

# **Kerberized NFSV4 Setup Tutorial**

Aime Le Rouzic  
([aime.le-rouzic@bull.net](mailto:aime.le-rouzic@bull.net))

## History

Version 1.0 created on june 26, 2007

Version 1.0 last updated on june 28,2007

Version 2.0 created on september 30, 2007

# Introduction

The goal of this tutorial is to gather all what need NFSV4 administrators and end-users to more easily configure and install NFSv4 within a secure environment. This document has been built from my own experience in testing the installation and robustness of NFSV4 with Kerberos and also from experiences of others who have published it on the Web. The list is at the end of this document.

It is certainly not complete. Let me know at [aimelero@bull.net](mailto:aimelero@bull.net) what is missing and also some real-life examples of Kerberos NFSV4 deployments.

In chapter 1, we present three commands to facilitate and controlling the setup of a linux kerberized NFS environment to allow the reader to use quickly NFS with Kerberos.

In chapter 2, we do a quick review of how NFSV4 and Kerberos work together in order to understand the different steps of the deployment and the impact on the configuration.

In chapter 3, we continue by presenting all the commands, files and daemons servers used to manage and monitor NFSV4 and Kerberos.

In chapter 4, we detail the methods, scripts and tools which make the setup and the control of the client and server configurations step by step by:

In chapter 5, we describe more the simplified and automatic setup listed in Chapter 1.

In chapter 6, we analyse and explain the common errors encountered by the administrators.

- How these appear in the logs files.
- How to detect them.
- Which recovery actions are necessary following troubleshootings.

In chapter 7, we provide a list of different kinds of distributions and operating systems offering a Kerberized NFSV4 environment.

In chapter 8, we consider the current limitations of Kerberized NFSV4 infrastructures.

In chapter 9, we look at future directions.

In chapter 10, we list the Frequently Asked Questions.

Finally in chapter 11, we give good documents related to the topic with the address where to find them.

# 1) Easy Setup

Here are three commands to facilitate and controlling the setup of a linux kerberized NFS environment.

- **krbkdcsv** : setup a Kerberos KDC and a Kerberos administration Server
- **krbnfssv** : setup a Kerberos NFS Server
- **krbnfsc1** : setup a Kerberos NFS Client

Those commands do the setup and also some controls about frequent kerberos and nfs errors happening during a kerberos nfs configuration:

- check client and server hosts are fully qualified name
- check REALM is UPPER CASE
- synchronize time with ntp when possible
- check time is synchronised (<300s) with the KDC Server machine
- check the /etc/hosts file lists the fully-qualified domain name as the first entry on the line with the machine's IP address,
- check in /etc/resolv.conf the name server is the same as in /etc/resolv.conf of the KDC Server
- check the /etc/services file lists the nfs service (port 2049) is fully qualified name
- check KDC, Kerberos Server and nfs Server are reachable
- check kerberos daemons are running (krb5kdc, kadmind) on the Kerberos Server
- check nfs server daemons are running on the NFS Server
- check rpc.gssd and rpc.svcgssd are running

You can find them at <http://nfsv4.bullopensource.org/>

## 1.1) To do a mount

First, **krbkdcsv start** has to be run on the machine choiced to be the Kerberos KDC and a Kerberos administration Server.

After, **krbnfssv start** is run on the machine choiced to be the Kerberos NFS Server.

Then, **krbnfsc1 start** is run on the machine to the Kerberos NFS Client.

Now the **kerberized nfs mount** can be done.

## 1.2) When rebooting or When a mount has failed

First, try **krbnfsc1 status** on the NFS client to check possible errors

After run **krbnfssv status** on the NFS Server to check other possible errors

Then **krbkdcsv status** on Kerberos Server to check other possible error

After that if the mount still fails try:

First run another **krbnfsc1 start** and try the **mount**

Next run another **krbnfssv start** and **krbnfsc1 start** and try the **mount**

## 1.3) Syntax

Default values are proposed from /etc/krb5.conf and from the configuration of the machine the command is running on. Nevertheless you change them by options in the command.

### 1.3.1) krbkdcsv usage

```
krbkdcsv start { -h | {-a kerberos administrator principal} {-b <kerberos administration server>} {-c <krb5.conf file directory>} {-C <kdc.conf file directory>} {-d <domain>} {-k <kdc server>} {-r <realm>} {-v}}
```

```
krbkdcsv status { -h | {-a admin name} {-c <krb5.conf file directory>} {-v}}
```

### 1.3.2) krbnfssv usage

```
krbnfssv start { -h | {-b <kerberos administration server>} {-c <krb5.conf file directory>} {-d <domain>} {-k <kdc server>} {-n <ntp server>} {-r <realm>} {-v}}
```

```
krbnfssv status { -h | {-b <kerberos administration server>} {-c <krb5.conf file directory>} {-k <kdc server>} {-n <ntp server>} {-v}}
```

### 1.3.3) krbnfscf usage

```
krbnfscf start { -h | {-b <kerberos administration server>} {-c <krb5.conf file directory>} {-d <domain>} {-k <kdc server>} {-n <ntp server>} {-r <realm>} {-s <nfs server>} {-u user name} {-v}}
```

```
krbnfscf status { -h | {-b <kerberos administration server>} {-c <krb5.conf file directory>} {-k <kdc server>} {-n <ntp server>} {-r <realm>} {-s <nfs server>} {-v}}
```

### 1.3.4) options

- b : kerberos administration server name
- c : directory where is located the krb5.conf file
- C : directory where is located the kdc.conf file
- d : domain name for the Kerberos realm
- h : help to display the command syntax
- k : KDC server name
- n : NTP server name
- r : realm for which the Kerberos server is to be configured
- s : nfs server name
- u : user name
- v : verbose mode

## 2 NFSV4 and Security

Let 's describe more what the above commands are doing and why they are doing it..

### 2.1) What is a Secure NFSV4

Goal is to have NFSV4 client and server using Kerberos V5 security with support for strong:

- Authentication

Kerberos authentication provides a mechanism for mutual authentication between a client and a server on an open network, and in which packets transmitted along the network can be monitored and modified at will. In order to provide secure authentication, Kerberos authentication uses symmetric keys, encrypted objects, and Kerberos services.

- Integrity

Integrity ensures that the data they send is valid and has not been tampered with during transit. Integrity is done through cryptographic checksumming of the data. Integrity also includes user authentication.

- Privacy

Privacy takes security a step further. Privacy not only includes verifying the integrity of transmitted data, but it encrypts the data before transmission protecting it from eavesdroppers. Privacy authenticates users, as well.

### 2.2) NFSv4 Security Plus

NFS V4 has been redesigned with Internet deployment and security in mind.

- NFS V4 consolidates all of its operations into a single protocol with one tcp port(2049). Coupled with using TCP as a transport, NFSv4 traffic is now capable of easier traversal for exchanging data over the internet. This makes it easier to deploy network security around NFSv4 deployments.
- All implementations of NFSv4 must implement the RPCSEC\_GSS protocol to be considered compliant with RFC 3530. For NFS versions 2 and 3 it is optional.
- What makes NFSv4 fundamentally more secure than previous NFS versions, it is because NFSv4 is only one protocol allowing RPCSEC\_GSS security to be applied to every NFS v4 transaction.
- Though NFSv4 provides Username Mapping Improvements by sending usernames and group names instead numeric UID and GID numbers across the network. Using Kerberos is strongly recommended when one user has different UIDs on two UNIX systems.
- When using Kerberos authentication, even if a client system is attacked , not all the system is down. It will need efforts client by client for the attacker to destroy all of them.

## 2.3) Notion of security services

**Access control:** Methods used to restrict the sources from which an operation may be performed.

**Authentication:** The process of proving or verifying that users are whom they say they are.

**Authorization:** The process of determining if a user is allowed to perform an operation.

**Credentials:** Ticket plus session key.

There are several different credential types for *users* and *machines*:

### User credentials

When someone is logged in to an client machine as a regular user, requests for services include that person's user credentials.

### Machine credentials

When a user is logged in to an client machine as root user, request for services use the client workstation's credentials.

**Directory Services:** A directory service provides information about users, groups and hosts within a computing environment. Examples include NIS,LDAP and DNS. Directory services are also sometimes provided by local databases on each host, typically stored in files such as /etc/passwd,/etc/group, and /etc/hosts on UNIX systems. Directory Services can be used by Authentication services(Kerberos, RADIUS); a user may be required to be authenticated before accessing a directory service or an authentication service may need information from directory service to identifying a user.

## Encryption

Different types of encryption exist:

- Symmetric key encryption

The same key is used to encrypt and decrypt the data.

Symmetric-key encryption is essentially the same as a secret code that each of the two computers must know in order to decode the information.

- Asymmetric key encryption /Public Key Exchange

Asymmetric key encryption requires two different unrelated keys. The public key is used to encrypt the data and the private key is used to decrypt the data. The basic idea behind this type of encryption is that the public key is freely distributed to anyone who wants it, but the private key is kept secret. Anyone can encrypt a message using a person's public key, but only that person can decrypt it because only he possesses the private key. One benefit of asymmetric key encryption over symmetric key encryption is that the key that must be communicated between parties will not decrypt the message. It will only encrypt the message. Because only one of the keys can decrypt the message, each person involved in the exchange must have a private and public key. We can consider the example of two people communicating via letter. Bob and Alice both have a lock and a key. Bob sends Alice his unlocked lock but keeps the key. Alice writes a message, puts it in a box, and locks it with Bob's lock. She sends Bob the box and her own unlocked lock. Bob opens the box with his private key, writes a message, places it in the box, and locks it with Alice's lock. Alice can unlock the box with her own private key upon receipt of the box.

- One-way hash functions and MACs

A cryptographic hash function takes any amount of data and applies an

algorithm that transforms it into a fixed-size output value.

**GSSAPI:** Stands for Generic Security Services Application Programming Interface. It is a security framework used by Kerberos 5. GSSAPI provides the application vendor with a common API so that applications can work with any number of security systems.

**Identification:** The process of communicating or determining the user ID associated with an operation

**KDC:**Key Distribution Center

To solve the problem of key distribution, the Kerberos protocol, similar to its namesake in Greek mythology, uses three “heads” — a client, a server, and a trusted third party that mediates between the other two.

The trusted intermediary in the protocol is the Key Distribution Center (KDC).The KDC is a service that runs on a physically secure server. The KDC maintains a database with account information for all security principals in its realm. Along with other information about each security principal, the KDC stores a cryptographic key known only to the security principal and the KDC. This key is used in exchanges between the security principal and the KDC.

Linux implements the KDC as a single process that provides two services:

**Authentication service (AS).** The AS issues TGTs good for admission to the ticket-granting service in its domain. Before network clients can get tickets for services, each client must get an initial TGT from the AS in the user’s account domain.

**Ticket-granting service (TGS).** The TGS issues tickets good for admission to other services in the TGS’s domain or to the ticket-granting service of a trusted domain. When a client wants access to a service, it must contact the ticket-granting service in the service’s account domain, present a TGT, and ask for a service ticket.

AS and TGS can run on separate physical machines.

The KDC is often referred to as the Kerberos server .

Kerberos Keys(secret keys): A key is an encryption key that is shared by a principal and the KDC. The key is distributed outside the system with a long lifetime, as opposed to a session key which has a much shorter lifetime. In the case of a user principal, the key is derived from a password. A keytab file contains the key for a service principal.

Kerberos authentication relies on different types of keys:

- **User, service, and system keys.** Long-term symmetric keys generated from passwords
- **Public keys.** Long-term asymmetric keys
- **Session keys.** Short-term symmetric keys created by KDC

**LDAP:** Kerberos offers strong authentication for verifying a client's true identity; the Lightweight Directory Access Protocol (LDAP) offers the mechanism to look up authorized users in an efficient and secure manner.

**Principal:** Every network entity in a Kerberos installation, including computer, users and networks services has a principal associated with it. In Kerberos 5, there are two types of principals: user and service principals. The user principal takes the form <user>/<administrative instance>@REALM, where the administrative instance is needed for an administrator who wants to be either a regular user or a Kerberos administrator. For example, an administrator named john can have a regular principal [john@KRB NFS.NET](#) or an administrative principal [john/admin@KRB NFS.NET](#). Likewise, if john has accounts on two different hosts, he can use two principal names with different instances, for example, john/host1.krbnfs.net@KRB NFS.NET and

john/host2.krbnfs.net@KRB NFS.NET

The service principal takes the form <service>/<server>@REALM, where service can be any Kerberized network service and server is the machine on which the service is running. For example: [nfs/nfssv.krbnfs.net@KRB NFS.NET](#) is the nfs service principal for the nfs server with a fully qualified domain name nfssv.krbnfs.net. Kerberos principals are also referred to as Kerberos clients.

**Realm:**

The purpose of realms is to allow an organization to have local control of authentication of its users and applications. All the users and applications that use a Kerberos server compose a **realm**.

The Kerberos realm is usually the DNS domain name converted to uppercase. A realm is case-sensitive.

**RPCSEC\_GSS:**

Enables NFS services to use Kerberos authentication. RPCSEC\_GSS is a security flavor that provides security services that are independent of the mechanisms being used. RPCSEC\_GSS sits on top of the GSS-API layer. Any pluggable GSS\_API-based security mechanism can be used by applications that use RPCSEC\_GSS.

**Service:** networked resource the client is trying to access

**ST:** Service Ticket

Generated by the ticket granting server(TGS), clients use service tickets to authenticate with the application server before it can access services in the network. A service ticket includes session key, client principal, ticket lifetime and client IP address.

**TGT:** Ticket Granting Ticket (Authentication ticket)

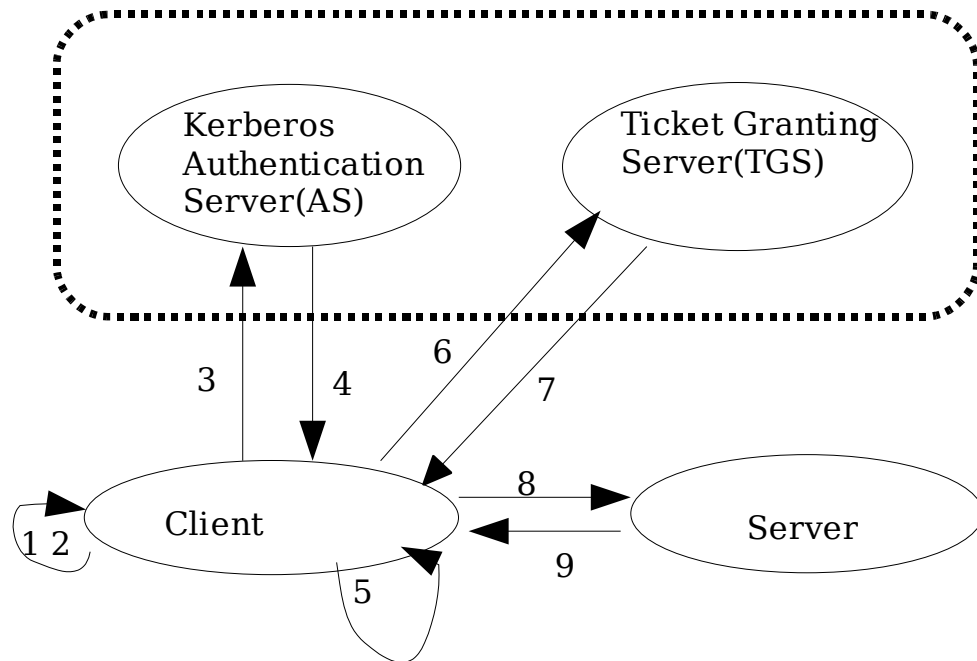
This ticket is used to get the ticket service to access a Kerberized service.

**Tickets:** The main component in Kerberos authentication is the ticket. Essentially, the goal of Kerberos messages is to request and deliver tickets. There are two types of tickets used in Kerberos authentication, ticket-granting tickets (TGTs) and service tickets

- A ticket can be forwardable: means the ticket can be used on another machine without a new authentication process.

## 2.4) How does Kerberos V5 authentication process work?

Linux implements the KDC as a single process that provides two services:



What follows is a simplified description of the protocol. The following shortcuts will be used: AS = Authentication Server, TGS = Ticket Granting Server, SS = Service Server.

In one sentence: the client authenticates itself to AS, then demonstrates to the TGS that it's authorized to receive a ticket for a service (and receives it), then demonstrates to the SS that it has been approved to receive the service.

In more details:

1. A user enters a username and password on the client.
2. The client performs a one-way hash on the entered password, and this becomes the secret key of the client.
3. The client sends a clear-text message to the AS requesting services on behalf of the user. Sample Message: "User XYZ would like to request services". Note: Neither the secret key nor the password is sent to the AS.
4. The AS checks to see if the client is in its database. If it is, the AS sends back the following two messages to the client:
  - \* Message A: Client/TGS session key encrypted using the secret key of the user.
  - \* Message B: Ticket-Granting Ticket (which includes the client ID, client network address, ticket validity period, and the client/TGS session key) encrypted using the secret key of the TGS.
5. Once the client receives messages A and B, it decrypts message A to obtain the client/TGS session key. This session key is used for further communications with TGS.

(Note: The client cannot decrypt the Message B, as it is encrypted using TGS's secret key.) At this point, the client has enough information to authenticate itself to the TGS.

6. When requesting services, the client sends the following two messages to the TGS:

- \* Message C: Composed of the Ticket-Granting Ticket from message B and the ID of the requested service.

- \* Message D: Authenticator (which is composed of the client ID and the timestamp), encrypted using the client/TGS session key.

7. Upon receiving messages C and D, the TGS decrypts message D (Authenticator) using the client/TGS session key and sends the following two messages to the client:

- \* Message E: Client-to-server ticket (which includes the client ID, client network address, validity period) encrypted using the service's secret key.

- \* Message F: Client/server session key encrypted with the client/TGS session key.

8. Upon receiving messages E and F from TGS, the client has enough information to authenticate itself to the SS. The client connects to the SS and sends the following two messages:

- \* Message G: the client-to-server ticket, encrypted using service's secret key.

- \* Message H: a new Authenticator, which includes the client ID, timestamp and is encrypted using client/server session key.

9. The server decrypts the ticket using its own secret key and sends the following message to the client to confirm its true identity and willingness to serve the client:

- \* Message I: the timestamp found in client's recent Authenticator plus 1, encrypted using the client/server session key.

10. The client decrypts the confirmation using its shared key with the server and checks whether the timestamp is correctly updated. If so, then the client can trust the server and can start issuing service requests to the server.

11. The server provides the requested services to the client.

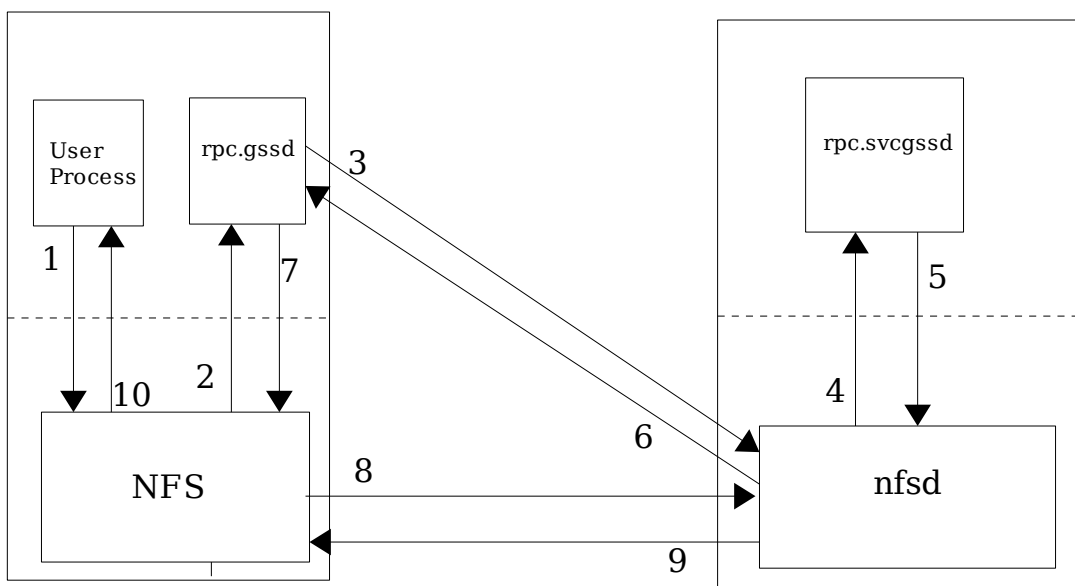
## 2.5) RPCSEC\_GSS

NFSV4 requires the RPCSEC\_GSS protocol for strong security.

RPCSEC\_GSS is based on the Generic Security Services API (GSS-API). GSS-API provides the ability to write applications that can authenticate clients and servers (RPC header is signed), integrity protect the messages they exchange (Header and Body are signed), and also privacy protect the messages they exchange (Header signed, Body Encrypted)

RPCSEC\_GSS leverages GSS-API capabilities to provide authentication, integrity, and privacy if the security mechanism provider supports those services. Currently Kerberos V5 is the mostly widely deployed GSS-API security provider, and it does support all three services. NFS over RPCSEC\_GSS can use Kerberos V5's capabilities.

See the following at <http://www.citi.umich.edu/projects/nfsv4/gssd/>



1) User process does initial operation requiring an NFS object which requires rpcsec\_gss authentication. (This can be any version of NFS -- 2, 3, or 4.)

2) Kernel code discovers it has no cached context for this user/server combination and does an upcall to obtain a security context. The upcall is handled by rpc.gssd.

The conduit between kernel and userspace is via a pipefs filesystem. The default mount location for this pipefs is `/var/lib/nfs/rpc_pipefs`. (This is referenced as `{pipefsdir}` below.) The kernel writes to:

**`{pipefsdir}/clntXX/info`**  
**`{pipefsdir}/clntXX/krb5`**  
**`{pipefsdir}/clntXX/spkm3`**

rpc.gssd monitors for changes (new files) in the `{pipefsdir}`.

rpc.gssd assumes that the user has already authenticated and has credentials available. It attempts to find credentials for the user (using only the UID) and calls `rpcsec_gss` routine `authgss_create()` to create an rpc context. `authgss_create()` calls `gss_init_sec_context()` to obtain a gss token.

3) rpc.gssd sends the gss token via a NULL rpc call to the server.

4) The server receives the NULL rpc call and does an upcall to rpc.svcgssd to handle it. The server side uses procs as a conduit between kernel and userspace

**`/proc/net/rpc/auth.rpcsec.context/channel`**  
**`/proc/net/rpc/auth.rpcsec.init/channel`**

rpc.svcgssd calls `gss_accept_sec_context()` to obtain a gss token and complete the gss context negotiation. The server now has a completed context which is ...

5) returned to the kernel (via the `auth.rpcsec.context` channel) as well as the response to the NULL rpc (via the `auth.rpcsec_init` channel).

6) The NULL rpc response (containing the gss token) is returned to the client which completes the context negotiation.

7) The gss context information is written to the kernel.

8) The original operation can now be performed using the security context cached by the kernel.

9) The response to the original operation.

10) Results of the original operation are returned to the user process.

## **2.6 How Kerberos Works with NFS**

When Kerberos is specified as the security style on an NFS-exported filesystem, clients are required to present valid Kerberos credentials(that is, a service ticket issued by the KDC) in order to access the file system.

While Kerberos provides authentication of users, it doesn't handle authorization. Authorization, for NFS, means determining if a user is allowed to perform a particular operation on a file or directory. While Kerberos tickets provide the NFS server with assurance that users are whom they say they are, it is still up to the NFS server to determine which operations should be allowed. This determination is made by consulting the ownership, permissions, and ACLs on the file or directory and comparing against the identity and group memberships of the requesting user.

Only the identity of the user is authenticated by the Kerberos mechanisms; the group memberships(and possibly username to UID number mapping) are obtained from local files or a directory service, while the file permissions and ACLs are stored in the file system.

## 3 Commands, files and daemons

This section to describe the Files, daemons and commands used to get a Kerberos NFS environment. You can use:

« man file/daemon/command » ex: man exports

to get more information

### 3.1 Files

#### 3.1.1 NFSV4

##### **/etc/exports**

In **/etc/exports** make an entry of your exported path with the export options for eg:-  
**/etc/exports** - contains a list of all directories that are to be exported via NFS.

The syntax is different from NFSv3:

Old syntax: supported before and after nfs-utils.1.1.0

```
/nfs *(rw,fsid=0,insecure,no_subtree_check,sync,no_root_squash)
/nfs gss/krb5(rw,fsid=0,insecure,no_subtree_check,sync,no_root_squash)
/nfs gss/krb5i(rw,fsid=0,insecure,no_subtree_check,sync,no_root_squash)
/nfs gss/krb5p(rw,fsid=0,insecure,no_subtree_check,sync,no_root_squash)
```

New syntax: supported only from nfs-utils.1.1.0

```
/exports *(sec=sys:krb5:krb5i,rw)
or, if you want to restrict access based both on client and on security flavor:
/exports 192.168.0.0/16(sec=sys:krb5:krb5i,rw)
```

##### **/etc/fstab**

mount NFSv4 exported volume

The NFS exported volume can also be mounted on the client just by making an entry in the **/etc/fstab** file. If your NFS server name is NFSserver and the mount point on the client is /mnt point then the entry in the **fstab** should look like something below.

The following entry is made in **/etc/fstab**

```
NFSserver:/ /mnt point nfs4 rw,user,noauto 0 0
```

##### **/etc/idmapd.conf**

configuration file for idmapd, the NFSv4 ID <-> Name Mapper

##### **/proc/net/rpc/nfsd, /proc/net/rpc/nfs, /proc/mounts**

Files to manage nfsv4

##### **/etc/resolv.conf**

DNS resolver configuration file.

#### 3.1.2 GSS

##### **/var/lib/nfs/rpc\_pipefs**

Used by rpc.gssd and rpc.svcgssd daemons (to get upcalls from the kernel)

##### **/etc/gssapi\_mech.conf**

This configuration file determines which GSS-API mechanisms the gssd code should

use. Usually no need to modify this file in 32 bit machines because the libraries are installed in /usr/lib.

Note:

- 1) In case of 64 bit machines this has to be modified to /usr/lib64.
- 2) Leaving the absolute path altogether and specifying just the library name might also work.

### **/proc/net/rpc/auth.rpcsec.init/channel**

### **/proc/net/rpc/auth.rpcsec.context/channel**

Used by the client and server side when using procs as a conduit between kernel and userspace for rpc.gssd and rpc.svcgssd

## **3.1.3 Kerberos**

### **/etc**

Directory to contain krb5.conf.

Somes implementations rather uses /etc/krb5 by default.

### **/etc/krb5.conf**

In this file, you must specify your realm, KDC's, administrative server, logging, default domain, and KDC information.

Every machine in Kerberos environment must have this config, it tells what is default realm for Kerberos libraries, and for each realm it tells where is KDC (master and its slaves) and administration server (admin server typically is on the same machine as master KDC). Also it tells how to map hosts and domains (domain names are precede with a dot ".") to realms.

### **/var/kerberos/krb5kdc**

This directory gathers all the KDC files(databases,keytab, acl..). The choice and the localisation of this directory is done in **krb5.conf**.

### **/var/kerberos/krb5kdc/kdc.conf**

That's KDC configuration (pretty self-explanatory), which defines realms to be served by this KDC, and for each realm defines where the principal's database is.

### **/var/kerberos/krb5kdc/principal\***

That is the **Kerberos database**.

The Kerberos database consists of principals and policies.

### **/var/kerberos/krb5kdc/kadm5.acl**

This file is used by kadmind to determine which principals have administrative access to the Kerberos database and their level of access

## **Key table**

The key table is used to store encryption keys. Normally, server applications provide the encryption keys that the Kerberos protocol uses when it needs to decrypt a request. Each key has an associated version number, and each time the key changes, the version is incremented. When the Kerberos protocol server encrypts a service ticket, it uses the latest encryption key stored in the **key table** and records the key version number in the ticket. Then, when the ticket is presented to the server, the key version number is used to retrieve the proper key from the key table. This allows the Server to change its key without invalidating existing tickets.

## **/etc/krb5.keytab**

Every host that provides a service must have a local file, called a keytab (short for “key table”). The keytab contains a serviced key for the appropriate service. A service key is used by a service to authenticate itself to the KDC and is known only by Kerberos and the service itself. For example, if you have a Kerberized NFS server, that server must have a keytab file that contains a service key for its nfs service principal.

To add a service key to a **keytab** file, you use the **ktadd** command of **kadmin**.

On application servers that provide Kerberized services, the keytab file is located at `/etc/krb5.keytab`, by default.

Nevertheless, be careful, following the operating system default are `/etc/krb5/krb5.keytab` or `/etc/krb5.keytab` for the kerberos commands.

## **/var/kerberos/krb5kdc/kadm5.keytab**

On the master KDC, the keytab file for keytab file for the kadmin/admin principal is located by the option **admin\_keytab** in **kdc.conf** file. It can be `/var/kerberos/krb5kdc/kadm5.keytab`

## **/tmp**

### **Credentials cache**

The credentials cache holds Kerberos protocol credentials (tickets, session keys, and other identifying information) in semi-permanent storage. The Kerberos protocol reads credentials from the cache as they are needed and stores new credentials in the cache as they are obtained. This relieves the application of the responsibility for managing the credentials itself.

- `/tmp/krb5cc_machine_REALM`  
Cache created by `rpc.gssd`. When I start `rpc.gssd` it creates its own `/tmp/krb5cc_machine` and doesn't use `/tmp/krb5cc_0`.
- `/tmp/krb5cc_{uid}`  
Default credentials cache for user `{uid}`.

Since `nfs-utils-1.1.0`, `gssd` has an option `< -n >` which says not to use machine credentials. In that case, `mount` will use whatever creds the root user has available.

### **/var/tmp/krb5kdc\_rcache: Replay cache**

The replay cache is used to detect duplicate requests. When the Kerberos protocol processes a request, it makes an entry in the replay cache. If it processes a later request that matches an entry already in the replay cache, an error will be returned to the application program. The replay cache is periodically purged to remove requests with expired lifetimes.

### **/var/tmp/kadmin\_0**

Created by **kadmin**

### **/var/kerberos/log/\***

Directory to put `kadmin` and `krb5kdc` daemons loggings.

The location is specified in `krb5.conf`

### **kadmin.log**

`kadmin` loggings

### **krb5kdc.log**

`krb5kdc` loggings

## 3.2 Daemons

### 3.2.1 NFSV4

#### **rpc.nfsd**

The `rpc.nfsd` program implements the user level part of the NFS service. The main functionality is handled by the `nfsd.o` kernel module; the user space program merely starts the specified number of kernel threads.

#### **rpc.idmapd**

`rpc.idmapd` is the NFSv4 ID <-> name mapping daemon. It provides functionality to the NFSv4 kernel client and server, to which it communicates via upcalls, by translating user and group IDs to names, and vice versa. It runs an client and server.

#### **rpc.mountd**

This process receives mount requests from NFS clients and verifies the requested file system is currently exported.

### 3.2.2 RCPSEC\_GSS

User level daemons used to handle context initiation phase

#### **rpc.gssd**

Client daemon that handles security contexts. This process provides the client transport mechanism for the authentication process (Kerberos Version 5) with NFSv4. This service is required for use with NFSv4.

#### **rpc.svcgssd**

Server daemon that handles security contexts. This process provides the server transport mechanism for the authentication process (Kerberos Version 5) with NFSv4.

### 3.2.3 Kerberos

#### **kadmind, krb5kdc**

The KDC runs two important Kerberos daemons. These daemons are `kadmind` and `krb5kdc`. These two daemons are run as root in user space.

`kadmind` is the administrative daemon for the Kerberos server. `kadmind` is used by a program named `kadmin` to maintain the database of principals and policy configuration.

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

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

## 3.3 Commands

### 3.3.1 NFSV4

#### Exportfs

The `exportfs` command is used to maintain the current table of exported file systems for NFS.

#### Mount/Umount

All files accessible in a Unix system are arranged in one big tree, the file hierarchy, rooted at `/`. These files can be spread out over several devices. The `mount` command serves to attach the file system found on some device to the big file tree. Conversely, the `umount(8)` command will detach it again

```
#mount -t nfs4 <servername>:/ <mntpath>
```

### 3.3.2 Kerberos

#### `kadmin.local`, `kadmin`

`kadmin` and `kadmin.local` are command-line interfaces to the Kerberos V5 KADM5 administration system. Both `kadmin` and `kadmin.local` provide identical functionalities; the difference is that `kadmin.local` runs on the master KDC and does not use Kerberos to authenticate to the database. `kadmin` provides for the maintenance of Kerberos principals, KADM5 policies, and service key tables (keytabs).

#### `Kadmin ktadd`

Add a Kerberos Service Key to a Keytab File

#### `kadmin ktremove`

Remove a Service key From a Keytab File

#### `Kdestroy`

`Kdestroy` clears ticket(credentials) cache

#### `Kinit`

`kinit` allows you to identify yourself to the Kerberos Server and obtain a TGT ticket. You must have a TGT ticket to use kerberized applications and will receive error messages when attempting to run applications without one.

The Kerberos `kinit` program forwards a request for a TGT to the KDC. The KDC then encrypts the TGT with your password and sends the encrypted TGT back to you. At your local client you type your password, and Kerberos decrypts the TGT and keeps it until the expiration time.

#### `Klist`

`Klist` lists the Kerberos principal and Kerberos tickets held in a credentials cache, or the keysheld in a keytab file.

#### `ktutil`

Another command that you can use to administer keytab files is the `ktutil` command. This interactive command enables you to manage a local host's keytab file without having Kerberos administration privileges, because `ktutil` doesn't interact with the Kerberos database as `kadmin` does.

So, after a principal is added to a keytab file, you can use `ktutil` to view the keylist in a keytab file or to temporarily disable authentication for a service.

With `ktutil` you can Temporarily Disable Authentication for a Service on a Host.

## 4 Step by step

On your network you will need to identify and set up three machines:

- the kerberos administration and KDC server
- the NFS server
- the NFS client

In this chapter we list the rules and the steps you need to respect to configure those machines with kerberos and to generate the principals to exploit the Kerberos services.

### 4.1 Network Environment and Rules

#### 4.1.1 TCP/IP Network Connectivity

Be careful, a user who can log on with cached credentials might not be aware of a connectivity issue.

- TCP and UDP Ports Required for Correct Operation of the Kerberos Protocol

- ◆ Kerberos ticket-granting service

88/TCP 88/UDP

All clients need to be able to connect to this port on the KDC servers.

- DNS Service

53/TCP 53/UDP

The internal DNS server needs to be accessible to all clients for the location of KDC computers

- NTP Time Service

123/TCP 123/UDP

All clients need to be able to connect to this port for time synchronization

- **Firewalls.** If you use a firewall, be sure that the Kerberos ticket-granting service ports (TCP port 88, UDP port 88) and also if used DNS, NTP ports are enabled on the network

#### 4.1.2 Domain Name System and Host Name resolution

Make sure to use for the NFS client and the NFS server the fully qualified domain name (FQDN) when you add them in the principal database. It is required for Kerberos to work.

#### 4.1.3 Time synchronisation between KDCs and Kerberos Clients

All hosts that participate in the Kerberos authentication system must have their internal clocks synchronized within a specified maximum amount of time (known as **clock skew**). This requirement provides another Kerberos security check. If the clock skew is exceeded between any of the participating hosts, client requests are rejected.

The clock skew also determines how long application servers must keep track of all Kerberos protocol messages, in order to recognize and reject replayed requests. So, the longer the clock skew value, the more information that application servers have to

collect.

The default value for the maximum clock skew is 300 seconds (five minutes). You can change this default in the `libdefaults` section of the `krb5.conf` file.

Nevertheless **it is strongly recommended to have exactly the same time between machines** to eliminate indirect effects for example when times are close to the expiration times. For that, it is recommended to use NTP (Network Time Protocol).

#### 4.1.4 FIRST entry of "hosts" database's record

To get Kerberos services working, the FIRST entry of "hosts" database's record must be the same on all machines (independently from a naming service used, whether it's ldap or nis or dns or just files).

It's because principals used by Kerberos admin are `kadmin/<admin host>@<REALM>` and `changepw/<admin host>@<REALM>` where `<admin host>` comes from the first entry queried from "hosts" database by the key taken from `admin_server` option of `krb5.conf` file.

#### 4.1.5 Realm name

Kerberos realm is case-sensitive. Make sure all the letters are uppercase.

The Kerberos realm is usually the DNS domain name converted to uppercase

#### 4.1.6 Keytab Files

The keytab should exist only on the machine's local disk and have the appropriate file access permissions (`chmod 600`). You will need to delete the keytab files from their original locations like the KDC server when not generated on the local machine. In that case also, you need to securely transport them to the desired destinations by using secure share (ssh) or some other means of secure file replication services. The keytab file is a potential point of entry for an attack. It should be readable only by root or the account designated to run the desired Kerberized service.

Be careful default following systems are `/etc/krb5/krb5.keytab` or `/etc/krb5.keytab`

### 4.2 KDC Machine Configuration

Knowing some network rules, first of all you need to identify the machine which will be the Kerberos Administration and Key Distribution Center (KDC). This machine will be the Kerberos Server.

No special options are required for the NFS integration. To the KDC, NFS is just another network service for which KDC basically issues various Kerberos tickets (TGT and Service tickets).

The Kerberos server runs the two following daemons:

- `krb5kdc` daemon -KDC service
- `kadmind` daemon -Kerberos administration service

1) On this machine you need to **edit `/etc/krb5.conf`** (Kerberos configuration file) by referring to « **man `krb5.conf`** »

Every machine in Kerberos environment must have this config file, it tells what is default realm for Kerberos libraries, and for each realm it tells where is the KDC (master and its slaves) and the administration server (admin server typically is on the

same machine as master KDC).

Example:

```
[libdefaults]
    default_realm = KRBNFS.NET
    default_keytab_name = FILE:/etc/krb5.keytab
    default_tkt_enctypes = des3-cbc-sha1 arcfour-hmac aes256-cts des-cbc-
md5 des-cbc-crc
    default_tgs_enctypes = des3-cbc-sha1 arcfour-hmac aes256-cts des-cbc-
md5 des-cbc-crc

[realms]
    KRBNFS.NET = {
        kdc = krbkdcsv.krbnfs.net:88
        admin_server = krbnfs.net:749
        default_domain = krbnfs.net
    }

[domain_realm]
    .krbnfs.net = KRBNFS.NET
    krbnfs.net = KRBNFS.NET

[kdc]
    profile = /var/kerberos/krb5kdc/kdc.conf

[logging]
    default = FILE:/var/kerberos/log/krb5libs.log
    kdc = FILE:/var/kerberos/log/krb5kdc.log
    admin_server = FILE:/var/kerberos/log/kadmin.log
```

2) Then **edit /etc/kdc.conf** (the Kerberos V5 KDC configuration file) referring to « **man kdc.conf** »

That's KDC configuration (pretty self-explanatory), which defines realms to be served by this KDC, and for each realm defines where the principal's database is.

Example:

```
[kdcdefault]
    acl_file = /var/kerberos/krb5kdc/kadm5.acl
    dict_file = /usr/share/dict/words
    admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab

[realms]
    KRBNFS.NET =
        master_key_type = des-cbc-crc
        supported_enctypes = arcfour-hmac:normal arcfour-hmac:norealm
arcfour-hmac:onlyrealm des3-hmac-sha1:normal des-hmac-sha1:normal des-cbc-
md5:normal des-cbc-crc:normal des-cbc-crc:v4 des-cbc-crc:afs3
    }
```

3) Then, create the principal's database by running **kdb5\_util create -s**

**#kdb5\_util create -s**

Initializing database '/var/kerberos/krb5kdc/principal' for realm  
'KRBNFS.NET',

master key name 'K/M@KRBNFS.NET'

You will be prompted for the database Master Password.

It is important that you NOT FORGET this password.

Enter KDC database master key::

Re-enter KDC database master key to verify::

the `-s` option specifies the stash file is also created. The stash file contains the master database password you can change later with the `kstash` command.

The command "**kdb5\_util destroy**" removes principal's database.

4) Set Access Control Level (ACL) for admin server.

Edit **kadm5.acl** (location specified in **kdc.conf** file(**acl\_file** option)), for instance give all permissions to "root/admin" principal.

```
root/admin@KRB NFS.NET *
```

5) Let **create** the **administrative user** « **root/admin** »

Add "root/admin" principal to the database:

```
# kadmin.local
```

```
kadmin.local: addprinc root/admin
```

```
WARNING: no policy specified for root/admin@KRB NFS.NET; defaulting to no policy
```

```
Enter password for principal "root/admin@KRB NFS.NET": :
```

```
Re-enter password for principal "root/admin@KRB NFS.NET": :
```

```
Principal "root/admin@KRB NFS.NET" created.
```

```
kadmin.local: listprincs
```

```
K/M@KRB NFS.NET
```

```
changepw/krbkdcsv.KRB NFS.NET@KRB NFS.NET
```

```
kadmin/admin@KRB NFS.NET
```

```
kadmin/changepw@KRB NFS.NET
```

```
kadmin/history@KRB NFS.NET
```

```
kadmin/krbkdcsv.krbnfs.net@KRB NFS.NET
```

```
krbtgt/KRB NFS.NET@KRB NFS.NET
```

```
root/admin@KRB NFS.NET
```

`kadmin.local` doesn't need KDC service running, it just edit principal's database directly.

We can see also, by default, `kdb5_util` creates several principals.

Double-check admin principal's (`kadmin` and `changepw`), they should correspond to KDC host name.

6) Start kdc service

```
service krb5kdc start
```

7) Start administration service

On some distributions the two following operations will be done by

```
service kadmin start
```

- Create service keys for `kadmin/admin` and `kadmin/changepw` principals in **kadm5.keytab**

Now we need starting the administration server (`kadmind`) for which we need to create a "keytab".

`kadmind(1M)` is like a normal Kerberos service, therefore:

- it must have one or more service principals,
- and besides, log-term Service's keys, corresponding to these principals, somehow must be shared with TGS, which

is krb5kdc.

Such long-term key sharing is implemented in the way that on KDC side keys are stored in principal's database, and on Service's side keys are in keytab files. According to man of **kadmind(1M)** it uses **kadmin/<admin host>** and **changepw/<admin host>** principals. Basically **kdb5\_util** generates them, so we can "export" these principals to a admin server's keytab. A keytab name is specified by **admin\_keytab** option in the **kdc.conf** file, which is often **/var/kerberos/krb5kdc/kadm5.keytab**

```
Run « kadmin.local -q "ktadd -k  
/var/kerberos/krb5kdc/kadm5.keytab kadmin/admin  
kadmin/changepw »
```

- Start kadmind daemon

```
Start « /usr/kerberos/sbin/kadmind »
```

**kadmind** creates some files e.g. **kadmin\_0,krb5kdc\_rcache**

## 9) A quick test of kerberos admin

```
# kadmin -p root/admin
```

```
Authenticating as principal root/admin@KRB NFS.NET with password.  
Password for root/admin@KRB NFS.NET:  
kadmin: listprincs  
K/M@KRB NFS.NET  
...
```

```
# kinit to obtain and cache Kerberos ticket-granting ticket, for this principal
```

```
# kinit root/admin
```

```
Password for root/admin@KRB NFS.NET:
```

```
# klist
```

```
Ticket cache: FILE:/tmp/krb5cc_0  
Default principal: root/admin@KRB NFS.NET  
Valid starting Expires Service principal  
06/06/07 14:06:24 06/07/07 14:06:24  
krbtgt/KRB NFS.NET@KRB NFS.NET  
Kerberos 4 ticket cache: /tmp/tkt0  
klist: You have no tickets cached  
"WARNING: no policy specified", a "policy" there means just a password  
policy.
```

## 4.3 Generating NFS Servers, NFS Client and Users Principals and exporting keys to keytab files

We have a KDC machine, let's go configuring the NFS Kerberos Server.

### 4.3.1 NFS KERBEROS SERVER Machine Configuration

1) First, the Kerberos configuration file **krb5.conf** must be the same as the KDC machine's one. Let's copy it by a secured way from the KDC machine.

2) Now, create machine credentials for the server. This means :

- creating a Kerberos V5 principal/instance name of the form **nfs/dns.name.of.server@REALM**,

- and either adding a key for this principal to an existing `/etc/krb5.keytab` or creating an `/etc/krb5.keytab`.

You can do it on the Kerberos server with `kadmin.local`

```
kadmin.local
```

```
Authenticating as principal root/admin@KRB NFS.NET with
password.
```

```
Password for root/admin@KRB NFS.NET
```

```
kadmin.local: addprinc -randkey nfs/dns.name.of.server
```

```
kadmin.local: ktadd -k /tmp/keytab nfs/dns.name.of.server
```

Now, `/tmp/keytab` needs to be copied with a securised tool to the nfs server in `/etc/krb5/keytab`

You can do it directly on the NFS server with `kadmin`:

```
kadmin
```

```
Authenticating as principal root/admin@KRB NFS.NET with
password.
```

```
Password for root/admin@KRB NFS.NET
```

```
kadmin: addprinc -randkey nfs/dns.name.of.server
```

```
kadmin: ktadd -k /etc/keytab nfs/dns.name.of.server
```

3) Check the principal exists with `« klist -k »`

4) Now start the server-side GSSAPI daemon `rpc.svcgssd`

5) Exporting

Let's fill the `/etc/exports` file for file systems which may be exported to NFS clients.

About security, four options are allowed:

- `sys`: UNIX(AUTH\_SYS) authentication

Based UID/GID authentication mechanism provided by NFS.

AUTH\_SYS is not considered a secure mechanism for authenticating users.

Nothing in RPC's AUTH\_SYS authentication protocol ensures that the user specified by the UID in the credential structure is permitted to use the RPC service, and nothing verifies that the user (or user running the application sending RPC requests) is really who the UID professes to be.

Use of Kerberos authentication is strongly encouraged as an alternative.

- `krb5`: Kerberos Authentication

Uses cryptography to prove mutually client and server identities.

- `krb5i`: Kerberos Integrity

Provides a cryptographic checksum of the data portion of each request and the response message to each request.

- `krb5p`: Kerberos Privacy

Encrypts all the contents of packets bi-directionally including user data

The data is encrypted only for transmission over the network; once data is received at the NFS server it is decrypted before being stored on disk.

Authentication is performed on each NFS request and response.

Knowing that, edit **/etc/exports** file to include the choiced Kerberos security option and run **exportfs -av** to export what it is contained in /etc/exports.

### 4.3.2 NFS KERBEROS CLIENT Machine Configuration

To finally access at a Kerberised NFS server, let's set up the client.

1) First, the Kerberos configuration file **krb5.conf** must be the same as the KDC machine's one. Let's copy it by a securised way from the KDC machine if not yet on the client machine.

2) Next, reate machine credentials for the client. This means :

- creating a Kerberos V5 principal/instance name of the form **nfs/dns.name.of.client@REALM**,
- and either adding a key for this principal to an existing **/etc/krb5.keytab** or creating an **/etc/krb5.keytab**.

You can do it on the Kerberos server with **kadmin.local**

```
kadmin.local
```

```
Authenticating as principal root/admin@KRB NFS.NET with password.
```

```
Password for root/admin@KRB NFS.NET
```

```
kadmin.local: addprinc -randkey nfs/dns.name.of.client
```

```
kadmin.local: ktadd -k /tmp/keytab nfs/dns.name.of.client
```

Now, **/tmp/keytab** needs to be copied with a securised tool to the nfs client in **/etc/krb5/keytab**

You can do it on the NFS client with **kadmin**:

```
kadmin
```

```
Authenticating as principal root/admin@KRB NFS.NET with password.
```

```
Password for root/admin@KRB NFS.NET
```

```
kadmin: addprinc -randkey nfs/dns.name.of.client
```

```
kadmin: ktadd -k /etc/keytab nfs/dns.name.of.client
```

3) Check the principal exits with **< klist -k >**

4) Now start the client-side GSSAPI daemon **rpc.gssd**

5) Check tickets with **klist /tmp/krb5cc\_machine\_REALM**

With Kerberos, access as root (UID 0) on the client uses the machine credentials.

```
klist /tmp/krb5cc_machine_REALM
```

```
Ticket cache: FILE:/tmp/krb5cc_machine_REALM
```

```
Default principal: nfs/dns.nfname.of.client@REALM
```

```
Valid starting Expires Service principal
```

```
06/17/07 14:47:14 06/18/07 14:44:02 krbtgt/REALM@REALM
```

```
06/17/07 14:47:14 06/18/07 14:44:02 nfs/dns.name.of.server@REALM
```

The above output shows us a number of things. The first line tells us that the

Kerberos tickets are being kept in the ticket cache file /tmp/krb5cc\_machine\_REALM.

The second line tells us that the tickets are for the principal nfs/dns.nfname.of.client in REALM's domain.

The rest of the lines show the tickets. The first one is the Kerberos ticket-granting-ticket (or krbtgt). This line tells three things, first that we have a valid ticket-granting-ticket. Second that it is valid starting at 2:22pm on March 24th. The time listed under "Expires" is the one you really care about. This is telling that at 3:22pm on March 25th, the ticket will expire and we will have to run kinit and get a new one. It will be automatically done by gssd if the machine credentials cache (/tmp/krb5cc\_machine\_REALM) is used. Underneath the ticket-granting-ticket you notice there is one other ticket. It is called service ticket. It is the one to access the nfs server.

### 4.3.3 USERS

#### User other than root:

To allow an user other than root to mount a partition you will need to:

- Create the Kerberos principal for that user using kadmin on NFS client machine or using kadmin.local on the KDC.
- Then on the client, as that user , run kinit.
- Edit /etc/fstab to add the **users** option in the line about the related mount

# 5 Automatically SETUP

Let's see now how with three commands we can do the same that it was just described above.

## 5.1 How to Automatically Configure a Kerbero Server

- Become superuser.
- Run the **krbkdcsv** installation script.

By default the following configuration will be proposed:

**Kerberos realm name:** the UPPERCASE DNS Domain Name

**KDC master host name:** the machine you are on

**Kerberos administration host name:** the machine you are on

**Dns name:** Current Dns name

**Kerberos Configuration File:** /etc/krb5.conf

**KDC Configuration File:** /var/kerberos/krb5kdc/kdc.conf

**Kerberos Administrator Name:** root

If you don't agree, you need to restart the command by specifying in the command the parameters you want to change:

- A command line call can be

### **krbkdcsv start**

The administrator will need to provide:

- ◆ the database Master Password
- ◆ password for principal "root/admin@REALM"

- **trace of an interactif example:**

```
[root@krbkdcsv SetUpKerberos]# ./krbkdcsv start
```

Here are the parameters from which the KDC and Kerberos administration Server configuration will be built:

```
-----  
Kerberos REALM:                KRBNFS.NET  
Domain Name:                   krbnfs.net  
KDC Server Name:               krbkdcsv.krbnfs.net  
Kerberos Administration Server Name:  krbkdcsv.krbnfs.net  
.....  
Kerberos Configuration File:   /etc/krb5.conf  
KDC Configuration File:       /var/kerberos/krb5kdc/kdc.conf  
.....  
Kerberos Administrator Name:   root
```

```
Do you agree with this KDC and Kerberos Administration configuration:  
yes/no[no]
```

```
yes
```

```
Stopping Kerberos 5 Admin Server
```

```
Deleting KDC database stored in '/var/kerberos/krb5kdc/principal', are you  
sure?
```

```
(type 'yes' to confirm)?
```

Yes

OK, deleting database '/var/kerberos/krb5kdc/principal'...

\*\* Database '/var/kerberos/krb5kdc/principal' destroyed.

Stopping ntpd

Starting ntpd

Loading random data

Initializing database '/var/kerberos/krb5kdc/principal' for realm 'KRB NFS.NET',  
master key name 'K/M@KRB NFS.NET'

You will be prompted for the database Master Password.

It is important that you NOT FORGET this password.

Enter KDC database master key:

Re-enter KDC database master key to verify:

Authenticating as principal root/admin@KRB NFS.NET with password.

WARNING: no policy specified for root/admin@KRB NFS.NET; defaulting to no  
policy

Enter password for principal "root/admin@KRB NFS.NET":

Re-enter password for principal "root/admin@KRB NFS.NET":

Principal "root/admin@KRB NFS.NET" created.

Starting Kerberos 5 KDC : [ OK ]

Extracting kadm5 Services Keys :Authenticating as principal  
root/admin@KRB NFS.NET with password.

Entry for principal kadmin/admin with kvno 3, encryption type ArcFour with  
HMAC/md5 added to keytab WRFIL E:/var/kerberos/krb5kdc/kadm5.keytab.

Entry for principal kadmin/admin with kvno 3, encryption type Triple DES cbc  
mode with HMAC/sha1 added to keytab

WRFIL E:/var/kerberos/krb5kdc/kadm5.keytab.

Entry for principal kadmin/admin with kvno 3, encryption type DES with  
HMAC/sha1 added to keytab

WRFIL E:/var/kerberos/krb5kdc/kadm5.keytab.

Entry for principal kadmin/admin with kvno 3, encryption type DES cbc mode  
with RSA-MD5 added to keytab

WRFIL E:/var/kerberos/krb5kdc/kadm5.keytab.

Entry for principal kadmin/changepw with kvno 3, encryption type ArcFour with  
HMAC/md5 added to keytab

WRFIL E:/var/kerberos/krb5kdc/kadm5.keytab.

Entry for principal kadmin/changepw with kvno 3, encryption type Triple DES  
cbc mode with HMAC/sha1 added to keytab

WRFIL E:/var/kerberos/krb5kdc/kadm5.keytab.

Entry for principal kadmin/changepw with kvno 3, encryption type DES with  
HMAC/sha1 added to keytab

WRFIL E:/var/kerberos/krb5kdc/kadm5.keytab.

Entry for principal kadmin/changepw with kvno 3, encryption type DES cbc  
mode with RSA-MD5 added to keytab

WRFIL E:/var/kerberos/krb5kdc/kadm5.keytab.

[ OK ]

Starting Kerberos 5 Admin Server : [ OK ]

Let's do some tests to see if the KDC Server works

- by running kinit root/admin without error

- by checking the krbtgt can be obtained

Password for root/admin@KRB NFS.NET:

klist: You have no tickets cached

Let's do some tests to see if the Kerberos Server works:

- by listing principals without error

Authenticating as principal root/admin with password.

Password for root/admin@KRB NFS.NET:

K/M@KRB NFS.NET  
kadmin/admin@KRB NFS.NET  
kadmin/changepw@KRB NFS.NET  
kadmin/history@KRB NFS.NET  
kadmin/nfs4\_gb.frec.bull.fr@KRB NFS.NET  
krbtgt/KRB NFS.NET@KRB NFS.NET  
root/admin@KRB NFS.NET

## 5.2 How to Automatically Configure a Kerberized NFS Server

- Become superuser.
- Run the **krbfnfssv** installation script. By default the command looks at an existing **/etc/krb5.conf** file to propose a **Kerberos administration server host name**. Then it gets by **scp** from this Kerberos server host name the **/etc/krb5.conf** file from which it proposes the:

***Kerberos realm name***

***Domain name***

***KDC Server Name***

***Kerberos Administration Name***

The KDC Configuration File proposed is **/etc/krb5.conf**

The default ntp server will be the *Kerberos administration host name*.

If you don't agree, you need to restart the command by specifying in the command another kerberos configuration file and/or the parameters you want to change.

The administrator will need to provide nevertheless the

***Password for the administrative principal***

At least the remote login is authorized by ssh, you will need:

- **Password for a ssh link with the Kerberos server**
- A command line call will be

**krbfnfssv start**

The administrator will need to provide:

- ◆ password for principal "[root/admin@REALM](#)"
- ◆ eventually ssh passwd with the Kerberos Server and the NFS server

- Trace of an interactif example

```
[root@krbfnfssv SetUpKerberos]# ./krbfnfssv start
```

Get by scp the **/etc/krb5.conf** file from the Kerberos server: **krbkdcsv**

Here are the parameters from which the Kerberos NFS Server configuration will be built:

Kerberos REALM:	KRB NFS.NET
Domain Name:	krbnfs.net
KDC Server Name:	krbkdcsv
Kerberos Administration Server Name:	krbkdcsv

```

NTP Server Name:                krbkdcsv
.....
Kerberos Configuration File:    /etc/krb5.conf
Do you agree with this Kerberos NFS Server configuration: yes/no[no]
yes

You need to load the rpcsec_gss_krb5 module if not already in the kernel
Do you want to have this command loading it: yes/no[no]
no
Let's check the krb5kdc daemon is running on the KDC and Administration
Server: krbkdcsv root@krbkdcsv's password:
krb5kdc is running

Let's check the kadmind daemon is running on the KDC and Kerberos
Administration Server: krbkdcsv
root@krbkdcsv's password:
kadmind is running

Let's check the nfsd daemon is running
nfsd daemon is running

ntp searching
host found : krbkdcsv

Create credentials for the server by creating a Kerberos V5
principal:nfs/krbnfssv.krbnfs.net Authenticating as principal
root/admin@KRBNFS.NET with password.          Password for
root/admin@KRBNFS.NET:
WARNING: no policy specified for nfs/krbnfssv.krbnfs.net@KRBNFS.NET;
defaulting to no policy
add_principal: Principal or policy already exists while creating
"nfs/krbnfssv.krbnfs.net@KRBNFS.NET".
Add a key for the nfs principal nfs/krbnfssv.krbnfs.net
Authenticating as principal root/admin@KRBNFS.NET with password.
Password for root/admin@KRBNFS.NET:
Entry for principal nfs/krbnfssv.krbnfs.net with kvno 4, encryption type DES cbc
mode with CRC-32 added to keytab WRFILE:/etc/krb5.keytab.

```

### 5.3 How to Automatically Configure a Kerberized NFS Client

- Become superuser.
- Run the **krbnfsc1** installation script.

By default the command looks at an existing **/etc/krb5.conf** file to propose a **Kerberos administration server host name**. Then it gets by **scp** from this Kerberos server host name the **/etc/krb5.conf** file from which it proposes:

```

Kerberos realm name
Domain name
KDC Server Name
Kerberos Administration Name

```

The KDC Configuration File proposed is **/etc/krb5.conf**

The default ntp server will be the *Kerberos administration host name*.

If you don't agree, you need to restart the command by specifying in the command another kerberos configuration file and/or the parameters you want to change.

The administrator will need to provide nevertheless the:

***NFS server name***

***Password for the administrative principal***

At least the remote login is authorized, you will need:

- **Password for a ssh link with the Kerberos server**
- **Password for assh link with the NFS server**
- A command line call will be  
**krb5fscl start**

The administrator will need to provide:

- ◆ password for principal "[root/admin@REALM](#)"
- ◆ eventually ssh passwd with the Kerberos Server and the NFS server

- Trace of an interactif example

```
[root@krb5fscl SetUpKerberos]# ./krb5fscl start
```

Get by scp the /etc/krb5.conf file from the Kerberos Server: krbkdcsv

Enter the NFS Server Name

krb5fssv

Here are the parameters from which the Kerberos NFS Client configuration will be built:

Kerberos REALM:	KRBNFS.NET	
Domain Name:	krb5fs.net	
KDC Server Name:	krbkdcsv	
Kerberos Administration Server Name:	krbkdcsv	
NFS Server Name:	krb5fssv	NTP Server
Name :	krbkdcsv	
User Name:	onlyrootuser	
.....		
Kerberos Configuration File:	/etc/krb5.conf	

Do you agree with those Kerberos configuration: yes/no[no]  
yes

You need to load the rpcsec\_gss\_krb5 module if not already in the kernel  
Do you want to have this command loading it: yes/no[no]  
no

Let's check the krb5kdc daemon is running on the KDC and Administration  
Server: krbkdcsv root@krbkdcsv's password:  
krb5kdc is running

Let's check the kadmind is running on the KDC and Kerberos Administration  
Server: krbkdcsv root@krbkdcsv's password:  
kadmind is running

Let's check nfsd daemon is running on the NFS Server: krb5fssv

root@krbnfssv's password:  
nfsd daemon is running

ntp searching

host found : krbkdcsv

Let's check time is synchronised (<300s) with the KDC Server machine  
krbkdcsv time root@krbkdcsv's password:

Time is Synchronized with the time of the KDC server:krbkdcsv

Get the /etc/krb5.conf file from krbkdcsv

root@krbkdcsv's password:

Authenticating as principal root/admin@KRB NFS.NET with password.

Password for root/admin@KRB NFS.NET:

WARNING: no policy specified for nfs/krbnfsc1.krbnfs.net@KRB NFS.NET;  
defaulting to no policy

add\_principal: Principal or policy already exists while creating  
"nfs/krbnfsc1.krbnfs.net@KRB NFS.NET".

Authenticating as principal root/admin@KRB NFS.NETwith password.

Password for root/admin@KRB NFS.NET:

Entry for principal nfs/krbnfsc1.krbnfs.net with kvno 4, encryption type ArcFour  
with HMAC/md5 added to keytab WRFILE:/etc/krb5.keytab.

Entry for principal nfs/krbnfsc1.krbnfs.netwith kvno 4, encryption type Triple  
DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.

Entry for principal nfs/krbnfsc1.krbnfs.net with kvno 4, encryption type DES  
with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.

Entry for principal nfs/krbnfsc1.krbnfs.net with kvno 4, encryption type DES cbc  
mode with RSA-MD5 added to keytab WRFILE:/etc/krb5.keytab.

## 5.4 When mount doesn't work

Each of the three commands has a status mode which checks if environment still OK.

Krbkdcsv status, krbnfssv status and krbnfsc1 status

It can be run in a interactive way or by giving parameters in the command line  
(run krbkdcsv/krbnfssv/krbnfsc1 status -h to get the possible parameters)

Here are the controls done:

- check client and server hosts are fully qualified name
- check REALM is UPPER CASE
- check time is synchronised (<300s) with the KDC Server machine
- check the /etc/hosts file lists the fully-qualified domain name as the first entry on the line with the machine's IP address,
- check in /etc/resolv.conf the name server is the same as in /etc/resolv.conf of the KDC Server
- check the /etc/services file lists the nfs service (port 2049) is fully qualified name
- check KDC, Kerberos Server and nfs Server are reachable
- check kerberos daemons are running (krb5kdc, kadmind) on the Kerberos Server
- check nfs server daemons are running on the NFS Server
- check rpc.gssd and rpc.svcgssd are running

**If still not working go to the next chapter: Troubleshooting Kerberos Errors.**

## 6 Troubleshooting Kerberos Errors

This chapter can help you troubleshoot NFS Kerberos authentication problems that might occur in a Linux operating system environment. It outlines some simple troubleshooting basics and explains the causes of common Kerberos errors. It also summarizes common tools used to troubleshoot problems with Kerberos authentication.

**Errors messages may not be explicit or the same cause can give different behavior or error messages ,so ALWAYS check you have well followed the all pre-requisites described in paragraph 4.1(Network Environment and Rules).**

### 6.1 Diagnostic Tools

#### 6.1.1 Log Files

The first place to look if you are experiencing a problem is to look at log files.

##### 6.1.1.1 systems logs: /var/log/messages

To get more NFSV4 messages:

Kernel NFS debugging can be enabled through /proc file system. All the debug messages will be logged in **/var/log/messages**.

```
echo "65535" > /proc/sys/sunrpc/nfsd_debug (debugging server)
echo "65535" > /proc/sys/sunrpc/nfs_debug (debugging client)
echo "65535" > /proc/sys/sunrpc/rpc_debug (RPC)
```

##### 6.1.1.2 KDC logs

When trouble shooting authentication issues, it can be very helpful to have a terminal open to the KDC running a *tail -f* on the KDC log. The location of the KDC log is specified in *krb5.conf*.

#### 6.1.2 Network Protocol Analysers

If the errors in the event logs do not help you solve the problem, or if you need more detailed information, use a Network Protocol Analyser to capture a network trace for inspection of the actual Kerberos packets being sent across the network. Here are two network analyser you can use.

- **Wireshark**

**Wireshark** is a GUI network protocol analyzer. It lets you interactively browse packet data from a live network or from a previously saved capture file. **Wireshark**'s native capture file format is **libpcap** format, which is also the format used by **tcpdump** and various other tools which Wireshark will be able to read/import file formats(tcpdump, AIX iptrace...)

- **Tshark**

It is the command line version of Wireshark.

- **Tcpdump**

Tcpdump is a command-line tool for monitoring network traffic. Tcpdump can capture and display the packet headers on a particular network interface or on all interfaces. Tcpdump can display all of the packet headers, or just the ones that match particular criteria.

Ex: tcpdump -s 256 -w tcpdump.out

### 6.1.2.1 Cases Examples

- **Show the Kerberos packets to/from the KDC**

Remove the credentials cache file (/tmp/krb5cc\_machine\_REALM) and capture during a restart of rpc.gssd and a subsequent mount request. That will allow the capture of the TGT request and the service ticket request.

### 6.1.3 Debug Output

You can use debug output associated with Kerberos authentication to obtain information if other troubleshooting tools fail to produce useful information.

#### 6.1.3.1 Client Side Debugging

- **rpc.gssd**

You will see more logging messages by starting the rpc.gssd process with -f -v -r options. This runs rpc.gssd in the foreground and increases the verbosity of the outputs printed on the console.

-v and -r can be asked several times: -vvv -rrr

#### 6.1.3.2 Server Side Debugging

- **rpc.svcgssd**

You will see more logging messages by starting the rpc.svcgssd process with -f -vvv -r options. This runs rpc.svcgssd in the foreground and increases the verbosity of the outputs.

#### 6.1.3.3 NFSv4 ID <-> Name Mapper Debugging

- **rpc.idmapd**

You will see more logging messages by starting the rpc.idmapd process with -f -vvv

#### 6.1.3.4 strace

strace is a useful diagnostic, instructional, and debugging tool.

Starting daemons or commands with it will help to find out problems.

Ex: sudo strace -o /tmp/rpc.gssd -f /usr/sbin/rpc.gssd -f -vvv

#### 6.1.3.5 gdb

The GNU debugger can help to resolve issues like for the following to determine a problem between rpc.idmapd and the libevent library.

Ex:

```
gdb rpc.idmapd
GNU gdb Red Hat Linux (6.5-8.fc6rh)
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu"...Using host libthread_db
library "/lib64/libthread_db.so.1".
```

```
(gdb) r
Starting program: /usr/sbin/rpc.idmapd
Error while reading shared library symbols:
DW_FORM_strp pointing outside of .debug_str section [in module
/usr/local/lib/libevent-1.1b.so.1]
```

Program exited with code 01.

```
(gdb) q
```

## 6.2 Common issues

The following section detail the most common problems encountered by users in Kerberos authentication environments, explain the possible causes of those problems, and suggest how to resolve those problems.

### 6.2.1 Bad Time Synchronization (Clock Skew)

#### 6.2.1.1 Issue

A bad time synchronisation can cause a lot of errors messages and bad behaviors.

*Clock skew too great*

*kadmin: GSS-API (or Kerberos) error while initializing kadmin interface*

.....

#### 6.2.1.2 Solution

Use NTP to synchronize.

Change skew in /etc/krb5.conf

### 6.2.2 Fully qualified domain name (FQDN) not used for client and NFS server when added in the principal database.

#### 6.2.2.1 Issue

*Permission denied* on the console

with the following messages in the /var/log/messages

*rpc.gssd[30631]: rpcsec\_gss: gss\_init\_sec\_context: (major) Unspecified GSS failure. Minor code may provide more information - (minor) Unknown code krb5 7*

*rpc.gssd[30631]: WARNING: Failed to create krb5 context for user with uid 0 with any credentials cache for server*

#### 6.2.2.2 Solution

Delete the principal not using the FQDN and add one respecting the FQDN rule.

### 6.2.3 FIRST entry of "hosts" database's record not the same on all machines

#### 6.2.3.1 Issue

On the server: *SERVER\_NOT\_FOUND* on KDC

On the client: *Required KADM5 principal missing while initializing kadmin interface*

It will happen for example in the following case:

If you have this on KDC machine:

```
# getent hosts nfs1.krbnfs.net
129.168.20.45 nfs1.krbnfs.net net
```

but that on a client machine:

```
# getent hosts nfs1.krbnfs.net
129.168.20.45 nfs1 nfs1.krbnfs.net
```

It's because principals used by Kerberos admin are `kadmin/<admin host>@<REALM>` and `changepw/<admin host>@<REALM>` where `<admin host>` comes from the first entry queried from "hosts" database by the key taken from `admin_server` option of `krb5.conf` file

### **6.2.3.2 Solution**

Check with « getent hosts » you have as first entry a FQDN name.

## **6.2.4 rpcsec\_gss\_krb5 kernel module not loaded**

### **6.2.4.1 Issue**

Rpc.gssd doesn't start

« failed to open /proc/net/rpc/auth.rpcsec.init/channel: No such file or directory »

### **6.2.4.2 Solution**

modprobe rpcsec\_gss\_krb5

## **6.2.5 Bad credentials cache used by the implementation**

Following the implementation a confusion can happen in the choice of the credentials used by rpc.gssd when there are several under /tmp.

Check rpc.gssd messages if other cache than /tmp/krb5cc\_machine\_REALM and keep only this one.

## **6.2.6 Bad file mode for /etc/krb5.keytab**

### **6.2.6.1 Issue**

When /etc/krb5.keytab has bad file mode (different to ) you can get the following message:

« Unsupported key table format version number while adding key to keytab »

### **6.2.6.2 Solution**

Remove /etc/krb5.keytab and let kadmin build it with the right file mode.

## 6.3 Errors

Here are some errors you can meet in log files. For some, there is a proposed way to recover.

### 6.3.1 Errors messages often met

**Message:**

*"add\_principal: Insufficient access to lock database while creating"  
before this one:  
"kadmin: Principal nfs/nfssv.krbnfs.net does not exist"*

**solution:**

This may due to a conflict between "selinux" and the kadmind started by /etc/service?

enforce "selinux" to permissive and try again.

you can try also by killing "kadmind" and restart it without using "service"  
(/usr/kerberos/sbin/kadmind)

**message:**

*create: No such file or directory while creating database  
'/var/kerberos/krb5kdc/principal'*

**solution:**

create /var/kerberos/krb5kdc directory

**message:** in /var/log/messages:

*failed to open /proc/net/rpc/auth.rpcsec.init/channel: No such file or directory*

**solution:** You need the gss kernel modules on nfs-servers and nfs-clients  
modprobe rpcsec\_gss\_krb5

**message:** in /var/log/messages on the client :

*rpc.gssd[xxxx] : Warning : failed to create krb5 context for user  
with uid 0 with any credentials cache for serveur krbnfs*

**solution:**

rpc.svdgssd not started

**message:** in /var/log/messages on the client :

*\* rpc.gssd[xxxx] : Warning : failed to create krb5 context for user  
with uid 0 with any credentials cache for serveur secu0*

**solution:**

KDC daemon not started

**message:** in /var/log/messages on the client :

*\* rpc.gssd[xxxx] : Warning : failed to create krb5 context for user  
with uid = 500 for server nfs*

**solution:**

user with uid=500 doesn't have valid TGT ticket (make kinit)

**message:** on the console

*mount /mnt/krb5*

*mount: only root can mount nfs4\_gb:/ on /mnt/krb5*

**solution:**

Check for the **users** option in /etc/fstab for this mount.

**message:** in /var/log/messages

on the console you have log on with an user other than root.

```
bash-3.1$ mount /mnt/krb5
bash-3.1$ cd /mnt/krb5
bash: cd: /mnt/krb5: Permission Denied
```

in /var/log/messages

```
rpc.gssd[30712]: ERROR: GSS-API: error in gss_acquire_cred(): Unspecified GSS failure. Minor code may provide more information - Unknown code krb5 195
```

```
rpc.gssd[30712]: WARNING: Failed to create krb5 context for user with uid 501 for server nfs4_gb.frec.bull.fr
```

**solution:**

You have managed to mount but you have Permission Denied to access the directory. You need to create a principal for the user and do a kinit for that user.

**Message:** in /var/log/messages

```
rpc.gssd[31856]: ERROR: No such file or directory while beginning keytab scan for keytab 'FILE:/etc/krb5.keytab'
```

```
last message repeated 2 times
```

```
rpc.gssd[31856]: ERROR: gssd_refresh_krb5_machine_credential: no usable keytab entry found in keytab /etc/krb5.keytab for connection with host nfs4_gb.frec.bull.fr
```

```
rpc.gssd[31856]: ERROR: No credentials found for connection to server nfs4_gb.frec.bull.fr
```

**solution:**

key missing in /etc/krb5.keytab of /etc/krb5.keytab not existing

Use kadmin to add this key.

**Message:** in /var/log/messages

```
WARNING: Decrypt integrity check failed while getting initial ticket for principal 'nfs/nfs1_gb.frec.bull.fr@FREC.BULL.FR' using keytab 'FILE:/etc/krb5.keytab'
```

**solution:**

Klist -k /etc/krb5.keytab will give more information on the contents but it can be better to remove it and recreate it.

## 6.3.2 Kerberos errors code.

Here are the kerberos errors code you will be appeared sometimes in the logging files under the form: « *Unknown code krb5 xxx* »

Here are the main ones:

- 0 KRB5KDC\_ERR\_NONE: No error
- 1 KRB5KDC\_ERR\_NAME\_EXP: Client's entry in database has expired
- 2 KRB5KDC\_ERR\_SERVICE\_EXP: Server's entry in database has expired
- 3 KRB5KDC\_ERR\_BAD\_PVNO: Requested protocol version not supported
- 4 KRB5KDC\_ERR\_C\_OLD\_MAST\_KVNO: Client's key is encrypted in an old master key
- 5 KRB5KDC\_ERR\_S\_OLD\_MAST\_KVNO: Server's key is encrypted in an old master key
- 6 KRB5KDC\_ERR\_C\_PRINCIPAL\_UNKNOWN: Client not found in Kerberos database
- 7 KRB5KDC\_ERR\_S\_PRINCIPAL\_UNKNOWN: Server not found in Kerberos database
- 8 KRB5KDC\_ERR\_PRINCIPAL\_NOT\_UNIQUE: Principal has multiple entries in Kerberos database
- 9 KRB5KDC\_ERR\_NULL\_KEY: Client or server has a null key
- 10 KRB5KDC\_ERR\_CANNOT\_POSTDATE: Ticket is ineligible for postdating  
*It may come from a time difference between the client and the KDC*
- 11 KRB5KDC\_ERR\_NEVER\_VALID: Requested effective lifetime is negative or too short  
*Requested start time is later than end time*  
*It may come from a time difference between the client and the KDC*
- 12 KRB5KDC\_ERR\_POLICY: KDC policy rejects request
- 13 KRB5KDC\_ERR\_BADOPTION: KDC can't fulfill requested option
- 14 KRB5KDC\_ERR\_ETYPE\_NOSUPP: KDC has no support for encryption type
- 15 KRB5KDC\_ERR\_SUMTYPE\_NOSUPP: KDC has no support for checksum type
- 16 KRB5KDC\_ERR\_PADATA\_TYPE\_NOSUPP: KDC has no support for padata type
- 17 KRB5KDC\_ERR\_TRTYPE\_NOSUPP: KDC has no support for transited type
- 18 KRB5KDC\_ERR\_CLIENT\_REVOKED: Clients credentials have been revoked
- 19 KRB5KDC\_ERR\_SERVICE\_REVOKED: Credentials for server have been revoked
- 20 KRB5KDC\_ERR\_TGT\_REVOKED: TGT has been revoked
- 21 KRB5KDC\_ERR\_CLIENT\_NOTYET: Client not yet valid - try again later
- 22 KRB5KDC\_ERR\_SERVICE\_NOTYET: Server not yet valid - try again later
- 23 KRB5KDC\_ERR\_KEY\_EXP: Password has expired  
*change password to reset*
- 24 KRB5KDC\_ERR\_PREAUTH\_FAILED: Preauthentication failed  
*It may come from a time difference between the client and the KDC*
- 25 KRB5KDC\_ERR\_PREAUTH\_REQUIRED: Additional pre-authentication required  
*This error can come when interoperability with Windows*
- 26 KRB5KDC\_ERR\_SERVER\_NOMATCH: Requested server and ticket don't match
- 27 KRB5PLACEHOLD\_27: KRB5 error code 27
- 28 KRB5PLACEHOLD\_28: KRB5 error code 28
- 29 KRB5PLACEHOLD\_29: KRB5 error code 29
- 30 KRB5PLACEHOLD\_30: KRB5 error code 30
- 31 KRB5KRB\_AP\_ERR\_BAD\_INTEGRITY: Decrypt integrity check failed  
*normally means the wrong password (or wrong key) was used*  
*check the krb5.keytab file or remove it and recreate it*
- 32 KRB5KRB\_AP\_ERR\_TKT\_EXPIRED: Ticket expired
- 33 KRB5KRB\_AP\_ERR\_TKT\_NYV: Ticket not yet valid
- 34 KRB5KRB\_AP\_ERR\_REPEAT: Request is a replay
- 35 KRB5KRB\_AP\_ERR\_NOT\_US: The ticket isn't for us
- 36 KRB5KRB\_AP\_ERR\_BADMATCH: Ticket/authenticator don't match
- 37 KRB5KRB\_AP\_ERR\_SKEW: Clock skew too great
- 38 KRB5KRB\_AP\_ERR\_BADADDR: Incorrect net address
- 39 KRB5KRB\_AP\_ERR\_BADVERSION: Protocol version mismatch
- 40 KRB5KRB\_AP\_ERR\_MSG\_TYPE: Invalid message type
- 41 KRB5KRB\_AP\_ERR\_MODIFIED: Message stream modified
- 42 KRB5KRB\_AP\_ERR\_BADORDER: Message out of order
- 43 KRB5KRB\_AP\_ERR\_ILL\_CR\_TKT: Illegal cross-realm ticket

44 KRB5KRB\_AP\_ERR\_BADKEYVER: Key version is not available  
 45 KRB5KRB\_AP\_ERR\_NOKEY: Service key not available  
 46 KRB5KRB\_AP\_ERR\_MUT\_FAIL: Mutual authentication failed  
 47 KRB5KRB\_AP\_ERR\_BADDIRECTION: Incorrect message direction  
 48 KRB5KRB\_AP\_ERR\_METHOD: Alternative authentication method required  
 49 KRB5KRB\_AP\_ERR\_BADSEQ: Incorrect sequence number in message  
 50 KRB5KRB\_AP\_ERR\_INAPP\_CKSUM: Inappropriate type of checksum in message  
 51 KRB5KRB\_AP\_PATH\_NOT\_ACCEPTED  
 52 KRB5KRB\_ERR\_RESPONSE\_TOO\_BIG  
 53-59 KRB5PLACEHOLD  
 60 KRB5KRB\_ERR\_GENERIC: Generic error (see e-text)  
 61 KRB5KRB\_ERR\_FIELD\_TOOLONG: Field is too long for this implementation  
 62-127 KRB5PLACEHOLD  
 128 KRB5\_ERR\_RCSID: \$Id: admin.texinfo,v 1.12.2.6 2001/02/23 00:31:34 tlyu Exp \$  
 129 KRB5\_LIBOS\_BADLOCKFLAG: Invalid flag for file lock mode  
 130 KRB5\_LIBOS\_CANTREADPWD: Cannot read password  
 131 KRB5\_LIBOS\_BADPWDMATCH: Password mismatch  
 132 KRB5\_LIBOS\_PWDINTR: Password read interrupted  
 133 KRB5\_PARSE\_ILLCHAR: Illegal character in component name  
 134 KRB5\_PARSE\_MALFORMED: Malformed representation of principal  
 135 KRB5\_CONFIG\_CANTOPEN: Can't open/find configuration file  
 136 KRB5\_CONFIG\_BADFORMAT: Improper format of configuration file  
 137 KRB5\_CONFIG\_NOTENUFSPACE: Insufficient space to return complete information  
 138 KRB5\_BADMSGTYPE: Invalid message type specified for encoding  
 139 KRB5\_CC\_BADNAME: Credential cache name malformed  
 140 KRB5\_CC\_UNKNOWN\_TYPE: Unknown credential cache type  
 141 KRB5\_CC\_NOTFOUND: Matching credential not found  
 142 KRB5\_CC\_END: End of credential cache reached  
 143 KRB5\_NO\_TKT\_SUPPLIED: Request did not supply a ticket  
 144 KRB5KRB\_AP\_WRONG\_PRINC: Wrong principal in request  
 145 KRB5KRB\_AP\_ERR\_TKT\_INVALID: Ticket has invalid flag set  
 146 KRB5\_PRINC\_NOMATCH: Requested principal and ticket don't match  
 147 KRB5\_KDCREP\_MODIFIED: KDC reply did not match expectations  
 148 KRB5\_KDCREP\_SKEW: Clock skew too great in KDC reply  
 149 KRB5\_IN\_TKT\_REALM\_MISMATCH: Client/server realm mismatch in initial ticket request  
 150 KRB5\_PROG\_ETYPE\_NOSUPP: Program lacks support for encryption type  
 151 KRB5\_PROG\_KEYTYPE\_NOSUPP: Program lacks support for key type  
 152 KRB5\_WRONG\_ETYPE: Requested encryption type not used in message  
 153 KRB5\_PROG\_SUMTYPE\_NOSUPP: Program lacks support for checksum type  
 154 KRB5\_REALM\_UNKNOWN: Cannot find KDC for requested realm  
 155 KRB5\_SERVICE\_UNKNOWN: Kerberos service unknown  
 156 KRB5\_KDC\_UNREACH: Cannot contact any KDC for requested realm  
 157 KRB5\_NO\_LOCALNAME: No local name found for principal name  
 158 KRB5\_MUTUAL\_FAILED: Mutual authentication failed  
 159 KRB5\_RC\_TYPE\_EXISTS: Replay cache type is already registered  
 160 KRB5\_RC\_MALLOC: No more memory to allocate (in replay cache code)  
 161 KRB5\_RC\_TYPE\_NOTFOUND: Replay cache type is unknown  
 162 KRB5\_RC\_UNKNOWN: Generic unknown RC error  
 163 KRB5\_RC\_REPLAY: Message is a replay  
 164 KRB5\_RC\_IO: Replay I/O operation failed XXX  
 165 KRB5\_RC\_NOIO: Replay cache type does not support non-volatile storage  
 166 KRB5\_RC\_PARSE: Replay cache name parse/format error  
 167 KRB5\_RC\_IO\_EOF: End-of-file on replay cache I/O  
 168 KRB5\_RC\_IO\_MALLOC: No more memory to allocate (in replay cache I/O code)  
 169 KRB5\_RC\_IO\_PERM: Permission denied in replay cache code  
 170 KRB5\_RC\_IO\_IO: I/O error in replay cache i/o code  
 171 KRB5\_RC\_IO\_UNKNOWN: Generic unknown RC/IO error  
 172 KRB5\_RC\_IO\_SPACE: Insufficient system space to store replay information

173 KRB5\_TRANS\_CANTOPEN: Can't open/find realm translation file  
174 KRB5\_TRANS\_BADFORMAT: Improper format of realm translation file  
175 KRB5\_LNAME\_CANTOPEN: Can't open/find lname translation database  
176 KRB5\_LNAME\_NOTRANS: No translation available for requested principal  
177 KRB5\_LNAME\_BADFORMAT: Improper format of translation database entry  
178 KRB5\_CRYPTO\_INTERNAL: Cryptosystem internal error  
179 KRB5\_KT\_BADNAME: Key table name malformed  
180 KRB5\_KT\_UNKNOWN\_TYPE: Unknown Key table type  
181 KRB5\_KT\_NOTFOUND: Key table entry not found  
182 KRB5\_KT\_END: End of key table reached  
183 KRB5\_KT\_NOWRITE: Cannot write to specified key table  
184 KRB5\_KT\_IOERR: Error writing to key table  
185 KRB5\_NO\_TKT\_IN\_RLM: Cannot find ticket for requested realm  
186 KRB5DES\_BAD\_KEYPAR: DES key has bad parity  
187 KRB5DES\_WEAK\_KEY: DES key is a weak key  
188 KRB5\_BAD\_ENCTYPE: Bad encryption type  
189 KRB5\_BAD\_KEYSIZE: Key size is incompatible with encryption type  
190 KRB5\_BAD\_MSIZ: Message size is incompatible with encryption type  
191 KRB5\_CC\_TYPE\_EXISTS: Credentials cache type is already registered.  
192 KRB5\_KT\_TYPE\_EXISTS: Key table type is already registered.  
193 KRB5\_CC\_IO: Credentials cache I/O operation failed XXX  
194 KRB5\_FCC\_PERM: Credentials cache file permissions incorrect  
195 KRB5\_FCC\_NOFILE: No credentials cache file found  
196 KRB5\_FCC\_INTERNAL: Internal file credentials cache error  
197 KRB5\_CC\_WRITE: Error writing to credentials cache file  
198 KRB5\_CC\_NOMEM: No more memory to allocate (in credentials cache code)  
199 KRB5\_CC\_FORMAT: Bad format in credentials cache  
200 KRB5\_INVALID\_FLAGS: Invalid KDC option combination (library internal error)  
[for dual tgt library calls]  
201 KRB5\_NO\_2ND\_TKT: Request missing second ticket [for dual tgt library calls]  
202 KRB5\_NOCREDS\_SUPPLIED: No credentials supplied to library routine  
203 KRB5\_SENDAUTH\_BADAUTHVERS: Bad sendauth version was sent  
204 KRB5\_SENDAUTH\_BADAPPLVERS: Bad application version was sent (via  
sendauth)  
205 KRB5\_SENDAUTH\_BADRESPONSE: Bad response (during sendauth exchange)  
206 KRB5\_SENDAUTH\_REJECTED: Server rejected authentication (during sendauth  
exchange)  
207 KRB5\_PREAUTH\_BAD\_TYPE: Unsupported preauthentication type  
208 KRB5\_PREAUTH\_NO\_KEY: Required preauthentication key not supplied  
209 KRB5\_PREAUTH\_FAILED: Generic preauthentication failure  
210 KRB5\_RCACHE\_BADVNO: Unsupported replay cache format version number  
211 KRB5\_CCACHE\_BADVNO: Unsupported credentials cache format version number  
212 KRB5\_KEYTAB\_BADVNO: Unsupported key table format version number  
213 KRB5\_PROG\_ATYPE\_NOSUPP: Program lacks support for address type  
214 KRB5\_RC\_REQUIRED: Message replay detection requires rcache parameter  
215 KRB5\_ERR\_BAD\_HOSTNAME: Hostname cannot be canonicalized  
216 KRB5\_ERR\_HOST\_REALM\_UNKNOWN: Cannot determine realm for host  
217 KRB5\_SNAME\_UNSUPP\_NAMETYPE: Conversion to service principal undefined  
for name type  
218 KRB5KRB\_AP\_ERR\_V4\_REPLY: Initial Ticket response appears to be Version 4  
error  
219 KRB5\_REALM\_CANT\_RESOLVE: Cannot resolve KDC for requested realm  
220 KRB5\_TKT\_NOT\_FORWARDABLE: Requesting ticket can't get forwardable tickets  
221 KRB5\_FWD\_BAD\_PRINCIPAL: Bad principal name while trying to forward  
credentials  
222 KRB5\_GET\_IN\_TKT\_LOOP: Looping detected inside krb5\_get\_in\_tkt  
223 KRB5\_CONFIG\_NODEFREALM: Configuration file does not specify default realm  
224 KRB5\_SAM\_UNSUPPORTED: Bad SAM flags in obtain\_sam\_padata

## 7 Linux Distributions examples

### 7.1 SLES

You will find a description of how to configure a NFSv4 Server and Client on a SLES 10 box at:

**<http://www.novell.com/coolsolutions/feature/17581.html>**

### 7.2 Redhat

You will find a description of how to configure a NFSv4 Server and Client on a RHEL5 box at:

**[www.redhat.co.hu/docs/manuals/enterprise/RHEL-5-manual/en-US/Deployment\\_Guide.pdf](http://www.redhat.co.hu/docs/manuals/enterprise/RHEL-5-manual/en-US/Deployment_Guide.pdf)**

## 8 Limitations

### 8.1) Kerberos v5 Pro

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

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

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

Here is a list of benefits:

- Single network sign on(SSO)  
Log onto your desk once and no more password prompt
- Use a central authentication server for all users and network services
- Heterogenous UNIX / Windows
- Resolve classic issues of client and services authentication inside a network
- It is a standard (RFC 1510) with several Open Sources implementations and also adopted by several other systems AIX, Solaris, Windows
- Passwords and Timestamps for expiration date
- Can be used on a open and unprotected network

Kerberos doesn't send any password across the network . Instead it uses session tickets that have expiration date.

- Mutual authentication:

In the Kerberos system the user principal must be authenticated by the server principal and also the server principal must prove to the user principal that it is indeed the server that the user intends to communicate with.

## **8.2) Compromise of Kerberos Infrastructure**

- **Protect your Kerberos Servers**

- The primary way in which an attacker will attempt to compromise a Kerberos Infrastructure would be to attack the Kerberos servers. If an attacker can gain root access to a KDC, the attacker will have access to the database of encrypted passwords of the principals. The attacker will also have access to the Kerberos software and configuration files both of which they can then modify to make the system perform authentications which should not be successful.
- Other methods of attacking Kerberos infrastructure include replay attacks and password-guessing attacks. A replay attack would involve intercepting or otherwise acquiring a Kerberos ticket and then fraudulently representing that ticket in an attempt to gain authentication. Password guessing in a Kerberos system could be done by intercepting Kerberos tickets from the network and then making a brute force attempt to decrypt the intercepted tickets.
- An attacker may exploit outdated software in the infrastructure. For example, there are many problems with Kerberos version 4. Most importantly, Kerberos version 4 has a basic protocol weakness in the encryption used. The design of Kerberos 4 included the use of DES in standard mode which allows attackers to intercept and modify the ciphertext of Kerberos tickets in an undetectable way. In order to prevent these attacks, Kerberos 5 has been modified to use triple DES in Cipher Block Chaining (CBC) mode.
- When discussing the strength of Kerberos 4, it is also important to note that many implementations of Kerberos version 4 have buffer overflow vulnerabilities. While the reference implementations of version 5 fixed the buffer overflow weaknesses in version 4, distributions of Kerberos 5 usually ship with software to provide backward compatibility to support legacy Kerberos 4 applications. Some of the backward compatibility code for version 4 support in Kerberos version 5 releases is still believed to be vulnerable to buffer overflow attacks.
- Therefore, due to the protocol weaknesses in version 4 and the potential for buffer overflow attacks with version 4 implementations and version 4 backward compatibility code, it is best not to support or use Kerberos version 4.
- In summary, from this description on how a Kerberos infrastructure can be compromised, we realize that we must give great priority to the security of the Kerberos servers themselves, run up to date Kerberos software, and remain vigilant in picking good passwords and in setting good password policy.
- It is important to note that Kerberos service should be run on dedicated hardware. Dedicating a machine to Kerberos means that only the Kerberos administrator will need to log in on those machines. It also means that no other services, except perhaps SSH, will be run on the machines. Since all of your users passwords are stored on the Kerberos servers, it is a good idea to limit access as much as possible to the hardware. Along with dedicating servers to Kerberos, you should also physically secure your servers as much as possible. For Kerberos servers, this may include locking the servers in a cabinet and having a dedicated terminal attached to them.

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

- **Protect the keys**

- Key sharing or key theft can allow impersonation attacks. If intruders somehow steal a principal's key, they will be able to masquerade as that user or service. To limit this threat, prohibit users from sharing their keys and document this policy in your security regulations.
- A keytab is analogous to a user's password. Just as it is important for users to protect their passwords, it is equally important for application servers to protect their keytab file. You should always store keytab files on a local disk, and make them readable only by the root user. Also, you should never send a keytab file over an unsecured network.

- **Protect the tickets**

- It is important to protect the credentials cache which contains tickets. Otherwise a non allowed user could access services he shouldn't to access.

- **Multi-user environment**

***Kerberos is more for single-user workstations***

In a multi-user environment, Kerberos is less secure. This is because it stores the tickets in the /tmp directory, which is readable by all users. If a user is sharing a computer with several other people simultaneously (i.e. multi-user), it is possible that the user's tickets can be stolen (copied) by another user.

This can be overwritten with the `-c filename` command-line option to or (preferably) the KRB5CCNAME environment variable, but this is rarely done.

## 9 Next/Future

The Version 1.0 of this tutorial was based on **linux-2.6.21-CITI\_NFS4\_ALL-1.diff** and **nfs-utils.1.1.0**.

The Version 2.0 is based on **linux-2.6.22-RC5-CITI\_NFS4\_ALL-1.diff**

Here are some evolutions which will be coming with the following releases you will be able to find at: <http://www.citi.umich.edu/projects/nfsv4/linux/>

### 9.1) Secinfo

In nfs-utils.1.1.0, the support for options of the form "sec=krb5:krb5i:krb5p" in the /etc/exports file is done.

An example of the new syntax:  
/exports \*(sec=sys:krb5:krb5i,rw)

or, if you want to restrict access based both on client and on security flavor:

/exports 192.168.0.0/16(sec=sys:krb5:krb5i,rw)

The client side is not yet implemented.

The client will get the list of flavors by performing a SECINFO call on mount.

It will be able possible to omit the "sec=" option on mount, or provide a list of flavors, in which case the client will restrict the choice of flavor to one of the flavors on your list.

## **9.2) SPKM-3(Simple Public-Key GSS-API Mechanism)**

SPKM is a GSS-API mechanism based on a public key technology, unlike Kerberos, which is based on symmetric key technology. SPKM provides authentication, key establishment, data integrity, and data confidentiality in an online distributed application environment using a public key infrastructure. SPKM data formats and procedures are designed to be as similar to those of the Kerberos mechanism as is practical, for easy implementation in those environments where Kerberos has already been implemented.

Because needed by LIPKEY(see below),the SPKM-3 support has been implemented. Nevertheless it is still not complete. There is experimental kernel code, but more work is needed for the user-level library.

## **9.3) LIPKEY**

The Low Infrastructure Public Key (LIPKEY) provides an alternative security flavor for Internet-based deployments of NFSV4. Although Kerberos is exceptionally good at intranet and local network deployments, scaling beyond a multirealm implementation under the control of a single organization is difficult because of the need to arrange manual trust relationships with external realms.

The NFS Version 4 protocol specifies an SSL-like GSS-API mechanism provider called LIPKEY as one of the two required security providers to RPCSEC\_GSS (Kerberos V5 being the other). LIPKEY uses asymmetric key algorithms. Like SSL, LIPKEY can be easily used through a firewall.

LIPKEY depends on SPKM-3. When the SPKM-3 support is complete, it should not be much more effort to add LIPKEY support.

## **9.4) NFSV4.1**

If the current implementations are based on RFC3530, changes are planned about Security in NFSV4.1 and are described in the draft-ietf-nfsv4-minorversion1-11 document so far (<http://tools.ietf.org/html/draft-ietf-nfsv4-minorversion1-11>)

## **9.5) SENFS(SELinux and NFSV4)**

A document of the integration of SELinux and NFSV4 is in progress at:

<http://namei.org/selinux/nfs/senfs-requirements-draft-05.txt>

## 10 FAQ

Have a look first at CITI FAQ:

<http://www.citi.umich.edu/projects/nfsv4/linux/faq/>

<http://www.faqs.org/faqs/kerberos-faq/general/preamble.html>

for other questions:

**Question) I am accessing an NFSv4 mount via Kerberos and then I do a kdestroy, but I am still able to access the NFS data. Why?**

**Answer)** *The kernel code caches the gssapi context that was negotiated using the Kerberos credentials. Destroying the credentials does not destroy the context in the kernel. There is a plan to change this behavior when moving to use the new key ring kernel support to store credentials and contexts.*

**Question) Does it work for both MIT and Heimdal Kerberos?**

**Answer)** *Yes*

## 11 Related Documents and Addresses Web Sites

- <http://www.connectathon.org/talks97/eisler1.pdf>
- [http://www.lwn.net/2001/features/OLS/pdf/pdf/nfsv4\\_ols.pdf](http://www.lwn.net/2001/features/OLS/pdf/pdf/nfsv4_ols.pdf)
- <http://www.ietf.org/rfc/rfc1510.txt>
- [http://blogs.sun.com/vl/entry/start\\_playing\\_with\\_kerberos](http://blogs.sun.com/vl/entry/start_playing_with_kerberos)
- <http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technologies/security/tkerberr.mspx>
- [www.netapp.com/library/tr/3481.pdf](http://www.netapp.com/library/tr/3481.pdf)
- <http://www.linux-tutorial.info/modules.php?name=Howto&pagename=Kerberos-Infrastructure-HOWTO/overview.html#howitworks>
- <http://www.citi.umich.edu/projects/nfsv4/>
- [http://wiki.linux-nfs.org/wiki/index.php/Enduser\\_doc\\_kerberos](http://wiki.linux-nfs.org/wiki/index.php/Enduser_doc_kerberos)
- [http://wiki.linux-nfs.org/wiki/index.php/General\\_troubleshooting\\_recommendations#Kerberos\\_issues](http://wiki.linux-nfs.org/wiki/index.php/General_troubleshooting_recommendations#Kerberos_issues)

# Table of Contents

1) Easy Setup.....	3
1.1) To do a mount.....	3
1.2) When rebooting or When a mount has failed.....	3
First, try <code>krbfnsc</code> status on the NFS client to check possible errors.....	3
1.3) Syntax.....	4
1.3.1) <code>krbkdcsv</code> usage.....	4
1.3.2) <code>krbnfssv</code> usage.....	4
1.3.3) <code>krbnfsc</code> usage.....	4
1.3.4) options.....	4
2 NFSV4 and Security.....	5
2.1) What is a Secure NFSV4.....	5
2.2) NFSv4 Security Plus.....	5
2.3) Notion of security services.....	6
2.4) How does Kerberos V5 authentication process work?.....	9
2.5) <code>RPCSEC_GSS</code> .....	11
2.6) How Kerberos Works with NFS.....	12
3 Commands, files and daemons.....	13
3.1) Files.....	13
3.1.1) NFSV4.....	13
3.1.2) GSS.....	13
Used by the client and server side when using <code>procf</code> s as a conduit between kernel and userspace for <code>rpc.gssd</code> and <code>rpc.svcgssd</code> .....	14
3.1.3) Kerberos.....	14
3.2) Daemons.....	16
3.2.1) NFSV4.....	16
3.2.2) <code>RPCSEC_GSS</code> User level daemons used to handle context initiation phase .....	16
3.2.3) Kerberos.....	16
3.3) Commands.....	17
3.3.1) NFSV4.....	17
3.3.2) Kerberos.....	17
4 Step by step.....	18
4.1) Network Environment and Rules.....	18
4.1.1) TCP/IP Network Connectivity.....	18
4.1.2) Domain Name System and Host Name resolution.....	18
4.1.3) Time synchronisation between KDCs and Kerberos Clients.....	18
4.1.4) FIRST entry of "hosts" database's record.....	19
4.1.5) Realm name.....	19
4.1.6) Keytab Files.....	19
4.2) KDC Machine Configuration.....	19
4.3) Generating NFS Servers, NFS Client and Users Principals and exporting keys to keytab files.....	22
4.3.1) NFS KERBEROS SERVER Machine Configuration.....	22
4.3.2) NFS KERBEROS CLIENT Machine Configuration.....	24
4.3.3) USERS .....	25
5 Automatically SETUP.....	26
5.1) How to Automatically Configure a Kerbero Server.....	26
5.2) How to Automatically Configure a Kerberized NFS Server .....	28

5.3	How to Automatically Configure a Kerberized NFS Client.....	29
5.4	When mount doesn't work .....	31
6	Troubleshooting Kerberos Errors.....	32
6.1	Diagnostic Tools.....	32
6.1.1	Log Files.....	32
6.1.1.1	systems logs: /var/log/messages.....	32
6.1.1.2	KDC logs.....	32
6.1.2	Network Protocol Analysers.....	32
	Wireshark.....	32
	Tshark.....	32
	Tcpdump.....	33
6.1.2.1	Cases Examples.....	33
	Show the Kerberos packets to/from the KDC.....	33
6.1.3	Debug Output.....	33
6.1.3.1	Client Side Debugging.....	33
6.1.3.2	Server Side Debugging.....	33
6.1.3.3	NFSv4 ID <-> Name Mapper Debugging.....	33
6.1.3.4	strace.....	33
6.1.3.5	gdb.....	33
6.2	Common issues.....	35
6.2.1	Bad Time Synchronization (Clock Skew).....	35
6.2.2	Fully qualified domain name (FQDN) not used for client and NFS server when added in the principal database.....	35
6.2.3	FIRST entry of "hosts" database's record not the same on all machines .....	35
6.2.4	rpcsec_gss_krb5 kernel module not loaded.....	36
6.2.5	Bad credentials cache used by the implementation.....	36
6.2.6	Bad file mode for /etc/krb5.keytab .....	36
6.3	Errors.....	37
6.3.1	Errors messages often met.....	37
6.3.2	Kerberos errors code.....	39
7	Linux Distributions examples.....	42
7.1	SLES.....	42
7.2	Redhat.....	42
8	Limitations.....	42
8.1)	Kerberos v5 Pro.....	42
8.2)	Compromise of Kerberos Infrastructure.....	43
	Protect your Kerberos Servers.....	43
	Protect the keys.....	44
	Protect the tickets.....	44
	Multi-user environment.....	44
9	Next/Future.....	44
9.1)	Secinfo.....	44
9.2)	SPKM-3(Simple Public-Key GSS-API Mechanism) .....	45
9.3)	LIPKEY.....	45
9.4)	NFSV4.1.....	45
9.5)	SENF(SLinuX and NFSV4).....	45
10	FAQ.....	46
11	Related Documents and Addresses Web Sites.....	46