OK**.
- 6) Set the default ticket lifetime to some non-zero value (26 hours is good).
- 7) Ignore the *Hosts* tab.
- 8) Now select the *Machine Defaults* tab and change the **DEFAULT TICKET LIFETIME** to 26 hours. Click **OK**.
- 9) Back on the **REFLECTION KERBEROS MANAGER** window, choose **CREDENTIALS**, then **NEW PRINCIPAL PROFILE...**
- 10) On the **ENTER PRINCIPAL** window, verify your principal name, and choose the realm `FNAL.GOV`. Click **OK**.
- 11) On the **CREATE NEW PRINCIPAL PROFILE** window, keep the default location for the credentials storage, and click **CREATE**.
- 12) Back on the **REFLECTION KERBEROS MANAGER** window, choose the tab with your new principal profile and check that your full `FNAL.GOV` principal is filled in.

## E.4 Kerberized Client-Server Issues during Transition

### E.4.1 How does a Client determine the Realm?

When a user invokes a Kerberized client program, the client program (e.g., Kerberized telnet, rsh, etc.) must decide which realm to ask for the service ticket. It bases this decision on (1) the hostname of the server for the client program, and (2) the rules which are now found in either or both of two places: the `[domain_realm]` sections of `/etc/krb5.conf` and TXT (text) records in DNS (the nameservers). The order in which these sources are tried for finding the realm of `somebox.fnal.gov` is<sup>1</sup>:

- 1) `somebox.fnal.gov = <REALM>` (in `krb5.conf`)
- 2) `_kerberos.somebox.fnal.gov txt <REALM>` (in DNS)
- 3) `.fnal.gov = <REALM>` (in `krb5.conf`)
- 4) `_kerberos.fnal.gov txt <REALM>` (in DNS)

---

1. This determines the realm of the `host/somebox.fnal.gov@<REALM>` ticket that the client requests.



Hosts which had service principals created before the realm migration began on May 10, 2001 have records matching number 2 in DNS, giving their realm as PILOT.FNAL.GOV, unless an email was sent to *nightwatch@fnal.gov* saying that the host's default realm has been changed. (See section D.5 *Finding out which Hosts have Migrated* for information on how to find the default realm of a given host.)

## E.4.2 What if the Service Exists in Both Realms?

If the service actually exists in both realms, then for the case of telnet and the r-commands it doesn't matter which realm the client believes the server to be in, the server will have the necessary key to check the presented credentials. FTP, unfortunately, is written to a different API which hides the details of Kerberos from the application and makes it quite difficult for the application to accept credentials which treat the service as being in any realm other than the one to which the server host would map itself according to the rules above. If client and server are both relying on DNS for realm information, this will very rarely be a problem.

## E.4.3 Migrated Client Hosts, Non-migrated Server Hosts

You may be running clients on your (migrated) system that commonly use servers on hosts that have not yet migrated to the production realm. This should present no problems for most clients. The telnet and r-command services will respond properly to clients who try to treat them as members of either realm, as long as the new kerberos and krb5conf products have been installed on the services' hosts.

However, an FTP server will accept Kerberos authentication only if the FTP client presents a service ticket from the service host's default realm<sup>1</sup>. To help out with that, the Kerberos code will now look for a DNS text record (or information in the `krb5.conf` file) for host-to-realm mapping information. This is discussed in section E.4.1 *How does a Client determine the Realm?*. Whenever *nightwatch@fnal.gov* is notified of a new host that's been migrated to FNAL.GOV, they will remove the explicit mapping of the host to the PILOT.FNAL.GOV realm from DNS. By this action, Kerberos is forced to look at the `.fnal.gov = <REALM>` line in the host's `krb5.conf` (or `_kerberos.fnal.gov txt REALM` in DNS) to determine the realm mapping, which will point to FNAL.GOV for this situation.

## E.4.4 Resolving Host Name when Requesting a Service Ticket

A service name includes the full, official name of the server host (e.g., `service/hostname.fnal.gov@REALM`). Aliases or nicknames of the server host are resolved before requesting the service ticket. But if there is a hosts file

---

1. The underlying requirement is that the server map its own name to the same realm to which the client maps the server's name.

or NIS map on the client machine that takes precedence over DNS, you might come up with the wrong name. Common errors include listing the “short” or one-word name or a nickname before the full name in a hosts file. This causes a request for a ticket for a service principal that usually does not exist in the Kerberos database.

## **E.5 KDC List**

As of July 18, 2001, the current KDC nodes are listed below. For WRQ, note that the :88 should be left off.

### **For PILOT.FNAL.GOV**

(Note the absence of “2”.)

krb-pilot-1.fnal.gov:88

krb-pilot-3.fnal.gov:88

krb-pilot-4.fnal.gov:88

krb-pilot-admin.fnal.gov (the master KDC for PILOT.FNAL.GOV)

### **For FNAL.GOV:**

krb-fnal-1.fnal.gov:88

krb-fnal-2.fnal.gov:88

krb-fnal-3.fnal.gov:88

krb-fnal-4.fnal.gov:88

krb-fnal-5.fnal.gov:88

krb-fnal-6.fnal.gov:88

krb-fnal-admin.fnal.gov (the master KDC for FNAL.GOV)

