Utilizing Kerberos authentication for Kerberized services and Web Single Sign-On

Table of contents

[Types of Single Sign-on 4](#_Toc262267832)

[Client based 4](#_Toc262267833)

[Server based 6](#_Toc262267834)

[Kerberos Authentication Protocol 7](#_Toc262267835)

[Kerberos terminology 7](#_Toc262267836)

[Kerberos Operation 8](#_Toc262267837)

[Benefits 11](#_Toc262267838)

[Drawbacks 11](#_Toc262267839)

[Implementing Kerberos V 12](#_Toc262267840)

[Pre-configuration 12](#_Toc262267841)

[Operating System 12](#_Toc262267842)

[Network details 12](#_Toc262267843)

[Users 12](#_Toc262267844)

[Services 13](#_Toc262267845)

[Useful utilities 13](#_Toc262267846)

[Step 1: Berkeley Internet Name Domain (BIND) 16](#_Toc262267847)

[DNS Operation 16](#_Toc262267848)

[BIND Installation 18](#_Toc262267849)

[BIND Configuration 18](#_Toc262267850)

[named.conf.local 18](#_Toc262267851)

[named.conf.options 18](#_Toc262267852)

[Zone files 18](#_Toc262267853)

[resolv.conf 19](#_Toc262267854)

[hostname files 19](#_Toc262267855)

[Testing BIND 20](#_Toc262267856)

[Step 2: MIT Kerberos V 22](#_Toc262267857)

[Kerberos Installation 22](#_Toc262267858)

[Kerberos Configuration 22](#_Toc262267859)

[krb5.conf 22](#_Toc262267860)

[kdc.conf 23](#_Toc262267861)

[kadm5.acl 24](#_Toc262267862)

[Kerberos Administration 24](#_Toc262267863)

[Initializing the Kerberos realm 24](#_Toc262267864)

[kadmin.local 24](#_Toc262267865)

[listprincs 24](#_Toc262267866)

[addprinc 25](#_Toc262267867)

[kadmin 25](#_Toc262267868)

[Adding users 25](#_Toc262267869)

[Setting up Kerberos clients 25](#_Toc262267870)

[Obtaining tickets manually 25](#_Toc262267871)

[Obtaining tickets automatically 26](#_Toc262267872)

[Step 3: Kerberized services 27](#_Toc262267873)

[telnet 27](#_Toc262267874)

[telnet configuration 27](#_Toc262267875)

[Testing telnet 28](#_Toc262267876)

[rlogin 29](#_Toc262267877)

[Testing rlogin 29](#_Toc262267878)

[Secure Shell (SSH) 29](#_Toc262267879)

[SSH Installation 29](#_Toc262267880)

[SSH Configuration 29](#_Toc262267881)

[Testing SSH 30](#_Toc262267882)

[File Transfer Protocol 30](#_Toc262267883)

[FTP Installation 30](#_Toc262267884)

[FTP Configuration 30](#_Toc262267885)

[Testing FTP 30](#_Toc262267886)

[References 32](#_Toc262267887)

Utilizing Kerberos authentication for Kerberized services and Web Single Sign-On

Single sign-on, abbreviated as “SSO”, is a way to work around the hassle of constantly having to authenticate into applications. Many employees and students will agree that constantly getting prompted to enter their password(s) over and over is more than just a nuisance and we can all agree that this proves to be detrimental for the overall efficiency of a company

When looking at Single Sign-on systems we can divide them in two main group, namely the client based and the server based ones.

# Types of Single Sign-on

## Client based

Client Based Single Sign-on systems prompt the user for his password the first time he wants to log in to a new website or application. The program then stores the password on the local computer and subsequently it will automatically enter these credentials whenever the user gets prompted for them.

Unfortunately this way of authentication requires some user input which makes room for human error. Another disadvantage to this is that in most cases all the data is stored on the local hard drive. It would be possible to work around this by for example bringing your authentication data with you everywhere you go on a stick, but this just creates further *complications*.

One big advantage of this method, however, is that once you have configured the application you can immediately gain access to multiple sites and applications that are not interconnected.

In the end this method of Single Sign-on authentication simply lacks the flexibility needed to be efficient in a production environment. This does prove to be a good method for simplifying authentication on a personal computer.

On the following page you can find an example of the Novell SecureLogin Single Sign-on tool. Several screenshots are depicted on which you can see the prompts presented to the end-user on the first time he authenticates to a new site.



http://www.novell.com/products/securelogin/media.html



http://www.novell.com/products/securelogin/media.html

## Server based

The key component in a server based single sign-on environment is one central authentication server within a certain production environment which takes care of all the authentication requests. Once you have logged in to the authentication server you will possess a ticket, whenever you will try to log in to a service you will send that ticket to the service. On its turn the service will crosscheck the ticket with the central authentication server to see if you have access to that service. If you do you will be authenticated automatically.

It can pretty much be compared to a ski resort, once you paid for your skipass you will get access to all the slopes you paid for.

Once the Single Sign-on system is in place an employee can for instance log into his workstation and from that point on he will possess a ticket which will allow him to authenticate easily via the central authentication server. When he opens outlook his login credentials will be passed on to the mail server and he will be authenticated without having to do a thing. He will just as well get logged in with a smooth transition when he connects to the intranet, a local fileserver, etc.

Implementing this will of course have a positive influence on the efficiency of the production environment and it will result in a rather fast Return on Investment.

Some Single Sign-on systems prompt the user for a smartcard instead of his credentials, he will then have to enter the card into a reader and his credentials will be loaded by the card reader. It is also possible to use a smartcard on top of normal login credentials to increase security.

The most secure authentication method right now uses OTP-tokens. OTP stands for One Time Password. The user has a small hardware device which generates a new password every time. These are either based on a mathematical algorithm or on time synchronization with the authentication server.

This thesis will further discuss the technical details of the Kerberos authentication protocol and show how to set up and configure a Kerberos server using normal login credentials in order to achieve an efficient Enterprise Single Sign-on system.

Kerberos Authentication Protocol

Kerberos is an authentication protocol which allows users to easily log into network services over an insecure network in a secure manner. It was developed in the early 80’s by MIT and has since then known great popularity, mainly thanks to the implementation of the protocol within the Windows server environments in the form of Active Directory.

The open-source nature of Kerberos also allows cross platform authentication which will be addressed later on in this thesis.

# Kerberos terminology

Following is a list with common Kerberos terminology. Most terms will be elaborated upon during the explanation of how the protocol works. This list has been quoted from HitMill.com [TODO01].

**Authentication Service (AS)** - Performs authentication and is a part of the Key Distribution Center (KDC).

**Key Distribution Center (KDC)** - Holds secret keys (the crytographic keys) for *"principals"*; provides authentication; creates and distributes session keys (crytographic keys). Session keys and secret keys are crytographic keys. The KDS utilizes symmetric cryptography. A KDC has a Ticket Granting Service (see TGS) and the Authentication Service.

**Principal** - Any object such as user, application, service, or resource which utilizes Kerberos authentication is referred to as principal. Collectively, the objects using Kerberos are principals. A Key Distribution Center (KDC) is responsible for one or more *"realms"* of principals. Any principal must "trust" the KDC. Principals do not directly trust each other. Only the KDC is supposed to have a copy of each principals "secret key".

**Realm** - A group or set of principals which are grouped together logically by a network administrator is called a realm. Again, a Key Distribution Center (KDC) is responsible for one or more realms.

**TGS (Ticket Granting Service)** - The part of the Key Distribution Center (KDS) which creates and distributes tickets to the objects (principals) containing session keys.

**Ticket** - A digital authentication token sent from the Authentication Service (AS). The first ticket sent from the AS to a principal (user, application, service or resource) is called the Ticket Granting Ticket (TGT).

**Secret keys and Session keys** - Symmetric cryptography keys used for both authentication and/or data encryption.

# Kerberos Operation

Kerberos authentication relies on trusted third parties to work and utilizes encrypted tickets controlled by those parties.

Three parties can be identified in the authentication scheme:

* The client that needs a service
* The server providing the service
* The Key Distribution Center, the trusted third party

For the sake of simplicity, the concepts of cross-realm authentication will be left out.

The Key Distribution Center is the heart of all Kerberos operations. It is composed of a authentication server, which cross-checks any credentials with a user database, and a ticket granting server, which is responsible for checking the validity of tickets and helping the client establish a connection with the server.

In order to gain access to a service, the following steps occur [TODO02, TODO03 **Fre, zet gij de bron van waar ge die images hebt gehaald hier ook bij?**]:



Figure 1: Kerberos authentication steps

Each step is described in detail on the next pages.

1. A user enters his username on a Kerberos-enabled client.
2. The client sends an Authentication Request to the Authentication Server, along with its username and a timestamp. This request is unencrypted.



Figure 1: Step 1 and 2

1. The Authentication Server checks if the username is present in its database. If so, it generates a secret key based on the found username and password. It also generates a Client/TGS session key, which is encrypted with the client secret key. It then sends a Ticket Granting Ticket (or TGT) encrypted with the secret key of the Ticket Granting Server. This initial ticket contains the username, the client network address, a timestamp, the validity period of the ticket and the Client/TGS session key as well as some other flags that the client may have requested.



Figure 2: Step 3

1. The user is prompted for his password. The client then generates a client secret key based on the username and password of the user.
2. The client attempts to decrypt the received packet with the client secret key. If successful, it gains access to the Client/TGS session key and knows that it has received a valid ticket from the Authentication Server.
3. The user attempts to access a service (e.g. a FTP server).
4. The client sends its TGT and an authenticator to the Ticket Granting Server. The authenticator comprises of the username and the timestamp of the client, and is encrypted with the Client/TGS session key. It also sends the requested service and a timestamp unencrypted.



Figure 3: Step 7

1. The Ticket Granting Server decrypts the TGS Request with the Client/TGS session key and checks if the client TGT has not expired and belongs to the authenticator using several techniques.
2. If the TGT is valid, the Ticket Granting Server creates a randomly generated Client/Service key. It then creates a Service Ticket, which contains the username of the client, the requested service, a timestamp, the Client/Service key and the lifetime of the ticket. It encrypts this ticket with the service secret key. Both the Client/Service key and the newly created Service Ticket are then encrypted with the Client/TGS key and are sent back to the client.



Figure 4: Step 8 and 9

1. The client receives the packet and decrypts it with the Client/TGS key to extract the Client/Service key. The Service Ticket remains encrypted.
2. The client creates an authenticator with the username and timestamp of the client. This authenticator is encrypted with the Client/Service key. It then sends an Application Request to the application server, which contains the authenticator and the Service Ticket.



Figure 5: Step 10 and 11

1. The application server decrypts the Service Ticket with its service secret key and extracts the Client/Service key. This key is used to decrypt the authenticator. The application server then checks if the Service Ticket is still valid and belongs to the authenticator using the same techniques as in step 8.
2. (OPTIONAL) The application server send a Application Reply back to the client to confirm it is indeed the server the client wished to contact. This step is executed when mutual trust is required.



Figure 6: Step 12 and 13

1. Communication between the client and the service server can now commence.



Figure 7: Step 14

Having seen how Kerberos performs secure authentication, several benefits and drawbacks can be concluded when using it.

## Benefits

* A user’s password is never sent over the network.
* All communication is encrypted and can only be cracked through brute-force methods.
* Single sign-on capabilities are achieved within a certain time span, usually one day.
* Passwords are stored in a single location: the Key Distribution Center.

## Drawbacks

* The Key Distribution Center is a single point of failure. If unavailable due to hardware problems or Denial-of-Service attack, secure authentication is made impossible. If a hacker manages to gain root access on the KDC, he will have access to the encrypted passwords and the Kerberos configuration files. Therefore, it is imperative the KDC is well protected.
* On a multi-access system, tickets stored locally in a user’s temporary folder can be possibly copied by another user, who can then use each ticket during its validity period.
* A hacker may intercept a ticket and then represent that ticket to gain access to the system without knowing the user password. This is called a replay attack, and is mitigated by methods such as time skewing and authenticator caching. However, to be completely safe from replay attacks, all communication between the client and the service server should be encrypted with the Client/Service key, which the hacker cannot acquire at any given time [TODO04].
* Sniffers on the client computer may steal the unencrypted user password. A strong company password policy should be enforced to prevent this situation.
* Weak passwords may be easily guessed or brute-forced. Again, a strong company policy should attempt to prevent weak passwords from being used.
* The previous version Kerberos, V4, contains several buffer overflow exploits and suffers from weak default encryption methods. A KDC that accepts V4 tickets will also be vulnerable to these attacks. Therefore, it is important that outdated services are upgraded to use Kerberos V5 and the KDC configured to ignore V4 requests [TODO05].
* Kerberos is an “all-or-nothing” approach: encrypting all remote logins, but sending e-mail passwords over the network unencrypted defeats the purpose of Kerberos encryption.

Implementing Kerberos V

At the end of this thesis, a proof-of-concept Kerberos authentication system for use in an intranet system will have been created. The system can be used for several Kerberized services, which allow a user to automatically use the credentials he used to log into the client computer for several services such as telnet, SSH, FTP and automatic authentication with a web server. This process is called Single Sign-On, as the user is only prompted for his password once. The concept of Web-SSO will be expanded upon, including the possibility to authenticate with principals from an Active Directory server when approaching the web server remotely, and using a PHP5 module to construct GSS-API security context with other applications.

This thesis will explain in detail the steps required to set up this system and many possibilities for extending the created system.

# Pre-configuration

Before actually configuring the system, some decisions had to be made concerning the operating system, the network details, which users were permitted to configure the server, which services would be served and where they would be located, and the installation of several useful utilities.

## Operating System

Ubuntu 9.04 was used as the Linux distribution, as we are well acquainted with this distribution and have used it to create a DNS server, web server and mail server during our studies at the Katholieke Hogeschool Mechelen. This operating system was installed on a VMware Workstation.

## Network details

The Virtual Machine was configured to use a bridged configuration, which means it was on the same subnet as the host computer. The KDC was configured to use 192.168.1.200 as its IP and 192.168.1.254 as its gateway. It also referred to itself as a DNS server. DNS zones it was not responsible for were directed towards the DNS servers of the ISP. The following files were changed to accommodate this:

/etc/network/interfaces:

auto lo

iface lo inet loopback

auto eth0

iface eth0 inet static

address 192.168.1.200

netmask 255.255.255.0

gateway 192.168.1.254

/etc/resolv.conf:

domain khm.lan

search khm.lan

nameserver 192.168.1.200

## Users

The main user, called ‘khmuser’, was given sudo powers.

## Services

The following services were “Kerberized” (use Kerberos authentication and possibly encryption):

* Telnet
* Rlogin
* SSH
* FTP
* Apache

Every service ran on the same machine as the KDC and the DNS server for the sake of simplicity.

## Useful utilities

Several utilities were used to debug, edit or showcase the functionality of a system feature. The following utilities were installed:

* The Ubuntu desktop package.

sudo apt-get install ubuntu-desktop

* Wireshark, a network packet sniffer.

sudo apt-get install wireshark

* Nmap, a network port scanner.

sudo apt-get install nmap

* ghg

Dfgf

root@ubuntu:/home/nojger# adduser khmuser

root@ubuntu:/home/nojger# adduser khmuser admin

khmuser@ubuntu:/home$ sudo apt-get install nmap wireshark gedit

khmuser@ubuntu:/home$ sudo apt-get install bind9

# OOK NOG MANUEEL IP INSTELLEN, verwijderen networkmanager enz

Named.conc.local wijzigen, zones toevoegen (zie mail fre)

khmuser@ubuntu:/home$ cd /etc/bind

khmuser@ubuntu:/etc/bind$ ls

db.0 db.255 db.local named.conf named.conf.options zones.rfc1918

db.127 db.empty db.root named.conf.local rndc.key

khmuser@ubuntu:/etc/bind$ sudo gedit named.conf.local

khmuser@ubuntu:/etc/bind$ cd /var/cache/bind

khmuser@ubuntu:/var/cache/bind$ sudo gedit khm.lan.zone

khmuser@ubuntu:/var/cache/bind$ sudo gedit /etc/bind/named.conf.local

khmuser@ubuntu:/var/cache/bind$ sudo gedit reverse-0.168.192.zone

khmuser@ubuntu:/var/cache/bind$ sudo /etc/init.d/bind9 restart

khmuser@ubuntu:/var/cache/bind$ ping krb.khm.lan

khmuser@ubuntu:/var/cache/bind$ sudo apt-get install krb5-admin-server krb5-kdc

khmuser@ubuntu:~$ sudo gedit /etc/krb5.conf

khmuser@ubuntu:~$ cd /var/log/

khmuser@ubuntu:/var/log$ sudo mkdir krb5

khmuser@ubuntu:/var/log$ sudo touch /var/log/krb5/krb5kdc.log

khmuser@ubuntu:/var/log$ sudo touch /var/log/krb5/kadmind.log

khmuser@ubuntu:/var/log$ sudo touch /var/log/krb5/krb5lib.log

Indien niet werkt : sudo krb5\_newrealm

khmuser@ubuntu:/var/log$ sudo gedit /etc/krb5kdc/kadm5.acl

kadmin.local: addprinc root/admin

kadmin.local: addprinc khmuser

khmuser@ubuntu:/var/log$ sudo apt-get install libpam-krb5

khmuser@ubuntu:~$ sudo gedit /etc/pam.d/common-session

khmuser@ubuntu:~$ sudo gedit /etc/pam.d/common-auth

khmuser@ubuntu:~$ sudo apt-get install krb5-telnetd

addprinc -randkey host/krb.khm.lan

ktadd host/krb.khm.lan

test:

telnet.krb5 -x krb.khm.lan

rlogin

khmuser@ubuntu:/etc$ sudo apt-get install openssh-server

khmuser@ubuntu:/etc$ sudo gedit /etc/ssh/sshd\_config

khmuser@ubuntu:/etc$ sudo gedit /etc/ssh/ssh\_config

ssh krb.khm.lan -l khmuser

sudo apt-get install krb5-ftpd

kadmin.local: addprinc -randkey ftp/krb.khm.lan

kadmin.local: ktadd ftp/krb.khm.lan

sudo gedit /etc/hostname wijzigen naar krb.khm.lan

# Step 1: Berkeley Internet Name Domain (BIND)

The first step involves installing and configuring a DNS server, as Kerberos relies heavily on properly configured DNS records in order to function correctly and in a secure manner. This chapter will discuss the operation of DNS and the installation and configuration of a Berkeley Internet Name Domain server, more commonly known as bind. This is one of the de facto standards on Unix-like systems.

## DNS Operation

Let us assume that a client wishes to go to the website <http://toledo.khm.be>.

When a DNS-enabled client tries to go to that address, its system will contact one of the root servers. These servers are authoritative for the DNS root zone. There are 13 root server clusters: when taking a closer look at the nameserver status, 13 zones are detected. The zones refer to the root servers.

Checking for nameserver BIND  
version: 9.6.1-P3  
CPUs found: 1  
worker threads: 1  
number of zones: 13  
debug level: 0  
xfers running: 0  
xfers deferred: 0  
soa queries in progress: 0  
query logging is OFF  
recursive clients: 0/0/1000  
tcp clients: 0/100  
server is up and running

Figure TODO: Output from the “rcnamed status” command

DNS addresses are read from back to front. One of the root servers will take a look at the address and will see that the first part is “be”.

It will then return a message to the client, forwarding it to one of the top level domain DNS servers. Top-level domain DNS servers are servers which are very high in the DNS hierarchy and are often responsible for namespaces such as .com, .net, .edu,…

In the case of our client, the top-level domain server it will be forwarded to is the server responsible for the .be zone. This server knows about the khm zone and will send the address of the khm DNS server back to the client.

The client will then contact the khm DNS server which will finally supply the client with an address record.

Issuing the dig command explains how this process works:

#dig toledo.khm.be  
  
; <<>> DiG 9.5.1-P2.1 <<>> toledo.khm.be  
;; global options: printcmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17431  
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 0  
  
;; QUESTION SECTION:  
;toledo.khm.be. IN A  
  
;; ANSWER SECTION:  
toledo.khm.be. 3600 IN CNAME athens.khm.be.  
athens.khm.be. 3600 IN A 193.191.150.39  
  
;; AUTHORITY SECTION:  
[khm.be](http://khm.be). 86400 IN NS dns.khm.be.  
[khm.be](http://khm.be). 86400 IN NS ns.belnet.be.  
  
;; Query time: 373 msec  
;; SERVER: 127.0.0.1#53(127.0.0.1)  
;; WHEN: Tue May 18 13:23:53 2010  
;; MSG SIZE rcvd: 110

Figure TODO: Output of the “dig Toledo.khm.be” command

In green is the top level domain server which supplied the client with the khm nameserver (which is colored in yellow). In red is the answer (an IP-address record) the client received from the nameserver. The dig command proves to be particularly useful for troubleshooting any DNS-related problems administrators may encounter.

The most important thing about DNS protocol is that it follows a highly hierarchical tree-like architecture and that it approaches addresses from back to front. For more information concerning DNS, please refer to the DNS RFCs [TODO06].



Figure TODO:

## BIND Installation

To install BIND, the following command was given:

sudo apt-get install bind9

## BIND Configuration

### named.conf.local

The named.conf.local file contains all local zones for which this DNS server contains information, and is located in /etc/bind/. The zone khm.lan was added to this file, as well as a reverse lookup zone called 1.168.192.in-addr.arpa :

zone "khm.lan" in {

type master;

file "khm.lan.zone";

};

zone "1.168.192.in-addr.arpa" {

type master;

notify no;

file "reverse-1.168.192";

};

The type master parameter makes this a master DNS server, which gets its information from a local source. The local source is specified right underneath after the file parameter. This refers to the zone file which contains the information to perform DNS services.

### named.conf.options

The named.conf.options file is used for specifying other configuration parameters that are unrelated to the zone specification. This file was configured to forward any DNS queries it is unable to answer to another DNS server to the local router:

forwarders {

192.168.1.254;

};

### Zone files

Two zone files were created in the /var/cache/bind directory, which contains all local zones of the DNS server: khm.lan.zone and reverse-1.168.192.

cd /var/cache/bind

sudo touch khm.lan.zone

sudo touch reverse-1.168.192

khm.lan.zone contained the following:

$TTL 86400 ;max TTL  
$ORIGIN khm.lan.  
@ IN SOA ns.khm.lan. root.khm.lan. (  
 2010040701 ;serial number  
 28800 ;refresh after 8 hours  
 7200 ;retry after 2 hours  
 604800 ;expire after a week  
 3600 ) ;minimum TTL of 1 hour  
@ IN A 192.168.0.152  
@ IN NS khm.lan.  
@ IN MX 10 mail  
www IN CNAME khm.lan

mail IN A 192.168.1.200  
krb IN A 192.168.1.200

reverse-1.168.192 contained the following:

$TTL 86400 ;max TTL

@ IN SOA ns.khm.lan. root.khm.lan. (

2010040701 ;serial number  
 28800 ;refresh after 8 hours  
 7200 ;retry after 2 hours  
 604800 ;expire after a week  
 3600 ) ;minimum TTL of 1 hour

NS ns.khm.lan.

200 IN PTR krb.khm.lan.

Reverse zones are zones that help services find back the FQDN (Fully Qualified Domain Name) linked to a certain address. This is – as the name clearly implies – a *reversed* DNS lookup.

These reverse zones are of vital importance for running Kerberos, as tickets sent across the network are linked to the FQDN of the Kerberos server.

### resolv.conf

The resolv.conf file is used in the Linux system to determine which DNS server to contact for any DNS queries. The server was configured in resolv.conf to have it contact itself:

domain khm.lan  
search khm.lan  
nameserver 192.168.1.200

### hostname files

Several files are used on Ubuntu to determine the hostname of the computer. When doing lookups for itself, it will use the following files: /etc/hosts, /etc/hostname and /etc/host.conf.

/etc/hosts was modified to contain the following:

127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts

::1 localhost ip6-localhost ip6-loopback

fe00::0 ip6-localnet

ff00::0 ip6-mcastprefix

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

ff02::3 ip6-allhosts

/etc/hostname was changed to contain the following value:

krb.khm.lan

The order parameter in /etc/host.conf was switched around:

# The "order" line is only used by old versions of the C library.

order bind,hosts

multi on

Finally, the hostname command was executed to also change the hostname:

sudo hostname krb.khm.lan

## Testing BIND

The following command was issued to test the zone files:

sudo rndc reload

server reload successful

Then, the BIND server was restarted:

sudo /etc/init.d/bind9 restart

Finally, DNS functionality was tested with the following commands:

ping krb.khm.lan

PING khm.lan (192.168.1.200) 56(84) bytes of data.

64 bytes from krb.khm.lan (192.168.1.200): icmp\_seq=1 ttl=64 time=0.028 ms

64 bytes from krb.khm.lan (192.168.1.200): icmp\_seq=2 ttl=64 time=0.051 ms

64 bytes from krb.khm.lan (192.168.1.200): icmp\_seq=3 ttl=64 time=0.050 ms

dig krb.khm.lan

; <<>> DiG 9.5.1-P2.1 <<>> krb.khm.lan

;; global options: printcmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32172

;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:

;krb.khm.lan. IN A

;; ANSWER SECTION:

krb.khm.lan. 86400 IN A 192.168.1.200

;; AUTHORITY SECTION:

khm.lan. 86400 IN NS khm.lan.

;; ADDITIONAL SECTION:

khm.lan. 86400 IN A 192.168.1.200

;; Query time: 0 msec

;; SERVER: 127.0.0.1#53(127.0.0.1)

;; WHEN: Fri May 21 16:09:21 2010

;; MSG SIZE rcvd: 80

dig -x 192.168.1.200

; <<>> DiG 9.5.1-P2.1 <<>> -x 192.168.1.200

;; global options: printcmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2006

;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:

;200.1.168.192.in-addr.arpa. IN PTR

;; ANSWER SECTION:

200.1.168.192.in-addr.arpa. 86400 IN PTR krb.khm.lan.

;; AUTHORITY SECTION:

1.168.192.in-addr.arpa. 86400 IN NS ns.khm.lan.

;; Query time: 0 msec

;; SERVER: 127.0.0.1#53(127.0.0.1)

;; WHEN: Fri May 21 16:11:26 2010

;; MSG SIZE rcvd: 91

# Step 2: MIT Kerberos V

The actual installation, configuration and administration of Kerberos is rather exhaustive, with many possible parameters. The contents of a configuration file will be explained when required.

## Kerberos Installation

MIT Kerberos V was compiled and installed from source to obtain This installation was then overwritten with the krb5-kdc and krb5-admin-server packages, as these contain several configuration files that make further configuration easier.

As Ubuntu 9.04 uses MIT Kerberos 1.6.4 Beta 1 in its packages, this version was downloaded from the MIT Kerberos site (<http://web.mit.edu/kerberos/>), compiled and installed:

apt-get install krb5-admin-server

apt-get install krb5-kdc

## Kerberos Configuration

Before creating a new realm, the krb5.conf configuration file must be modified.

### krb5.conf

The krb5.conf file, located in the /etc directory, contains Kerberos configuration information concerning the locations of KDCs and admin servers for the Kerberos realms of interest, defaults for the current realm and for Kerberos applications, and mappings of hostnames onto Kerberos realms. It in fact identifies the different realms to the Kerberos server.

The krb5.conf contains several default entries about the standard MIT realms. These entries were deleted and replaced with the following entries:

[libdefaults]

default\_realm = KHM.LAN

forwardable = true

proxiable = true

krb4\_convert = false

krb4\_get\_tickets = false

[realms]

KHM.LAN = {

kdc = krb.khm.lan

admin\_server = krb.khm.lan

default\_domain = khm.lan

}

[domain\_realm]

.khm.lan = KHM.LAN

khm.lan = KHM.LAN

[logging]

kdc = FILE:/var/log/krb5/kdc.log

admin\_server = FILE:/var/log/krb5/admin.log

default = FILE:/var/log/krb5/general.log

Note that Kerberos realms are case sensitive. It is best practice to name a Kerberos realm after the used domain with all letters capitalized. For instance, our domain khm.lan would use as its realm name KHM.LAN. There are different sections in this file all headed by the name of the section in square brackets, similar to the structure of a Windows .INI file.

**[libdefaults]**

The libdefaults section contains default values used by the Kerberos library. Note that unlike the DNS namespace, MIT Kerberos realms are not hierarchical: it is impossible to make “subrealms” such as accounting.khm.lan and expect that they inherit the principals of khm.lan. In this case, multiple realms would have to be made.

**[realms]**

The realms section contains subsections containing information about the different realms. In the above example, only one realm is specified.

The subsection specifies that the KDC and the admin server can both be found on krb.khm.lan.  
It is unnecessary to specify the entire path for the kdc and admin\_server parameter if the default\_domain has been set.

**[domain\_realm]**

The domain\_realm section contains relations which map domain names and subdomains onto Kerberos realm names. In the example above, the khm.lan domain is linked to the KHM.LAN realm.

**[logging]**

The logging section can be used to specify where Kerberos will keep its logs. By default, Kerberos logs to syslog. Keeping things separated allows for a better overview. These log files are not created by default. Touching them with the touch command will create them:

cd /var/log

sudo mkdir krb5

sudo touch krb5/admin.log

sudo touch krb5/kdc.log

sudo touch krb5/general.log

### kdc.conf

The kdc.conf configuration file, found in /etc/krb5kdc, contains several parameters concerning the Kerberos KDC. The default values were kept, changing only the realm from EXAMPLE.COM to KHM.LAN:

[kdcdefaults]

kdc\_ports = 750,88

[realms]

KHM.LAN = {

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 = aes256-cts:normal arcfour-hmac:normal des3-hmac-sha1:normal des-cbc-crc:normal des:normal des:v4 des:norealm des:onlyrealm des:afs3

default\_principal\_flags = +preauth

}

### kadm5.acl

kadm5.acl, found in /etc/krb5kdc, contains the access list Kerberos utilizes when authenticating users. It defines the user access rights within Kerberos. There is no need to specify entries in the access lists for normal users, as they only need the standard privileges.

Admin users, however, require all privileges. After initializing the Kerberos realm (see below), the kadm5.acl file was edited to contain the following:

\*/admin@KHM.LAN \*

The access list abides by a fixed syntax. The general naming syntax is spec@realm, where spec is composed of different components separated by a “/”. The first component defines the username and the second – if specified – defines the user role. Because of this, an admin can also log in with normal user privileges.

An asterisk is used as a wildcard. The configuration above gives all users with the role admin all privileges in the realm KHM.LAN.

## Kerberos Administration

### Initializing the Kerberos realm

Having configured the krb5.conf file correctly, the realm now needs to be initialized. By running the krb5\_newrealm command, the realm specified as the default realm is initialized.

**sudo krb5\_newrealm**

You will be prompted for a password. Enter one and confirm it. This password can be used to decrypt the Kerberos database if needed, and is used as the main administrative password for the realm.

### kadmin.local

The kadmin.local command is a command that can only be used on the administrative server itself. Instead of accessing the Kerberos server over the network with the Kerberos protocol to authenticate, the kadmin.local directly reads the Kerberos database present in the local file system. This is of course only possible as a user with sufficient rights on the local machine. This command was issued to gain access to the administration server:

sudo kadmin.local

### listprincs

This command lists all the principals in the database.

kadmin.local:  listprincs  
K/M@KHM.LAN  
kadmin/admin@KHM.LAN  
kadmin/changepw@KHM.LAN  
kadmin/history@KHM.LAN  
krbtgt/KHM.LAN@KHM.LAN

### addprinc

This command is used to add a new principal.

The following command was issued to create a new principal called admin with all privileges:

kadmin.local:  addprinc root/admin  
WARNING: no policy specified for root/admin@KHM.LAN; defaulting  
to no policy  
Enter password for principal "root/admin@KHM.LAN":  
Re-enter password for principal "root/admin@KHM.LAN":  
Principal "root/admin@KHM.LAN" created.

### kadmin

Now that an administrative principal has been created, it can be used to log in using kadmin instead of kadmin.local. As we said before this authenticates using the Kerberos protocol and will therefore prompt you for a password. Note that Kerberos will try to authenticate as the user that initiates the authentication. In this example, it will log in as root because this command is issued as a superuser.

sudo kadmin  
Authenticating as principal root/admin@KHM.LAN with password.  
Password for root/admin@KHM.LAN:  
kadmin: listprincs  
K/M@ KHM.LAN  
kadmin/admin@KHM.LAN  
kadmin/changepw@ KHM.LAN  
kadmin/history@ KHM.LAN  
krbtgt/KHM.LAN@ KHM.LAN  
root/admin@ KHM.LAN

The listprincs command was issued again. Note that the principal root/admin was added to the list.

### Adding users

As mentioned previously in this thesis, Kerberos does not keep any information about the user itself. Its only task is to authenticate principals. Applications are usually configured in such a way to search through a database to determine the privileges a certain user has for that application.

To continue, a principal with the same username and password as the main user was created:

kadmin:  addprinc khmuser

## Setting up Kerberos clients

### Obtaining tickets manually

The klist command can be used to observe which tickets the current user has:

klist  
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc\_1000)

The kinit command is used to acquire a Ticket Granting Ticket:

kinit  
Password for khmuser@KHM.LAN:

The klist command was issued again. Note that the user khmuser received a Ticket Granting Ticket in the form of krbtgt@KHM.LAN from the ticket granting server.

klist

Ticket cache: FILE:/tmp/krb5cc\_1000

Default principal: khmuser@KHM.LAN

Valid starting Expires Service principal

05/21/10 18:33:05 05/22/10 04:33:05 krbtgt/KHM.LAN@KHM.LAN

renew until 05/22/10 18:33:03

The kdestroy command is used to remove all tickets in the credential cache:

kdestroy

klist

klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc\_1000)

Note that it is possible to kinit as any user by simply issuing their user principal name after the kinit command. If a kinit is initiated without a username as a parameter, Kerberos will by default attempt to kinit with the username of the user who issued the command.

### Obtaining tickets automatically

Ideally, a kinit will be done when a user logs in, so he or she no longer needs to supply any more passwords. However, this is advised for single access machines, as the tickets a user receives are stored locally can be copied and used on another machine or by another use on the same machine for the duration of the validity of each ticket.

In order to have the system perform a kinit on startup, certain PAM files have to be edited. PAM (Pluggable Authentication Module) is a mechanism which integrates multiple authentication systems in one centralized API [TODO07]. The PAM mechanism will not be discussed in-depth as it is beyond the scope of this thesis. However, it is advised to have a look at the PAM documentation to understand how it functions.

The PAM module package libpam-krb5 was installed:

sudo apt-get install libpam-krb5

This package assures the interconnectivity between Kerberos and PAM.

Then, the following files were modified to configure the PAM module, all of which are in /etc/pam.d:

common-auth: This file contains a list of the authentication modules that define the central authentication scheme for use on the system. The file was modified to include the following line:

auth [success=2 default=ignore] pam\_krb5.so minimum\_uid=1000

common-session: This file contains information on the tasks that should be performed at the start and at the end of a session. In order to pass on the user password to Kerberos, the following line was added:

session optional pam\_krb5.so minimum\_uid=1000 use\_first\_pass

After these modifications, the khmuser was logged out and logged in again. The klist command was issued to confirm the functionality of the PAM module:

klist

Ticket cache: FILE:/tmp/krb5cc\_1000\_DmOE8l

Default principal: khmuser@KHM.LAN

Valid starting Expires Service principal

05/21/10 18:55:19 05/22/10 04:55:19 krbtgt/KHM.LAN@KHM.LAN

renew until 05/22/10 18:55:18

# Step 3: Kerberized services

This chapter will devote its attention to the installation and configuration of several Kerberized services. The first example will be explained more extensively in order to understand the steps necessary to configure a Kerberized service.

All Kerberized clients that MIT has provided as examples were downloaded using the following command:

sudo apt-get install krb5-clients

## telnet

### telnet configuration

The following command was issued to install a Kerberized network daemon:

sudo apt-get install krb5-telnetd

Then, a connection to the Kerberized telnet server was attemped:

telnet.krb5 -x krb.khm.lan

Trying 192.168.1.200...

Connected to krb.khm.lan (192.168.1.200).

Escape character is '^]'.

Waiting for encryption to be negotiated...

Negotiation of authentication, which is required for encryption,

has failed. Good-bye.

A connection could not be made because the service telnet/www.khm.lan is not known with the KDC. Looking in the KDC log file at /var/log/krb5/kdc.log, we find the following:

May 21 19:15:27 krb.khm.lan krb5kdc[2811](info): TGS\_REQ (1 etypes {1}) 192.168.1.200: UNKNOWN\_SERVER: authtime 1274493616, khmuser@KHM.LAN for host/krb.khm.lan@KHM.LAN, Server not found in Kerberos database

The KDC requires a principal in the form of *service@SERVER* to be present in order to provide authentication for that service. As you can also see in the log, the telnet service is known to Kerberos as “host”. Therefore, the following principal was added:

kadmin.local: addprinc -randkey host/krb.khm.lan

Contrary to the user principals that have a USER/ROLE@REALM layout, the service principals have a SERVICE/SERVER@REALM layout. Note how the –randkey parameter was used to generate a random key. There is no need for the key to be known by the administrator because it only needs to be handled by the KDC and the telnet server.

This secret key may only be known by the KDC and the telnet server. The telnet server uses a keytab file to store its secret key. In order to create a keytab file, the ktadd command was issued in the administrative interface of the KDC:

ktadd host/krb.khm.lan

Entry for principal host/krb.khm.lan with kvno 3, encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5.keytab.

Entry for principal host/krb.khm.lan with kvno 3, encryption type ArcFour with HMAC/md5 added to keytab WRFILE:/etc/krb5.keytab.

Entry for principal host/krb.khm.lan with kvno 3, encryption type Triple DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.

Entry for principal host/krb.khm.lan with kvno 3, encryption type DES cbc mode with CRC-32 added to keytab WRFILE:/etc/krb5.keytab.

The krb5.keytab file should then have been copied in a secure manner to the telnet server, but as the KDC server will also house the available services for the sake of simplicity, this was not necessary.

Using the klist command, it is possible to look at the contents of a keytab file:

sudo klist -kte /etc/krb5.keytab

Keytab name: FILE:/etc/krb5.keytab

KVNO Timestamp Principal

---- ----------------- --------------------------------------------------------

3 05/21/10 19:20:00 host/krb.khm.lan@KHM.LAN (AES-256 CTS mode with 96-bit SHA-1 HMAC)

3 05/21/10 19:20:00 host/krb.khm.lan@KHM.LAN (ArcFour with HMAC/md5)

3 05/21/10 19:20:00 host/krb.khm.lan@KHM.LAN (Triple DES cbc mode with HMAC/sha1)

3 05/21/10 19:20:00 host/krb.khm.lan@KHM.LAN (DES cbc mode with CRC-32)

### Testing telnet

As of now it is possible telnet to krb.khm.lan. The following command was issued to test this functionality:

telnet.krb5 -x krb.khm.lan

Trying 192.168.1.200...

Connected to krb.khm.lan (192.168.1.200).

Escape character is '^]'.

Waiting for encryption to be negotiated...

[ Kerberos V5 accepts you as ``khmuser@KHM.LAN'' ]

done.

Last login: Fri May 21 19:26:14 from krb

Linux krb.khm.lan 2.6.28-11-generic #42-Ubuntu SMP Fri Apr 17 01:57:59 UTC 2009 i686

Note that either the –a or the -x parameter needs to be issued. The –a parameter attempts automatic authentication, but does not encrypt the telnet data. The –x parameter attempts to ensure encrypted communication, authenticating the user in the process. Otherwise, the server will refuse the connection.

## rlogin

### Testing rlogin

rlogin requires the same kind of configuration as telnet. As such, everything was already set up to use this Kerberized service. The following command was issued to confirm this:

rlogin -x krb.khm.lan

This rlogin session is encrypting all data transmissions.

Last login: Fri May 21 19:33:24 from krb

Linux krb.khm.lan 2.6.28-11-generic #42-Ubuntu SMP Fri Apr 17 01:57:59 UTC 2009 i686

Note that rlogin, too, will fail to connect when not specifying the –x parameter.

## Secure Shell (SSH)

Secure Shell is preferred over telnet and rlogin considering the security it provides [TODO08]. There is no preconfigured Kerberized package for SSH. The OpenBSD SSH daemon will be downloaded and configured for use with Kerberos authentication.

### SSH Installation

The following command was issued to install the SSH daemon:

sudo apt-get install openssh-server

### SSH Configuration

Both the client and server files have to be configured to use Kerberos authentication. Both files can be found at /etc/ssh.

The sshd\_config configuration file is responsible for the server configuration. The following lines were added or uncommented in order to enable Kerberos authentication:

# Kerberos options  
KerberosAuthentication yes  
KerberosTicketCleanup yes

# GSSAPI options  
GSSAPIAuthentication yes  
GSSAPICleanupCredentials yes

These parameters specify that GSSAPI and Kerberos authentication will be enabled. Users will thus henceforth be authenticated through the KDC.

The parameters also specify that the user’s ticket and credential cache files will automatically be destroyed upon logging out.

The ssh\_config file needs to be reconfigured on every client that will be using Kerberized SSH. The file was modified to include the following parameters:

GSSAPIAuthentication yes  
GSSAPIDelegateCredentials yes

These parameters specify that GSSAPI authentication will be used and that the user credentials will be delegated to the server.

SSH uses the same “host” principal as telnet and rlogin to authenticate, so further configuration is not needed.

The SSH daemon was then restarted with the following command:

sudo /etc/init.d/ssh restart

### Testing SSH

In order to test if SSH is properly Kerberized, the following command was issued:

ssh krb.khm.lan -l khmuser

Linux krb.khm.lan 2.6.28-11-generic #42-Ubuntu SMP Fri Apr 17 01:57:59 UTC 2009 i686

The programs included with the Ubuntu system are free software;

the exact distribution terms for each program are described in the

individual files in /usr/share/doc/\*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by

applicable law.

To access official Ubuntu documentation, please visit:

http://help.ubuntu.com/

Last login: Fri May 21 19:48:42 2010 from krb.khm.lan

The –l parameter specifies the user account with which a user wishes to log in. It is also possible to log in as another user via SSH, but the original user will then be prompted for that user’s password.

## File Transfer Protocol

### FTP Installation

A Kerberized version of the FTP daemon is available for download. The following command was issued to download and install the package:

sudo apt-get install krb5-ftpd

### FTP Configuration

The FTP daemon requires both the “host” principal as well as a “ftp” principal in order to function. The following commands were issued to add the “ftp” principal to the KDC and the keytab:

kadmin.local: addprinc -randkey ftp/krb.khm.lan

kadmin.local: ktadd ftp/krb.khm.lan

### Testing FTP

The following command was issued to test the functionality of Kerberized FTP:

ftp krb.khm.lan

Connected to krb.khm.lan.

220 krb.khm.lan FTP server (Version 5.60) ready.

334 Using authentication type GSSAPI; ADAT must follow

GSSAPI accepted as authentication type

GSSAPI authentication succeeded

Name (krb.khm.lan:khmuser):

232 GSSAPI user khmuser@KHM.LAN is authorized as khmuser

Remote system type is UNIX.

Using binary mode to transfer files.

ftp>

It is possible that the FTP server asks for a name; in that case, just press the enter key.

# References

[TODO01] http://www.hitmill.com/computers/kerberos.html

Todo02 <http://www.kerberos.org/software/tutorial.html>

Todo03 Bronnen van images van Kerberos stappen

Todo04 <http://users.tkk.fi/u/autikkan/kerberos/AIWSC03_kerberos_replay_attacks.pdf>

Todo05 <http://tldp.org/HOWTO/Kerberos-Infrastructure-HOWTO/overview.html>

Todo06 <http://www.dns.net/dnsrd/rfc/>

Todo07 <http://en.wikipedia.org/wiki/Pluggable_Authentication_Modules>

Todo08 <http://en.wikipedia.org/wiki/Secure_Shell>

[1] Novel, “Novel securelogin”, <http://www.novell.com/products/securelogin/media.html>, 2010

###### [2] Cheryl Gribble, “Tutorial for Kerberos”, <http://www.hitmill.com/computers/kerberos.html>, 1 februari 2009

###### [3]MIT, “Kerberos: The network authentication protocol”, <http://web.mit.edu/Kerberos/>, 8 april 2010

###### [4] Ninja, “Ninja lezingendavond over internetinfrastructuur”, <http://ninja4ru.wordpress.com/2008/02/25/ninja-lezingenavond-over-internet-infrastructuur/>, 25 februari 2008

###### [5]Docelic, “MIT Kerberos installation on Debian”, <http://www.debian-administration.org/articles/570#krb-install>, 5 december 2007

[6]MIT, “Install the master KDC”, <http://web.mit.edu/Kerberos/krb5-1.3/krb5-1.3.5/doc/krb5-install.html#Install%20the%20Master%20KDC>, 1995-2002

[7]OpenSuse community, “Example Configuration of Kerberos 5 Server on OpenSUSE 10.2”, <http://en.opensuse.org/Howto_KRB_server>, 25 april 2009

[8] Romain Lenglet, “How to install Kerberos 5”, <http://www.berabera.info/oldblog/lenglet/howtos/kerberoshowto/index.html>, 8 november 2005

[9] Raxxal, “DNS Server with Suse Linux 10.0”, <http://en.opensuse.org/DNS_server_with_SUSE_Linux_10.0>, 13 januari 2006

[10] Lars M. Hansen, “Writing zone files”, <http://www.hansenonline.net/Networking/zonefile.html>, 1999-2005

[11]Wiki community, “Domain Name System”, <http://en.wikipedia.org/wiki/Domain_Name_System>

[12]zytrax.com, “Chapter 5: OpenLDAP Samples”, <http://www.zytrax.com/books/ldap/ch5/index.html#step1-ldif>

[13] Stone, “Automatic kinit upon login”, <http://yoten.blogspot.com/2009/04/automatic-kinit-upon-login.html>, 22 april 2009