

- Partners
- Support
- Community
- Ubuntu.com

- [Login to edit](#)

# Kerberos

Kerberos is an authentication protocol using a combination of secret-key cryptography and trusted third parties to allow secure authentication to network services over untrusted networks. More information about the Kerberos protocol is available from MIT's Kerberos site. Designing an Authentication System is an accessible introduction to the principals of Kerberos' authentication scheme.

## Intended Audience

This guide aims to supplement the documentation available in the <https://help.ubuntu.com/> by re-iterating certain key concepts in more detail and providing information on network service configuration. It is directed at system administrators that need to supplement their understanding of Kerberos and its advanced configuration.

Several Kerberos implementations exist. Two common open-source implementation of the Kerberos protocol are the original MIT implementation, and Heimdal, an implementation that was created to avoid United States export regulations. These regulations are no longer a concern, and for sake of brevity, this guide will provide instructions for the MIT version. The same concepts apply to Heimdal, although the syntax of various commands is different.

Microsoft's **Active Directory** is a common closed-source implementation of a Kerberos authentication realm. The following guide contains several notes that give specific configuration information for Active Directory. In an Active Directory environment, the KDC is typically one of the services provided by the **Domain Controller (DC)**. Readers wishing to integrate with an Active Directory Domain can skip directly to the Client Configuration section.

## Pre-installation Decisions

A central part of Kerberos' trusted third party authentication scheme is the **Key Distribution Center (KDC)**, which is a centralized repository for users' password information. Before deploying Kerberos,

### Sommaire

1. Intended Audience
2. Pre-installation Decisions
3. Prerequisites
  1. Host Names
  2. Connectivity
  3. Time Synchronization
  4. Firewalls
4. KDC Setup
  1. Principals
  2. Realm Administration: kadmin
  3. Publish Realm Info Via DNS
5. Client Configuration
  1. Package Installation
  2. Configuration
6. Testing
7. Kerberizing Local Authentication
  1. Installation
  2. Configuration
  3. Credential Caching
8. Configuring Network Services
  1. Generating Keytabs
  2. Security
  3. Service Configuration
  4. Remote Login
  5. Apache
  6. Integration with SASL
9. Troubleshooting
10. Further Reading
11. Acknowledgements

a server must be selected to take on the role of KDC. Physical and network security will be critical for this server, since a compromised KDC compromises the security of the entire realm.

Selecting a descriptive name for the Kerberos authentication realm is also important. Conventionally, the realm name is the site's domain name fully capitalized. For instance, the site `example.com` would conventionally use `EXAMPLE.COM` for its realm name.

## Prerequisites

All servers and clients that participate in a Kerberos realm must be able to communicate with each other and have accurate system clocks. The following section describes how to meet these requirements.

### Host Names

Each server in a Kerberos authentication realm must be assigned a **Fully Qualified Domain Name (FQDN)** that is forward-resolvable.

Kerberos also expects the server's FQDN to be reverse-resolvable. If reverse domain name resolution is not available, set the `rdns` variable to `false` in `clients' krb5.conf`

**Note:** Active Directory depends heavily on **DNS**, so it is likely that the Active Directory Domain Controller is also running the Microsoft DNS server package. If this is the case, verify that each server has a FQDN assigned to it before performing the tests outlined in this section.

If the server already has an FQDN assigned to it, test forward and reverse look-up with the following commands:

```
$ nslookup server.example.com
$ nslookup <server ip address>
```

The output of the first command should contain the IP address of the server. The output of the second command should contain the FQDN of the server.

If the server does not already have a FQDN assigned to it and DNS services are not available, name resolution can be implemented by editing the local hosts file (typically this is located in `/etc`) on each server and client, adding the following line:

```
127.0.0.1 linuxwork.example.com localhost linuxwork
```

Test the local DNS name resolution using the `nslookup` technique described at the beginning of the section.

### Connectivity

To verify connectivity between hosts, ping each host's FQDN:

```
$ ping kerberos.example.com
```

```
PING kerberos.example.com (10.0.0.1) 56(84) bytes of data:
64 bytes from kerberos.example.com (10.0.0.1): icmp_seq=1 ttl=128 time=0.176ms
```

The output of the ping command shows successful resolution of the FQDN to an IP Address, and a sample reply from the server. A reply from the server serves as confirmation that the host and the server can communicate.

Connectivity failures while pinging the KDC's FQDN usually point to DNS server or client configuration errors. See the Network Configuration for information on correct network settings.

### Time Synchronization

The Kerberos protocol requires the time of the client and server to match: if the system clocks of the client does not match that of the server, authentication will fail. The simplest way to synchronize the system clocks is to use a **Network Time Protocol (NTP)** server. See [UbuntuTime](#) for details.

**Note:** Active Directory Domain Controllers are typically also NTP servers.

## Firewalls

As with all network services, Kerberos must be allowed to pass through any firewalls between hosts. The Kerberos System Administration Manual has a detailed section on this topic.

## KDC Setup

```
$ sudo apt-get install krb5-kdc krb5-admin-server
$ sudo dpkg-reconfigure krb5-kdc
```

These packages are available from the Main repository. See [InstallingSoftware](#) for details on software installation, repositories and package management.

The package installation process will step through defining the basic Kerberos configuration parameters. Recommended settings are:

- disable Kerberos 4 compatibility mode
- do not run `krb524d` (daemon to convert Kerberos tickets between versions)
- defaults for the other settings are acceptable

The default location of the KDC's configuration file is `/etc/krb5kdc/kdc.conf`. Important settings are the locations of the KDC's data files, and the default settings for the durations that tickets are valid. The realm name must be configured; other configuration options should only be changed by knowledgeable users.

Here is an example configuration file:

```
[kdcdefaults]
    kdc_ports = 750,88
    default_realm = EXAMPLE.COM

[realms]
EXAMPLE.COM = {
    database_name = /var/lib/krb5kdc/principal
    admin_keytab = FILE:/etc/krb5kdc/kadm5.keytab
    acl_file = /etc/krb5kdc/kadm5.acl
    key_stash_file = /etc/krb5kdc/stash
    kdc_ports = 750,88
    max_life = 10h 0m 0s
    max_renewable_life = 7d 0h 0m 0s
    master_key_type = des3-hmac-sha1
    supported_enctypes = des3-hmac-sha1:normal des-cbc-crc:normal des:normal des:v4 des:r
    default_principal_flags = +preauth
}
```

Kerberos uses an Access Control List (ACL) to specify the per-principal access rights to the Kerberos admin daemon. This file's default location is `/etc/krb5kdc/kadm5.acl`. The default as shown below is sufficient for most realms, but additional ACLs may be necessary depending on the network configuration.

```
# This file is the access control list for krb5 administration.
# When this file is edited run /etc/init.d/krb5-admin-server restart to activate
# One common way to set up Kerberos administration is to allow any principal
# ending in /admin is given full administrative rights.
# To enable this, uncomment the following line:
*/admin@EXAMPLE.COM *
```

Create the Kerberos database with the following command:

```
$ krb5_newrealm
```

## Principals

Principals are entries in the Kerberos database that represent users or services on the network. Each user or service that participates in a Kerberos authentication realm **must** have a principal defined in the Kerberos database.

User principals take the form `principal@REALM.NAME`. For instance, user `tom` on the `EXAMPLE.COM` realm should have a principal `tom@EXAMPLE.COM` in the Kerberos database.

Service principals take the form `service/server.fqdn@REALM.NAME`. For example, the FTP service on `lab.example.com` in the `EXAMPLE.COM` realm would have the principal `ftp/lab.example.com@EXAMPLE.COM` in the Kerberos database.

Principal names are case sensitive.

## Realm Administration: kadmin

The Kerberos realm is administered using the `kadmin` utility. The `kadmin` utility is an interactive interface that allows the administrator to create, retrieve, update, and delete realm principals. `kadmin` can be run on any computer that is part of the Kerberos realm, provided the user has the proper credentials. However, for security reasons, it is best to run `kadmin` on a KDC.

An alternative program, `kadmin.local`, is available as well. Running `kadmin.local` as the root user on the KDC allows the administrator to authenticate without having an existing principal. This is generally not necessary in a properly configured environment.

To start the `kadmin` utility, issue the following command:

```
$ kadmin -p <principal>
```

Replace `<principal>` with a valid principal name. By default, the principal `admin/admin` has administrative rights to the realm, so to launch `kadmin` as the realm administrator, use:

```
$ kadmin -p admin/admin
```

Type `?` and press Enter at the `kadmin` prompt to see a list of valid commands.

### Common tasks

- Add a user:

```
kadmin: addprinc user
```

The default realm name is appended to the principal's name by default

- Delete a user:

```
kadmin: delprinc user
```

- List principals:

```
kadmin: listprincs
```

- Add a service:

```
kadmin: addprinc service/server.fqdn
```

The default realm name is appended to the principal's name by default

- Delete a user:

```
kadmin: delprinc service/server.fqdn
```

## Publish Realm Info Via DNS

Client configuration can be simplified by publishing information about the Kerberos realm via DNS. This is accomplished by adding SRV records that point to the Kerberos KDC.

To do this in a DNS zone controlled by BIND requires entries like the following. Here the DNS domain is `example.com`, and the Kerberos realm is `EXAMPLE.COM`.

```
$ORIGIN example.com.
_kerberos          TXT          "EXAMPLE.COM"
$ORIGIN _udp.example.com.
_kerberos          SRV          0 0 88 kdc.example.com.
_kerberos-adm     SRV          0 0 749 kdc.example.com.
```

See [BIND9ServerHowto](#) for information on configuring DNS services with BIND.

With this DNS configuration in place, it should not be necessary to add information to the `/etc/krb5.conf` on client computers: the Kerberos software should get all necessary realm information from the DNS server. The `kadmin` utility still seems to require the `[domain_realm]` mappings, though.

## Client Configuration

**Note:** Verify the clients meet the criteria laid out in the [Prerequisites](#) section before installing client software.

## Package Installation

Kerberos services and utilities require several separate packages:

- **krb5-user:** Basic programs to authenticate using MIT Kerberos.
- **krb5-config:** Configuration files for Kerberos Version 5.
- **libkadm55:** MIT Kerberos administration runtime libraries. (No longer available in Karmic)

These packages are available from the Main repository. See [InstallingSoftware](#) for details on software installation, repositories and package management.

## Configuration

### krb5-config

The `krb5-config` installation process customizes the `/etc/krb5.conf` file. Supply your realm name when prompted to enter a default realm. If Kerberos SRV records aren't available via DNS, the installation process will prompt for basic information about the Kerberos realm:

```
What are the Kerberos servers for your realm?
kerberos.example.com
```

```
What is the administrative server for your Kerberos realm?
kerberos.example.com
```

Substitute the FQDN of the KDC for the example answers.

### Manual configuration: `/etc/krb5.conf`

It is also possible to configure the Kerberos client manually by editing `/etc/krb5.conf`. This

approach allows greater customization of the file, but lacks the automation of the `krb5-config` package. The end result is the same. The following is a sample configuration.

```
[logging]
    default = FILE:/var/log/krb5.log

[libdefaults]
    default_realm = EXAMPLE.COM
    kdc_timesync = 1
    ccache_type = 4
    forwardable = true
    proxiable = true

[realms]
    EXAMPLE.COM = {
        kdc = kerberos.example.com
        admin_server = kerberos.example.com
        default_domain = EXAMPLE.COM
    }

[domain_realm]
    .example.com = EXAMPLE.COM
    example.com = EXAMPLE.COM
```

## Testing

To test the operation of Kerberos, request a Ticket-Granting Ticket (TGT) with the `kinit` command, as shown. Any valid Kerberos principal can be substituted for "Administrator". Omit the realm name from the command if the `default_realm` directive is properly specified in the `/etc/krb5.conf` file or DNS SRV records.

**Note:** The realm name is **CASE SENSITIVE**.

```
$ kinit -p Administrator@LAB.EXAMPLE.COM
Password for Administrator@LAB.EXAMPLE.COM: ****
```

Use the `klist` command to verify the TGT is valid:

```
$ klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: Administrator@LAB.EXAMPLE.COM

Valid starting      Expires            Service principal
01/21/05 10:28:51  01/21/05 20:27:43  krbtgt/LAB.EXAMPLE.COM@LAB.EXAMPLE.COM
    renew until 01/21/05 20:28:51
```

If the above tests work, congratulations! You have successfully obtained a Kerberos ticket. This ticket can be used to access network services that use Kerberos for authentication.

The `kinit` program will attempt to obtain a TGT for the principal `<username>@REALM.NAME` if run without a `-p` argument where `<username>` is the username of the local user, and `REALM.NAME` is the name of the default realm as configured in `/etc/krb5.conf`

Release the test ticket by issuing the `kdestroy` command.

## Kerberizing Local Authentication

Kerberos is frequently used as a source for local authentication through the `pam-krb5` module. Adding this module to a host's PAM stack allows users to authenticate to the local host using their Kerberos password. Given a username and password, `pam-krb5` will check the supplied password against the password for principal `<username>@REALM.NAME`. The `pam-krb5` module will inform the local system that the user authenticated successfully if the supplied password matches the password of the principal.

The module also establishes a credentials cache when a user has authenticated successfully, allowing

the user to utilize Kerberized network services without entering their credentials a second time.

## Installation

**Note:** Before deploying Kerberized PAM authentication, make sure that each each user has a valid account, either on the local host (via `adduser` or similar), or through a shared source such as LDAP.

To install the `pam-krb5` PAM module, issue the following command from a command prompt:

```
$ sudo apt-get install libpam-krb5
```

## Configuration

In Ubuntu release 9.04 (Jaunty Jackalope) and newer, the details of PAM configuration are handled by the `pam-auth-update` utility. To enable or disable Kerberos authentication, run `pam-auth-update` from a command prompt. More information on `pam-auth-update` is available in its documentation.

For releases prior to Jaunty, a basic configuration can be implemented by adding the following line to the top of the stack in `/etc/pam.d/common-auth`:

```
auth    sufficient      pam_krb5.so minimum_uid=1000
... 
```

PAM is a highly configurable framework; consult the PAM documentation for more information on advanced configuration options, such as falling back on local password services.

## Credential Caching

For roaming hosts such as laptops that may not always have access to the KDC, it is useful to cache credentials using the `libpam-ccreds` package. Installation is simple:

```
$ sudo apt-get install libpam-ccreds
```

## Configuring Network Services

Once a user has obtained a TGT using `kinit`, they can use it to prove their identity to a network service such as file sharing or printing. This authentication process is automatic: no password is required to access network services as long as the user's TGT is valid (for security purposes, tickets expire after a period of time, and must be renewed).

Services use files called **keytabs** that contain a secret known only to the service and the KDC. The user authenticates themselves to the KDC, and then requests information from the KDC that is encrypted using the service's shared secret. This encrypted message is sent to the client, which sends it to the service. If the service can decrypt and read the message (and the user passes other security checks), the service accepts the user's identity.

## Generating Keytabs

To create a service keytab, first create a service principal of the form `service/host.fqdn@REALM.NAME` using `kadmin`. For example, an FTP service on `lab.example.com` in the `EXAMPLE.COM` realm would have a principal named `ftp/lab.example.com@EXAMPLE.COM`, and would be added like this:

```
kadmin: addprinc -randkey ftp/lab.example.com@EXAMPLE.COM
```

The realm name is optional if the principal is part of the default Kerberos realm as configured in `krb5.conf`.

Once the principal has been created, create or add to a keytab with the `ktadd` command in the `kadmin` interface. To create a keytab for the FTP service from the preceding example, the command would look like this:


```
kadmin: ktadd -k /etc/ftp.keytab ftp/lab.example.com@EXAMPLE.COM
```

**Note:** the user running `kadmin` must have write access to the path specified by the `-k` argument.

As before, the realm name is optional for principals in the default realm. The keytab's path is also variable, although keytabs typically reside in the host's `/etc` directory.

It is not necessary to generate the keytab on host for which the keytab is intended. For example, one could simply generate the keytab as indicated on the KDC, save it in a temporary location, and **securely** transport it (encrypted transport such as `scp`, or physically via Flash drive) to the intended host.

## Security

 **The security of a keytab is vital: malicious users with access to keytabs can impersonate network services.** To avoid this, secure the keytab's file permissions with `chmod` (this is a system command, not a `kadmin` command). Using the FTP example, the command looks like this:

```
$ chmod 0700 /etc/ftp.keytab
```

The file ownership may also require adjustment to allow access by the FTP daemon.

## Service Configuration

Once the keytab is generated, the service must be configured to use it. Consult the documentation of the software package for details on this configuration process.

## Remote Login

The "host" service is the name of the service provided by remote access services such as telnet or SSH. To support Kerberized remote login:

1. Create a service principal with the name `host/client.fqdn@REALM.NAME` (e.g. `host/lab.example.com@EXAMPLE.COM`).
2. Save a keytab for this principal in `/etc/krb5.keytab`
3. Set the file permissions to `0700` and change the owner to `root`:

```
$ sudo chown root:root /etc/krb5.keytab
$ sudo chmod 0700 /etc/krb5.keytab
```
4. Configure the remote access service to use Kerberos. Refer to the service's documentation for instructions.

## Apache

Apache supports Kerberized authentication via `mod_auth_kerb`. The basic configuration utilizes Apache's Basic Auth mechanism. Transport layer encryption (i.e. SSL/TLS) should be deployed if the basic configuration is used; the Basic Auth mechanism does not encrypt login credentials during transmission. <sup>1</sup>

It is also possible to configure web browsers to use an existing credentials cache (instead of requiring a user to re-enter their Kerberos username and password) by enabling GSSAPI SPNEGO. Transport layer encryption is not necessary if SPNEGO is used, but the client's browser must be properly configured.



## Server

1. Create a principal and keytab for the web server:

```
$ sudo kadmin.local
kadmin> addprinc -randkey HTTP/webserver.example.com
kadmin> ktadd -k /etc/apache2/auth/apache2.keytab HTTP/webserver.example.com
kadmin> quit
```

2. Set the keytab's permissions and ownership:

```
$ sudo chown www-data:www-data /etc/apache2/auth/apache2.keytab
$ sudo chmod 400 /etc/apache2/auth/apache2.keytab
```

3. Install mod\_auth\_kerb:

```
# sudo apt-get install libapache2-mod-auth-kerb
```

4. Update the Apache configuration file (generally available in /etc/apache2/sites-enabled/). Here access is restricted to /var/www.

```
<Directory /var/www/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all

    AuthType Kerberos
    AuthName "Kerberos Login"
    KrbAuthRealm EXAMPLE.COM
    Krb5Keytab /etc/apache2/auth/apache2.keytab
    KrbMethodK5Passwd off #optional--makes GSSAPI SPNEGO a requirement
    Require valid-user
</Directory>
```

## Browser

To utilize existing an existing credentials cache (generated by kinit or a SSO login), the client's browser must be configured to use GSSAPI SPNEGO. The details of browser configuration varies. The following sections list resources for common browsers.

### Firefox

- <http://www.grolmsnet.de/kerbtut/firefox.html>
- <http://mbechler.eenterphace.org/blog/index.php?/archives/6-Doing-GSSNegotiate-SSO-using-Mozilla-Firefox,-MIT-Kerberos-and-PHP.html>

### Chrome

- <http://www.chromium.org/developers/design-documents/http-authentication>

### Internet Explorer

- [http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tsec\\_SPNEGO\\_config\\_web.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tsec_SPNEGO_config_web.html)

## Integration with SASL

Several popular open source software packages (most notably OpenLDAPServer) use Kerberos as an authentication source via the SASL pluggable authentication framework. The following section describes how to install and configure SASL, and verify it is utilizing Kerberos correctly. The details of configuring a service to utilize SASL depend on the package; refer to the service's documentation for details.

## Package Installation

We will need the SASL pluggable authentication framework, and the GSSAPI module for the Kerberos implementation in use. Here the MIT implementation is used. Run the following command on the server:

```
$ sudo apt-get install sasl2-bin libsasl2-2 libsasl2-modules libsasl2-modules-gssapi-mit
```

## Setup

1. Create a Kerberos service principal. Follow the naming conventions outlined in the Principals section.
2. Generate a keytab for the service principal and securely transfer it to the server running the service. Restrict the keytab's file permissions as necessary.
3. Configured the service to access the correct keytab. Many services will use the system keytab (`/etc/krb5.keytab`) by default. If the keytab for the service principal is not stored in the system keytab, the service must be configured to read the correct keytab. This is generally done by setting the environment variable `KRB5_KTNAME`:

```
$ export KRB5_KTNAME=/path/to/keytab
```

This command can often be added to the service's startup procedure by adding it to the service's defaults file in `/etc/default`.

## Test and Troubleshoot

Configuration issues are common at this stage, so a test is in order, using the SASL sample client and server.

1. Make sure the user running the sample server has permission to access to the appropriate keytab. If the keytab for the service principal is stored in a file other than the default system principal (`/etc/krb5.keytab`), you will need to set the environment variable `KRB5_KTNAME`:

```
$ export KRB5_KTNAME=/path/to/keytab
```

2. Start the SASL sample server on the server with the GSSAPI mechanism and the specified service:

```
$ sasl-sample-server -m GSSAPI -s servicename
```

3. In a separate terminal **on the server**, initialize a credentials cache for a user using `kinit`:

```
$ kinit
```

4. In the same terminal, start the SASL sample client:

```
$ sasl-sample-client -s servicename -n server.fqdn -u user
```

Replace "servicename" with the name of the service that is being tested.

5. Copy the line beginning with "S: " from the server window to the client window and hit Enter. The client will generate a line beginning with "C: " that must be copy/pasted into the server window. This exchange must be repeated several times. The process should completely with no errors. Here is an example of a successful run from the server side (some lengthy strings elided):

```
admin@server:~$ sasl-sample-server -m GSSAPI -s ldap
Forcing use of mechanism GSSAPI
Sending list of 1 mechanism(s)
S: R1NTQVBJ
Waiting for client mechanism...
C: R1N...
got 'GSSAPI'
Sending response...
S: YIG...
Waiting for client reply...
```

```

C:
got ''
Sending response...
S: YD8G...
Waiting for client reply...
C: YEc...
got '...'
Negotiation complete
Username: user
Realm: (NULL)
SSF: 56
sending encrypted message 'srv message 1'
S: AAAAS...
Waiting for encrypted message...
C: AAAAU...
got ''
recieved decoded message 'client message 1'

```

The same run, from the client's side:

```

user@server:~$ sasl-sample-client -s ldap -n ldap.server.fqdn -u user
service=ldap
Waiting for mechanism list from server...
S: RlNTQVBJ
recieved 6 byte message
Choosing best mechanism from: GSSAPI
returning OK: user
Using mechanism GSSAPI
Preparing initial.
Sending initial response...
C: RlN...
Waiting for server reply...
S: YIG...
recieved 153 byte message
C:
Waiting for server reply...
S: YD8...
recieved 65 byte message
Sending response...
C: YEc...
Negotiation complete
Username: user
SSF: 56
Waiting for encoded message...
S: AAAAS...
recieved 77 byte message
recieved decoded message 'srv message 1'
sending encrypted message 'client message 1'
C: AAAAU...

```

Address any error messages before continuing. The Troubleshooting section of <http://mah.everybody.org/docs/sasl-gssapi/> contains several common failure modes and their solutions.

6. Repeat the same process with the sample server running on the server and the sample client running on a host that will need to access the service.

## Troubleshooting

Kerberos is fairly fault-tolerant, if the requisite services are in place. If Kerberos authentication fails, check the following:

1. The user has a valid ticket (use `klist`).
2. Basic network connectivity is available (use `ping`).
3. Forward DNS hostname lookup succeeds on both the KDC and the local machine.
4. Reverse DNS lookup succeeds on both the KDC and local machine, or `rdns` is set to `false` in `krb5.conf`

5. The clocks of the KDC and local machine are synchronized.
6. The file permissions for various critical files such as `/etc/krb5.keytab` are correct and accessible by the user or service in question. `sudo -u <username>` can be helpful for this.

Following the KDC logs while attempting an operation can also be very informative. The following lines in `/etc/krb5kdc/kdc.conf` will enable KDC logging:

```
[logging]
    kdc = FILE:/var/log/krb5kdc.log
```

If you had to edit `kdc.conf` to enable logging, restart the KDC to apply the changes:

```
$ sudo service krb5-kdc restart
```

Once logging is enabled, you can follow the log while attempting an operation:

```
$ sudo tail -f /var/log/krb5kdc.log
```

## Further Reading

- The Ubuntu Server Guides from the official Ubuntu documentation provide a great deal of useful information.
- OpenSSH & Kerberos A detailed guide to SSH authentication with Kerberos.
- MIT's Kerberos site
- Designing an Authentication System
- NCSA Kerberos Troubleshooting for Unix

## Acknowledgements

Parts of this document are derived from the SingleSignOn and Samba/Kerberos documentation. Many thanks to the authors of both documents.

1. <http://httpd.apache.org/docs/current/howto/auth.html> (1)

Kerberos (dernière édition le 2013-09-11 01:35:16 par enlightened-despot @ localhost[127.0.0.1]:enlightened-despot)